From: admin@xwiki.org
Subject: [xwiki-notifications] XWiki updates, 108 pages have been modified since 2017/09/05
Date: 6 September 2017 at 09:48
To: notifications@xwiki.org

Hello

Contents

- InternalDataModel
- Documentation.DevGuide
  - Data Model
- Documentation.DevGuide.DatabaseSchema
  - XWiki Attachment (xwikiattachment)
- Documentation.DevGuide.DatabaseSchema
  - XWiki List Classes (xwikislistclasses)
- Documentation.DevGuide.DatabaseSchema
  - XWiki Space (xwikispace)
- Documentation.DevGuide
  - Database Schema
- Documentation.DevGuide
  - DsXWikiListClasses
- Documentation.DevGuide.FAQTutorial
  - FAQTutorialAWM
  - FAQTutorialManual
  - FAQTutorial
- Documentation.DevGuide.FrontendResources
  - Integrating JavaScript Libraries in XWiki
- Documentation.DevGuide.FrontendResources
  - LiveValidation UI Component
- Documentation.DevGuide.FrontendResources
  - Using jQuery and jQuery UI
- Documentation.DevGuide
  - Front-end Resources
- Documentation.DevGuide
  - GroovyClassHelloWorldTutorial
  - GroovyNotificationTutorial
  - IncludeInVelocity
  - IntegratingJavaScriptLibraries
  - InternalDataModel
  - InternationalizingApplications
  - LiveValidation
  - MasterDetailTutorial
  - PerformingAsynchronousTasks
  - SavingData
  - Scripting
- Documentation.DevGuide.Scripting
  - Scripting API Guide
- Documentation.DevGuide
  - Scripting
- Documentation.DevGuide.Scripting
  - Velocity Training
- Documentation.DevGuide.Scripting
  - HQL Scripting Examples in Velocity
- Documentation.DevGuide
  - SkinExtensionsTutorial
  - Skins
  - Standard URL Format
  - TranslationsTutorial
  - Tutorials
- Documentation.DevGuide.Tutorials
  - Creating a form with validation and tooltips
- Documentation.DevGuide.Tutorials
  - Creating a Tree View
- Documentation.DevGuide.Tutorials
  - Creating an Extension

- Documentation.DevGuide.Tutorials
  - Creating New XClass Property Types
- Documentation.DevGuide.Tutorials
  - Creating Plugins
- Documentation.DevGuide.Tutorials
  - Custom Mapping
- Documentation.DevGuide.Tutorials.FAQTutorial
  - Creating a FAQ application (AWM)
  - Creating a FAQ Application (Manual)
  - FAQ Tutorial
- Documentation.DevGuide.Tutorials
  - Creating a Groovy Class
- Documentation.DevGuide.Tutorials
  - Old Notifications Tutorial
- Documentation.DevGuide.Tutorials
  - Writing Internationalized XWiki Applications
- Documentation.DevGuide.Tutorials
  - Master Detail Tutorial
- Documentation.DevGuide.Tutorials
  - Performing Asynchronous Tasks
- Documentation.DevGuide.Tutorials
  - Saving Data
- Documentation.DevGuide.Tutorials
  - Skin Extension Tutorial
- Documentation.DevGuide.Tutorials
  - Skins
- Documentation.DevGuide.Tutorials
  - Tutorial for Internationalizing a Form
- Documentation.DevGuide.Tutorials
  - Extension Point Tutorial
- Documentation.DevGuide
  - Tutorials & Guides
- Documentation.DevGuide.Tutorials
  - Creating XWiki Components
- Documentation.DevGuide.Tutorials
  - Writing a custom SOLR search for an AppWithinMinutes application
- Documentation.DevGuide.Tutorials
  - Writing an Event Listener
- Documentation.DevGuide.Tutorials
  - Writing XWiki Rendering Macros
- Documentation.DevGuide.Tutorials.WritingMacros
  - Writing XWiki Rendering Macros in wiki pages
- Documentation.DevGuide
  - UIXTutorial
  - URL Architecture
- Documentation.DevGuide.URL Architecture
  - Standard URL Format
- Documentation.DevGuide
  - URL Architecture
- Documentation.DevGuide
  - UsingJQuery
  - Developer Guide
  - WikiMacroTutorial
  - WritingComponents
  - WritingCustomSolrSearchForAWM
  - WritingEventListenerTutorial
  - WritingMacros
  - XWikiLinks

- ~~XWIKILINKS~~
- XWikiVelocityTraining
- Documentation.DevGuide
  - XWikiVelocityTraining
  - XWikiVelocityTrainingCodeMacro
  - XWikiVelocityTrainingImageMacro
- Documentation.DevGuide
  - XWikiVelocityTrainingCodeMacro
  - XWikiVelocityTrainingImageMacro
  - velocityHqlExamples
- FAQ
  - How to include a velocity page into another page?
- XWiki
  - Profile of Vincent Massol
  - Profile of Ecaterina Moraru (Valica)
  - Profile of Ilie Andriuta

Developers,

This message is sent by XWiki. Here are the pages in your watchlist that have been modified since the last notification:

dev

Drafts

User Cookbook

Drafts.XEDocUserCookbook

Between 2017/09/05 and 2017/09/05, the page has been modified 5 times, by 1 user: Vincent Massol

If you have a great recipe ([[How-to>>http://en.wikipedia.org/wiki/How-to]]), share with other chefs ~~;)~~!! Please consider our
~~[[Guidelines>>#HGuidelinesforaddingnewrecipes]]~~
[[Guidelines>>||anchor="HGuidelinesforaddingnewrecipes"]] below.
~~{{code}}~~**{{code}}{{info}} The recipe is valid for:**
~~{{info}} The recipe is valid for:~~
~~{{/info}}~~**{{/info}}{{/code}}**
~~{{/code}}~~
~~{{info}} The recipe is valid for:~~**{{info}}**
The recipe is valid for:
~~{{code}}~~**{{code}}{{velocity}}**
~~{{velocity}}~~
~~{{/velocity}}~~**{{/velocity}}{{/code}}**
~~{{/code}}~~
~~{{warning}}Maybe the macro "includeForm()" is not the best choice here, see [[including macros>>~~
~~http://platform.xwiki.org/xwiki/bin/view/DevGuide/IncludeInVelocity]]. Please improve this recipe if you know it better.{{/warning}}~~
{{warning}}
Maybe the macro "includeForm()" is not the best choice here, see [[including macros>>xwiki:FAQ.IncludeInVelocity]]. Please improve this recipe if you know it better.
{{/warning}}

~~{{code}}~~**{{code}}{{velocity}}**
~~{{velocity}}~~
~~{{/velocity}}~~**{{/velocity}}{{/code}}**
~~{{/code}}~~
~~{{info}} The recipe is valid for:~~**{{info}}**
The recipe is valid for:

- author: xwiki:XWiki.krokosjablik ⇨ xwiki:XWiki.VincentMassol
- syntax: XWiki 2.0 ⇨ XWiki 2.1

IRC

IRC Archive for channel #xwiki-notifications on 05 September 2017

IRC.xwiki-notificationsArchive20170905

On 2017/09/05, the page has been created by Vincent Massol

<xwikibot> has joined #xwiki-notifications

IRC.IRCBotArchiveClass
channel:
xwiki-notifications

- title: ⇨ IRC Archive for channel #xwiki-notifications on 05 September 2017
- parent: ⇨ IRC.WebHome
- author: XWiki.XWikiGuest ⇨ xwiki:XWiki.VincentMassol
- syntax: XWiki 2.1 ⇨ Plain 1.0

extensions

Extension

Help Application

Extension.Help Application

On 2017/09/05, the page has been modified by Admin

ExtensionCode.ExtensionClass
description:

properties:
maven.groupid=org.xwiki.platform maven.artifactid=xwiki-platform-help-ui
maven.Model=org.xwiki.platform:xwiki-platform-help-ui:xar:9.7
~~xwiki.extension.recommendedVersions.platform=org.xwiki.commons:.*/[9.7], org.xwiki.rendering:.*/[9.7], org.xwiki.platform:.*/[9.7]~~
~~xwiki.extension.recommendedVersions.commons=org.xwiki.commons:.*/[9.7] xwiki.extension.recommendedVersions=org.xwiki.commons:.*/[9.7], org.xwiki.rendering:.*/[9.7], org.xwiki.platform:.*/[9.7]~~**xwiki.extension.recommendedVersions.platform=org.xwiki.commons:.*/[9.7],
org.xwiki.rendering:.*/[9.7],
org.xwiki.platform:.*/[9.7]
xwiki.extension.recommendedVersions.commons=org.xwiki.commo**

ns:.*/[9.7]
xwiki.extension.recommendedVersions=org.xwiki.commons:.*/[9.7],
org.xwiki.rendering:.*/[9.7],
org.xwiki.platform:.*/[9.7]

- author: xwiki:XWiki.evalica ⇨ xwiki:XWiki.Admin

XWiki Multipage PDF Export

Extension.Multipage PDF Export

On 2017/09/05, the page has been modified by Anca Luca

ExtensionCode.ExtensionClass
description:

ExtensionCode.ExtensionCompatibilityClass
notes:

ExtensionCode.ExtensionCompatibilityClass
notes:

XWiki.XWikiComments
comment:

XWiki.XWikiComments
comment:

XWiki.XWikiComments
author:
xwiki:XWiki.lucaa
comment:
> It would be nice to be able to omit the title of each (but first) document.
Well, this looks like quite a specific use-case and I doubt that such a rule could have room in a generic tool like what this intends to be. The purpose of this pdf export tool is to export a dynamic list of pages where all pages are "equal", they're just a collection of pages. If your usecase is about //assembling// a bigger document from its chapters which are divided in individual documents, you could actually create, as a wiki document, the document that represents the assembled document (using the [[include macro>>Extension.Include Macro]] and/or [[display macro>>Extension.Display Macro]] and custom headings when they're needed) and then export this assembled document using the standard XWiki export function. This way you can fine grain the hierarchy between the different sub-documents and also fully control the titles that appear in the table of contents. This works, of course, if you know all the documents you're including in the assembled document and you can handle that "statically".
date:
2017-09-05 15:58:37.0
replyto:
2

- author: xwiki:XWiki.ppantiru ⇨ xwiki:XWiki.lucaa

platform

AdminGuide

Installation_ExportImport09

AdminGuide.Installation_ExportImport09

On 2017/09/05, the page has been deleted by Vincent Massol

UpgradeXWiki09Installation

AdminGuide.UpgradeXWiki09Installation

On 2017/09/05, the page has been deleted by Vincent Massol

Test

Cluster

Test.Cluster.WebHome

Between 2017/09/05 and 2017/09/05, the page has been modified 4 times, by 1 user: Thomas Mortagne

> change ~~1~~**2**

> XWiki.Mail
> text:
> **toto**

On 2017/09/05, the page has been created by Thomas Mortagne

> **change 1**

- author: XWiki.XWikiGuest ⇨ xwiki:XWiki.ThomasMortagne
- defaultLanguage: ⇨ en

xwiki

DevGuide

DatabaseSchema

DevGuide.DatabaseSchema.WebHome

On 2017/09/05, the page has been modified by Vincent Massol
On 2017/09/05, the page has been created by Vincent Massol

Documentation.DevGuide

API

Documentation.DevGuide.API

Documentation.DevGuide.API

On 2017/09/05, the page has been deleted by Vincent Massol

Documentation.DevGuide

API Reference

Documentation.DevGuide.API.WebHome

On 2017/09/05, the page has been created by Vincent Massol

~~{{velocity}}~~**1 XWiki API Reference**
~~#set($versionStable = "9.7")~~
~~##set($versionDev = "9.7-rc-1")~~
~~#set($versionOldStable = "8.4.5")~~
~~{{/velocity}}~~
XWiki's ~~APIs~~**API** are the ~~methods that the XWiki development team consider safe for~~**objects and properties** you ~~to use (i.e. for which backward compatibility is guaranteed). They~~ can ~~be called from Java or~~ directly from your ~~wiki pages [[using~~**documents [using** a scripting ~~language>>Documentation.DevGuide.Scripting]] (Velocity, Groovy, Python, Ruby, etc).~~**language>DevGuide.Scripting] (either Velocity or Groovy).**
~~{{info}}~~**The API documentation is [available as JavaDoc>http://build.xpertnet.biz/doc/api/com/xpn/xwiki/api/Api.html].**
~~As users of the XWiki API you need to be aware of some important points:~~
~~* You should never use a class having ##internal## in its package. It means this class is not a public API and XWiki developers can change it at any time. If you depend on such a class you're on your own and the XWiki Development Team doesn't guarantee its stability.~~
~~* You should pay attention to the ##@Unstable## annotation. When you see code marked with this annotation it means it's a new public API that it still considered unstable and that can change at any time too. See [[Unstable Annotation>>dev:Community.DevelopmentPractices||anchor="H40UnstableAnnotation"]] for more details.~~
~~{{/info}}~~
~~== Scripting Reference Documentation ==~~
~~* For XWiki versions >= 6.2.5+ you should install the [[Scripting Documentation Application>>extensions:Extension.Scripting Documentation Application]] in your wiki. You can [[browse a version of it installed on xwiki.org>>platform:ScriptingDocumentation.WebHome]] (but beware, it'll show the Scripting API available on the XWiki version installed for xwiki.org, which may differ from your locally installed XWiki version).~~
~~* For XWiki versions <= 5.0.x you can browse the old [[Scripting Reference Documentation>>platform:SRD.Navigation||queryString="xpage=embed"]] (a.k.a SRD).~~
~~== How to find a class ==~~
~~You can get the maven module where to find a Java class by using [[XWiki Nexus Search page>>http://nexus.xwiki.org/nexus/index.html#nexus-search;classname~~]]. Put the complete class name and you will get all modules containing it. Nexus also allows you to see what's in those artifacts and read the javadoc.~~
~~== JavaDoc ==~~
~~XWiki is made of various extensions/modules and each module offers API for both developers coding in Java and for Scripts written in wiki pages. In the future our goal is to have Javadoc links directly from each extension page located in the [[Extensions Wiki>>extensions:Main.WebHome]].~~

{{velocity}}
Right now we're providing all Javadocs for all modules:
* [[Javadoc for XWiki ${versionStable} (stable)>>Javadoc||queryString="versionId=$versionStable&versionName=stable"]]
#if ($versionDev)
* [[Javadoc for XWiki ${versionDev} (dev)>>Javadoc||queryString="versionId=$versionDev&versionName=dev"]]
#end
#if ($versionOldStable)
* [[Javadoc for XWiki ${versionOldStable} (LTS)>>Javadoc||queryString="versionId=$versionOldStable&versionName='LTS'"]]
#end
* (((
{{html}}
<form action="$xwiki.getURL('Javadoc')" class="xformInline">
<label for="javadocVersionId">Older version:</label>
<input id="javadocVersionId" type="text" size="30" name="versionId" class="withTip" value="e.g., 3.2.1, 4.1-milestone-1, 5.1-rc-1"/>
<input type="hidden" name="versionName" value="LTS"/>
<input class="button" type="submit" value="View"/>
</form>
{{/html}}
)))
{{/velocity}}
The [[full Javadoc>>http://maven.xwiki.org/site/docs/]] is also available ({{info}}but only for old versions, we need to fix this{{/info}}).
Older Javadocs of XWiki can be found in our [[Maven Release repository>>http://maven.xwiki.org/releases/]]. For example for XWiki Platform Core Javadocs, check the files suffixed by ##javadoc## for the version you wish under [[this directory>>http://maven.xwiki.org/releases/org/xwiki/platform/xwiki-platform-oldcore/]].

XWiki.XWikiComments
- author:
XWiki.msenn1
- comment:
Javadoc link is broken; needs single quoted parameters within the HREF double quotes.
URL effectively ends at getPlugin(
Also, I am having lots of difficulty adding this comment. I keep getting Velocity errors.
- date:
2007-05-03 16:28:58.0

XWiki.XWikiComments
- author:
xwiki:XWiki.VincentMassol
- comment:
Mark, the problem wasn't the single quote. It was simply that in our last upgrade we didn't turn on the zipexplorer plugin...
- date:

~~2007-05-04 11:42:58.0~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.Alex~~
- comment:
~~This application doesn't works. After some seconds exploring, it doesn't show anything. And all the site starts to show error processing Html macro. The error's details says something like zip closed.~~
~~Vicent, is it related with this zipexplorer plugin? How can I turn it on in XE 7.3?~~
~~Alex~~
- date:
~~2015-11-23 09:39:27.0~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.VincentMassol~~
- comment:
~~@Alex: Please use the [[Forum/Mailing List>>dev:Community.MailingLists]] to discuss.~~
- date:
~~2015-11-23 09:45:04.0~~
- replyto:
~~2~~

- title: API Reference ⇨
- parent: xwiki:Documentation.DevGuide.WebHome ⇨ DevGuide.WebHome
- syntax: XWiki 2.1 ⇨ XWiki 1.0

Documentation.DevGuide

APIGuide
Documentation.DevGuide.APIGuide

On 2017/09/05, the page has been deleted by Vincent Massol

Architecture
Documentation.DevGuide.Architecture

On 2017/09/05, the page has been deleted by Vincent Massol

Documentation.DevGuide.Architecture.URL Architecture

Standard URL Format
Documentation.DevGuide.Architecture.URL Architecture.Standard URL Format.WebHome

On 2017/09/05, the page has been modified by Vincent Massol
- parent: xwiki:Documentation.DevGuide.URL Architecture.WebHome ⇨

xwiki:Documentation.DevGuide.Architecture.URL Architecture.WebHome
On 2017/09/05, the page has been created by Vincent Massol

~~{{box cssClass="floatinginfobox" title="**Contents**"}}~~**1 URLs resolution**
~~{{toc/}}~~
~~{{/box}}~~
~~The general format~~**In a single instance** of XWiki **you have many sections and the** URLs ~~is {{code language="none"}}<protocol>://<server>/<context path>/<type>/<path>{{/code}}, where:~~**need to reflect all that. This documentation details the way URLs are parsed by XWiki and what means each part of it.**
~~* ##<protocol>##: by default it's ##http##. It's possible to use ##https## (See [[how to modify the wiki descriptor>>manager:AdminGuide.EditWikiDescriptor]]).~~**http://wiki1.xwiki.org/xwiki/bin/view/SomeSpace/SomePage/attachment.ext?param1=value1&param2=value2**
~~* ##<server>##: the XWiki instance address and port, e.g. ##localhost:8080##, ##xwiki.org##. In domain-mode the server also contains the subwiki name. See [[Accessing a wiki>>platform:AdminGuide.AccessWiki]]. In path-based mode, the server is the address of the main wiki.~~
~~* ##<context path>##: the name under which the XWiki web application is deployed in your Servlet Container, e.g. ##xwiki##. See [[Short URLs>>platform:Main.ShortURLs#HI.Applicationname]] if you wish to deploy XWiki as ROOT.~~
~~* ##<type>##: the type of URL depending on the XWiki subsystem accessed. The following types are supported:~~
~~** ##bin##: points to a dynamic resource. Examples: an Entity to display (wiki, page, space, attachment, object, property, etc), a dynamic skin resource (image, css, js), a dynamic template resource (image, js, css, icon, etc) and more. By dynamic we mean that some computation needs to happen to compute the resource (for example some Velocity needs to be evaluated).~~
~~** ##wiki##: this is similar to ##bin## but specifying the subwiki to access, in path-based mode (See [[Accessing a wiki>>platform:AdminGuide.AccessWiki]]).~~
~~** ##rest##: points to a [[REST Resource>>platform:Features.XWikiRESTfulAPI]]~~
~~** ##skins##: points to a static filesystem skin resource (image, js, css, etc), i.e. a resource located inside a skin directory on the filesystem.~~
~~** ##resources##: points to a static template resource (image, js, css, etc) on the filesystem.~~
~~** ##xmlrpc##: points to the now deprecated XMLRPC API (see [[extensions:Extension.XML-RPC Integration]] for more details).~~
~~** ##webdav##: points to a WebDAV resource (see [[Web DAV>>platform:Features.WebDAV]] for more details).~~
~~* ##<path>##: the path completely depends on the ##<type>## used. See below for more details.~~
~~{{info}}~~**1.1 Protocol**
~~If you're interested~~**By default "http" is used when generating an url** in ~~configuring the XWiki URLs, check the [[URL Architecture page>>URL Architecture]].~~**XWiki. You can set "secure" parameter to true in a wiki descriptor to generate urls with "https" in "http".**
~~{{/info}}~~
~~= Type: ##bin## =~~**1.1 Host**
*** The first thing XWiki do to find the right wiki the URL is pointing to is by trying to find <tt>wiki1.xwiki.org</tt> in the list of wiki descriptors.**

* The descriptor can be found:
** if the host is an IP (127.0.0.1, 85.65.12.36, etc.) XWiki goes to main wiki
** if the first part of the host name (what is before the first point) is "www" XWiki goes to main wiki
** if the first part of the host name match a wiki identifier XWiki goes to this wiki
{{info}}1.1 \[OPTIONNAL\] Application context
XWiki 7.2: Added support for Nested Spaces#warning("TODO")
{{/info}}
There are various ##bin## URLs. The generic format of ##bin## URLs is ##bin/<action>/<path>##. The full list of ##<action>## is defined in the Strut Configuration file (located in ##WEB-INF/struts-config##).1.1 \[OPTIONNAL\]/bin/
#warning("TODO")
Here are some well-known actions:
* [[##skin##>>platform:DevGuide.Standard URL Format#HAction:skin]]: Dynamic Skin resource, e.g. ##http:~/~/<server>/xwiki/bin/skin/resources/uicomponents/search/searchSuggest.css##1.1 \[OPTIONNAL\] Action
* ##jsx##: [[Javascript Skin Extension>>platform:DevGuide.SkinExtensionsTutorial]] URL, e.g. ##http:~/~/<server>/xwiki/bin/jsx/AnnotationCode/Settings?language=en##XWiki use "action" identifiers to determine what is the goal a the user when going to a page.
* ##ssx##: [[StyleSheet Skin Extension>>platform:DevGuide.SkinExtensionsTutorial]] URL, e.g. ##http:~/~/<server>/xwiki/bin/ssx/AnnotationCode/Style?language=en##
* ##webjars##: URL when [[accessing some resources located in a WebJar>>extensions:Extension.WebJars Integration]], e.g. ##http:~/~/<server>/xwiki/bin/webjars/resources/path?value=requirejs%2F2.1.15%2Frequire.min.js##
* [[##download##>>platform:DevGuide.Standard URL Format#HAction:download]]: URL to access an attachment from a wiki page, e.g. ##http:~/~/<server>/xwiki/bin/download/XWiki/JohnDoe/john.png?width=30&height=30&keepAspectRatio=true##
* [[##view##>>platform:DevGuide.Standard URL Format#HAction:view]]: Display Entity content (wiki, page, space, attachment, object, property, etc) with the skin applied, e.g. ##http:~/~/<server>/xwiki/bin/view/Sandbox/WebHome##
* ##edit##: Edits wiki content (page, object, property, etc), e.g. ##http:~/~/<server>/xwiki/bin/edit/Sandbox/WebHome##
* [[##get##>>platform:DevGuide.Standard URL Format#HAction:get]]: Accessing Entity content (wiki, page, space, attachment, object, property, etc) without any skin applied, e.g. ##http:~/~/<server>/xwiki/bin/get/Main/WebHome?xpage=xpart&vm=commentsinline.vm##. The ##get## action should be used to render the content of the page and nothing else (mainly used for code pages). This action is similar to the ##view## action, but without the UI and with "text/plain" mime type set by default for the response.
* ##export##: Exports wiki pages to various formats (PDF, HTML, etc). See [[platform:Features.Exports]].
* [[##viewrev##>>platform:DevGuide.Standard URL Format#HAction:viewrev]]: This action is the same as the ##view## action but allowing to address a given revision of the document asked.
* [[##create##>>platform:DevGuide.Standard URL Format#HAction:create]]: Provides the UI for creating new documents, e.g.

~~Format/Path/Action.create]]. Provide the URL for creating new documents, e.g. ##http:~/~/<server>/xwiki/bin/create/Space/NewDocument##~~
~~== Action: ##view## ==~~**For example if the action is "edit" XWiki will print a editor the user use to modify a page content but if the "view" action is used XWiki will execute the page content to print/render the content.**
~~The additional URL paths for the ##view## action is ##<space1>/.../<spaceN>/<page>?<query string>## or ##<space1>/.../<spaceN>?<query string>##~~**1.1 Space**
All XWiki page are located in what is called a "space". You have only one level of space but you can indicate a "parent space" for a space.
~~Examples:~~**1.1 Page**
The page is what contains wiki contents and objects/classes and attachments.
~~* ##http:~/~/<server>/xwiki/bin/view/Sandbox/WebHome##: Views the ##Sandbox.WebHome## page.~~**1.1 Attachment**
~~* ##http:~/~/<server>/xwiki/bin/view/Sandbox##: Views~~**This part indicate** the ~~##Sandbox.WebHome##~~**attachment you want to download from a particular** page.
~~* ##http:~/~/<server>/xwiki/bin/view/Space1/Space2##: Views the ##Space1.Space2.WebHome## page (if the ##Space2## page doesn't exist. If it exists then it views the ##Space1.Space2## page).~~
~~* ##http:~/~/<server>/xwiki/bin/view/Space1/Space2/##: Force to always views the ##Space1.Space2.WebHome## page (even if the ##Space2## page exists in the ##Space1## space. Note the ##/## at the end).~~
~~* ##http:~/~/<server>/xwiki/bin/view/Space1/Space2?spaceRedirect=false##: Force to always view the ##Space1.Space2## terminal page (even if the ##Space1.Space2.WebHome## page exists and even if the ##Space1.Space2## page may not exist).~~
~~Here are some of the supported query string parameters.~~**1.1 Parameters**
Actions can use URL parameters to modify there action depending of there value.
~~=== Parameter: ##xpage## ===~~**For example you can make a velocity script that print the content of this value with:**
~~* ##plain##: this value indicates that only the content of the page is printed (no UI). With a document that's not using the XWiki Syntax 1.0 you can control the output syntax using ##outputSyntax## and ##outputSyntaxVersion##. You can also pass other parameters to allow fine-grained control:~~**{code}**
~~** ##raw=1##: Returns~~**The value of** the ~~page source as is without any formatting but XML-escaped~~**parameter is: $request.param1**
~~** ##raw=2##: Returns the page source as is without any formatting (without any escaping)~~**{code}**
~~** ##htmlHeaderAndFooter=true##: {{info}}Since 6.3RC1{{/info}} Display~~**if you have a parameter "param1=Hello World"** the ~~HTML headers (##<html>##,##<head>##,##<body>##, etc...) and footers (##</body>##, ##</html>##, etc...).~~**result will be**
~~** ##outputTitle##: {{info}}Since 7.3M2{{/info}} If ##true## then also display the title in addition to the content. Defaults to ##false## if not specified.~~
{code}
~~* ##xml##: print the exported form~~**The value** of ~~a document. This is what you can find in~~ the ~~xar for each document.~~**parameter is: Hello World**
~~* ##rev##: when specified, redirects to the ##viewrev## action (see below)~~
{code}
~~* ##children##: {{info}}Since 7.2M3{{/info}} display a page that lists all the children of the document. You can also add the following parameters:~~
~~** ##hierarchy=parentchild##: do the same but display the children according to the (old) parent/child mechanism~~

** ##displayHidden=true##: display the hidden documents (no matter what the user's configuration is)
* ##backlinks##: {{info}}Since 7.2M3{{/info}} display a page that lists all the documents having a link to the current document. You can also add the following parameters:
** ##displayHidden=true##: display the hidden documents (no matter what the user's configuration is)
* ##siblings##: {{info}}Since 7.2RC1{{/info}} display a page that lists all the children of the parent of the document. You can also add the following parameter:
** ##displayHidden=true##: display the hidden documents (no matter what the user's configuration is)
* ##hierarchy_reference##: {{info}}Since 7.3M2{{/info}} display the hierarchy of the page, as raw HTML, without any style. You can also add the following parameters:
** ##id=someId##: add an HTML id to the hierarchy breadcrumb.
** ##displayTitle=true|false##: display the title of the elements in the breadcrumb (default: true)
** ##excludeSelf=true|false##: display the hierarchy without the page itself (default: false)
** ##limit=##: maximum number of hierarchy elements displayed (default: none)
** ##local=true|false##: do not display the main wiki reference for documents located in sub-wikis. (default: false)
=== Parameter: ##outputSyntax## and ##outputSyntaxVersion## ===
With a document that's not using the XWiki 1.0 syntax it's possible to explicitly indicate the renderer to use for the page content. For example with ##outputSyntax=plain## you will get the result in plain text.
##outputSyntaxVersion## is an optional parameter, if it's not provided XWiki will search for the last version of the renderer for the given ##outputSyntax## syntax name. Note that ##outputSyntax## controls only the document content output syntax. If you only want the document content you can use ##xpage=plain##.
See [[extensions:Extension.Rendering Module]] for more details on renderers.
=== Parameter: ##viewer## ===
Note: if the document does not exist, you will have a 404 error page, and the following parameters will not be taken into account. Use ##xpage## if you need to have the desired behavior even if the document does not exist.
* ##children##: {{info}}Since 7.2RC1{{/info}} display a page that lists all the children of the document. You can also add the following parameters:
** ##hierarchy=parentchild##: do the same but display the children according to the (old) parent/child mechanism
** ##displayHidden=true##: display the hidden documents (no matter what the user's configuration is)
* ##backlinks##: {{info}}Since 7.2RC1{{/info}} display a page that lists all the documents having a link to the current document. You can also add the following parameter:
** ##displayHidden=true##: display the hidden documents (no matter what the user's configuration is)
* ##siblings##: {{info}}Since 7.2RC1{{/info}} display a page that lists all the children of the parent of the document. You can also add the following parameter:
** ##displayHidden=true##: display the hidden documents (no matter what the user's configuration is)
=== Parameter: ##spaceRedirect## ===
Accepted values:

* ##true## (default): By default, an URL not ending in ##/## resolves to a non-terminal page, unless the equivalent terminal page already exists.
* ##false##: An URL not ending in ##/## will always resolve to a terminal page.
Not applicable (i.e. no effect) for top-level pages and URLs ending in ##/##.
Examples:
* ##http:~/~/<server>/xwiki/bin/view/Space1/Space2[?spaceRedirect=true]##: Views the ##Space1.Space2.WebHome## page (if the ##Space2## page doesn't exist. If it exists then it views the ##Space1.Space2## page).
* ##http:~/~/<server>/xwiki/bin/view/Space1/Space2?spaceRedirect=false##: Force to always view the ##Space1.Space2## terminal page (even if the ##Space1.Space2.WebHome## page exists and even if the ##Space1.Space2## page may not exist).
* ##http:~/~/<server>/xwiki/bin/view/Space1[?spaceRedirect=false|?spaceRedirect=true|/]##: Will always view the ##Space1.WebHome## non-terminal page because it is a top-level page and we can never have a top-level terminal page.
== Action: ##viewrev## ==
=== Parameter: ##rev## ===
The version to get, e.g. ##2.3## in ##http:~/~/<server>/xwiki/bin/viewrev/MySpace/MyPage?rev=2.3##
== Action: ##save## ==
The additional URL paths for the ##save## action is ##<space>/<page>?<query string>##.
Examples:
* ##http:~/~/<server>/xwiki/bin/save/Sandbox/WebHome##: Save the ##Sandbox.WebHome## page.
Here are some of the supported query string parameters.
=== Parameter: ##creator## ===
Change the creator of the current page. Should be a string as a Document Reference like ##JohnDoe##, ##XWiki.JohnDoe## or ##xwiki:XWiki.JohnDoe## which is a page containing an ##XWiki.Users## object.
=== Parameter: ##language## ===
Will target a specific language. You must use a specific 2-letters code (see [[I18N>>http://platform.xwiki.org/xwiki/bin/view/Features/I18N]]).
=== Parameter: ##objectPolicy## ===
{{info}}Since 7.1M1{{/info}} The ##objectPolicy## parameter is a request configuration parameter that will decide how to handle custom query string parameters (see [[Custom Parameters>>platform:DevGuide.Standard URL Format#HCustomParameters]]):
* ##update##
** Each custom parameter will update a property in an existing object of the current page.
** If the custom parameter refers to a property or an object that don't exist, it will simply ignore it.
* ##updateOrCreate##
** Each custom parameter will update a property in an existing object of the current page.
** If the custom parameter refers to a property that doesn't exist, it will ignore it.
** If the custom parameter refers to an object that doesn't exist, it will create it (and pad will ##null## objects if needed).(((
//Example//: If there is already 2 objects ##XWiki.XWikiRights## in the page, ##XWiki.XWikiRights_0_levels## and ##XWiki.XWikiRights_1_levels## will update these existing objects. ##XWiki.XWikiRights_2_levels## and ##XWiki.XWikiRights_3_levels## will

~~##XWiki.XWikiRights_2_levels## and ##XWiki.XWikiRights_3_levels## will create 2 new objects. ##XWiki.XWikiRights_42_levels## will be created and 37 empty objects too (42 minus the 4 already existing and minus the 42th which will be created).~~
~~//Note//: All this information about empty/null objects is currently an implementation detail of how we currently handle object numbers and it may be subject to future change so, as much as possible, do not rely on it and, instead, use the public available APIs to handle these things.~~
~~)))~~
~~=== Parameter: ##parent## ===~~
~~Allows you to set the parent of the page. Should be a string as a Document Reference like ##WebHome##, ##Main.WebHome## or ##xwiki:Main.WebHome##.~~
~~=== Parameter: ##syntaxId## ===~~
~~Specify which syntax is used for the content of the current page (see [[Supported syntaxes>>http://rendering.xwiki.org/xwiki/bin/view/Main/#HSupportedSyntaxes]]).~~
~~=== Parameter: ##template## ===~~
~~Allows you to have a specific template to apply for the current page. Should be a string as a Document Reference like ##MyTemplate##, ##Main.MyTemplate## or ##xwiki:Main.MyTemplate##.~~
~~=== Parameter: ##title## ===~~
~~Allows you to modify the title of the page.~~
~~=== Custom Parameters ===~~
~~Custom parameters have the format ##<Space>.<ClassPage>_<number>_<propertyname>## (e.g. ##XWiki.XWikiUsers_0_password## or ##XWiki.WatchListClass_3_interval##):~~
~~* ##<Space>## must refer to an existing space of the wiki~~
~~* ##<ClassPage>## must refer to a class~~
~~* ##<number>## is a positive number~~
~~* ##<propertyname>## is a name of a property of the class ##<Space>.<ClassPage>##~~
~~== Action: ##inline## ==~~
~~The additional URL paths for the ##inline## action is ##<space>/<page>?<query string>##.~~
~~Examples:~~
~~* ##http:~/~/<server>/xwiki/bin/inline/Sandbox/WebHome##: Edit in inline mode the ##Sandbox.WebHome## page.~~
~~For the supported query string parameters, you can refer to the [[##save## action>>platform:DevGuide.Standard URL Format#HAction:save]].~~
~~== Action: ##get## ==~~
~~=== Parameter: ##htmlHeaderAndFooter## ===~~
~~{{info}}Since 6.3RC1{{/info}} If ##htmlHeaderAndFooter=true## is passed then display the HTML headers (##<html>##,##<head>##,##<body>##, etc...) and footers (##</body>##, ##</html>##, etc...).~~
~~== Action: ##download## ==~~
~~The additional URL paths for the ##download## action is ##<space1>/.../<spaceN>/<page>/<attachment>?<query string>##.~~
~~Example:~~
~~*~~
~~##http:~/~/<server>/xwiki/bin/download/Space1/Space2/WebHome/image.png##: Views the ##image.png## attachment located in the ##Space1.Space2.WebHome## page.~~
~~=== Parameter: ##force-download## ===~~
~~If set, force the attachment to be downloaded instead of being opened in the browser (it sets the ##Content-Disposition## header to ##attachment##~~

instead of ##inline##).
== Action: ##create## ==
The additional URL paths for the create action is ##<space1>/.../<spaceN>/<page>?<query string>##.
Examples:
http:~/~/<server>/xwiki/bin/create/Space1/Space2/NewPage: UI for creating the Space1.Space2.NewPage document (also known as a terminal document).
http:~/~/<server>/xwiki/bin/create/Space1/Space2/WebHome: UI for creating the Space1.Space2.WebHome document (which is also known as a space homepage or a non-terminal document).
http:~/~/<server>/xwiki/bin/create/Main/WebHome: Generic UI for creating a new document (asking for location and document name since Main.WebHome already exists)
Here are some of the supported query string parameters.
=== Parameter: ##spaceReference## ===
{{info}}Since 7.2M1{{/info}} A local reference string serialization of the space in which to create the new document, e.g. "Space1.Space2"
//Note//: Since it's a serialized string reference, entity names need to be already escaped. For a space name "A.B", the equivalent local space reference would be "A\.B". Another example of serialized local space reference would be "A\.B.C\.D", meaning space "C.D" inside space "A.B". See the [[extensions:Extension.Model Module]]'s documentation for more information on EntityReferences.
=== Parameter: ##name## ===
{{info}}Since 7.2M1{{/info}} A string representing the name of the new entity, e.g. "NewDocument".
Depending on the type of document to create, this will be used as:
* a space name, when creating non-terminal documents, e.g. spaceReference=Space&name=Page results in creating "Space.Page.WebHome"
* a document name, when creating terminal documents, e.g. spaceReference=Space&name=Page&tocreate=terminal results in creating "Space.Page"
=== Parameter: ##tocreate## ===
Accepted values:
* empty or not specified:
** if the current document does not exist, a terminal document will be created having the current document's reference, e.g. ##/create/Space/NewPage## creates "Space.NewPage"
** if the current document exists, the new document (specified using the ##spaceReference## + ##name## parameters pair) is created as a non-terminal document, e.g. ##/create/Main/WebHome?spaceReference=Space&name=NewPage## creates "Space.NewPage.WebHome".
** {{warning}}Deprecated since 7.2M1{{/warning}} if the current document exists, the new document (specified using the deprecated ##space## + ##pageName## parameters pair) is created as a terminal document, e.g. ##/create/Main/WebHome?space=Space&page=NewPage## creates "Space.NewPage".
* "terminal" - forces the new document to be a terminal document.
** Example: If "Space1.Space2.Page" is to be created, then it is already a terminal document and is not affected in any way by this setting, but if "Space1.Space2.WebHome" is to be created then it would be a non-terminal document by default. In this later case, specifying the tocreate=terminal parameter would force the creation of a terminal document instead, result in the creation of ##Space1.Space2## (i.e. the

page named "Space2" in the space "Space1"), effectively going up one level in the Nested Spaces hierarchy.
** In the case of a top level space's WebHome document (e.g. "Space.WebHome"), this setting is ignored, since the top limit of the hierarchy is reached and there is nowhere else to go up. The resulting document to be created is thus "Space.WebHome".
* "space" - {{warning}}Deprecated since 7.2M1{{/warning}} The document to be created will always be a space WebHome (a.k.a a non-terminal document). This value is considered only when used in conjunction with the deprecated ##space## parameter. In this case, the deprecated ##page## parameter is completely ignored. For example: ##tocreate=space&space=MySpace## will result in "MySpace.WebHome".
=== Parameter: ##template## ===
A string serialized document reference of the document to use as template when creating the new document.
If empty, the new document will be a blank page.
If not specified, and existing template providers are available to choose from in the current space and the ##templateprovider## parameter is also not specified, no document will be created yet and the create UI will show up asking the user to choose a template provider from a list.
This parameter has precedence over the ##templateprovider## parameter, meaning that if both are specified, the ##templateprovider## will be ignored.
=== Parameter: ##templateprovider## ===
A string serialized document reference of the template provider document to use when creating the new document. The actual template document to use will be extracted from the specified template provider.
If the specified template provider is not allowed to be used in the space where the new document is created, then the create UI will be displayed with an error.
//Note//: This check is done for the ##spaceReference## of the new document to be created. Example: If we are creating the terminal document "Space.NewPage", the check is done for the space "Space". However, if we are creating the non-terminal document "Space.NewPage.WebHome", the check is done for the non-terminal document's space "Space" and not for actual document's reference space "NewPage".
If empty, the new document will be a blank page.
If not specified, and existing template providers are available to choose from in the current space and the ##template# parmeter is also not specified, no document will be created yet and the create UI will show up asking the user to choose a template provider from a list.##
If both this parameter and the ##template## parameter are specified, the ##template## parameter will be used instead of and this parameter will be ignored.
=== Parameter: ##title## ===
A string to use as the title of the new document.
=== Parameter: ##parent## ===
{{warning}}Deprecated since 7.2M1{{/warning}} A string serialized document reference of the document to use as parent for the new document.
=== Parameter: ##space## ===
{{warning}}Deprecated since 7.2M1{{/warning}} A string representing an unescaped space name where the new document will be created, Example: "MySpace", "SpaceNameWith.Dots", etc.
//Note//: Because this is just a space name and not a space local reference, only top level space documents can be created, i.e. "TopLevelSpace.Document" is possible but not "TopLevelSpace.SecondLevelSpace.Document"

~~TopLevelSpace.SecondLevelSpace.Document~~.
~~=== Parameter: ##pageName## ===~~
~~{{warning}}Deprecated since 7.2M1{{/warning}} A string representing the name of the new document. Example: "NewDocument"~~
~~//Note//: When using ##tocreate=space## in conjunction with the ##space## parameter, this parameter will be ignored and the document "<space>.WebHome" will be created.~~
~~== Action: ##skin## ==~~
~~Right now dynamic Skin Resources are handled as Entity Resources ([[which is not correct>>https://jira.xwiki.org/browse/XWIKI-12449]]).~~
~~The additional URL paths for the ##skin## action is~~
~~* ##<filesystem path to resource>?<optional query string>## for Filesystem-based Skins~~
~~* ##<space1>/.../<spaceN>/<page>/<attachment name>## for a resource attached to a wiki page. Note that Velocity is executed only if the attachment has a CSS or Javascript extension name.~~
~~* ##<space1>/.../<spaceN>/<page>/<xobject property name>## for Skins defined in wiki pages and having an ##XWiki.XWikiSkins## xobject with a property named ##<xobject property name>##. Note that Velocity is executed only if the ##<xobject property name>## name has a CSS or Javascript extension name.~~
~~Examples:~~
~~* ##http:~/~/<server>/xwiki/bin/skin/skins/flamingo/style.css?skin=XWiki.DefaultSkin&colorTheme=playground%3AFlamingoThemes.Charcoal##: Points to the ##style.css## resource located in the ##[xwiki webapp location]/skins/flamingo## directory on the filesystem (and if not found look for the resource in the base skin and default base skin).~~
~~*~~
~~##http:~/~/<server>/xwiki/bin/skin/resources/uicomponents/search/searchSuggest.css##: Points to the ##searchSuggest.css## resource located in the ##[xwiki webapp location]/resources/uicomponents/search## directory on the filesystem.~~
~~*~~
~~##http:~/~/<server>/xwiki/bin/skin/uicomponents/search/searchSuggest.css##: Similar as just above except that the ##resources## part has been omitted and if the resource doesn't exist in ##[xwiki webapp location]/uicomponents/search## then it's also searched for in ##[xwiki webapp location]/resources/uicomponents/search##~~
~~* ##http:~/~/<server>/xwiki/bin/skin/XWiki/DefaultSkin/somefile.css##: Points to the ##somefile.css## attachment in the ##XWiki.DefaultSkin## wiki page (and if not found look for the resource in the base skin and default base skin).~~
~~* ##http:~/~/<server>/xwiki/bin/skin/XWiki/DefaultSkin/edit.vm##: Points to the content of the ##edit.vm## property in the ##XWiki.XWikiSkins## xobject in the ##XWiki.DefaultSkin## wiki page (and if not found look for the resource in the base skin and default base skin).~~
~~= Type: ##wiki## =~~
~~{{info}}~~
~~XWiki 7.2: Added support for Nested Spaces~~
~~{{/info}}~~
~~The format is ##wiki/<wiki name>/<bin subpath>## where:~~
~~* ##<wiki name>##: the name of the subwiki, e.g. ##platform##.~~
~~* ##<bin subpath>##: the same path as for the ##bin## action (see above).~~
~~For example, if you use ##http:~/~/<main wiki server>/xwiki/bin/download/XWiki/JohnDoe/john.png## you would use ##http:~/~/<main wiki server>/xwiki/wiki/<subwiki name>/download/XWiki/JohnDoe/john.png## to access the same attachment but located in a subwiki named ##<subwiki>##.~~

~~= Type: ##resources## =~~
~~Static template resources, e.g.~~
~~##http:~/~/<server>/xwiki/resources/js/prototype/prototype.js##.~~
~~= Type: ##skins## =~~
~~Static skin resources, e.g.~~
~~##http:~/~/<server>/xwiki/skins/flamingo/logo.png##.~~
~~= Type: ##temp## =~~
~~Dynamic temporary resources, bound to a specific document and a specific module.~~
~~The format is ##temp/<Space>/<Document>/<module>/<file.ext>##, where:~~
~~* ##<Space>## and ##<Document>## identify the target document; currently only the last part of the space is supported~~
~~* ##<module>## is the name of the module that generated the resource, a short string used for avoiding conflicts between different modules generating temporary files with the same name~~
~~* ##<file.ext>## is the name of the temporary file~~
~~== Parameter: ##force-download## ==~~
~~If set to ##1##, force the file to be downloaded instead of being opened in the browser (it sets the ##Content-Disposition## header to ##attachment##).~~
~~== Parameter: ##force-filename## ==~~
~~[since 9.0RC1]~~
~~If ##force-download## is set, the value of ##force-filename## is used in the ##Content-disposition## HTTP header to indicate the name of the file to the browser.~~
~~= Resource Handler based types =~~
~~Since 6.1 any extension can dynamically register (as components) "resource handlers" (not to mix with the "resources" type).~~
~~See [[known resource types>>extensions:Extension.Resource API||anchor="#HKnownResourceTypes"]]~~

url.png: 1.2 ⇨ ()

- title: Standard URL Format ⇨
- parent: xwiki:Documentation.DevGuide.WebHome ⇨
- author: XWiki.VincentMassol ⇨ XWiki.ThomasMortagne
- syntax: XWiki 2.0 ⇨ XWiki 1.0

Documentation.DevGuide.Architecture

URL Architecture

Documentation.DevGuide.Architecture.URL Architecture.WebHome

On 2017/09/05, the page has been created by Vincent Massol

~~{{box cssClass="floatinginfobox" title="**Contents**"}}~~
~~{{toc/}}~~
~~{{/box}}~~
{{image ~~reference="url-architecture-overall.png" width="800px"/}}~~**reference="url-architecture-overall.png"/}}**
~~= Standard URL Scheme =~~**== Customizing URLs ==**
~~This is the default URL format used by XWiki. See [[Standard URL Scheme>>Standard URL Format]].~~
~~= Customizing URLs =~~
* You can simply register new Resource Types or new Entity Actions. See the ~~[[Resource API>>extensions:Extension.Resource~~**[[URL**

~~the [[Resource API~~ **API>>extensions:Extension.URL** API]].
Note that using Tuckey's [[Rewrite Filter>>http://tuckey.org/urlrewrite/]] is very powerful as it also allows using dynamic rules written in Java by using the [[##<class-rule>##>>http://tuckey.org/urlrewrite/manual/4.0/index.html#class-rule]] ~~feature. However, [[dynamic outbound rules are not yet possiblet>>https://code.google.com/p/urlrewritefilter/issues/detail?id=73]].~~**feature.**
~~[[Here's an example of writing a custom ##class-rule##>>https://github.com/xwiki-contrib/url-rewrite-reference]].~~
~~= Struts Servlet Architecture =~~
~~As shown above, when there's no Resource Reference Handler registered to handle a Resource Type, the rest of ##web.xml## is processed and if the URL type is ##bin## or ##wiki##, the Struts Servlet is then called and the following happens:~~
~~{{image reference="xwiki-struts.png" width="800px"/}}~~
~~For example for the ##http:~/~/www.xwiki.org/xwiki/bin/view/Main/WebHome## URL, the following happens:~~
~~* The URL path contains ##/view/## and thus the ##ViewAction.java## class is executed (configured in ##struts-config.xml##) and called by the Struts Action, itself configured in ##web.xml##.~~
~~* The wiki is the main wiki in this example (since the server name starts by ##www##). See [[Accessing a wiki>>AdminGuide.AccessWiki]] for more details on how wikis are accessed from URLs.~~
~~* ##ViewAction## specifies that ##view.vm## will be rendered using Velocity~~
~~* The space asked is ##Main## and the document's name is ##WebHome##~~

url-architecture-overall.png: 1.1 ⇨ ()

xwiki-struts.png: 1.1 ⇨ ()

parent: xwiki:Documentation.DevGuide.WebHome ⇨

On 2017/09/05, the page has been modified by Vincent Massol

parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Architecture.WebHome

Documentation.DevGuide

Architecture

Documentation.DevGuide.Architecture.WebHome

On 2017/09/05, the page has been created by Vincent Massol

~~{{box cssClass="floatinginfobox" title="**Contents**"}}~~**= General Architecture =**
~~{{toc/}}~~
~~{{/box}}~~
~~= What is XWiki? =~~
~~XWiki is a [[Second Generation Wiki>>platform:Main.SecondGenerationWiki]] and as such it's a full-fledged wiki but it's also a **runtime platform for developing collaborative web applications**. It's very versatile and you can modify any part of the XWiki~~

~~applications~~. It's very versatile and you can modify any part of the XWiki runtime to match your requirements.
When you use a traditional development language such as Java, Python, Perl, etc you start from scratch and assemble building blocks to create the application you wish:
{{image reference="traditional.png" width="650px"/}}
XWiki takes a different approach: it's already a running application (we call it a "runtime platform") and you can tune the various modules it's made of to morph this runtime into your target application: you configure existing modules, you remove some, you add some, till you get what you want.
{{image reference="morphing.png" width="650px"/}}
This approach has several advantages:
* Works from day one, immediate feedbacks
* Iterative, Need-based development: You have something working from day one and you can have users using it right away, and modify it as you see needs emerge or user request features, at the rythm you wish.
* Continuous Delivery: Tuning it, adding new modules, removing some other, are all operations that you can do live on a running XWiki. Of course you could have 2 XWiki instances: a staging one and a production one, and copy from one to another if you want to control your changes (and put changes in a SCM).
* Work collaboratively on creating applications: Since everything is done in wiki pages, designers can style while devs create logic.
* Efficient: what you get with XWiki is akin to what you get when you edit content in a WYSIWYG editor: fast productivity since you immediately see the results of your actions (no need to compile, deploy, etc).
= General Architecture =
Architecture Concepts:
* XWiki is made of [[Extensions>>extensions:Extension.WebHome]].
[[Extensions>>Extension.WebHome]].
* XWiki ~~Standard~~**Enterprise** is a generic Flavor that allow users to collaboratively author content. In the near future, several other Flavors will be available.
* There are 2 types of ~~Extensions (and it's possible to extends Extension Manager to support other kinds):~~**Extensions:**
{{image reference="container.png" width="650px"/}}
There are two options for integrating XWiki into your environment:
* Accessing your environment (other software for example) from XWiki and displaying information inside XWiki's UI. This can be achieved by [[scripting>>platform:DevGuide.Scripting]] inside Wiki pages or by developing Java code. See also the following [[Integration Extensions>>extensions:Main.Tags||queryString="do=viewTag&tag=integration"]].
* Access XWiki's content (wiki pages, user data, etc) from other software by accessing XWiki remotely using [[REST APIs>>platform:Features.XWikiRESTfulAPI]] (It's also possible to use [[XMLRPC>>extensions:Extension.XML-RPC Integration]] or {{scm path="xwiki-platform-core/xwiki-platform-gwt/xwiki-platform-gwt-api/src/main/java/com/xpn/xwiki/gwt/api/server/XWikiServiceImpl.java"}}GWT{{/scm}}).
XWiki being a development platform for web applications there are plenty of ways to extend XWiki. The most obvious one is to [[look for an Extension and install it>>extensions:Extension.WebHome]].
However if there's no Extension for what you wish to achieve, here are main options:
* Use [[scripts>>platform:DevGuide.Scripting]] in wiki pages and access all APIs made available by other Extensions. This allows you to develop Applications in your wiki. Make sure to check the [[FAQ

~~Tutorial>>platform:DevGuide.FAQTutorial]] which shows how to do this.~~
~~* Develop some Java code to make available [[Java APIs>>platform:DevGuide.API]] which can then be accessed from wiki pages using Scripting. [[XWiki uses a Component approach to develop Java code>>platform:DevGuide.WritingComponents]]. It's also possible to override any existing XWiki Component with your own implementation, thus changing the default behavior of whatever portion of XWiki that you wish.~~
~~* [[Create some Macros>>platform:DevGuide.WritingMacros]] to use in your wiki pages.~~
~~* [[Create new Skins>>platform:DevGuide.Skins]] or modify existing one to match your needs.~~
~~* [[Create or modify Skin Themes>>extensions:Extension.Flamingo Theme Application]].~~
~~* Use [[Skin Extensions>>platform:DevGuide.SkinExtensionsTutorial]] to customize the CSS or JavaScript used to display wiki pages.~~
~~* Configure what is displayed in various locations of the screen by contributing content for existing [[UI Extension Points>>platform:ExtensionPoint.WebHome]].~~
~~Check the [[Developer Guide>>platform:DevGuide.WebHome]] to learn about all this and more.~~

container.png: 1.1 ⇨ ()

morphing.png: 1.1 ⇨ ()

- title: Architecture ⇨ XWiki Architecture
- parent: xwiki:Documentation.DevGuide.WebHome ⇨

Documentation.DevGuide

BestPractices
Documentation.DevGuide.BestPractices

On 2017/09/05, the page has been deleted by Vincent Massol

Documentation.DevGuide

Best Practices
Documentation.DevGuide.BestPractices.WebHome

On 2017/09/05, the page has been created by Vincent Massol

~~{{box cssClass="floatinginfobox" title="**Contents**"}}~~**1 Best Practices**
~~{{toc/}}~~
~~{{/box}}~~
~~= Where to put code? =~~**1.1 Naming Best Practices**
~~Since xwiki allows you to put code both in wiki pages and in Java you might wonder where you should put your code. Here are some general guidelines:~~**1.1.1 Application**
*** Each application must have its own Space if a user interface is needed.**
~~* Don't put "business logic" code in wiki pages. Use Java for that. This gives you nice IDEs, the ability to easily debug the code and the ability to write automated unit tests. Generally speaking it makes it easy on maintenance.~~**1.1.1 Classes**

* ~~In general put the minimum amount of scripts~~**Classes must be** in ~~your wiki pages since that makes them harder to maintain.~~**the XWiki space**
* The ~~only scripts that you should put in wiki pages (and not in Java code!) are "presentation logic" scripts, i.e. scripts in charge of presenting the data retrieved by using the Java/REST APIs.~~**name must end with "Class" like "TagClass"**
~~Said differently you should use the [[MVC>>http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller]] pattern by separating your Model (what we called "business logic" above) from your View (what we called "presentation logic" above).~~**1.1.1 Macro**
*** Macro must be in the XWiki space**
*** The name must end with "Macro" like "TagMacro"**
~~= XWiki Application Organization =~~**1.1.1 Sheets**
*** Sheets must be in the XWiki space**
*** The name must end with "Sheet" like "TagSheet"**
~~* [[XWiki Development Team best practices>>dev:Community.ApplicationDevelopmentBestPractices]]~~**1.1.1 Templates**
* ~~[[Best Practices~~**Templates must be in the** XWiki ~~Application Organization>>https://goo.gl/1epAJn]] (proposed by Ludovic Dubost)~~**space**
*** The name must end with "Templates" like "TagTemplates"**
~~= Check for Object existence in Class Sheets documents =~~**1.1.1 Objects**
*** Objects must be in their own space**
~~Class sheet documents should be written using~~**TODO: Explain** the ~~following construct (this is an example for displaying documents containing XWiki.XWikiUsers objects):~~**reasons behind these practices**
~~{{code language="velocity"}}~~
~~#set($obj = $doc.getObject("XWiki.XWikiUsers"))~~
~~#if(!$obj)~~
~~1 User Sheet~~
~~This stylesheet must be applied on a document containing a XWiki.XWikiUsers object.~~
~~#else~~
~~1 $msg.get("userProfile", [$xwiki.getUserName($doc.fullName, false)])~~
~~...~~
~~#end~~
~~{{/code}}~~
~~{{info}}~~
~~The 'if' tests first for the non existence. This is so that XWiki can extract the title from the //1 User Sheet//, which is a proper title to display when viewing the sheet page, instead of the computed name which will usually display something wrong.~~
~~{{/info}}~~
~~= Handling errors when using xredirect for non-Javascript UIs =~~
~~When writing an UI with JavaScript, AJAX takes care of forwarding your action to a background service replying with success or with an error that is then displayed to the user in the same page.~~
~~Without JavaScript, we usually use the xredirect query parameter to specify the current page (and state) to which we want to come back after performing an action (by pressing a button, link, submitting a form, etc.).~~
~~One common problem when writing UIs without JavaScript in this way is error handling. In other words, how do you handle the situation when the service that you use to perform your action throws an error?~~
~~A simplified code for this in the background service that produces the error is:~~

{{code language="velocity"}}
#handleRequest($success)
#if ($success)
#if ($request.action == 'get' || $request.xpage == 'plain')
JavaScript call here.
Action was successful.
#elseif ("$!request.xredirect" != '')
No JavaScript here. Redirect.
$response.sendRedirect($request.xredirect)
#end
#else
#if ($request.action == 'get' || $request.xpage== 'plain')
JavaScript call here.
Action was NOT successful.
$response.setStatus(403)
#elseif ("$!request.xredirect" != '')
No JavaScript here. What now!? Redirect?
#handleErrorHere($request.xredirect)
#end
#end
{{/code}}
The idea is that you want to pass the error message to the UI but you don`t have a clear way of doing it, like you have for AJAX calls (response code and response text). A solution is to use the Session in order to pass your error message. You set the error in the service and, in the UI, you read and remove it so that it is only displayed once.
For the background service, it translates to:
{{code language="velocity"}}
...
#elseif ("$!request.xredirect" != '')
No JavaScript here. Redirect and forward error message.
#set ($errorMessageKeyPrefix = "myModule.error.")
$request.session.setAttribute("${errorMessageKeyPrefix}${request.xredirect}", 'Action was NOT successful')
$response.sendRedirect($request.xredirect)
#end
...
{{/code}}
On the UI side:
{{code language="velocity"}}
...
#set ($xredirect = $doc.getURL($context.action, $!{request.queryString}))
#set ($errorMessageKeyPrefix = "myModule.error.")
#set ($errorMessage =
$request.session.getAttribute("${errorMessageKeyPrefix}${xredirect}"))
#if ("$!errorMessage" != '')
Clean the error and display the message.
#set ($discard =
$request.session.removeAttribute("${errorMessageKeyPrefix}${xredirect}"))
{{error}}$errorMessage{{/error}}
#end
...
{{/code}}
Note that using xredirect's value as session key (prefixed or not) is a good idea because:
1. it's already there in both the UI (for sending it as parameter) and the background service (received as parameter)
1. it acts like a namespace, ensuring that the error will only be displayed for

~~1. it acts like a namespace, ensuring that the error will only be displayed for the current page/request.~~
~~Using a prefix as above allows you to have multiple components (wiki macros, gadgets, etc.) in the same page using the same mechanism without collisions.~~
~~This method works together with the whole purpose for which we are doing the redirect in the first place (so that the user can refresh the page without re-sending the action or re-posting a form), ensuring that after the first display, on a refresh, the error goes away.~~

- title: Best Practices ⇨
- parent: xwiki:Documentation.DevGuide.WebHome ⇨ DevGuide.WebHome
- author: XWiki.evalica ⇨ XWiki.VincentMassol
- syntax: XWiki 2.0 ⇨ XWiki 1.0

Documentation.DevGuide

Creating XWiki Plugin Java
Documentation.DevGuide.Creating XWiki Plugin Java

On 2017/09/05, the page has been deleted by Vincent Massol

Creating a form with validation and tooltips
Documentation.DevGuide.Creating a form with validation and tooltips

On 2017/09/05, the page has been deleted by Vincent Massol

CreatingATreeView
Documentation.DevGuide.CreatingATreeView

On 2017/09/05, the page has been deleted by Vincent Massol

Documentation.DevGuide

CreatingExtensions
Documentation.DevGuide.CreatingExtensions.WebHome

On 2017/09/05, the page has been deleted by Vincent Massol

Documentation.DevGuide

CreatingNewXClassPropertyTypes
Documentation.DevGuide.CreatingNewXClassPropertyTypes

On 2017/09/05, the page has been deleted by Vincent Massol

CreatingPlugins
Documentation.DevGuide.CreatingPlugins

On 2017/09/05, the page has been deleted by Vincent Massol

CustomMapping

Documentation.DevGuide.CustomMapping

On 2017/09/05, the page has been deleted by Vincent Massol

Documentation.DevGuide

CustomMapping

Documentation.DevGuide.CustomMapping.WebHome

On 2017/09/05, the page has been deleted by Vincent Massol
On 2017/09/05, the page has been created by Vincent Massol

> XWiki.RedirectClass
> location:
> **xwiki:Documentation.DevGuide.Tutorials.CustomMapping.WebHome**

- hidden: false ⇨ true

Documentation.DevGuide

DataModel

Documentation.DevGuide.DataModel

On 2017/09/05, the page has been deleted by Vincent Massol

Documentation.DevGuide.DataModel

InternalDataModel

Documentation.DevGuide.DataModel.InternalDataModel.WebHome

On 2017/09/05, the page has been modified by Vincent Massol

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.DataModel.WebHome

On 2017/09/05, the page has been created by Vincent Massol

- author: XWiki.VincentMassol ⇨ XWiki.sdumitriu

Documentation.DevGuide

Data Model

Documentation.DevGuide.DataModel.WebHome

On 2017/09/05, the page has been created by Vincent Massol

> ~~{{box cssClass="floatinginfobox" title="**Contents**"}}~~**1 XWiki Data Model**
> ~~{{toc/}}~~
> ~~{{/box}}~~
> XWiki is not just your average wiki engine; it exposes a flexible data model that can be leveraged at the presentation level. Combined with its powerful presentation level scripting capabilities, XWiki's data model can help you build simple to complex Web applications with little or no need to access the ~~XWiki~~**Xwiki** core. In other words, you can build custom applications through the XWiki web interface without having to compile, package, and deploy software components.

This area explains XWiki's data model of Classes, Properties, and Objects for those users who want to build applications primarily at the presentation level. It will help you understand how you can achieve significant ~~features~~**functionality** by programming at this level ~~alone.~~**alone. For information about XWiki's underlying database schema (tables and fields), see: [More on the Data Model>Internals Data Model].**

~~For information about XWiki's underlying database schema (tables~~**1.1 Xwiki Classes, Objects,** and ~~fields), see: [[XWiki's Database Schema>>platform:DevGuide.DatabaseSchema]].~~**Properties = XWiki Classes, Objects,**

If you are familiar with object oriented programming, you understand the concept of classes, properties, and ~~Properties =~~**and objects. The presentation level data model that XWiki exposes is much the same.**

~~If you are familiar with object oriented programming, you understand the concept of classes, properties, and objects. The presentation level data model that XWiki exposes is much the same.~~**<dl>**

<dt>*Classes*</dt>

<dd>An XWiki class is like a template that defines a unique type of object. It defines what properties an object can have. When you define a custom class, you are saying that your application may need to create one or many objects (instances) of that class. The class itself is like a cookie cutter and the objects are unique instances of that class. So, in this analogy, if the class is the cookie cutter, the objects are the cookies (and we are talking about real edible cookies in this analogy; not Web cookies).<dd>

~~; Classes~~**<dt>*Properties*</dt>**

~~: An XWiki class is like a template~~**<dd>Properties are the characteristics** that ~~defines a unique type of object. It defines what properties~~ an object can have. ~~When you define~~**In** a ~~custom class, you are saying that your application may need to create one or many objects (instances) of that class. The~~ class ~~itself is like a cookie cutter and~~**definition,** the ~~objects are~~**properties define the data fields that each** unique ~~instances~~**instance** of ~~that class. So, in this analogy, if~~ the class ~~is the cookie cutter, the objects are the cookies (and we are talking about real edible cookies in this analogy; not Web cookies).~~**can have values for.</dd>**

~~: A class is attached to a page. There can be at most one class per page. The class name is the name of the document it is attached to.~~**<dd>**

XWiki currently supports the following kinds of properties (datatypes) which will be explained later in greater detail:

~~; Properties~~*** Number**

~~: Properties are the characteristics that an object can have. In a class definition, the properties define the data fields that each unique instance of the class can have values for. Properties have displayers which control how the property value is displayed in view or edit mode. Some displayers use advanced pickers to help you input a valid property value.~~*** String**

~~: XWiki currently supports the following kinds of properties (data types):~~*****

TextArea

*** Password**

*** Boolean**

*** Static List**

*** Database List**

*** Date**

</dd>

<dd>You may find the labels for the property types somewhat confusing because in the XWiki Class Editor, they also contain the word 'Class'. For example, you will see Number Class, String Class, TextArea Class, and so forth. It is best if you just ignore the word

'Class' and think of them as simple property datatypes in the traditional sense for now.</dd>
(((<dt>*Objects*</dt>
|= Field Type |= Since Version |= Description |= Screenshots<dd>Objects are unique instances of a class with unique values defined for each of the properties that make up the class.</dd>
|//String// | |Allows to store and display one line texts |\\</dl>
|//TextArea// | |Allows to store and display large text fields (text and wysiwyg)|\\
|//Number// | |Allows to store and display Numbers of type integer,long,float,double|\\
|//Password// | |Allows to store password fields which can be encrypted or hashed |\\
|//EMail// |new in 4.2M3 |Allows to store email fields that can be obfuscated at display time |\\
|//Boolean// | |Allows to store and display boolean values (yes/no or 1/0) which can be displayed as select or checkbox fields |\\
|//Static List// | |Allows to store and display single-select or multi-select fields that can be displayed as select, checkbox, radio or suggests fields. The possible values of static list fields are configured in the field definition.
|//Database List// | |Allows to store and display single-select or multi-select fields that can be displayed as select, checkbox, radio or suggests fields. The possible values of static list fields are taken from a query from other XWiki data.|\\
|//Database Tree List// | |Same as a Database List field, with the addition that the data can be displayed as a tree, using the Yahoo Javascript Tree library |\\
|//Page// |new in 4.2M3 |Same as a Database List field, but made to store XWiki page names. |\\
|//Date// | |Allows to store and display date or datetime values. A date picker is used automatically for this field as shown below. |(((
{{image reference="datePicker.png"/}}
)))
|//User List// | |Allows to store and display single-select or multi-select users. The field uses a user picker as shown below.|(((
{{image reference="userPicker-editSingle-objectEditor.png"/}} {{image reference="userPicker-editMultiple.png"/}} {{image reference="userPicker-viewMultiple.png"/}}
)))
|//Group List// | |Allows to store and display single-select or multi-select groups. The field uses a user picker as shown below.|(((
{{image reference="groupPicker-editMultiple.png"/}}
)))
|//Computed Pseudofield// |new in 4.2M3|Allows to create a pseudo field for which the display can be configured using a custom displayer and the value retrieved by using a script. It allows to combine other fields together.|\\
|//Time Zone//|new in 7.3M1|Allow to display and edit Time Zones.|\\
: Starting with version 4.3M2 you can define [[your own property types>>platform:DevGuide.CreatingNewXClassPropertyTypes]].~~In summary~~:
; Objects
: Objects are unique instances of a class with unique values defined for each of the properties that make up the class.
: An object is attached to a specific page. Each page can have multiple objects.
: Using objects, you insert **structured** information in the wiki, while in a traditional wiki you can only enter unstructured information (plain text). This is one of the reasons why XWiki is a second generation wiki

~~is one of the reasons why XWiki is a second generation wiki.~~
~~//In summary//:~~
~~= Suggested Reading =~~
~~* Armed with your knowledge of Classes and Objects you could try creating a small application, such as a FAQ application, by following this [[FAQ Tutorial>>platform:DevGuide.FAQTutorial.WebHome]].~~
~~* You could also start understanding how to display properties of Objects in wiki pages by using [[platform:DevGuide.Scripting]].~~
~~* You could also try using the [[App Within Minutes extension>>extensions:Extension.App Within Minutes Application]] which simplifies how to get started developing an application with little required knowledge. After your application is generated, if you wish to go further and develop new features for your app, you'll need to use your knowledge about Classes, Objects and [[platform:DevGuide.Scripting]].~~
~~* More information on the [[Custom Display field>>xwiki:FAQ.Bindings available inside the Custom Display property]] and an [[example>>snippets:Extension.User Property Custom Display]]~~
~~)))~~

datePicker.png: 1.1 ⇨ ()

groupPicker-editMultiple.png: 1.1 ⇨ ()

userPicker-editMultiple.png: 1.1 ⇨ ()

userPicker-editSingle-objectEditor.png: 1.1 ⇨ ()

userPicker-viewMultiple.png: 1.1 ⇨ ()

XWiki.XWikiComments
- author:
~~XWiki.XWikiGuest~~
- comment:
~~Hi,~~
~~It would be nice to know, how to create a class (and add properties to it). How to associate "the" class to a page. What's the relation between "the" class and the may different? objects associated to a page. Is a page a class ? How to edit and present/view the values of properties on a page ? best wishes Martin~~
- date:
~~2008-06-09 09:16:42.0~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.BrianJones~~
- comment:
~~I see above all the properties/datatypes that are supported by XWiki, however where can we find more documentation on these. It is mentioned that "Xwiki supports the following kinds of properties (datatypes) which will be explained later in greater detail."~~
~~Where exactly can we find this more in depth look at the datatypes, their respective options and how to configure them?~~
- date:
~~2008-10-23 17:13:18.0~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.VincentMassol~~
- comment:
~~Hi Brian,~~
~~The best way to ask question is through our [[mailing list/forum>>dev:Community.MailingLists]]~~
- date:
~~2008-10-23 17:16:39.0~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.BrianJones~~
- comment:
~~Ahh, thank you kindly for your speedy response Vincent, I wasn't aware of this feature.~~
- date:
~~2008-10-23 17:27:13.0~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.tullio0106~~
- comment:
~~In my opinion is not clear how classes can and object can be used.~~
~~Is it possible reference objects outside the page in which the class is defined ? How ?~~
~~Is it possible create more than one instance of a given class ?~~
- date:
~~2014-05-29 16:14:45.0~~

- title: XWiki Data Model ⇨
- parent: xwiki:Documentation.DevGuide.WebHome ⇨ DevGuide.WebHome
- syntax: XWiki 2.0 ⇨ XWiki 1.0

Documentation.DevGuide.DatabaseSchema

XWiki Attachment (xwikiattachment)
Documentation.DevGuide.DatabaseSchema.DsXWikiAttachment.WebHome

Between 2017/09/05 and 2017/09/05, the page has been modified 3 times, by 2 users: Vincent Massol, Thomas Mortagne

~~|[[image:DevGuide.DatabaseSchema@db_key.gif]]|XWA_ID|BIG_INT(20)|~~
~~[[image:DevGuide.DatabaseSchema@db_check.gif]]||~~**|(%class="fa fa-key"%)((()))|XWA_ID|BIG_INT(20)|(%class="fa fa-check"%)((()))||** |0|
~~|~~
~~[[image:DevGuide.DatabaseSchema@db_field.gif]]|XWA_DOC_ID|BIG_INT(20)|~~**||XWA_DOC_ID|BIG_INT(20)|** || |null|
~~|~~
~~[[image:DevGuide.DatabaseSchema@db_field.gif]]|XWA_FILENAME|VARCHAR(255)|~~

~~[[image:DevGuide.DatabaseSchema@db_check.gif]]~~||XWA_FILENAME|VARCHAR(255)|(%class="fa fa-check"%)((()))|| | |
~~|~~
~~[[image:DevGuide.DatabaseSchema@db_field.gif]]~~||XWA_SIZE|INTEGER|||XWA_SIZE|INTEGER| || |null| Deprecated since 9.0RC1, use XWA_LONGSIZE instead
~~|~~
~~[[image:DevGuide.DatabaseSchema@db_field.gif]]~~||XWA_DATE|DATETIME|~~[[image:DevGuide.DatabaseSchema@db_check.gif]]~~|| |~~0000-00-00 00:00:00~~|||XWA_LONGSIZE|LONG| || |null|
~~|~~
~~[[image:DevGuide.DatabaseSchema@db_field.gif]]~~||XWA_AUTHOR|VARCHAR(255)| || |null~~|~~||XWA_DATE|DATETIME|(%class="fa fa-check"%)((()))|| |0000-00-00 00:00:00|
~~|~~
~~[[image:DevGuide.DatabaseSchema@db_field.gif]]~~||XWA_VERSION|VARCHAR(255)|
~~[[image:DevGuide.DatabaseSchema@db_check.gif]]~~||XWA_AUTHOR|VARCHAR(255)| || ~~|~~null|
~~|~~
~~[[image:DevGuide.DatabaseSchema@db_field.gif]]~~||XWA_COMMENT|VARCHAR(~~1023~~)||XWA_VERSION|VARCHAR(255)|(%class="fa fa-check"%)((()))| || | ~~||null||~~
~~|~~
~~[[image:DevGuide.DatabaseSchema@db_field.gif]]~~||XWA_MIMETYPE|VARCHAR(255)~~||~~||XWA_COMMENT|VARCHAR(1023)| | ~~|Since 7.1M1~~||null| ||XWA_MIMETYPE|VARCHAR(255)||| | |Since 7.1M1

Documentation.DevGuide.DatabaseSchema

XWiki List Classes (xwikislistclasses)
Documentation.DevGuide.DatabaseSchema.DsXWikiListClasses.WebHome

On 2017/09/05, the page has been created by Vincent Massol
On 2017/09/05, the page has been modified by Vincent Massol

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.DatabaseSchema.WebHome

Documentation.DevGuide.DatabaseSchema

XWiki Space (xwikispace)
Documentation.DevGuide.DatabaseSchema.DsXwikiSpace.WebHome

On 2017/09/05, the page has been modified by Vincent Massol

- title: DsXwikiSpace ⇨ XWiki Space (xwikispace)

Documentation.DevGuide

Database Schema
Documentation.DevGuide.DatabaseSchema.WebHome

On 2017/09/05, the page has been modified by Vincent Massol

- title: XWiki Database Schema ⇨ Database Schema

Documentation.DevGuide

DsXWikiListClasses

Documentation.DevGuide.DsXWikiListClasses

On 2017/09/05, the page has been deleted by Vincent Massol

Documentation.DevGuide.FAQTutorial

FAQTutorialAWM

Documentation.DevGuide.FAQTutorial.FAQTutorialAWM

On 2017/09/05, the page has been deleted by Vincent Massol

FAQTutorialManual

Documentation.DevGuide.FAQTutorial.FAQTutorialManual

On 2017/09/05, the page has been deleted by Vincent Massol

FAQTutorial

Documentation.DevGuide.FAQTutorial.WebHome

On 2017/09/05, the page has been deleted by Vincent Massol

Documentation.DevGuide.FrontendResources

Integrating JavaScript Libraries in XWiki

Documentation.DevGuide.FrontendResources.IntegratingJavaScriptLibraries.WebHome

On 2017/09/05, the page has been modified by Vincent Massol

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.FrontendResources.WebHome

On 2017/09/05, the page has been created by Vincent Massol

{{toc ~~depth="1"/}}~~**depth="1" /}}**
~~1*. \\~~
~~1*. \\~~
1. Deploy moment.js as a ~~[[**WebJar**>>extensions:Extension.WebJars Integration]]~~
[[WebJar**>>http://extensions.xwiki.org/xwiki/bin/view/Extension/WebJars+Integration]]** and load it using ~~[[**RequireJS**>>JavaScriptAPI||anchor="HRequireJSandjQueryAPIs"]]~~
[[RequireJS**>>http://platform.xwiki.org/xwiki/bin/view/DevGuide/JavaScriptAPI#HRequireJSandjQueryAPIs]]** and the WebJar Script Service.
= Why should we ~~packag~~**pacakge** WebJars? =
* See [[Why should we use ~~WebJars?>>]]~~**WebJars?>>||anchor=""]]**

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ DevGuide.WebHome

parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.WebHome

author: XWiki.VincentMassol ⇨ XWiki.mflorea

Documentation.DevGuide.FrontendResources

LiveValidation UI Component

Documentation.DevGuide.FrontendResources.LiveValidation.WebHome

On 2017/09/05, the page has been created by Vincent Massol

> XWiki bundles the LiveValidation library. You can read its ~~[[examples>>http://livevalidation.com/examples]]~~ **[[examples>>documentation>>http://livevalidation.com/examples]]** or the [[full documentation>>http://livevalidation.com/documentation]] on their ~~website. You may also be interested by the [[LiveValidation Macros>>extensions:Extension.LiveValidation Macros]]~~ ~~extensions.~~**website.**
> ~~===~~ Exemple of use in XWiki ~~===~~
> ~~});~~

author: XWiki.VincentMassol ⇨ XWiki.jvelociter

On 2017/09/05, the page has been modified by Vincent Massol

parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.FrontendResources.WebHome

Documentation.DevGuide.FrontendResources

Using jQuery and jQuery UI

Documentation.DevGuide.FrontendResources.UsingJQuery.WebHome

On 2017/09/05, the page has been created by Vincent Massol

> jQuery and jQuery UI are provided as Webjars in XWiki, and so can be easily integrated into XWiki using RequireJS ([[see ~~here>>doc:DevGuide.IntegratingJavaScriptLibraries]]~~**here>>http://platform.xwiki.org/xwiki/bin/view/DevGuide/IntegratingJavaScriptLibraries]]** for details).
> ~~{{code}}require(['jquery'], function($) { jQuery code });{{/code}}~~**{{code}}**
> **require(['jquery'], function($) { jQuery code });**
> **{{/code}}**
> For other webjars such as jQuery UI the location needs to be supplied to require. This is made easier with the webjars service, which can give the location of a specific library with $services.webjars.url() ([[API>>http://nexus.xwiki.org/nexus/service/local/repositories/releases/archive/org/xwiki/platform/xwiki-platform-webjars/7.0/xwiki-platform-webjars-7.0-javadoc.jar/!/org/xwiki/webjars/script/WebJarsScriptService.html]]). See [[this ~~snippet>>doc:snippets:Extension.Requirejs and Webjars workaround~~**snipper>>http://extensions.xwiki.org/xwiki/bin/view/Extension/Requirejs+and+Webjars+workaround+for+relative+path+dependencies]]** for ~~relative path dependencies]] for~~ move infromation.
> == ~~Example ==~~**Example==**
> * [[Integrating Javascript Libraries in ~~XWiki>>doc:DevGuide.IntegratingJavaScriptLibraries]]~~**XWiki>>http://platform.xwiki.org/xwiki/bin/view/DevGuide/IntegratingJavaScriptLibraries]]**
> * More information about using [[XWiki Javascript

~~events>>doc:DevGuide.JavaScriptAPI]]~~**events>>http://platform.xwiki.org/xwiki/bin/view/DevGuide/JavaScriptAPI#HRequireJSandjQueryAPIs]]**
* [[Incorporating Javascript into skin ~~extensions>>doc:DevGuide.SkinExtensionsTutorial]]~~**extensions>>http://platform.xwiki.org/xwiki/bin/view/DevGuide/SkinExtensionsTutorial]]**

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ DevGuide.FrontendResources
- author: XWiki.VincentMassol ⇨ XWiki.beldaz

On 2017/09/05, the page has been modified by Vincent Massol

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.FrontendResources.WebHome

Documentation.DevGuide

Front-end Resources

Documentation.DevGuide.FrontendResources.WebHome

Between 2017/09/05 and 2017/09/05, the page has been modified 3 times, by 1 user: Vincent Massol

~~{{info}}~~
~~This section is a work in progress.~~
~~{{/info}}~~
See [[Using jQuery and jQuery UI>>platform:DevGuide.UsingJQuery]] for an example of using jQuery and jQuery UI in an XWiki ~~page.~~ **page, and more generally check [[how to integrate JavaScript Libraries in XWiki>>Documentation.DevGuide.FrontendResources.IntegratingJavaScriptLibraries.WebHome]].**

XWiki.XWikiComments
comment:

- author: XWiki.evalica ⇨ XWiki.VincentMassol

Documentation.DevGuide

GroovyClassHelloWorldTutorial

Documentation.DevGuide.GroovyClassHelloWorldTutorial

On 2017/09/05, the page has been deleted by Vincent Massol

GroovyNotificationTutorial

Documentation.DevGuide.GroovyNotificationTutorial

On 2017/09/05, the page has been deleted by Vincent Massol

IncludeInVelocity

Documentation.DevGuide.IncludeInVelocity

On 2017/09/05, the page has been deleted by Vincent Massol

IntegratingJavaScriptLibraries
Documentation.DevGuide.IntegratingJavaScriptLibraries

On 2017/09/05, the page has been deleted by Vincent Massol

InternalDataModel
Documentation.DevGuide.InternalDataModel

On 2017/09/05, the page has been deleted by Vincent Massol

InternationalizingApplications
Documentation.DevGuide.InternationalizingApplications

On 2017/09/05, the page has been deleted by Vincent Massol

LiveValidation
Documentation.DevGuide.LiveValidation

On 2017/09/05, the page has been deleted by Vincent Massol

MasterDetailTutorial
Documentation.DevGuide.MasterDetailTutorial

On 2017/09/05, the page has been deleted by Vincent Massol

PerformingAsynchronousTasks
Documentation.DevGuide.PerformingAsynchronousTasks

On 2017/09/05, the page has been deleted by Vincent Massol

SavingData
Documentation.DevGuide.SavingData

On 2017/09/05, the page has been deleted by Vincent Massol

Scripting
Documentation.DevGuide.Scripting

On 2017/09/05, the page has been deleted by Vincent Massol

Documentation.DevGuide.Scripting

Scripting API Guide
Documentation.DevGuide.Scripting.APIGuide.WebHome

Between 2017/09/05 and 2017/09/05, the page has been modified 2 times, by 1 user: Vincent Massol

> This guide covers the main XWiki APIs that you can use from scripts in wiki pages. It's not meant to be comprehensive. For that you'll need to check the [[XWiki Reference API ~~page>>Documentation.DevGuide.API]].~~**page>>Documentation.DevGuide.API.WebHome]].**

parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Scripting.WebHome

On 2017/09/05, the page has been created by Vincent Massol

~~{{box cssClass="floatinginfobox" title="**Contents**"}}~~**1 XWiki API Guide**

~~{{toc/}}~~

~~{{/box}}~~

This guide covers the main XWiki ~~APIs that you can use from scripts in wiki pages.~~**APIs.** It's not meant to be comprehensive. For that you'll need to check the ~~[[XWiki~~**[XWiki** Reference API ~~page>>Documentation.DevGuide.API]].~~**page>API].**

~~Note that while most examples are written in Velocity you can use [[any other scripting language>>Documentation.DevGuide.Scripting]] to access the same APIs.~~**#toc("" "" "")**

~~= Querying documents =~~**1.1 Getting external content**

~~See the [[Query Module>>extensions:Extension.Query Module]] for examples on how to perform queries on the wiki using a scripting language.~~

~~= Create a new Document =~~

~~For example in Velocity:~~

~~{{code}}~~

~~## Note that getDocument() creates a Document if it doesn't exist. This can be checked with $newDoc.isNew()~~

~~#set ($newDoc = $xwiki.getDocument('MySpace.MyPage'))~~

~~## Set its content (for example)~~

~~#set ($discard = $newDoc.setContent("new content"))~~

~~## The second parameter to save() indicates whether the save is a minor edit or not~~

~~#set ($discard = $newDoc.save("some comment explaining the save", true)~~

~~{{/code}}~~

~~= Accessing the request =~~

~~You can access the HTTP Request by accessing the ##com.xpn.xwiki.web.XWikiServletRequest## object through the ##request## script variable.~~

~~For example in Velocity, to access an ##action## HTTP parameter passed in the request you would write:~~

~~{{code language="velocity"}}~~

~~$request.action~~

~~{{/code}}~~

~~Note that this is a shortcut to:~~

~~{{code language="velocity"}}~~

~~$request.get("action")~~

~~{{/code}}~~

~~= Getting external content =~~

~~{{code language="java"}}~~**{code:none}**

~~{{/code}}~~**{code}**

For ~~example in Velocity:~~**example:**

{code:none}

$xwiki.getURLContent("http://teamcity.xwiki.org/externalStatus.html")

{code}

~~{{code language="velocity"}}~~

~~$xwiki.getURLContent("http://google.com")~~

~~{{/code}}~~

~~= Add objects to a page =~~

~~Here is a piece of Velocity script to show how is possible to store a new object in one page:~~

~~{{code language="velocity"}}~~

~~## Create an object~~

~~#set($obj = $doc.newObject("XWiki.SomeClass"))~~

~~#set($obj = $doc.newObject("XWiki.SomeClass"))~~
~~$obj.set("field1",$value1)~~
~~$obj.set("field2",$value2)~~
~~## Save the object in the page~~
~~$doc.save()~~
~~{{/code}}~~
~~The "XWiki.SomeClass" class has to be created (through the class editor): field1 and field2 are property of the class. At the moment, this code works fine only if the user currently logged in has editing rights on the page, otherwise the Document.save() will not work.~~
~~= Access objects in a page =~~
~~Here is a piece of Velocity script to show how it is possible to access an object attached to the page, and read its fields :~~
~~{{code language="velocity"}}~~
~~## Retrieve the first object (index [0]) among all objects attached to this page and of a certain class~~
~~#set($obj = $doc.getObject('SomeSpace.SomeClass'))~~
~~## Retrieve the raw value of the propertty "field1" for this object, provided ## a property called "field1" is actually defined in the class of this object~~
~~#set($rawValue = $obj.getProperty('field1').value)~~
~~SomeSpace.SomeClass[0] : field1 = "$rawValue"~~
~~## or value displayed (supports I10n or key-value properties)~~
~~#set($ValueDisplayed = $obj.field1)~~
~~#set($class = $obj.xWikiClass) ## access the class object representing SomeSpace.SomeClass~~
~~## return the property type: String, TextAreaDate, Boolean, ...~~
~~$class.get('field1').classType~~
~~{{/code}}~~
~~You can also go through all the properties of an object, without knowing their respective names. That's how the default Class Sheet works, when you create a class using the Class Wizard.~~
~~{{code language="velocity"}}~~
~~#set($class = $obj.xWikiClass) ## access the class object representing SomeSpace.SomeClass~~
~~#foreach($prop in $class.properties) ## go through all properties~~
~~<dt> *${prop.prettyName}* </dt>~~
~~<dd>$doc.display($prop.getName())</dd>~~
~~#end~~
~~{{/code}}~~
~~Actually the line {{code}}$doc.display(propertyName){{/code}} can either display the property value, or generate an input field in the page, mapped to the property whose name is passed as argument (when you edit the page in inline mode). If you have a Velocity script that uses the ##display(propertyName)## method to access properties of an object attached to the including page and you want to include it somewhere else you have to use the ##includeForm()## Velocity macro in the including script:~~
~~{{code language="velocity"}}~~
~~#includeForm("spacename.docname")~~
~~{{/code}}~~
~~See [[the includeForm() macro>>extensions:Extension.Include Form Macro]] for more information.~~
~~= Access objects from any page and loop over all objects of same Class =~~
~~Here is a piece of Velocity script to show how it is possible to access an object attached to the page from another page, and read its fields:~~
~~(It is similar than previous code except you must "call" page before with ##$xwiki.getDocument##.)~~
~~{{code language="velocity"}}~~

get the document which has the object (only one here) - this is the page where I can see things in the object editor
Retrieve the first object (index [0]) among all objects attached to the page MySpace.MyDocWithMyClassObjects and of a certain class MySpace.MyClass
#set($MyDoc =
$xwiki.getDocument("MySpace.MyDocWithMyClassObjects"))
get the document wich contains the class definition: this page has entries in the class editor
#set($class = $xwiki.getClass("MySpace.MyClass"))
#foreach($prop in $class.properties) ## go through all properties
* ${prop.prettyName} : $MyDoc.display($prop.getName())
#end
{{/code}}
If ##MySpace.MyDocWithMyClassObjects## have many attached objects of ##MySpace.MyClass## class (with different value)
{{image reference="APIGuide-MyDocWithMyClassObjects.png"/}} {{image reference="APIGuide-ResultOfLoops.png"/}}
{{code language="velocity"}}
if you have more than one object on a page, you will have to loop over them and use "$doc.use"
#set($MyDoc =
$xwiki.getDocument("MySpace.MyDocWithMyClassObjects"))
#set($class = $xwiki.getClass("MySpace.MyClass"))
loop over all objects
#foreach($obj in $MyDoc.getObjects("MySpace.MyClass"))
* Object number $velocityCount
#set($discard = $MyDoc.use($obj))
#foreach($prop in $class.properties) ## go through all properties
** ${prop.prettyName} : $MyDoc.display($prop.getName())
#end
#end
{{/code}}
= Include a Velocity page into another Velocity page =
See the [[Include In Velocity tutorial>>FAQ.IncludeInVelocity]].
= Redirecting to another page =
It's possible to redirect the user to another page. This is useful for example when a page has been removed and you have given the URL to someone so you want the old page to redirect to the new page.
Example:
{{code language="velocity"}}
$response.sendRedirect($xwiki.getURL("Space.Page"))
{{/code}}
For more examples, see the [[Redirect Snippet>>snippets:Extension.Redirect]].
= Add an Attachment to a page =
For example, in Velocity:
{{code}}
{{velocity}}
#set($content = "This is some small arbitrary content")
#set($discard = $doc.addAttachment("myfile.txt", $content.getBytes()))
#set($discard = $doc.save("some comment"))
{{/velocity}}
{{/code}}
= Add a new user to a List of Users xobject =
Let's imagine that ##Space.Page## contains an xobject having a ##users## xproperty of type "List of Users" (and configured to be a

~~multiselect).~~
~~The following code adds a new user reference to the list:~~
~~{{code}}~~
~~{{velocity}}~~
~~#set ($usersObject = $doc.getObject('Space.Page'))~~
~~#set ($referenceList = $usersObject.getValue('users'))~~
~~#set ($referenceList = "${referenceList},XWiki.NewUser")~~
~~#set ($discard = $usersObject.set('users', $referenceList))~~
~~#set ($discard = $doc.save('Added new user', true))~~
~~{{velocity}}~~
~~{{/code}}~~
~~= Create an XClass =~~
~~The following example creates an XClass with a single ##mytextarea## xproperty of type Text Area in the document ##Sandbox.TestClass##.~~
~~{{warning}}~~
~~Requires Programming Rights~~
~~{{/warning}}~~
~~{{code}}~~
~~{{velocity}}~~
~~#set ($mydoc = $xwiki.getDocument('Sandbox.TestClass'))~~
~~#set ($myinternaldoc = $mydoc.getDocument())~~
~~#set ($myclass = $myinternaldoc.getXClass())~~
~~#set ($discard = $myclass.addTextAreaField("mytextarea", "My Text Area", 100, 5))~~
~~#set ($discard = $mydoc.save('created doc + xclass'))~~
~~{{/velocity}}~~
~~{{/code}}~~

APIGuide-MyDocWithMyClassObjects.png: 1.1 ⇨ ()

APIGuide-ResultOfLoops.png: 1.1 ⇨ ()

XWiki.XWikiComments
- author:
~~xwiki:XWiki.PatrickWhittingham~~
- comment:
~~Do anyone know how to view the 'velocity' code for "Editing class". It is part of the classsheet or something else?~~
- date:
~~2012-11-15 16:08:07.0~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.Sergiu~~
- comment:
~~It's in a template on the filesystem, ##templates/editclass.vm##.~~
- date:
~~2012-11-16 00:22:45.0~~
- replyto:
~~0~~

- title: Scripting API Guide ⇨
- parent: xwiki:Documentation.DevGuide.WebHome ⇨ DevGuide.WebHome
- syntax: XWiki 2.0 ⇨ XWiki 1.0

Documentation.DevGuide

Scripting

Documentation.DevGuide.Scripting.WebHome

Between 2017/09/05 and 2017/09/05, the page has been modified 7 times, by 1 user: Vincent Massol

> The API is documented in Javadoc format and can be accessed here: [[XWiki API ~~Javadoc>>Documentation.DevGuide.API]].~~**Javadoc>>Documentation.DevGuide.API.WebHome]].** If you are not familiar with Java or object oriented programming, you will probably be confused by the API documentation. It is not within the scope of our documentation to teach you all the details about Java, or object oriented programming. You can find all of that information already online. You can also explore the page code found throughout the [[Extensions wiki>>extensions:Main.WebHome]] area to see how others have figured out how to achieve a variety of results.
> We're also making available an [[API ~~Guide>>Documentation.DevGuide.APIGuide]]~~**Guide>>Documentation.DevGuide.Scripting.APIGuide.WebHome]]** with examples about using the XWiki API.
> **See also this [[HQL tutorial in Velocity>>Documentation.DevGuide.Scripting.velocityHqlExamples.WebHome]].**
> You can [[gain access to the XWiki ~~core>>Documentation.DevGuide.Scripting#HXWikiCoreAccess]]~~**core>>Documentation.DevGuide.WebHome#HXWikiCoreAccess]]** from Velocity but this will require Programming Rights. Strictly speaking, protected APIs are only available when the page that contains them was last saved by someone who had Programming Rights (see above).
> In Velocity you can't import classes and as such you cannot gain direct access to XWiki components as shown ~~[[above>>Documentation.DevGuide.Scripting#HXWikiComponentAccess]].~~ **[[above>>Documentation.DevGuide.WebHome#HXWikiComponentAccess]].** This leaves you with the provided ~~[[bindings>>Documentation.DevGuide.Scripting#HBindings]]~~ **[[bindings>>Documentation.DevGuide.WebHome#HBindings]]** (NOTE: In Velocity, these bindings all start with **##$##** as with all other Velocity variables)
> For more information about programming in the Velocity language, you can refer to the [[Velocity User Guide>>http://velocity.apache.org/engine/releases/velocity-1.7/user-guide.html]]. See also this [[Velocity Training ~~Presentation>>Documentation.DevGuide.XWikiVelocityTraining.WebHome]].~~**Presentation>>Documentation.DevGuide.Scripting.XWikiVelocityTraining.WebHome]].**

XWiki.XWikiComments
comment:

XWiki.XWikiComments
comment:

XWiki.XWikiComments
comment:

On 2017/09/05, the page has been created by Vincent Massol

{{box cssClass="floatinginfobox" title="**Contents**"}}**1 Scripting**
{{toc/}}
{{/box}}
Scripting allows**XWiki integrates both Velocity and Groovy scripting. Together, these two mechanisms allow** you to create basic to complex web applications at the XWiki page (or view) layer without the need for compiling code or deploying software components. In other words, you can use scripting**Velocity and Groovy script** syntax in addition to wiki and HTML syntax as the contents of an XWiki page **page.**
XWiki integrates [[jsr-223>>http://jcp.org/en/jsr/detail?id=223]] scripting. You can script using several available languages by using the generic [[Script Macro>>extensions:Extension.Script Macro]] or one of following specific macros:**1.1 XWiki's Velocity API**
* [[Velocity Macro>>extensions:Extension.Velocity Macro]] (installed by default**The concept of the 'context' is central to Velocity. The context is a 'carrier' of data between the Java layer of the XWiki engine and the template or page layer. The programmers of the XWiki core have gathered objects of various types and placed them** in XWiki)**the Velocity context. These objects, and their methods and properties, are accessable via template elements called references and effectively form an API for XWiki.**
* [[Groovy Macro>>extensions:Extension.Groovy Macro]] (installed by default in XWiki)
* [[Python Macro>>extensions:Extension.Python Macro]] (installed by default in XWiki)
* [[Ruby Macro>>extensions:Extension.Ruby Macro]] (not installed by default in XWiki)
* [[PHP Macro>>extensions:Extension.PHP Macro]] (not installed by default in XWiki)
= Permissions =**The API is documented in Javadoc format and can be accessed here: [XWiki API Javadoc>http://build.xpertnet.biz/doc/api/com/xpn/xwiki/api/Api.html]. If you are not familiar with Java or object oriented programming, you will probably be confused by the API documentation. It is not within the scope of our documentation to teach you all the details about Velocity, Java, or object oriented programming. You can find all of that information already online. Instead, we provide only a high-level overview in [Basic Velocity Scripting Tutorial>Basic Velocity]; it will give you a good sense for how Velocity scripting works in XWiki. You should then refer to the [Velocity User Guide>http://jakarta.apache.org/velocity/docs/user-guide.html#What%20is%20Velocity] as an ongoing reference. Finally, you can explore the page code found throughout the [Samples and Snippets] area to see how others have figured out how to achieve a variety of results.**
Starting with XWiki 7.2M1, a user needs to have the [[Script Permission**You can access** in order**your velocity script** to be able to write Script>>extensions:Extension.Script Macro||anchor="HRights"]].:
*** The current document: *\$doc*, see the [XWiki javadoc 0.9.840>http://build.xpertnet.biz/doc/api/com/xpn/xwiki/api/Document.html] [XWiki javadoc SVN Version>http://build.xpertnet.biz/doc/nightly/api/com/xpn/xwiki/api/Document.html]**

*** The Context of the request: *\$context*, see the [XWiki javadoc 0.9.840>http://build.xpertnet.biz/doc/api/com/xpn/xwiki/api/Context.html] [XWiki javadoc SVN Version>http://build.xpertnet.biz/doc/nightly/api/com/xpn/xwiki/api/Context.html]**
*** the XWiki Object: *\$xwiki*, see the [XWiki javadoc 0.9.840>http://build.xpertnet.biz/doc/api/com/xpn/xwiki/api/XWiki.html] [XWiki javadoc SVN Version>http://build.xpertnet.biz/doc/nightly/api/com/xpn/xwiki/api/XWiki.html]**
~~In addition, all scripting languages other than Velocity also require Programming Rights (see below for more details).~~**Each of this objects are the wrapped version of original ones. You can acces to it if you have the programming right by \$xwiki.xwiki, \$context.context and \$doc.document.**
~~= Choosing a Scripting language =~~**You can also [use hql to query the XWiki database>velocityHqlExamples] from your velocity scripts.**
~~Since XWiki supports several scripting languages you might be wondering which one to use. Most of the code written by XWiki developers is in Velocity, with a few more complex extensions written in Groovy; these two are thoroughly tried and tested, so they are both a safe bet. The other languages //should// work just as well, but there are less developers that could help answering any questions.~~**1.1 XWikis Groovy API**
~~== Velocity ==~~Currently Groovy is only allowed for admins of a wiki (or users having the 'programming' right).
~~The first thing to know is that Velocity is different from the other scripting languages on 2 aspects:~~*** Groovy examples: See [Dev.XWikiGroovy], [Dev.GroovyXML]**
*** [Feeling Groovy>http://www-128.ibm.com/developerworks/java/library/j-alj08034.html]**
* ~~It's a templating language rather than a pure scripting language, which means that its content is actually wiki markup interspersed~~**[MVC programming** with ~~Velocity directives, whereas pure scripting languages are written in that language and they need to explicitly output wiki markup. For example:(((~~**Groovy templates>http://www-128.ibm.com/developerworks/java/library/j-pg02155/]**
~~Velocity:~~*** [Guillaume Laforge on Groovy, XWiki etc.>http://www.vanwardtechnologies.com/glaforge01.php]**
~~{{code}}~~
~~{{velocity}}~~
~~Your username is \$xcontext.getUser(), welcome to the site.~~
~~{{/velocity}}~~
~~{{/code}}~~
~~Groovy:~~**TO DO: We need to move all of the Groovy pages from the "Dev" Web Space to this area. Then, with all the Velocity and Groovy docs in this Web Space, we can reorganize and cross reference and such.**
~~{{code}}~~
~~{{groovy}}~~
~~println("Your username is " + xcontext.getUser() + " welcome to the site.");~~
~~{{/groovy}}~~
~~{{/code}}~~
~~)))~~
~~* It doesn't require special permissions (other than Script Permission starting with XWiki 7.2M1) since it runs in a Sandbox, with access to only a few safe objects, and each API call will check the rights configured in the wiki, forbidding access to resources or actions that the current user~~

~~shouldn't be allowed to retrieve/perform. Other scripting language require the user that wrote the script to have Programming Rights to execute them, but except this initial precondition, access is granted to all the resources on the server. Note that starting with XWiki 4.1 we've introduced a [[Sandbox for Groovy>>platform:AdminGuide.Configuration#HSecuringGroovyScripts]] too, but it's still in an early stage and is currently very restrictive.~~
~~Being a templating engine, Velocity doesn't offer many means of structuring code. In fact, there's only one useful directive in this regard, ###macro##. However, because it is a templating engine, its syntax is much simpler and easier to understand by non-developers, which means that it's accessible to a wider range of users, without a serious background in programming.~~
~~Without Programming Rights, it's impossible to instantiate new objects, except literals and those safely offered by the XWiki APIs. Nevertheless, the XWiki API is powerful enough to allow a wide range of applications to be safely developed, if "the XWiki way" is properly followed.~~
~~Velocity is also available in some other parts of XWiki: it is the language in which all the templates that generate the HTML UI of XWiki are written, it can be optionally activated in skin extensions, and it is executed when sending CSS and JavaScript skin resources from the filesystem.~~
~~In conclusion, **Velocity is suited for projects with small to medium complexity, and which don't require access to other resources except the provided XWiki API and registered script services. It allows very quick and easy development, offers good security and decent performance, and can easily be packaged and distributed as a XAR.**~~
~~== Groovy ==~~
~~Groovy is a full-fledged scripting language, which supports almost the entire Java syntax, and provides its own syntax delicacies and custom APIs that enhance the Java language even further. While it is recommended that complex code be written in Java as components accessible via script services, Groovy has the advantage that it is written live in the wiki, without requiring compilation, deployment and server restarts, thus enabling faster development.~~
~~The XWiki API is available in the context when executing Groovy scripts, but unlike in Velocity, the code isn't limited to this API, and any other classes or objects can be accessed freely. New classes can be defined in Groovy, compatible with Java classes, and this allows more structured code to be written, unlike in Velocity. A particular case of classes is new component roles and component implementations, which allows, for example, new script services or new event listeners to be defined in the wiki. It is possible to load attached jar files into the classpath of an executing script, which means that a wiki document can contain a complex program AND its required libraries not already provided by the platform.~~
~~Other than being available as a scripting language for writing custom code, it is also the language in which scheduler tasks are written.~~
~~In conclusion, **Groovy is suited for complex projects or for custom wiki enhancement through new components, when speedy live development is required. Being written in wiki documents, it can also be easily packaged and distributed as a XAR.**~~
~~After taking into account these considerations and if requiring Programming Rights isn't an issue for you, you should pick the script language that you're most familiar with!~~
~~= XWiki Scripting API =~~
~~The API is documented in Javadoc format and can be accessed here: [[XWiki API Javadoc>>Documentation.DevGuide.API]]. If you are not familiar with Java or object oriented programming, you will probably be confused by the API documentation. It is not within the scope of our documentation to teach you all the details about Java, or object oriented programming. You can find all of that information already online. You can~~

~~programming. You can find all of that information already online. You can also explore the page code found throughout the [[Extensions wiki>>extensions:Main.WebHome]] area to see how others have figured out how to achieve a variety of results.~~
~~We're also making available an [[API Guide>>Documentation.DevGuide.APIGuide]] with examples about using the XWiki API.~~
~~== [[Bindings>>extensions:Extension.Script Macro#HBindings]] ==~~
~~The main objects available to you in scripting languages are:~~
~~* The current Document: **##doc##**~~
~~* The Context of the request: **##xcontext##**~~
~~* The Request object: **##request##**~~
~~* The Response object: **##response##**~~
~~* The XWiki object: **##xwiki##**~~
~~* The XWiki utils: **##util##** (this is deprecated)~~
~~* Various [[Script Services>>extensions:Extension.Script Module]]: **##services##**~~
~~See [[Scripting Reference Documentation>>platform:SRD.Navigation]] for a complete list.~~
~~== [[XWiki Component>>extensions:Extension.Component Module]] Access ==~~
~~Since XWiki 4.1M2+ there's a Script Service to access the Component Manager (see also: [[Accessing components from Groovy>>Documentation.DevGuide.Tutorials.WritingComponents.WebHome#HFromwikipages]]).~~
~~For example using Groovy you'd write:~~
~~{{code language="java"}}~~
~~{{groovy}}~~
~~def greeter = services.component.getInstance(org.xwiki.component.HelloWorld.class)~~
~~println greeter.sayHello()~~
~~{{/groovy}}~~
~~{{/code}}~~
~~You can also get the ComponentManager with:~~
~~{{code language="java"}}~~
~~{{groovy}}~~
~~def cm = services.component.componentManager~~
~~{{/groovy}}~~
~~{{/code}}~~
~~{{info}}~~
~~With versions of XWiki older than 4.1M2 you'd use (in Groovy):~~
~~{{code language="java"}}~~
~~{{groovy}}~~
~~def greeter = com.xpn.xwiki.web.Utils.getComponent(org.xwiki.component.HelloWorld.class)~~
~~println greeter.sayHello()~~
~~{{/groovy}}~~
~~{{/code}}~~
~~{{/info}}~~
~~== XWiki Core Access ==~~
~~Sometimes the XWiki Api doesn't provide the methods which you need for your application. You can gain raw access the core of XWiki but it presents an increased security risk and requires **Programming Rights** to be able to save the page containing the script (Programming Rights are not required for viewing a page containing a script requiring Programming Rights, rights are only needed at save time). Using the core should be avoided if at all possible.~~

{{code language="java"}}
{{groovy}}
def xc = xcontext.getContext();
def wiki = xc.getWiki();
def xdoc = doc.getDocument();
{{/groovy}}
{{/code}}
After using this snippet, you will have 3 new objects:
* {{scm path="xwiki-platform-core/xwiki-platform-oldcore/src/main/java/com/xpn/xwiki/XWikiContext.java"}}The underlying XWikiContext behind the Context object{{/scm}}: **##xc##**
* {{scm path="xwiki-platform-core/xwiki-platform-oldcore/src/main/java/com/xpn/xwiki/XWiki.java"}}The underlying XWiki object which backs the **##xwiki##** object{{/scm}}: **##wiki##**
* {{scm path="xwiki-platform-core/xwiki-platform-oldcore/src/main/java/com/xpn/xwiki/doc/XWikiDocument.java"}}The underlying XWikiDocument behind the current Document{{/scm}}: **##xdoc##**
You will find that many of the methods in **##wiki##** and **##xdoc##** require an instance of the XWikiContext, this is the underlying xcontext **##xc##** not the Api context **##xcontext##**.
Again, these methods are only for the rare cases when functionality is not provided by the public Api. We put a lot of effort into preserving the behavior of the public Api and much less into preserving the behavior of core methods so you may find that core methods are deprecated, removed, or their behavior is changed in subsequent versions.
== Querying XWiki's Model ==
From your script you can query the full XWiki Model. Check the [[Query Module>>extensions:Extension.Query Module]] for more information.
{{id name="velocity"/}}
= Velocity Specific Information =
Velocity is currently the only scripting language which can be used without Programming [[platform:AdminGuide.Access Rights]]. This means you can save Velocity scripts using a user with less permissions and nobody will be able to exploit your script to perform a security breach.
You can [[gain access to the XWiki core>>Documentation.DevGuide.Scripting#HXWikiCoreAccess]] from Velocity but this will require Programming Rights. Strictly speaking, protected APIs are only available when the page that contains them was last saved by someone who had Programming Rights (see above).
In Velocity you can't import classes and as such you cannot gain direct access to XWiki components as shown [[above>>Documentation.DevGuide.Scripting#HXWikiComponentAccess]]. This leaves you with the provided [[bindings>>Documentation.DevGuide.Scripting#HBindings]] (NOTE: In Velocity, these bindings all start with **##$##** as with all other Velocity variables)
For more information about programming in the Velocity language, you can refer to the [[Velocity User Guide>>http://velocity.apache.org/engine/releases/velocity-1.7/user-guide.html]]. See also this [[Velocity Training Presentation>>Documentation.DevGuide.XWikiVelocityTraining.WebHome]].
For more details on using Velocity check the [[Velocity Module Documentation>>extensions:Extension.Velocity Module]] which also contains the full list of Velocity Tools that you can use in your scripts.
{{info}}

If you wish to add new Velocity tools you'll need to edit your ##xwiki.properties## file and follow the instructions in there.
{{/info}}
To include Velocity scripts in other Velocity scripts, see [[How to include a Velocity page into another page>>Documentation.DevGuide.IncludeInVelocity]].
== Other Velocity Variables ==
{{warning}}
These variables can be used but are subject to change in the future.
{{/warning}}
=== Controlling Which Sections to Display ===
You can control whether to display Comments/History/Attachment/Information sections or not by setting some velocity variables to ##false##:
{{code language="velocity"}}
#set ($showcomments = false)
#set ($showattachments = false)
#set ($showhistory = false)
#set ($showinformation = false)
{{/code}}
To remove them all you can set:
{{code language="velocity"}}
#set($docextras = [])
{{/code}}
=== Information about the current user ===
The following variables (set in the {{scm path="xwiki-platform-core/xwiki-platform-web/src/main/webapp/templates/xwikivars.vm"}}xwikivars.vm{{/scm}} template) are shortcuts for checking various information **for the current user**:
* ##$isGuest##: checks if the current user is ##XWiki.XWikiGuest##
* ##$isSuperAdmin##: checks if the current user is the special user ##superadmin##
* ##$hasComment##: checks comment rights on the current document
* ##$hasEdit##: checks edit rights on the current document
* ##$hasWatch##: checks if the user is authenticated and the watch service is available
* ##$hasAdmin##: checks admin rights on the current document
* ##$hasSpaceAdmin##: checks admin rights on the ##XWikiPreferences## document of the current space
* ##$hasGlobalAdmin##: checks admin rights on ##XWiki.XWikiPreferences##
* ##$hasCreateSpace##: checks edit rights on that page that does not exist, in a space that doesn't exist
* ##$hasCreatePage##: checks edit rights on that page that does not exist, in the current space
* ##$hasProgramming##: checks if the current user has programming rights
* ##$isAdvancedUser##: advanced users: ##superadmin##, users with the ##usertype## property set to "Advanced", guest users with admin rights
Example:
{{code language="velocity"}}
{{velocity}}
#if ($hasAdmin)
This link will only be visible to users that have admin rights on this document
[[Do some admin action>>Some.Document]]
#end
{{/velocity}}

{{/velocity}}
{{/code}}
=== Information about the current wiki ===
The following variables (set in the {{scm path="xwiki-platform-core/xwiki-platform-web/src/main/webapp/templates/xwikivars.vm"}}xwikivars.vm{{/scm}} template) are shortcuts for checking various information **about the current wiki**:
* ##$isReadOnly##
* ##$isInServletMode##
* ##$isInPortletMode##
= Groovy Specific Information =
{{info}}
Currently all non Velocity scripting languages are only allowed to be used by users having Programming Rights.
{{/info}}
* See Groovy snippets in the [[Extensions wiki>>extensions:Main.WebHome]] (click on the "Groovy" tag in the Tag Cloud)
* [[Groovy web site>>http://groovy.codehaus.org/]]
== Groovy Example ==
The following example demonstrates how to use a groovy script to interact with velocity code in your page. This example performs a DNS lookup from the velocity variable ##$hostname## and stores the result in the variable ##$address##.
Using XWiki Syntax 2.0:
Objects can be passed back and forth between scripting languages by storing them in commonly available objects. One such commonly available object which only lasts the length of the request is the context object, known as xcontext.
{{code language="velocity"}}
{{velocity}}
#set($hostname = "www.xwiki.org")
Host Name: $hostname
$xcontext.put("hostname", $hostname)
{{/velocity}}
{{groovy}}
import java.net.InetAddress;
host = xcontext.get("hostname");
InetAddress addr = InetAddress.getByName(host);
String address = addr.getHostAddress();
xcontext.put("address", address);
{{/groovy}}
{{velocity}}
IP Address: $xcontext.get("address")
{{/velocity}}
{{/code}}
Using XWiki Syntax 1.0:
Because Groovy and Velocity code are parsed together, variables defined in Groovy can be used directly in velocity without storing in and retrieving from the context.
{{code language="velocity"}}
#set ($hostname = "www.xwiki.org")
Host Name: $hostname
<%
import java.net.InetAddress;
vcontext = context.get("vcontext");
host = vcontext.get("hostname");

~~InetAddress addr = InetAddress.getByName(host);~~
~~String address = addr.getHostAddress();~~
~~%>~~
~~IP Address: $address~~
~~{{/code}}~~
~~= Python Specific Information =~~
~~You can run Python code in XWiki just like Velocity or Groovy.~~
~~{{code language="python"}}~~
~~{{python}}~~
~~print "The full name of this document is " + doc.getFullName()~~
~~{{/python}}~~
~~{{/code}}~~
~~= Share variable between languages =~~
~~Most JSR223 based scripting languages reinject the created variable in the current ##ScriptContext## which means you can define a variable in a Groovy script and reuse it in a following Python script for example.~~
~~Since the Velocity implementation isn't based on JSR223, we have also set up a bridge so that Velocity scripts can access the current ##ScriptContext## variables. This allows for example to define a variable in a Groovy script and then reuse it in a Velocity one:~~
~~{{code}}~~
~~{{groovy}}~~
~~var = "foo"~~
~~{{/groovy}}~~
~~{{velocity}}~~
~~$var~~
~~{{/velocity}}~~
~~{{/code}}~~
~~Starting with XWiki 8.3 the Velocity bridge has been improved and it's now possible to do the opposite, i.e. define a variable in Velocity and access it from another scripting language. For example:~~
~~{{code}}~~
~~{{velocity}}~~
~~#set($myvar = "foo")~~
~~{{/velocity}}~~
~~{{groovy}}~~
~~print myvar~~
~~{{/groovy}}~~
~~{{/code}}~~
~~Note that you can also share variables by setting them in the XWiki Context (##xcontext## binding).~~
~~= Scripting In XWiki Syntax 1.0 =~~
~~XWiki Syntax 1.0 is rendered by an old rendering engine which is still supported but for which no further development is planned (it will eventually be removed). Syntax 1.0 has some idiosyncrasies which were solved by syntax 2.0.~~
~~* The only scripting languages available to you are Velocity and Groovy.~~
~~* In Groovy, the context is known as: **##context##** not **##xcontext##**~~
~~* The beginning and end of Groovy scripts are denoted by <% and %> rather than through the [[extensions:Extension.Groovy Macro]] (using ~{~{groovy}} and ~{~{/groovy}})~~
~~* Velocity is parsed in a page no matter what (there is no need to invoke the [[extensions:Extension.Velocity Macro]] using ~{~{velocity}} and ~{~{/velocity}})~~
~~The last part is important because it means you need to be careful when using **$** and **#** in your document. This is still true inside of **<%** and **%>** so you have to be careful when writing Groovy.~~

XWiki.XWikiComments
- author:
xwiki:XWiki.brandone
- comment:
Guys,
Also something I'm bumping into a lot lately is the need to use Math tools - searching on various topics (abs, math, div, add, etc) yields only one math topic which only shows multiplication and addition - but nothing else. Or, is the velocity math tool exposed to the user in some way?
- date:
2007-01-15 14:20:44.0

XWiki.XWikiComments
- author:
XWiki.snowedice
- comment:
From this page, it is totally confusing what role velocity or groovy play. It does not explain what the doc, xwiki,...
- date:
2007-07-03 03:23:33.0

XWiki.XWikiComments
- author:
XWiki.ricaforrica
- comment:
The link to Guillaume's interview is not working :)
It is located in http:~//www.stelligent.com/content/view/44/71/
Regards!
- date:
2007-08-20 14:19:07.0

XWiki.XWikiComments
- author:
xwiki:XWiki.wychmiss
- comment:
$services.localization.render('platform.dashboard.wiki.welcome')
How can I customize the default value of 'platform.dashboard.wiki.welcome' ?
- date:
2014-05-13 04:34:02.0

XWiki.XWikiComments
- author:
xwiki:XWiki.ThomasMortagne
- comment:
You can overwrite it in a wiki translation, see [[extensions:Extension.Localization Module#HRegisterawikitranslation]].
- date:
2014-05-14 09:33:45.0

- replyto:
~~3~~

- title: Scripting ⇨
- parent: xwiki:Documentation.DevGuide.WebHome ⇨ DevGuide.WebHome
- syntax: XWiki 2.0 ⇨ XWiki 1.0

Documentation.DevGuide.Scripting

Velocity Training

Documentation.DevGuide.Scripting.XWikiVelocityTraining.WebHome

On 2017/09/05, the page has been modified by Vincent Massol

Presentation.PresentationClass
- layout:
layout-widescreen

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Scripting.WebHome

On 2017/09/05, the page has been created by Vincent Massol

~~{{code language="velocity"}}~~**{{codePres}}**
~~{{velocity}}~~
~~{{/velocity}}~~**{{/codePres}}**
~~{{/code}}~~
~~{{image reference="sample1.png"/}}~~
{{imagePres}}sample1.png{{/imagePres}}
=== ===
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
~~{{image reference="serrure.jpg"/}}~~
{{imagePres}}serrure.jpg{{/imagePres}}
=== ===
~~{{image reference="detective.jpg"/}}~~
{{imagePres}}detective.jpg{{/imagePres}}
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
~~{{image reference="nobody.jpg"/}}~~
{{imagePres}}nobody.jpg{{/imagePres}}
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
~~{{image reference="variables.png"/}}~~
{{imagePres}}variables.png{{/imagePres}}
=== ===
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
=== ===
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
~~{{image reference="class.jpg"/}}~~**{{imagePres}}class.jpg{{/imagePres}}**

=== ===
~~{{image reference="javadoc.png"/}}~~
{{imagePres}}javadoc.png{{/imagePres}}
=== ===
~~{{image reference="java.jpg"/}}~~
~~== Basic instructions ==~~
~~=== Conditions ===~~**{{imagePres}}java.jpg{{/imagePres}}**
== Basic instructions ==
=== Conditions ===
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
~~{{image reference="visitors.jpg"/}}~~
{{imagePres}}visitors.jpg{{/imagePres}}
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
~~{{image reference="teacher.jpg"/}}~~
{{imagePres}}teacher.jpg{{/imagePres}}
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
~~{{image reference="goodmark.jpg"/}}~~
{{imagePres}}goodmark.jpg{{/imagePres}}
~~{{code language="velocity"}}~~**{{codePres}}**
~~{{/code}}~~**{{/codePres}}**
~~{{image reference="foreach1.png"/}}~~
{{imagePres}}foreach1.png{{/imagePres}}
=== ===
~~{{image reference="API.jpg"/}}~~**{{imagePres}}API.jpg{{/imagePres}}**
~~{{image reference="SRD.png"/}}~~**{{imagePres}}SRD.png{{/imagePres}}**
~~[[http:~~/~~/platform.xwiki.org/xwiki/bin/view/SRD/Navigation?~~
~~xpage=embed>>doc:platform:SRD.Navigation||queryString="xpage=embed~~
~~"]]~~**http://platform.xwiki.org/xwiki/bin/view/SRD/Navigation?**
xpage=embed
=== Is the API a ~~mess? ===~~**mess?===**
~~{{image reference="mess.jpg"/}}~~**{{imagePres}}mess.jpg{{/imagePres}}**
~~{{image reference="datacenter.jpg"/}}~~

{{imagePres}}datacenter.jpg{{/imagePres}}
=== ===
{{code language="velocity"}}{{codePres}}
{{/code}}{{/codePres}}
{{image reference="wayne.jpg"/}}{{imagePres}}wayne.jpg{{/imagePres}}
=== ===
{{code language="velocity"}}{{codePres}}
{{/code}}{{/codePres}}
{{image reference="future.jpg"/}}{{imagePres}}future.jpg{{/imagePres}}
{{image reference="edit_objects.png"/}}
{{imagePres}}edit_objects.png{{/imagePres}}
=== ===
{{imagePres}}objects.png{{/imagePres}}
{{image reference="objects.png"/}}
{{imagePres}}class_editor.png{{/imagePres}}
{{image reference="class_editor.png"/}}
{{imagePres}}appwithinminutes.png{{/imagePres}}
{{image reference="appwithinminutes.png"/}}
{{imagePres}}comments.png{{/imagePres}}
{{image reference="comments.png"/}}
{{imagePres}}user.png{{/imagePres}}
{{image reference="user.png"/}}
{{code language="velocity"}}{{codePres}}
{{/code}}{{/codePres}}
{{imagePres}}rollback.png{{/imagePres}}
{{image reference="rollback.png"/}}
{{code language="velocity"}}{{codePres}}
{{/code}}{{/codePres}}
{{code language="velocity"}}{{codePres}}
{{/code}}{{/codePres}}
=== Solution: use references ! ===!===
{{code language="velocity"}}{{codePres}}
{{/code}}{{/codePres}}
* It is better! The reference tool allow us to strictlystricty separe what is the space and what is the page
{{code language="velocity"}}{{codePres}}
{{/code}}{{/codePres}}
{{code language="velocity"}}{{codePres}}
{{/code}}{{/codePres}}
{{code language="velocity"}}{{codePres}}
{{/code}}{{/codePres}}
=== Use case: changing the name of a userthe boss ===
{{code language="velocity"}}{{codePres}}
{{/code}}{{/codePres}}
=== But remember we know who changes what! ===
{{imagePres}}harry.jpg{{/imagePres}}
{{image reference="history.png"/}}=== But remember we know who change what! ===
{{imagePres}}history.png{{/imagePres}}
=== ===
* See the [[XWiki Script Guide>>doc:Documentation.DevGuide.Scripting]]Guide>>http://platform.xwiki.org/xwiki/bin/view/DevGuide/Scripting]] to have some informations about how to make scripts in XWiki
* See the [[XWiki Query Module>>doc:extensions:Extension.Query Module]]Module>>http://extensions.xwiki.org/xwiki/bin/view/Extension/Query+Module]] to learn how to do advanced queries
* See the [[XWiki Data

~~Model>>doc:Documentation.DevGuide.DataModel]]~~**Model>>http://platform.xwiki.org/xwiki/bin/view/DevGuide/DataModel]]** to learn how to create your XWiki classes
=== ===
~~{{image reference="velocity.jpg"/}}~~
{{imagePres}}velocity.jpg{{/imagePres}}
=== Any question? ===
{{imagePres}}questions.jpg{{/imagePres}}

API.jpg: 1.1 ⇨ ()

SRD.png: 1.1 ⇨ ()

appwithinminutes.png: 1.1 ⇨ ()

class.jpg: 1.1 ⇨ ()

class_editor.png: 1.1 ⇨ ()

comments.png: 1.1 ⇨ ()

datacenter.jpg: 1.1 ⇨ ()

detective.jpg: 1.1 ⇨ ()

edit_objects.png: 1.1 ⇨ ()

foreach1.png: 1.1 ⇨ ()

future.jpg: 1.1 ⇨ ()

goodmark.jpg: 1.1 ⇨ ()

history.png: 1.1 ⇨ ()

java.jpg: 1.1 ⇨ ()

javadoc.png: 1.1 ⇨ ()

mess.jpg: 1.1 ⇨ ()

nobody.jpg: 1.1 ⇨ ()

objects.png: 1.1 ⇨ ()

rollback.png: 1.1 ⇨ ()

sample1.png: 1.1 ⇨ ()

serrure.jpg: 1.1 ⇨ ()

teacher.jpg: 1.1 ⇨ ()

user.png: 1.1 ⇨ ()

variables.png: 1.1 ⇨ ()

velocity.jpg: 1.1 ⇨ ()

visitors.jpg: 1.1 ⇨ ()

wayne.jpg: 1.1 ⇨ ()

Presentation.PresentationClass

- title: Velocity Training ⇨ XWiki Velocity Training
- parent: xwiki:Documentation.DevGuide.WebHome ⇨ DevGuide.WebHome
- author: XWiki.evalica ⇨ XWiki.gdelhumeau

Documentation.DevGuide.Scripting

HQL Scripting Examples in Velocity

Documentation.DevGuide.Scripting.velocityHqlExamples.WebHome

On 2017/09/05, the page has been modified by Vincent Massol

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Scripting.WebHome

On 2017/09/05, the page has been created by Vincent Massol

~~{{box cssClass="floatinginfobox" title="**Contents**"}}~~
~~##~~
~~{{toc start="2" depth="4"/}}~~**## Macro to display the result of any query in a table**
~~{{/box}}~~**#macro(displayQuery $query)**
~~= HQL Query Examples in Velocity =~~
~~##~~
~~{{warning}}~~**#set($results = $xwiki.search($query,10,0))**
~~The examples on this page are using deprecated search APIs (##$xwiki.searchDocuments## and ##$xwiki.search##). If you want to query the XWiki database then you should use the [[Query Module>>extensions:Extension.Query Module]] instead. If you want to perform a full text search on the XWiki model then you can use the [[Solr Search API>>extensions:Extension.Solr Search Query API]].~~**1.1.1**
xwiki.search(${query} ,10,0) returns:
~~{{/warning}}~~**{table}**
~~XWiki allows user to access documents and objects with [[HQL>>http://docs.jboss.org/hibernate/stable/core/reference/en/html/queryhql.html]] queries in [[Velocity>>http://jakarta.apache.org/velocity/docs/user-guide.html]] scripts.~~
~~== Public API (##searchDocuments##) ==~~
~~{{velocity}}~~
~~{{html}}~~
~~#info("With this API the query consist in the WHERE condition of a full HQL query. Any user with edit rights can write a script using this API. Any user with view rights can view the result of such a query.")~~
~~{{/html}}~~
~~{{/velocity}}~~

{{/velocity}}
General example showing how to display the first 5 results of a given query:
{{code language="none"}}
#set($hql = "<query here>")
#set($results = $xwiki.searchDocuments($hql, 5, 0))
#foreach ($item in $results)
* $item
#end
{{/code}}
The examples below will show you various HQL queries that you can write.
=== Simple Query ===
Displays all documents who have been created by the user ##XWiki.JohnDoe##:
{{code language="none"}}
#set($hql = "where doc.creator='XWiki.JohnDoe'")
{{/code}}
=== Ordered Query ===
Displays all documents who have been created by the user ##XWiki.JohnDoe## and sorted by document's last modification date, in ascending order:
{{code language="none"}}
#set($hql = "where doc.creator='XWiki.VincentMassol' order by doc.date asc")
{{/code}}
=== Advanced Query (date & time) ===
{{velocity}}
{{html wiki="true"}}
#info("Since there is no [[standard way to calculate dates interval in HQL>>http://opensource.atlassian.com/projects/hibernate/browse/HHH-2434]] those queries are a bit unnatural.")
{{/html}}
{{/velocity}}
{{code language="none"}}
#set($hql = "where year(doc.date) = year(current_date()) and month(doc.date) = month(current_date()) and day(doc.date) = day(current_date()) and hour(doc.date) > (hour(current_time()) - 1) order by doc.date desc")
{{/code}}
=== Query documents older than 3 days ===
The following snippet deletes all documents in the space TempDocs that are older than 3 days.
For the computation of the relative age, a calendar object created by the $datetool is used and passed to the hibernate query. Tested with MySQL only.
{{code language="none"}}
#set($hql="select doc.fullName from XWikiDocument as doc where doc.web = ? and doc.creationDate < ? ")
#set($cal = $datetool.systemCalendar)$cal.add(7, 3)
#set($olderTempDocs=$xwiki.search($hql,5000, 0, ["TempdDocs",$datetool.toDate($cal)]))
#foreach($x in $olderTempDocs)$xwiki.getDocument($x).deleteWithProgrammingRights()#end
{{/code}}
Other examples:
* Listing all documents modified during the current day: {{code language="none"}}where year(doc.date) = year(current_date()) and month(doc.date) = month(current_date()) and day(doc.date) >

(day(current_date()) - 1) order by doc.date desc{{/code}}
* Listing all documents modified during the current week: {{code language="none"}}where year(doc.date) = year(current_date()) and month(doc.date) = month(current_date()) and day(doc.date) > (day(current_date()) - 7) order by doc.date desc{{/code}}
* Listing all documents modified during the current month: {{code language="none"}}where year(doc.date) = year(current_date()) and month(doc.date) > (month(current_date()) - 1) order by doc.date desc{{/code}}
== Privileged API (##search##) ==
{{velocity}}
{{html wiki="true"}}
#warning("Calls to the privileged API are only executed when the calling page has been saved by a user with **Programming Rights**. The reason is that search can be used to send dangerous HQL command like update, delete, etc.")
{{/html}}
{{/velocity}}
General example showing how to display the first 5 results of a given query:
{{code language="none"}}
#set($hql = "<query here>")
#set($results = $xwiki.search($hql, 5, 0))
#foreach ($item in $results)
* $item
#end
{{/code}}
The examples below will show you various HQL queries that you can write.
=== Simple Query ===
{{code language="none"}}
#set($hql = "select doc.name from XWikiDocument doc")
{{/code}}
=== Count Query ===
{{code language="none"}}
#set($results = $xwiki.search("select count(doc) from XWikiDocument doc"))
Since $xwiki.search is returning a list, we get its first element
Count : $results.get(0)
{{/code}}
=== Simple Query with multiple fields ===
{{code language="none"}}
#set($results = $xwiki.search("select doc.name, doc.date from XWikiDocument doc", 5, 0))
#foreach ($col in $row)#if ($velocityCount==1) ### only once per table
#if ($velocityCount == 1)##-------------------------
#set($docName = $col)## establish num columns:
#elseif ($velocityCount == 2)##-------------------------
#set($docDate = $col)#set($numCols=0)
#foreach ($col in $row)
#set($numCols=$velocityCount)
##-------------------------
print header:
##-------------------------
#if ($numCols>0)
#foreach ($col in [1..$numCols]) field $col | #end \
#else
Single field
#end

#end ### end only once per table
##-------------------------
table body:
##-------------------------
#if ($numCols>0)
#foreach ($column in $row) $column | #end \
#else
$row
$docName : $docDate
{{/code}}{table}
#####################################
#end
#####################################
=== Getting objects of a specific class ===1 Velocity HQL query examples.
{{code language="none"}}XWiki allows user to access documents and objects with [hql>http://www.hibernate.org/hib_docs/reference/en/html/queryhql.html] queries in [Velocity>http://jakarta.apache.org/velocity/docs/user-guide.html] scripts.
#set($hql = "select obj.name from BaseObject obj where obj.className='XWiki.XWikiUsers'")
{{/code}}
=== Getting objects' properties ===You can execute queries as follows:
{code}
\#set(\$query="select doc.name from XWikiDocument doc")
\#set(\$results = \$xwiki.search(\$query))
\#foreach (\$item in \$results)
* \$item
\#end
{code}
{{code language="none"}}Here are a few examples:
#set($hql = "select obj.name, prop.value from BaseObject obj, StringProperty prop where obj.className='XWiki.XWikiUsers' and prop.id.id=obj.id and prop.name='first_name'")
{{/code}}
=== Getting documents where objects' properties equals some value ===1.1 Simply Query:
#set($hql="select doc.name from XWikiDocument doc")
#displayQuery($hql)
{{code language="none"}}
#set($hql = "select doc.fullName from XWikiDocument doc, BaseObject obj, StringProperty prop where doc.fullName=obj.name and obj.className='XWiki.XWikiUsers' and prop.id.id=obj.id and prop.name='first_name' and prop.value='Jean-Vincent'")1.1 Simply Query with multiple fields:
{{/code}}Queries with multiple fields require a bit of trickery to access the columns. Please edit if you have a better solution.
{code}
\#set(\$results=\$xwiki.search("select doc.name, doc.date from XWikiDocument doc"))
\#foreach (\$row in \$results)
\#foreach (\$col in \$row)
\#if (\$velocityCount==1)
\#set(\$docName=\$col)
\#elseif (\$velocityCount==2)
\#set(\$docDate=\$col)
\#end

\#end
\#end
\$docName : \$docDate
\#end
{code}
{{code language="none"}}#set($hql="select doc.name, doc.date from XWikiDocument doc")
#set($hql = ", BaseObject as obj, StringProperty as firstName, StringProperty as lastName where doc.fullName = obj.name and obj.className='XWiki.XWikiUsers' and obj.id=firstName.id.id and firstName.id.name='first_name' and obj.id=lastName.id.id and lastName.id.name='last_name' and firstName.value like 'A%' and lastName.value like 'B%' order by doc.fullName asc")#displayQuery($hql)
{{/code}}

1.1 Getting objects of a specific class
#set($hql="select obj.name from BaseObject obj where obj.className='XWiki.XWikiUsers'")
#displayQuery($hql)
{{code language="none"}}
#set($hql = "select doc.fullName from XWikiDocument doc, BaseObject obj, StringProperty prop where doc.fullName=obj.name and obj.className='XWiki.XWikiUsers' and prop.id.id=obj.id and prop.name='first_name' and prop.value='Jean-Vincent'")1.1 Getting objects' properties
{{/code}}#set($hql="select obj.name, prop.value from BaseObject obj, StringProperty prop where obj.className='XWiki.XWikiUsers' and prop.id.id=obj.id and prop.name='first_name'")
If property is multiselect Page type, relational storage must be checked, if not, the content is not stored the same way and we cannot do "join prop.list as list"#displayQuery($hql)
{{code language="none"}}#set ($hql ="select distinct doc.fullName from XWikiDocument as doc, BaseObject as obj, DBStringListProperty as prop join prop.list list where obj.name=doc.fullName and obj.className='MySpace.MyClass' and obj.id=prop.id.id and prop.id.name='MyPropertyName' and list = 'MyValueInMyPropertyName'")
{{/code}}
=== List users currently editing pages ===
{{code language="none"}}
#set($hql = "select distinct lock.userName from XWikiLock lock")
{{/code}}
=== List attachments of a page ===
{{code language="none"}}
#set($hql = "select att.filename from XWikiAttachment att, XWikiDocument doc where doc.fullName='Main.WebHome' and att.docId=doc.id")
{{/code}}
=== Statistics ===
* [[Most Viewed Articles Snippet>>snippets:Extension.Most Viewed Articles]]
* [[Most Active Contributors In Group Snippet>>snippets:Extension.Most Active Contributors In Group]]
* [[Number Of Active Users Per Day And Per Week Snippet>>snippets:Extension.Number Of Active Users Per Day And Per Week]]
* [[Number Of Edited Articles Per Day And Per Week Snippet>>snippets:Extension.Number Of Edited Articles Per Day And Per Week]]
* [[Number Of Created Articles Per Day And Per Week

~~Snippet>>snippets:Extension.Number Of Created Articles Per Day And Per Week]]~~
~~* [[Number Of Deleted Articles Per Day And Per Week~~
~~Snippet>>snippets:Extension.Number of Deleted Articles Per Day And Per Week]]~~
~~== Non-exhaustive list of queryable Object Fields ==~~
~~The full list of available fields can be found in the Hibernate mapping files (*.hbm.xml), which can be found inside WEB-INF/lib/xwiki-core-x.y.jar (where x.y is the XWiki version).~~
~~=== XWikiDocument ===~~
~~* **XWikiDocument.fullName** : full name, including space and page name. Example value: ##Main.WebHome##~~
~~* XWikiDocument.author : last editor. Example value: ##XWiki.Admin##~~
~~* XWikiDocument.creator : first editor. Example value: ##XWiki.Admin##~~
~~=== BaseObject ===~~
~~* **BaseObject.id** : arbitrary unique id of the object. Example value: ##123456789##~~
~~* BaseObject.className : class. Example value: ##XWiki.XWikiUsers##~~
~~=== *Property (StringProperty, IntegerProperty, etc) ===~~
~~* **Property.id.id** : unique id of the object the property belongs to. Example value: ##123456789##~~
~~* Property.name : name of the property. Example value: ##first_name##~~
~~* Property.value : value. Example value: ##John##~~

XWiki.TagClass
tags:
~~HQL~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.StefanBachert~~
- comment:
~~"Property (StringProperty, IntegerProperty, etc)"~~
~~Where do I find a complete list of "etc".~~
~~Neither google anything, and xwiki lucene comes up with an error~~
~~Stefan Bachert~~
- date:
~~2010-03-01 19:18:20.0~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.Sergiu~~
- comment:
~~The actual mapping file defines all the properties that can be used in queries; look at the subclasses of com.xpn.xwiki.objects.BaseProperty inside {{scm path="xwiki-platform-core/xwiki-platform-oldcore/src/main/resources/xwiki.hbm.xml"}}xwiki.hbm.xml{{/scm}}.~~
~~The current list contains:~~
~~* IntegerProperty~~
~~* LongProperty~~
~~* FloatProperty~~
~~* DoubleProperty~~
~~* StringProperty~~
~~* DateProperty~~
~~* LargeStringProperty~~

~~* LargeStringProperty~~
~~* StringListProperty~~
~~* DBStringListProperty~~

- date:
~~2010-03-01 23:25:11.0~~
- replyto:
~~0~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.MilindKamble~~
- comment:
~~The link to HQL in the first line needs to be updated to http://docs.jboss.org/hibernate/stable/core/reference/en/html/queryhql.html~~
- date:
~~2010-03-31 16:09:54.0~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.jvdrean~~
- comment:
~~Done. Note that it's a wiki, you could have done it directly :)~~
- date:
~~2010-03-31 16:16:44.0~~
- replyto:
~~2~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.olaf~~
- comment:
~~If you want to use criteria on a multiselect static list field:~~
~~1. make sure you have switched on 'relational storage' on that field~~
~~2. in the from part of your query use "DBStringListProperty as prop join prop.list list"~~
~~3. in the where clause use "obj.id=prop.id.id and prop.id.name='<property name>' and list='<property value>'"~~
~~Sorry for not feeling competent enough to incorporate this directly into the text above yet. As it took me some time to find out I still thought I'd rather put it in as a comment than not at all.~~
- date:
~~2011-10-29 19:07:16.0~~

XWiki.XWikiRights
- allow:
~~1~~
- groups:
~~xwiki:XWiki.XWikiAdminGroup~~
- levels:
~~edit~~

XWiki.XWikiRights
- allow:
~~1~~
- groups:
~~xwiki:XWiki.XWikiCommittersGroup~~
- levels:
~~edit~~

XWiki.XWikiRights
- allow:
~~1~~
- groups:
~~xwiki:XWiki.XWikiPowerUsers~~
- levels:
~~edit~~

XWiki.XWikiRights
- allow:
~~1~~
- groups:
~~xwiki:XWiki.XWikiAllGroup~~
- levels:
~~edit~~

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ DevGuide.Scripting
- author: XWiki.evalica ⇨ XWiki.jvdrean
- syntax: XWiki 2.1 ⇨ XWiki 1.0

Documentation.DevGuide

SkinExtensionsTutorial
Documentation.DevGuide.SkinExtensionsTutorial

On 2017/09/05, the page has been deleted by Vincent Massol

Skins
Documentation.DevGuide.Skins

On 2017/09/05, the page has been deleted by Vincent Massol

Standard URL Format
Documentation.DevGuide.Standard URL Format

On 2017/09/05, the page has been deleted by Vincent Massol

TranslationsTutorial
Documentation.DevGuide.TranslationsTutorial

On 2017/09/05, the page has been deleted by Vincent Massol

Tutorials
Documentation.DevGuide.Tutorials

On 2017/09/05, the page has been deleted by Vincent Massol

Documentation.DevGuide.Tutorials

Creating a form with validation and tooltips
Documentation.DevGuide.Tutorials.Creating a form with validation and tooltips

On 2017/09/05, the page has been modified by Vincent Massol

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WebHome

On 2017/09/05, the page has been created by Vincent Massol

~~{{box cssClass="floatinginfobox" title="**Contents**"}}~~**1 Validation Sample**
~~{{toc/}}~~
~~{{/box}}~~
~~Download~~**{attach:validation-sample-1.0.xar|Download** the
~~[[sample~~**sample** in xar ~~format>>attach:validation-sample-1.0.xar]].~~**format}**
~~You will need to add ##ValidationSample.Translations## to your translations bundles in the advanced section of the global preferences.~~**1.1 Known issues:**
~~= How does my form look like when validation errors are shown =~~**The following patches need to be appied on top of XWiki 1.3.2 or later:**
~~{{image reference="formvalidationedit.png"/}}~~*****
http://jira.xwiki.org/jira/browse/XWIKI-2383: "Document.displayPrettyName should look for translations" otherwise the pretty names of fields will not how up.
*** http://jira.xwiki.org/jira/browse/XWIKI-2384: "Add optional validation in SaveAction" otherwise the validation will not work when editing a document (using the inline mode).**
*** http://jira.xwiki.org/jira/browse/XWIKI-2385: "displayPrettyName should show mandatory if param is true regardless of a regular expression setting" otherwise the mandatory modified will not appear for the "age" field.**
~~= Documents, Class and Scripts =~~**1.1 How does my form look like when validation errors are shown**
~~* //ValidationSample.WebHome// Home page where you can create a new document and access the existing ones~~**{image:formvalidationedit.png}**
~~* //ValidationSample.ValidationSampleClassSheet// Sheet presenting the document in create, edit and view mode including validation error messages~~
~~* //ValidationSample.CreateDoc// Page called on the submission of the create document form. This will validate and show the form with errors or will save the document.~~
~~* //ValidationSample.ValidationSampleClass// Class with definitions of fields, regular expressions and error message translations strings~~
~~* //ValidationSample.ValidationGroovy// Groovy validation script for complex validations~~
~~* //ValidationSample.ValidationSampleClassTemplate// Template of a document~~
~~* //ValidationSample.Translations// Translations of the texts, tooltips and error messages. This shows an example of the naming conventions for~~

~~error messages. This shows an example of the naming conventions for tooltips and pretty names~~
~~* //ValidationSample.Val//, //ValidationSample.Val_0//, //ValidationSample.Val_1// Sample documents~~
~~= How to create validations using regular expressions =~~**1.1 Documents, Class and Scripts:**
*** ValidationSample.WebHome Home page where you can create a new document and access the existing ones**
*** ValidationSample.ValidationSampleClassSheet Sheet presenting the document in create, edit and view mode including validation error messages**
*** ValidationSample.CreateDoc Page called on the submission of the create document form. This will validate and show the form with errors or will save the document.**
*** ValidationSample.ValidationSampleClass Class with definitions of fields, regular expressions and error message translations strings**
*** ValidationSample.ValidationGroovy Groovy validation script for complex validations**
*** ValidationSample.ValidationSampleClassTemplate Template of a document**
*** ValidationSample.Translations Translations of the texts, tooltips and error messages. This shows an example of the naming conventions for tooltips and pretty names**
*** ValidationSample.Val ValidationSample.Val_0 ValidationSample.Val_1 Sample documents**
1.1 How to create validations using regular expressions
~~Then,~~**The following code should be called** to ~~perform~~**launch a** validation ~~after~~**on** a ~~standard "Save" in the form, following code is needed:~~**document**
~~{{code language="html"}}~~**{code}**
~~<input type="hidden" name="xvalidate" value="1" />~~**## this will launch a validation on a document. All errors are added to the context**
~~{{/code}}~~**$doc.validate()**
{code}
~~{{info}}~~**The following code is need in the form to launch the validation for a standard Save to validate the form**
~~The code above is sufficient to perform validation with regular expressions defined in the Class properties.~~
~~{{/info}}~~
~~Pay attention to the Validation Regular Expression and Validation Message fields. The first one is [[a Java Regular Expression pattern>>http://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html]] and the second one is a translation string:~~**{code}**
<input type="hidden" name="xvalidate" value="1" />
{code}
Pay attention to the Validation Regular Expression and Validation Message fields. The first one is [a Java Regular Expression pattern>http://java.sun.com/j2se/1.4.2/docs/api/java/util/regex/Pattern.html] and the second one is a translation string:
~~**/^.{2,20}$/****~~** **/.{2,20}/** -> ~~this~~ field needs to be between 2 characters and 20 ~~characters. If the field can have new lines, enable the [[dotall mode>>http://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html#DOTALL]] by adding an **s** at the end of the regex (##/^.{2,20}$/s##), otherwise a regex that contains a new line will not pass validation.~~**characters.**
~~**val_firstname_toolong****~~** **val_firstname_toolong** -> XWiki will lookup this translation string in the translations pages
* ~~last_name~~ **last_name**
~~**/^.{2,20}$/****~~** **/.{2,20}/** -> ~~this~~ field needs to be between 2 **characters** and

20 characters.
* ~~email~~ **email**
~~**{{{/.*@.*.com$/}}}**~~ **/.*@.*\.com/** -> ~~this~~ field must contain ~~**@**~~**@** and finish with ~~**.com**~~**.com**
* ~~age~~ **age**
** no validation set for age. This will be handled by the groovy script
* ~~usphone~~ **usphone**
~~**{{{/^[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]$/}}}**~~** **{pre} /[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]/ {/pre}** -> the phone number must be made of digits separated by - in the form 000-000-0000
~~Other Validation Regular Expression examples:~~
{image:formvalidationclass.png}
~~* do not match //XWiki.XWikiGuest//, but allow it~~**1.1 How** to ~~be inside the string: **{{{/^(?!XWiki.XWikiGuest$).*/}}}**~~**create validations using a groovy script**
~~* forbid //XWiki.XWikiGuest// anywhere in the string: **~~
~~{{{/^(?!.*XWiki.XWikiGuest).*/}}}**~~
~~{{image reference="formvalidationclass.png"/}}~~**To create complex validation you can use a groovy script.**
~~To call validation dynamically, the~~**The** following ~~Velocity~~ code should be ~~called:~~**called to launch a validation on a document**
~~{{code}}~~**{code}**
add the groovy script to the validation (the xvalidation parameter in the request also works to set the groovy script validation)
$doc.setValidationScript("ValidationSample.ValidationGroovy")
~~{{/code}}~~**{code}**
~~= How to create validations using a groovy script =~~
~~To create complex validations you have~~**The following code is need in the form** to ~~use~~**launch the validation on** a ~~Groovy script.~~ **standard Save of a document:**
~~== Invoking Groovy script ==~~**{code}**
~~Groovy validation script can be invoked using two approaches:~~**## Force validation**
~~~~1. Using HTML code in the form, the validation on a standard "Save" of a document:~~
~~{{code language="html"}}~~**## Invoke validation**
Set the ~~page, which Groovy script will be used for validation~~**validation script. This is necessary until the bug http://jira.xwiki.org/jira/browse/XWIKI-2382**
<input type="hidden" name="xvalidation" value="ValidationSample.ValidationGroovy" ~~/>{{/code}}~~**/>**
{code}
~~2. Or dynamically using Velocity code:~~
~~{{code}}## set the page, which Groovy script will be used for validation~~
~~$doc.setValidationScript("ValidationSample.ValidationGroovy")~~
~~## invoke document validation.~~
~~$doc.validate()~~~~{{/code}}~~
~~{{info}}~~
~~After document validation all errors are added to the context.~~
~~{{/info}}~~
~~== Groovy script sample ==~~
~~{{warning}}~~**{code}**
~~Do not use the **~~~~{~~~~{groovy}}** macro when creating your script, just paste your code in the wiki editor.~~
~~{{/warning}}~~
~~{{code language="java"}}~~
~~{{/code}}~~
~~~~

= How to display validation error messages ={code}
== Display all1.1 How to display validation error message at the top of the document ==messages
1.1.1 Display all validation error message at the top of the document
{{code}}{code}
#if(($context.validationStatus.errors&&$context.validationStatus.errors.size()>0)||
($context.validationStatus.exceptions&&$context.validationStatus.exceptions.size()>0))#if(($context.validationStatus.errors&&$context.validationStatus.errors.size()>0)||
($context.validationStatus.exceptions&&$context.validationStatus.exceptions.size>0))
{{/code}}{code}
This is called using {{code}}#showallerrors(){{/code}}.using
== Display the validation error messages next to the field ==
{code}
#showallerrors()
{code}
The following macro can be used to show1.1.1 Display the validation error called ##$message##:messages next to the field
{{code}}The following macro can be used to show the validation error called $\message
{code}
{{/code}}{code}
This is called using
{{code}}#showvalidationmessage("val_firstname_toolong"){{/code}}.using
= How to create tooltips, mandatory icon and the pretty name ={code}
#showvalidationmessage("val_firstname_toolong")
{code}
The following macro shows a field including the1.1 How to create tooltips, mandatory field, the tooltipicon and the pretty name:name
{{code}}The following macro shows a field including the mandatory field, the tooltip and the pretty name
{code}
displayPrettyName will get the translation only with patch in
https://jira.xwiki.org/browse/XWIKI-2383http://jira.xwiki.org/jira/browse/XWIKI-2383
{{/code}}{code}
This is called using {{code}}#showfield("first_name",true){{/code}}.using:
{code}
#showfield("first_name",true)
{code}
{{code}}{code}
{{/code}}{code}
==1.1.1 The tooltip ==tooltip
{{code}}{code}
{{/code}}{code}
==1.1.1 The pretty name with mandatory icon ==icon
{{code}}{code}
{{/code}}{code}
==1.1.1 The pretty name without mandatory icon ==icon
{{code}}{code}
{{/code}}{code}
=1.1 How to validate and save the document in CreateDoc =CreateDoc
{{code}}{code}
this does not work yet because of bug
https://jira.xwiki.org/browse/XWIKI-2382http://jira.xwiki.org/jira/browse/XWIKI-2382
{{/code}}{code}

~~{{/code}}~~**{code}**
=**1.1** Complete presentation sheet of the ~~document =~~**document**
~~{{code}}~~**{code}**
Set the validation script. This is necessary until the bug
~~https://jira.xwiki.org/browse/XWIKI-2382~~**http://jira.xwiki.org/jira/browse/XWIKI-2382**
displayPrettyName will get the translation only with patch in
~~https://jira.xwiki.org/browse/XWIKI-2383~~**http://jira.xwiki.org/jira/browse/XWIKI-2383**
~~{{/code}}~~
~~= Client side validation with LiveValidation =~~**{code}**
~~It's important to have server side validation for security and because not everyone uses Javascript but client side validation while the user types can improve the user's experience and save server load from forms submitted over and over again.~~
~~To do validation on the client side you have to use the LiveValidation Javascript code. You can define your own style for validation error messages or you can use the style sheet which is used by [[XWiki.Registration>>extensions:Extension.Administration Application#HVerifiedRegistrationPage28SinceEnterprise2.2M229]].~~
~~Here is a simple example of how to use LiveValidation in XWiki:~~
~~{{code}}~~
~~{{velocity}}~~
~~$xwiki.get('jsfx').use('uicomponents/widgets/validation/livevalidation_prototype.js')~~
~~$xwiki.get('ssfx').use('uicomponents/widgets/validation/livevalidation.css')~~
~~{{/velocity}}~~
~~{{html}}~~
~~<form action="">~~
~~<label for="helloField">Say hello to LiveValidation:</label>~~
~~<input id="helloField" length="20" type="text">~~
~~</form>~~
~~<script>~~
~~/* <![CDATA[*/~~
~~document.observe('dom:loaded', function() {~~
~~var helloField = new LiveValidation("helloField", { validMessage: "Hi There.", wait: 500});~~
~~helloField.add(Validate.Presence, { failureMessage: "Say Something, anything..."});~~
~~helloField.add(Validate.Format, { pattern: /^[Hh]ello$/, failureMessage: "How about saying 'Hello'?"});~~
~~});//]]>~~
~~</script>~~
~~{{/html}}~~
~~{{/code}}~~
~~The result is this:~~
~~{{velocity}}~~
~~$xwiki.get('jsfx').use('uicomponents/widgets/validation/livevalidation_prototype.js')~~
~~$xwiki.get('ssfx').use('uicomponents/widgets/validation/livevalidation.css')~~
~~{{/velocity}}~~
~~{{html}}~~
~~<form action="">~~
~~<label for="helloField">Say hello to LiveValidation:</label>~~
~~<input id="helloField" length="20" type="text">~~
~~</form>~~
~~<script>~~
~~/* <![CDATA[*/~~

~~document.observe('dom:loaded', function() {~~
~~var helloField = new LiveValidation("helloField", { validMessage: "Hi There.", wait: 500});~~
~~helloField.add(Validate.Presence, { failureMessage: "Say Something, anything..."});~~
~~helloField.add(Validate.Format, { pattern: /^[Hh]ello$/, failureMessage: "How about saying 'Hello'?"});~~
~~});//]]>~~
~~</script>~~
~~{{/html}}~~
~~This example shows validation of [[presence>>http://livevalidation.com/documentation#ValidatePresence]] (something must be written) and validation of [[format>>http://livevalidation.com/documentation#ValidateFormat]] (testing against a regular expression).~~
~~Notice how the first line of Javascript says **new LiveValidation("helloField"**. This binds the validation to field with the id "helloField". Text validation also works on TextAreas.~~
~~There are more types of validation which are documented in the [[LiveValidation documentation>>http://livevalidation.com/documentation]].~~
~~To change the look and feel of the error message, you may define your own CSS [[skin extension>>platform:DevGuide.SkinExtensionsTutorial]] instead of using the one provided.~~
~~All validation messages are of the class (% class="LV_validation_message" %)##LV_validation_message##(%%). The error messages are of the class (% class="LV_invalid" %)##LV_invalid##(%%) and the valid messages are of class (% class="LV_valid" %)##LV_valid##(%%). LiveValidation will also detect submit buttons and bind to them, blocking them if fields are invalid. There are ways around this such as using prototype's [[Event.stopObserving>>http://prototypejs.org/doc/latest/dom/Event/stopObserving/]] function.~~

formvalidationclass.png: 1.2 ⇨ ()

formvalidationedit.png: 1.2 ⇨ ()

validation-sample-1.0.xar: 1.1 ⇨ ()

XWiki.XWikiComments

- author:
~~xwiki:XWiki.zed0~~
- comment:
~~This validation appears to break if "Save and continue" instead of "Save and view" is pressed and the fields are not correct. Is there any chance of a fix for this?~~
- date:
~~2008-08-08 17:00:54.0~~

- title: Creating a form with validation and tooltips ⇨
- author: XWiki.VincentMassol ⇨ XWiki.LudovicDubost
- syntax: XWiki 2.0 ⇨ XWiki 1.0

Documentation.DevGuide.Tutorials

Creating a Tree View

Documentation.DevGuide.Tutorials.CreatingATreeView.WebHome

On 2017/09/05, the page has been created by Vincent Massol

~~{{template name="diff_macros.vm"/}}~~
#macro (unifiedDiff $previous $next)
#set ($discard = $xwiki.ssfx.use('uicomponents/viewers/diff.css', true))

<table>
#foreach ($block in $services.diff.display.unified($previous, $next))
<tr>
<td class="diff-line-number">...</td>
<td class="diff-line-number">...</td>
<td class="diff-line diff-line-meta">@@ -$mathtool.add($block.previousStart, 1),$block.previousSize +$mathtool.add($block.nextStart, 1),$block.nextSize @@</td>
</tr>
#foreach ($line in $block)
<tr>
#set ($lineNumber = $line.index + 1)
<td class="diff-line-number">#if (!$line.added)$lineNumber#end</td>
<td class="diff-line-number">#if (!$line.deleted)$lineNumber#end</td>
<td class="diff-line diff-line-$line.type.toString().toLowerCase()">#unifiedDiffLine($line)</td>
</tr>
#end
#end
</table>

#end
#macro (unifiedDiffLine $line)
${line.type.symbol}##
#if ($line.chunks)##
#inlineDiffLine($line.chunks)##
#else##
$escapetool.xml($line.value)##
#end
#end
#macro (inlineDiffLine $chunks)
#foreach ($chunk in $chunks)
#set ($escapedChunk = $escapetool.xml($chunk))
#if ($chunk.added)##
<ins>$escapedChunk</ins>##
#elseif ($chunk.deleted)##
~~$escapedChunk~~##
#else##
$escapedChunk##
#end
#end
#end
{{toc ~~depth="1"/}~~depth="1" /}}
The most easiest way to create a tree is by using the [[Tree ~~Macro>>extensions:Extension.Tree Macro~~]]Macro>>http://extensions.xwiki.org/xwiki/bin/view/Extension/Tree+Macro]] with wiki syntax.

As you can see we're using ~~[[Solr>>extensions:Extension.Solr Search Application]]~~
[[Solr>>http://extensions.xwiki.org/xwiki/bin/view/Extension/Solr+Search+Application]] to retrieve the suggestions.
= Dynamic Team Hierarchy Tree ~~v4~~**v1** (context menu) =
= Dynamic Team Hierarchy Tree ~~v5~~**v1** (drag & drop) =

- parent: DevGuide.WebHome ⇨
- author: XWiki.VincentMassol ⇨ XWiki.mflorea

On 2017/09/05, the page has been modified by Vincent Massol

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WebHome

Documentation.DevGuide.Tutorials

Creating an Extension

Documentation.DevGuide.Tutorials.CreatingExtensions.WebHome

On 2017/09/05, the page has been created by Vincent Massol

~~{{box cssClass="floatinginfobox" title="**Contents**"}}~~
~~{{toc/}}~~
~~{{/box}}~~
The simplest strategy is to create the wiki pages in a running XWiki instance and then to [[export them as a ~~XAR>>platform:Features.Exports||anchor="HXARExport"]].~~**XAR>>Features.Exports||anchor="HXARExport"]].**
* ~~[[scripts>>platform:DevGuide.Scripting]],~~
[[scripts>>DevGuide.Scripting]],
* [[wiki ~~macros>>platform:DevGuide.WikiMacroTutorial]],~~**macros>>DevGuide.WikiMacroTutorial]],**
If you wish to save your XAR sources under [[version control>>https://en.wikipedia.org/wiki/Version_control]] and be able to build it, we recommend using [[Maven>>https://maven.apache.org/]]. ~~There's~~**We have developed** a plugin called the [[XAR Maven plugin>>dev:Community.XARPlugin]] to help with this.
Here's an example of how your ##pom.xml## could look like (adapt to your ~~need).~~**need):**
~~Notice the usage of a Contrib Parent POM which automatically provides configuration for lots of things. To know more about how to use the Contrib Parent POM check the [[documentation for it on GitHub>>https://github.com/xwiki-contrib/parent]].~~
~~<version>8.4-6</version>~~**<version>7.4-6</version>**
~~<!-- Issue management -->~~
~~<xwiki.release.jira.skip>false</xwiki.release.jira.skip>~~
~~<xwiki.issueManagement.jira.id>(your jira project id)~~
~~</xwiki.issueManagement.jira.id>~~
<distributionManagement>
<repository>
<id>xwiki-staging</id>
<name>XWiki Staging Repository</name>
<url>http://nexus.xwiki.org/nexus/service/local/staging/deploy/maven2/</url>
</repository>

</distributionManagement>
<issueManagement>
<system>jira</system>
<url>http://jira.xwiki.org/jira/browse/(your jira project id)</url>
</issueManagement>
* **{{info}}New{{/info}}** Category (using ##<xwiki.extension.category>##). Valid values are [[listed here>>extensions:Extension.Repository ~~Application||anchor="HCategories"]].~~**Application#HCategories]].**
* Issue ~~Management (most of the time only the jira id as a Maven property like in the example)~~**Management**
LTS are always provided for https://github.com/xwiki-contrib/parent, in case the version you want is not listed you ~~can:~~**can either ask for it to be released in the Mailing List or use ##org.xwiki.commons:xwiki-commons-pom## instead.**
~~* use the closest LTS as parent but use a different version for the dependencies~~
~~* ask for it to be released in the Mailing List~~
~~* use ##org.xwiki.commons:xwiki-commons-pom## instead~~
* [[Write a ~~Listener>>platform:DevGuide.WritingEventListenerTutorial]],~~**Listener>>DevGuide.WritingEventListenerTutorial]],**
~~Follow~~**We recommend following** the [[Creating an XWiki Component ~~tutorial>>platform:DevGuide.WritingComponents]]~~**tutorial>>DevGuide.WritingComponents]]** to learn how to both develop a Component and use Maven to build it.
~~There are 2 ways but the recommended one is to use the [[Extension Manager>>extensions:Extension.Extension Manager Application]] which you can find in the XWiki instance where you wish to install the extension in.~~**{{todo/}}**
~~== Using the Extension Manager ==~~
~~The advantage over the Manual way is that you don't need to regularly start/stop your XWiki instance and thus you don't occur the start wait times.~~
~~* Have a running XWiki instance configured with a local Extension Repository pointing to your Maven local Repository. Edit ##xwiki.properties## and make sure you have the following set:(((~~
~~{{code language="none"}}~~
~~extension.repositories=local:maven:file://${sys:user.home}/.m2/repository~~
~~extension.repositories=maven-xwiki:maven:http://nexus.xwiki.org/nexus/content/groups/public~~
~~extension.repositories=extensions.xwiki.org:xwiki:http://extensions.xwiki.org/xwiki/rest/~~
~~{{/code}}~~
~~)))~~
~~* Build your component and deploy it in your local Maven repository with ##mvn install##~~
~~* Inside your running XWiki instance, go to the Extension Manager in the Admin UI (e.g. ##~~
~~{{{http://localhost:8080/xwiki/bin/admin/XWiki/XWikiPreferences?editor=globaladmin&section=XWiki.AddExtensions}}}##) and click on Advanced Search and enter your extension's id and version and follow the instructions.~~
~~{{warning}}~~
~~If you want to redeploy an extension and it's already installed with the same version, the Extension Manager won't let you do so. Thus you'll need to uninstall it first using the Extension Manager. You'll also need to remove metadata in memory using the [[Extension Tweak>>extensions:Extension.Extension Tweak]].~~
~~{{/warning}}~~

{{/warning}}
== Manually ==
=== For a XAR ===
* To build the component, issue ##mvn install##. This generates a XAR in the ##target## directory of your project.
* Inside your XWiki instance, go the Admin and [[Import the XAR>>platform:Features.Imports]].
=== For a JAR ===
* To build the component, issue ##mvn install##. This generates a JAR in the ##target## directory of your project.
* To install it into a XWiki Enterprise instance, just copy that JAR file in ##XE_WAR_HOME/WEB-INF/lib## where ##XE_WAR_HOME## is where the XWiki Enterprise WAR is deployed.
If you wish to make your extension available to others to use, the best is to contribute it on [[extensions.xwiki.org>>extensions:Main.WebHome]]. Go there and enter a name for your extension in the Contribute box and submit. Then document nicely your extension with instructions on how to use. The more screenshots the better! Here are some [[documentation guidelines>>contrib:Main.WebHome||anchor="HDocumenting"]].**{{todo/}}**
Once your extensions is there, it means that any user of XWiki in the world will be able to find it and install it directly from his/her wiki!
You may also want to join a family and contribute your extension on [[XWiki Contrib>>contrib:Main.WebHome]] so that it is developed and maintained collaboratively.

- parent: xwiki:Documentation.DevGuide.WebHome ⇨

On 2017/09/05, the page has been modified by Vincent Massol

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WebHome
- author: XWiki.evalica ⇨ XWiki.VincentMassol

Documentation.DevGuide.Tutorials

Creating New XClass Property Types

Documentation.DevGuide.Tutorials.CreatingNewXClassPropertyTypes.WebHome

On 2017/09/05, the page has been modified by Vincent Massol

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WebHome

On 2017/09/05, the page has been created by Vincent Massol

{{box cssClass="floatinginfobox" title="**Contents**"}}
{{toc start="2"/}}
{{/box}}
This tutorial guides you through the creation of a new [[XClass property type>>DataModel||anchor="HXWikiClasses2CObjects2CandProperties"]], which is a way to extend the [[class editor>>platform:Features.PageEditing||anchor="HClasseseditingmode"]].
{{warning}}
This tutorial applies to XWiki version 4.3M2 and above. Don't try on older versions!
{{/warning}}
You should start by reading the [[XWiki Data Model>>DataModel]] to understand XWiki Classes, Objects and Properties, and then the [[Writing

~~XWiki components>>platform:DevGuide.WritingComponents]] tutorial because new property types are implemented as components.~~
~~== Create a new property type ==~~
~~As an example, we will create an "External Image" property type which can be used to store URLs to external images. Let's start by creating the actual property type which must extend ##PropertyClass##. Unfortunately this means that your Maven project will have to depend on XWiki's old core.~~
~~{{code language="java"}}~~
~~public class ExternalImageClass extends PropertyClass~~
~~{~~
~~/**~~
~~* Default constructor.~~
~~*/~~
~~public ExternalImageClass()~~
~~{~~
~~// Specify the default name and pretty name of this XClass property. They can be overwritten from the class~~
~~// editor when adding a property of this type to an XClass.~~
~~super("externalImage", "External Image", null);~~
~~}~~
~~@Override~~
~~public BaseProperty fromString(String value)~~
~~{~~
~~BaseProperty property = newProperty();~~
~~// The stored value can be different than the value set by the user. You can do the needed transformations here.~~
~~// In our case the value is an image URL so we keep it as it is. The reverse transformation, from the stored~~
~~// value to the user friendly value, can be done in the property displayer.~~
~~property.setValue(value);~~
~~return property;~~
~~}~~
~~@Override~~
~~public BaseProperty newProperty()~~
~~{~~
~~// The value of this XClass property is stored as a String. You have to use raw types here like StringProperty~~
~~// because they are mapped to the database. Adding a new raw type implies modifying the Hibernate mapping and is~~
~~// not the subject of this tutorial.~~
~~BaseProperty property = new StringProperty();~~
~~property.setName(getName());~~
~~return property;~~
~~}~~
~~@Override~~
~~public <T extends EntityReference> void mergeProperty(BaseProperty<T> currentProperty,~~
~~BaseProperty<T> previousProperty, BaseProperty<T> newProperty,~~
~~MergeConfiguration configuration,~~
~~XWikiContext context, MergeResult mergeResult)~~
~~{~~
~~if (!Objects.equals(previousProperty, newProperty)) {~~
~~if (Objects.equals(previousProperty, currentProperty)) {~~
~~currentProperty.setValue(newProperty.getValue());~~
~~} else {~~
~~// Found conflict~~
~~mergeResult.getLog().error("Collision found on property [{}] current has~~

been modified", getName());
}
}
}
}
{{/code}}
Notice that we have used a ##StringProperty## to store the value of our XClass property. The available raw property types that can be used for storing the value in the database are:
* ##DateProperty##
* ##DBStringListProperty##
* ##DoubleProperty##
* ##FloatProperty##
* ##IntegerProperty##
* ##LargeStringProperty##
* ##LongProperty##
* ##StringListProperty##
* ##StringProperty##
Extending this list is not possible without modifying the Hibernate mapping and is not the subject of this tutorial. You can create high level XClass property types but in the end their values will be stored as one of these raw types.
Also as you can see we have overwritten the default 3-way-merge implementation for this kind of properties to make sure the URL is not merged character by character but instead compared as a whole.
Next let's create a provider for our "External Image" property type. This is going to be used whenever a property of type "External Image" is added to an XClass (e.g. from the class editor).
{{code language="java"}}
@Component
// Note that the component hint matches the name of the property class without the "Class" suffix. The reason is that
// the component hint must match the value returned by the #getClassType() method of your property class, which by
// default strips the "Class" suffix from the Java class name of your property class. If you want to use a different
// hint that doesn't follow this naming convention you need to override #getClassType().
@Named("ExternalImage")
@Singleton
public class ExternalImageClassProvider implements PropertyClassProvider
{
@Override
public PropertyClassInterface getInstance()
{
return new ExternalImageClass();
}
@Override
public PropertyMetaClassInterface getDefinition()
{
PropertyMetaClass definition = new PropertyMetaClass();
// This text will appear in the drop down list of property types to choose from in the class editor.
definition.setPrettyName("External Image");
definition.setName(getClass().getAnnotation(Named.class).value());
// Add a meta property that will allows us to specify a CSS class name for the image HTML element

the image HTML element.
// NOTE: We define meta properties using XClass property types. This means for instance that you can define meta
// properties of External Image type or whatever XClass property type you create.
StringClass styleName = new StringClass();
styleName.setName("styleName");
styleName.setPrettyName("Style Name");
definition.safeput(styleName.getName(), styleName);
// The alternative text is required for a valid image HTML element so we add a meta property for it.
StringClass placeholder = new StringClass();
placeholder.setName("placeholder");
placeholder.setPrettyName("Alternative Text");
definition.safeput(placeholder.getName(), placeholder);
// Add more meta properties here.
return definition;
}
}
{{/code}}
The provider acts like a factory for our property type but it also defines the list of meta properties. Each XClass property type has a list of meta properties that control how the property is displayed, how it's value is parsed, and so on. The values of these meta properties are shared by all instances of an XClass. So for instance, if you create a XClass with an "External Image" property and set, from the class editor, the "styleName" meta property to "icon" then all objects of that XClass will use that value.
The final step is to add the provider component to ##components.txt##.
{{code language="none"}}
org.xwiki.example.internal.ExternalImageClassProvider
{{/code}}
Now you can build your Maven project and copy the generated jar to the ##WEB-INF/lib## folder of your XWiki instance. Restart the server and you're done.
== Use the new property type ==
Let's create a class that has a property of type "External Image". You should see "External Image" listed in the drop down list box in the [[class editor>>platform:Features.PageEditing||anchor="HClasseseditingmode"]].
{{image reference="addProperty.png"/}}
After you add the property to the class you can set all its meta properties. You'll notice that each property has a list of standard meta properties, like name or pretty name, and some specific meta properties, like style name and alternative text in our case.
{{image reference="editProperty.png"/}}
Save the class and let's add an object (instance) of this class to a wiki page. For this you have to edit the wiki page in [[object mode>>platform:Features.PageEditing||anchor="HObjectseditingmode"]].
{{image reference="editObject.png"/}}
Save the page and let's create a [[sheet>>extensions:Extension.Sheet Module]] for our class.
{{code language="none"}}
{{velocity}}
; $doc.displayPrettyName('screenshot', false, false)
: $doc.display('screenshot')
{{/velocity}}
{{/code}}
Now if you look at the wiki page that has the object you'll see that it doesn't look too good. This is because the "External Image" property type uses a

~~default displayer that only displays the value (the image URL).~~
~~{{image reference="viewModeNoDisplayer.png"/}}~~
~~== Write a displayer for the new property type ==~~
~~We can improve the display of our "External Image" properties by creating a custom displayer. One way to achieve this is by creating a Velocity template ##displayer_externalimage.vm## under the ##/templates/## folder.~~
~~{{code language="none"}}~~
~~#if ($type == 'edit')~~
~~#set($id = $escapetool.xml("${prefix}${name}"))~~
~~<input type="text" id="$!id" name="$!id" value="$!escapetool.xml($value)" />~~
~~#elseif ($type == 'view')~~
~~~~
~~#else~~
~~## In order for the custom displayer to be taken into account, the result of its evaluation with an unknown display~~
~~## mode must not be empty. Let's output something.~~
~~Unknown display mode.~~
~~#end~~
~~{{/code}}~~
~~You can read more about [[custom displayers>>xwiki:ReleaseNotes.ReleaseNotesXWikiEnterprise42M2||anchor="HDefaultcustomdisplayersforEasiercustomizationofthewayobjectfieldsaredisplayed"]]. The wiki page should look better in view mode now.~~
~~{{image reference="viewMode.png"/}}~~
~~In edit mode it will look the same but you can extend the displayer to provide image preview for instance.~~
~~{{image reference="editMode.png"/}}~~

addProperty.png: 1.1 ⇨ ()

displayer_externalimage.vm: 1.1 ⇨ ()

editMode.png: 1.1 ⇨ ()

editObject.png: 1.1 ⇨ ()

editProperty.png: 1.1 ⇨ ()

viewMode.png: 1.1 ⇨ ()

viewModeNoDisplayer.png: 1.1 ⇨ ()

xclass-property-image.zip: 1.1 ⇨ ()

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ DevGuide.WebHome
- author: XWiki.VincentMassol ⇨ XWiki.mflorea

Documentation.DevGuide.Tutorials

Creating Plugins
Documentation.DevGuide.Tutorials.CreatingPlugins.WebHome

On 2017/09/05, the page has been modified by Vincent Massol

XWiki.XWikiComments
comment:

parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WebHome

On 2017/09/05, the page has been created by Vincent Massol

~~{{box cssClass="floatinginfobox" title="**Contents**"}}~~**1 Creating Plugins**
~~{{toc/}}~~
~~{{/box}}~~
~~{{warning}}~~**Plugins are quite handy when you want to interact with third-party code from the Velocity context. Check the [Code Zone>Code.WebHome] for a list of existing plugins.**
~~Plugins are the old way of writing XWiki extensions. The new way is to [[write a Component>>platform:DevGuide.WritingComponents]].~~
~~{{/warning}}~~
~~Plugins~~**Here** are ~~quite handy when you want to interact with third-party code from~~ the ~~Velocity context. Check the [[Extensions wiki>>extensions:Main.WebHome]] for~~**steps to develop** a ~~list of existing plugins.~~**Hello World plugin:**
~~Here are the steps to develop a "Hello World"~~**1.1 The** plugin ~~and more.~~**architecture**
~~= The plugin architecture =~~
* The ~~**plugin**~~***plugin*** itself: ~~it must implement the [[XWikiPluginInterface>>https://fisheye2.atlassian.com/browse/xwiki/xwiki-platform/core/trunk/xwiki-core/src/main/java/com/xpn/xwiki/plugin/XWikiPluginInterface.java]] interface. For simplicity you can also extend the [[XWikiDefaultPlugin>>https://fisheye2.atlassian.com/browse/xwiki/xwiki-platform/core/trunk/xwiki-core/src/main/java/com/xpn/xwiki/plugin/XWikiDefaultPlugin.java]] class which is an adapter to the XWikiPluginInterface. The plugin~~**inherits from [XWikiDefaultPlugin>http://build.xpertnet.biz/doc/api/com/xpn/xwiki/plugin/XWikiDefaultPlugin.html].** contains the core functions of your plugin. They will not be accessible ~~from scripting (without programming rights).~~**form the outside**
* Its ~~**API**: it should extend the [[Api>>https://fisheye2.atlassian.com/browse/xwiki/xwiki-platform/core/trunk/xwiki-core/src/main/java/com/xpn/xwiki/api/Api.java]] class.~~***API*: inherits from [Api>http://build.xpertnet.biz/doc/api/com/xpn/xwiki/api/Api.html].** Will contain all the ~~public methods,~~**methods** accessible ~~from scripting.~~**form the Velocity context.**
~~Although you can write~~**1.1 Build** the ~~functionality inside the API, this is not recommended; the plugin functionality is written in the //hidden// part ("hidden" as in "not publicly accessible"), and the API can filter the access to privileged users, beautify the method names or parameter list, etc., or simply forward the call to the hidden part.~~**plugin**
~~== Plugin lifecycle ==~~**First of all let's *declare our plugin class*:**
~~When the XWiki engine is initialized, the Plugin Manager calls the class constructor for all the enabled plugins (classes implementing the com.xpn.xwiki.plugin.XWikiPluginInterface). For each plugin the class constructor is called only once, and the plugin manager calls the init(XWikiContext) method of the plugin. Each time a plugin is referenced by~~

a Velocity script, for example, when you call a method served by the plugin API:{code:java}
{{code}}#set($helloWorldText = "$xwiki.helloworld.hello()"){{/code}}
or when you ask the XWiki instance for the plugin API object :
{{code}}
#set($pluginObject = $xwiki.getPlugin("helloworld")
#* the name given as argument of getPlugin() should be
the one returned by the getName() method of the Plugin class.
*#
{{/code}}
XWiki calls the getPluginApi() method for the plugin's instance, which itself creates an instance of the class com.xpn.xwiki.plugin.PluginApi. This is why you should not store things in fields of the class extending PluginApi in your plugin, since the usual behavior for the getPluginApi() method is to create a new instance of the PluginApi class every time Velocity needs to access the API for your plugin. This is not true if you store the returned plugin API in a variable, for example:
{{code}}
#set($myPluginApi = $xwiki.helloworld)
{{/code}}
The myPluginApi variable will point to the same object as long as the variable exists. You can declare fields in your plugin class instead, since there is only one instance of this class, whose lifecycle spans over the entire servlet's lifecycle.
= Write the plugin =
First of all let's **declare our plugin class**:
{{code language="java"}}
{{/code}}{code}
Then let's **implement*overload the needed constructor**:inherited constructor*:
{{code language="java"}}{code:java}
} init(context);
{{/code}}{code}
Set*Set a method to get the name of the plugin:plugin*. That's how we will call it from Velocity. For example, we will be able to use our plugin with ##$xwiki.helloworld.myMethod()##;\$xwiki.helloworld.myMethod()
{{code language="java"}}{code:java}
{{/code}}{code}
Write*Write a method to get the plugin API:API*. Don't forget to cast the plugin.
{{code language="java"}}{code:java}
{{/code}}{code}
Overload*Overload the cache flush method (optional):method* (optional)
{{code language="java"}}{code:java}
{{/code}}{code}
Optionally, we can **create*create a [[log4j>>http://logging.apache.org/log4j/1.2/apidocs/index.html]] instance** [log4j>http://logging.apache.org/log4j/docs/index.html] instance* for the plugin:
{code:java}
private static Log mLogger =
LogFactory.getFactory().getInstance(com.xpn.xwiki.plugin.helloworld.
HelloWorldPlugin.class);
{code}
{{code language="java"}}
private static final Log LOG = LogFactory.getLog(HelloWorldPlugin.class);
{{/code}}

{{/code}}
{{code language="java"}}{code:java}
LOG.debug("EnteredmLogger.debug("Entered method getName");
{{/code}}{code}
Then, to enable logging at a specific level for your plugin, edit //webapps/xwiki/WEB-INF/classes/log4j.properties//webapps/xwiki/WEB-INF/classes/log4j.properties and add, for example:
{{code}}{code}
{{/code}}{code}
You'll then be able to follow your plugin's log messages by tailing your ##xwiki.log##xwiki.log file. Note that you'll need to restart the app server for changes to ##log4j.properties##log4j.properties to take effect.
And finally, **write*write a method to init the context**:context*.
{{code language="java"}}{code:java}
{{/code}}{code}
{{code language="java"}}{code:java}
private static Log LOGmLogger =
LogFactory.getLog(HelloWorldPlugin.class);LogFactory.getFactory().getInstance(com.xpn.xwiki.plugin.helloworld.HelloWorldPlugin.class);
{{/code}}{code}
= Write1.1 Build the API =API
Firstly, **class declaration**:*class declaration*:
{{code language="java"}}{code:java}
{{/code}}{code}
Then, **plugin*plugin field declaration**.declaration*. It will let our APIapi to call backend methods.
{{code language="java"}}{code:java}
{{/code}}{code}
Required constructor**Constructor overloading*
{{code language="java"}}{code:java}
super(context);
setPlugin(plugin);
{{/code}}{code}
Classic **plugin*plugin getter and setter**. These methods are not required at all, on the contrary, they should not be defined, unless they are really needed.setter*
{{code language="java"}}{code:java}
return (hasProgrammingRights() ? plugin : null);plugin;
// Uncomment for allowing unrestricted access to the plugin
// return plugin;
this.plugin = plugin;
{{/code}}{code}
Here is the key **API method**.*API method*. Here is the one that you will call from velocity. You can define any number of them and call your plugin backend from them.
{{code language="java"}}{code:java}
{{/code}}{code}
You can also have void methods:
{{code language="java"}}
public void updatePage() {
//...
}
{{/code}}
{{code language="java"}}{code:java}
public public HelloWorldPluginApi(HelloWorldPlugin plugin, XWikiContext context) {
super(context); super(context);
setPlugin(plugin); setPlugin(plugin);

~~} }~~
~~public~~ **public** HelloWorldPlugin getPlugin(){
~~return (hasProgrammingRights() ? plugin : null);~~ **return plugin;**
~~// Uncomment for allowing unrestricted access to the plugin~~ **}**
~~// return plugin;~~
~~}~~ **public void setPlugin(HelloWorldPlugin plugin) {**
~~public void setPlugin(HelloWorldPlugin plugin) {~~ **this.plugin = plugin;**
~~this.plugin = plugin;~~ **}**
~~}~~
~~public~~ **public** String hello() {
~~return~~ **return** "Hello World!";
~~} }~~
~~public void updatePage() {~~
~~//...~~
~~}~~
~~{{/code}}~~**{code}**
~~=~~**1.1** Integrate the plugin in your XWiki ~~installation =~~**installation**
First of all you need to ~~**copy~~***copy** your classes to the XWiki servlet ~~installation**.~~**installation*.** Don't forget to be consistent with your package tree. With a Linux Tomcat installation, ~~you'll need to follow these steps~~ ~~which~~ you **do this. You** should be able to reproduce **these steps** easily in your favourite operating ~~system:~~**system.**
~~{{code}}~~**{code}**
go to the tomcat installation ~~folder (or whatever container you are using)~~**folder**
$ cp myPluginsFolder/HelloWorldPlugin.class
~~classes/com/xpn/xwiki/plugin/helloworld~~**classes/com/xpn/xwiki/plugin/myWonderfulPackage**
$ cp myPluginsFolder/HelloWorldPluginAPI.class
~~classes/com/xpn/xwiki/plugin/helloworld~~**classes/com/xpn/xwiki/plugin/myWonderfulPackage**
~~{{/code}}~~**{code}**
Alternatively, you can jar up your classes (with the required directory structure) and place the jar in ~~//webapps/xwiki/WEB-INF/lib//.~~**webapps/xwiki/WEB-INF/lib.** This is a more agreeable way of distributing your plugin.
Finally you need to ~~**register~~***register** your ~~plugin**~~**plugin*** in the ~~##xwiki.cfg##~~**~~xwiki.cfg~~~~** file located in ~~//WEB-INF//:~~**~~WEB-INF~~~~**
~~{{code}}~~**{code}**
~~xwiki.plugins=com.xpn.xwiki.plugin.calendar.CalendarPlugin,\~~**xwiki.plugins=com.xpn.xwiki.plugin.calendar.CalendarPlugin, ..., com~~~~...,\~~**.xpn.xwiki.plugin.mail.MailPlugin,**
com.xpn.xwiki.plugin.helloworld.HelloWorldPlugin
~~com.xpn.xwiki.plugin.helloworld.HelloWorldPlugin~~**{code}**
~~{{/code}}~~
Don't forget to restart your ~~servlet container~~**tomcat server** after this. XWiki ~~has~~**have** to re-read the configuration file.
~~=~~**1.1** Use the ~~plugin =~~**plugin**
Here is the simplest part. Edit a page and ~~write: {{code}}My plugin says: "$xwiki.helloworld.hello()"{{/code}}.~~**write:**
~~It should be rendered like this: {{code}}My plugin says: "Hello World!"~~
~~{{/code}}.~~**{code}**
~~You can also call void methods specified in the API class :~~**My plugin says: "$xwiki.helloworld.hello()"**
~~{{code}}~~**{code}**
~~$xwiki.helloworld.updatePage()~~**It should be rendered like this**
~~The page has been updated.~~**{code}**

{{/code}}**My plugin says: "Hello World!"**
= Examples ={code}
Here are some examples of what you can do with plugins. You should actually check the [[API Guide>>DevGuide.APIGuide]], since it contains examples on how to use the XWiki API. The examples in the API Guide are written in Velocity, and are thus easily applicable to Java.
== Accessing pages, objects and object properties from pages ==
This is something you can do from Velocity as well, but when you need to perform complex treatments on your XWiki pages, you need to do it from a java plugin.
The class representing a document in the XWiki Java model is ##com.xpn.xwiki.doc.XWikiDocument##. The class representing an object in the XWiki Java model is ##com.xpn.xwiki.objects.BaseObject##.
If you need to access existing documents from your plugin, you use the XWiki class, ##com.xpn.xwiki.XWiki##, which has a getDocument() method. You can retrieve the current Xwiki instance by using the ##com.xpn.xwiki.XWikiContext## class, which has a getWiki() method.
The rule, in plugin programming, is to pass the current context as a ##com.xpn.xwiki.XWikiContext## function parameter, between the different methods of your plugin class. The plugin API class also has a context property pointing to the current context.
{{code}}
// You need the current context, which you always have in a plugin anyway
com.xpn.xwiki.doc.XWikiDocument doc = context.getDoc(); // current document;
com.xpn.xwiki.doc.XWikiDocument doc = context.getWiki().getDocument("theSpace.theDoc", context); // any document
com.xpn.xwiki.objects.BaseObject meta;
meta = doc.getObject("fooSpace.fooClass");
String docType = (String)meta.getStringValue("type"); //if the class of the object has a property named "type", which can accept a text value...
meta.set("type", "newValue", context);
{{/code}}
If you need to access the parent of an XWiki document, you should use the getDocument() method of the XWiki class, as seen in the example above, with, as parameter value, the parent's full name returned by the getParent() method of the XWikiDocument class.
{{code}}
com.xpn.xwiki.doc.XWikiDocument parentDocument = context.getWiki().getDocument(childDocument.getParent());
{{/code}}
You should not use ##XWikiDocument.getParentDoc## since it only returns a blank XWikiDocument object set with the same full name as the parent's full name.

XWiki.XWikiComments

- author:

xwiki:XWiki.mbirukov

- comment:

All these classes from this topic are deprecated now! Tell me please what do you advice for developers as new solutions?

- date:

2016-02-09 15:44:42.0

XWiki.XWikiComments

- author:
~~xwiki:XWiki.VincentMassol~~
- comment:
~~Check the first line on this doc. There's link to writing Components...~~
- date:
~~2016-02-09 15:49:51.0~~
- replyto:
~~0~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.mbirukov~~
- comment:
~~Thanks for your answer! But I have read this topic.~~
~~So! I cannot get access to XWiki sheets from Java code?~~
~~I mean... I would like to get access to XWikiContext class for getting properties from xwiki sheet. And now it isn't possible to get them without platform-old core.~~
- date:
~~2016-02-10 08:47:05.0~~
- replyto:
~~1~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.VincentMassol~~
- comment:
~~Hi, could you please send your questions to the [[xwiki users mailing list>>dev:Community.MailingLists]], that will be better to discuss? Also I'd like to talk to you about your Extension that you're trying to contribute, in order to help you. Thanks!~~
- date:
~~2016-02-10 21:50:59.0~~
- replyto:
~~2~~

- title: Creating Plugins ⇨
- parent: xwiki:Documentation.DevGuide.WebHome ⇨ DevGuide.BestPractices
- syntax: XWiki 2.0 ⇨ XWiki 1.0

Documentation.DevGuide.Tutorials

Custom Mapping

Documentation.DevGuide.Tutorials.CustomMapping.WebHome

On 2017/09/05, the page has been modified by Vincent Massol

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WebHome

~~xwiki.Documentation.DevGuide.Tutorials.WebHome~~

- author: XWiki.evalica ⇨ XWiki.VincentMassol

On 2017/09/05, the page has been created by Vincent Massol

> Custom ~~Mapping~~**Mappings** allows to map a XWiki Class (XClass) to its own database table (as opposed to XClasses that are not mapped and that use the standard, predefined, XWiki table scheme). Custom mapping can be useful for improving performances (when expecting a class to have a large number of instances for example), or for sharing external data ~~(sensitive~~**(enterprise** data, or other software data for example) with XWiki.
> ~~{{info}}You can also use the [[Custom Mapping Application>>extensions:Extension.Custom mapping tool]] extension to generate the ##hbm## file.{{/info}}~~
> <id name="id" ~~type="long" unsaved-value="undefined">~~**type="integer" unsaved-value="any"><column name="mam_id" not-null="true" /> <generator class="assigned" /></id>**
> ~~<column name="xwo_id" not-null="true" /> <generator class="assigned" /> </id>~~
> 1. Last, the XClass for which the mapping has been written should be set as containing a custom mapping. Unfortunately there's currently no way to set this using the XWiki Enterprise UI so you'll have to set it programmatically. The following Groovy snippet will do the trick (Remember that for Groovy code to be executed, the page that contains the code should be saved by a user having the programming right allowed on that document). Note that this could also be done in a Java ~~component.~~**component.(((**
> ~~((({{code}}~~
> ~~{{code language="java"}}~~
> ~~classDocumentName = "Mailing.MailingMemberClass";~~**import java.util.*;**
> classDoc = ~~xwiki.getDocument(classDocumentName).getDocument();~~**xwiki.getDocument("Mailing.MailingMemberClass")**
> ~~xml~~**classDoc2** = ~~classDoc.getxWikiClassXML();~~**classDoc.document**
> ~~if (xml == null || "".equals(xml)) {~~**println "Before: "**
> ~~println("The document [" + classDocumentName + "] doesn't seem to ";~~**println classDoc2.xWikiClass.customMapping**
> ~~+ ((classDoc.isNew()) ? "exist." : "contain a class."));~~**classDoc2.xWikiClass.setCustomMapping("internal")**
> ~~} else {~~**xwiki.xWiki.saveDocument(classDoc2, context.context)**
> ~~classDoc.getxWikiClass().setCustomMapping("internal");~~**classDoc = xwiki.getDocument("Mailing.MailingMemberClass")**
> ~~xcontext.getContext().getWiki().saveDocument(classDoc, xcontext.getContext());~~**println " After: "**
> ~~classDoc =~~
> ~~xwiki.getDocument(classDocumentName).getDocument();~~**println classDoc2.xWikiClass.customMapping**
> ~~if ("internal".equals(classDoc.getxWikiClass().getCustomMapping())) {~~
> ~~println("Success.");~~
> ~~} else {~~
> ~~println("Failed to alter the custom mapping field.");~~
> ~~}~~
> ~~}~~
> ~~{{info}}~~
> ~~Since XWiki 3.5.2, 4.1.4, and 4.2M3, copying a document containing a custom mapped class does not make the copied class custom mapped anymore. This has been introduced to avoid the newly created class to be unusable and inconsistent with the available mapping.~~

~~{{/info}}~~

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ DevGuide.WebHome
- author: XWiki.evalica ⇨ XWiki.VincentMassol

Documentation.DevGuide.Tutorials.FAQTutorial

Creating a FAQ application (AWM)

Documentation.DevGuide.Tutorials.FAQTutorial.FAQTutorialAWM

Between 2017/09/05 and 2017/09/05, the page has been modified 2 times, by 1 user: Vincent Massol

- title: FAQ Application Tutorial ⇨ Creating a FAQ application (AWM)
- parent: xwiki:Documentation.DevGuide.FAQTutorial.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.FAQTutorial.WebHome
- author: XWiki.evalica ⇨ XWiki.VincentMassol

On 2017/09/05, the page has been created by Vincent Massol

This tutorial will show you how to build a Frequently Asked Questions ~~(FAQ)~~ **(FAQs)** Application much like the one you can find on the [[FAQ page>>xwiki:FAQ.WebHome]]. This is a very simple application that is made using App Within Minutes ~~Application. You~~**Application, you** may frequently use this technique as the basis for several different kinds of applications.
~~{{warning}}~~
~~If you've already followed the [[Manual FAQ Tutorial>>xwiki:Documentation.DevGuide.FAQTutorial.FAQTutorialManual]] or installed the [[FAQ Application>>extensions:Extension.FAQ Application]] you'll need to do one of 2 things before you can take this tutorial:~~
~~* Delete the ##FAQ## space (if you've followed the Manual FAQ Tutorial) or uninstall the FAQ Application (if you've installed the FAQ Application)~~
~~* or simply follow the tutorial below but instead of using ##FAQ## as the application name, use another name such as ##MyFAQ##.~~
~~{{/warning}}~~
= Application ~~Overview =~~**Overview=**
= Creating the FAQ ~~Application =~~**Application=**
{{image reference='FAQ-Go1.png'/}}
{{image reference='FAQ-Go2.png'/}}
~~{{image reference="FAQ-Go1.png"/}}~~
~~{{image reference="FAQ-Go2.png"/}}~~
{{image ~~reference="FAQ-Go3.png"/}}~~**reference='FAQ-Go3.png'/}}**
{{image ~~reference="FAQ-Go4.png"/}}~~**reference='FAQ-Go4.png'/}}**
In the second step you define the application structure. Drag and drop a ~~"Long Text"~~**Long Text** field from ~~the~~ Field Palette. Then change the name from "Long Text" to "Answer". This will be the ~~"Answer"~~**Answer** field.
{{image ~~reference="FAQ-NEW1.png"/}}~~**reference='FAQ-NEW1.png'/}}**
In the last step you customize the way the home page of your application will look like. Click on the drop-down menu and select the "Creation date" item then click on the ~~"add"~~**add** button. This will add a "Creation date" column in your application live table.
{{image ~~reference="FAQ-NEW2.png"/}}~~**reference='FAQ-NEW2.png'/}}**
~~= Creating new FAQ entries =~~
= Creating new FAQ entries=
{{image ~~reference="FAQ-NEW3.png"/}}~~**reference='FAQ-NEW3.png'/}}**

Write your question and click on the ~~"add"~~ **add** button.
{{image ~~reference="FAQ-NEW4.png"/}}~~**reference='FAQ-NEW4.png'/}}**
~~Click on the "Save & View" button. Go back to the "FAQ.WebHome" page to see the list of existing questions.~~
~~= Answering~~**Click on "SAVE & VIEW" button. Go back** to ~~FAQ questions~~
=the "FAQ.WebHome" page to see the list of existing questions.
~~To answer to a question click on the "edit" button in the "Action" column from the live table, enter your answer in the "Answer" field and press the "Save & View" button.~~
~~{{image reference="FAQ-NEW5.png"/}}~~**= Answering to FAQ questions=**
To answer to a question click on the edit button in the "Action" column from the live table, enter your answer in the "Answer" field and press "SAVE & VIEW" button.
{{image reference='FAQ-NEW5.png'/}}

FAQ-Go1.png: 1.3 ⇨ ()

FAQ-Go2.png: 1.3 ⇨ ()

FAQ-Go3.png: 1.3 ⇨ ()

FAQ-Go4.png: 1.3 ⇨ ()

FAQ-NEW1.png: 1.3 ⇨ ()

FAQ-NEW2.png: 1.3 ⇨ ()

FAQ-NEW3.png: 1.3 ⇨ ()

FAQ-NEW4.png: 1.3 ⇨ ()

FAQ-NEW5.png: 1.3 ⇨ ()

- title: FAQ Application Tutorial ⇨
- parent: xwiki:Documentation.DevGuide.FAQTutorial.WebHome ⇨
- author: XWiki.evalica ⇨ XWiki.Sorin

Creating a FAQ Application (Manual)

Documentation.DevGuide.Tutorials.FAQTutorial.FAQTutorialManual

Between 2017/09/05 and 2017/09/05, the page has been modified 2 times, by 1 user: Vincent Massol

XWiki.XWikiComments
comment:

XWiki.XWikiComments
comment:

XWiki.XWikiComments
comment:

- title: Creating a FAQ Application ⇨ Creating a FAQ Application (Manual)
- parent: xwiki:Documentation.DevGuide.FAQTutorial.WebHome ⇨

parent: xwiki:Documentation.DevGuide.FAQTutorial.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.FAQTutorial.WebHome

author: XWiki.evalica ⇨ XWiki.VincentMassol

On 2017/09/05, the page has been created by Vincent Massol

~~{{box cssClass="floatinginfobox" title="**Contents**"}}~~**1 Building a FAQ Application**

~~{{toc/}}~~

~~{{/box}}~~

This tutorial will show you how to build a Frequently Asked Questions (FAQs) ~~[[Application>>platform:Features.Applications]]~~**Application** much like the one you can find on the ~~[[FAQ page>>xwiki:FAQ.WebHome]].~~**[FAQ page>FAQ].** This is a very simple application that makes use of XWiki's ~~[[classes,~~**classes,** properties, and ~~objects>>platform:DevGuide.DataModel||anchor="HXWikiClasses2CObjects2CandProperties"]].~~**objects.** It also uses a technique that you may frequently use as the basis for several different kinds of ~~applications.~~**applications. In addition to the data model, this tutorial will introduce the concept of Authoring Templates and Page Design Sheets. Completing this tutorial is a recommended prerequisite for anyone who wants to build custom applications on the XWiki engine.**

~~= Prerequisites for following the tutorial =~~**1.1 Application Overview**

~~{{warning}}~~**The FAQ application allows users to post a question by entering the question into a simple form field and then submitting the form. The question then appears in a list along with all other questions that have been previously posted. Users can click on the questions to view both the question and answer in a consistently styled view. If the question has not yet been answered, any user can post an answer to the question by editing the page. In edit mode, the page will display a web form that is always consistent for every FAQ page.**

~~If you've already followed the [[AWM FAQ Tutorial>>xwiki:Documentation.DevGuide.FAQTutorial.FAQTutorialAWM]] or installed the [[FAQ Application>>extensions:Extension.FAQ Application]] you'll need to do one of 2 things before you can take this tutorial:~~

~~* Delete the ##FAQ## space (if you've followed the AWM FAQ Tutorial) or uninstall the FAQ Application (if you've installed the FAQ Application)~~

~~* or simply follow the tutorial below but replace the space ##FAQ## with another value everywhere it's used. For example use ##MyFAQ## as the space name instead.~~

~~{{/warning}}~~

~~You should have [[installed XWiki>>platform:AdminGuide.Installation]] and have~~**Let us begin by taking** a ~~[[basic understanding of how~~**look at what we are going** to ~~use it>>platform:Features.WebHome]].~~**build. The system has the following views. Click any view link below to preview a screenshot. (Note: Firefox users can middle click or click down on the mouse scroll-wheel to open the links in a new tab.)**

~~All through this tutorial you should refer to the [[XWiki Data Model>>platform:DevGuide.DataModel]] for information on XWiki's data model. You might also use the [[XWiki Scripting Guide>>platform:DevGuide.Scripting]] to get you started with scripting in XWiki and manipulating XWiki objects. In addition, this tutorial will introduce the concepts of Authoring Templates and Page Design Sheets, patterns that you will find particularly useful in creating XWiki applications. Completing this tutorial is a recommended prerequisite for anyone who wants to build custom applications on the XWiki engine.~~*** The FAQs Summary View**

*** A Question and Answer Page in Display Mode**

~~* A Question and Answer Page in Display Mode~~
* A Question and Answer Page in Edit Mode
~~{{warning}}~~1.1 Authoring Templates and Page Design Sheets
~~Make sure that your user is an [[Advanced user>>platform:Features.PageEditing||anchor="HAdvancedMode"]] before following this tutorial since you'll need for example to use the wiki editor (##Wiki > Edit## menu).~~
~~{{/warning}}~~
~~= Application Overview =~~An Authoring Template is a template for creating documents of a specific type. Unlike a regular content page in edit mode with one field for freeform editing, an Authoring Template presents a custom set of form fields. Each form field can have different types of form elements. These form elements are defined by the properties of a class.
~~The FAQ application allows users to post~~In object oriented programming, remember that a ~~question by entering~~~~class~~ is a template for an object. Using the ~~question into~~analogy of a ~~simple form field~~cookie cutter, the ~~class~~ is the ~~cookie cutter~~ and ~~then submitting~~ the ~~form. The question then appears in~~~~objects~~ are the actual ~~cookies~~. An Authoring Template provides one way to represent a ~~list along with all other questions~~class visually so that ~~have been previously posted. Users~~users can ~~click on the questions~~fill out a form to ~~view both the question and answer~~set unique properties (values in ~~a consistently styled view. If~~form fields). When the ~~question has not yet been answered, any~~ user ~~can post an answer to~~submits the ~~question by editing the page. In edit mode, the page will display~~form, they are creating a ~~web form that is~~unique ~~object~~ of the ~~same for every FAQ page.~~~~class~~ type.
~~Let us begin by taking~~~~In this sample application,~~ a ~~look at what we are going to build. The~~ new ~~application will have~~document is created and the ~~following views:~~object instance is automatically added to the new document. So, in truth, objects in XWiki are not really documents; they are added to a document. Once an object is added to a document, it's properties can easily be accessed by scripting in that document's content.~~
~~* The FAQ [[Class>>xwiki:FAQ.WhatIsAClass]]~~The Page Design Sheet is like a style sheet that defines each document will look when it is rendered. Even though the unique object instances will have different values for their properties, they will always have a consistent presentation display because they will be rendered through the Page Design Sheet.
~~{{image reference="FAQsSummary.png"/}}~~
~~* An FAQ entry in View mode~~
~~{{image reference="FAQSheetView.PNG"/}}~~
~~* An FAQ entry in Edit mode~~
~~{{image reference="FAQSheetEdit.PNG"/}}~~
~~= Authoring Templates and Page Design Sheets =~~1.1 Get Set with the Class Editor Wizard
~~An Authoring Template is~~Five pages have been developed which collectively make up a ~~template for creating documents of a specific type. Unlike a regular content page~~Class Editor Wizard application that can assist you in ~~edit mode with one field for freeform editing, an Authoring Template presents~~this process. Your first task is to see if you have these pages in your XWiki already. If so, you'll make a ~~custom set of form fields for creating a document with specific type of data. These form elements are defined by~~link to the ~~properties~~wizard app so that you can always access it easily from an administrative page. If you do not have all four pages of ~~a class.~~the wizard, you'll need to make the missing

pages using code provided by this tutorial.
~~In object oriented programming, remember that~~* Enter a ~~//class// is a template~~search query for ~~an object. Using~~ the ~~analogy of~~keyword "XWikiClasses". This should return a ~~cookie cutter,~~document called XWikiClasses in the ~~//class//~~XWiki space (i.e. XWiki.XWikiClasses). This is the ~~//cookie cutter// and~~first of the ~~//objects// are the actual //cookies//.~~ ~~An Authoring Template provides one way to represent a class visually so~~five pages that ~~users can fill out a form to set unique properties (values in form fields). When~~make up the ~~user submits~~wizard and it is the ~~form, they are creating a unique //object// of~~main entry page to the ~~//class// type.~~wizard application.
* Take a moment to bookmark this page or, even better, create a link to it from an administrative page so you can access it more easily in the future.
* Navigate to the XWiki.XWikiClasses page.
~~Precisely, an Authoring Template is a prototype document used~~Now, you just need to ~~create other specific instances of documents of~~verify that you have the ~~same type, along with a method of exposing the creation process to the user: the properties edit form. Remember~~other four pages that ~~a XWiki Document can contain objects and this is~~make up the ~~case of an authoring template: it is a XWiki Document with an empty object of a specific class, which is duplicated to create more~~wizard app and ~~more documents based on that model, using~~create them if you do not. If you have Administrative rights, you should see four links under the ~~data inserted by the user in the editing form as specific values for the particular object instance in the current copy.~~"Models" heading:
~~The Page Design Sheet is like a style sheet that defines what each document will look like when rendered. Even though the unique object instances will have different values for their properties, they will always have a consistent presentation display because they will be rendered through the Page Design Sheet. The XWiki API available in scripting languages provides a mechanism that will help us use the same sheet for both editing and view modes. We will see how we can achieve this once we get to the FAQ Design Sheet section.~~* ClassSheet
* ClassTemplate
* ClassItemSheet
* ClassItemTemplate
~~= Go~~If any of the links are showing up with question marks, you do not have the page in your XWiki database. Copy the page code from the links below and paste it into the respective pages in your XWiki instance to ensure that you have all the ~~Class Editor Wizard =~~parts of the wizard in tact. Once you have all of these pages in place, you are ready to start building your FAQs application.
~~Five pages, which collectively make up a [[XClass application>>extensions:Extension.XClass Application]] (a.k.a Class Wizard or Class Editor Wizard), have been developed to assist you in this process. Those page are technical pages hidden by default. To navigate to~~* Get the ~~wizard home page, go to your profile page, select the Preferences tab, edit the~~[XWikiClasses>http://www.xwiki.com/xwiki/bin/view/XWiki/XWikiClasses?xpage=code] page ~~and choose to view Hidden Documents. Enter a search query for the keyword "XWikiClasses". This should return a document called ##XWikiClasses## in the ##XWiki## space (i.e. ##XWiki.XWikiClasses##). This is the homepage of the class wizard creator: you are now ready to start building your FAQs application.~~ code.
* Get the [ClassSheet>http://www.xwiki.com/xwiki/bin/view/XWiki/ClassSheet?xpage=code] page code

*** Get the [ClassTemplate>http://www.xwiki.com/xwiki/bin/view/XWiki/ClassTemplate?xpage=code] page code.**
*** Get the [ClassItemSheet>http://www.xwiki.com/xwiki/bin/view/XWiki/ClassItemSheet?xpage=code] page code.**
*** Get the [ClassItemTemplate>http://www.xwiki.com/xwiki/bin/view/XWiki/ClassItemTemplate?xpage=code] page code.**
=1.1 Create the FAQ ~~Class =~~**Class**
* On the Class Editor wizard entry page (XWiki.XWikiClasses), under the heading "Create a new ~~data type", in the "Title" field,~~**Class",** enter ~~##FAQ## as~~ the ~~name of the page to create:(((~~**following web space and class name:**
~~{{image reference="CreateANewClass1.png"/}}~~**** Web: FAQs**
~~)))~~**** Class: FAQ**
* ~~As you can see in~~**Click** the ~~Breadcrumb below the new~~**"Create this Class" button. You should then see a code** page ~~will be created at location ##XWiki > FAQ##. In practice the Class Wizard will automatically prefix the page name~~ with ~~##Class## (you could also enter ##FAQClass## as~~ the ~~page name directly).~~**following code:**
~~* Now it would be nice to have it created in a new location such as ##FAQ > FAQ Class##. Since the ##FAQ## parent doesn't exist we cannot use the Tree picker button. Thus click the Pencil button as shown in the following image and replace ##XWiki## by ##FAQ##.(((~~
~~{{image reference="CreateANewClass2.png"/}}~~
~~)))~~
~~* In technical terms you're creating a page named ##FAQClass## (with a title of "FAQ Class") located in a space also called ##FAQ## and thus the technical reference is ##FAQ.FAQClass##.~~
~~* Click the "Create this Class" button. You should then see a page with the following content:(((~~
~~{{code language="none"}}~~
~~{{velocity}}~~
~~## Replace the default space with the space where you want your documents to be created.~~
~~## Replace the default parent with the one of your choice and save the document.~~
~~##~~
~~#set($defaultSpace = $doc.space)~~
~~#set($defaultParent = $doc.fullName)~~
~~{{/velocity}}~~
~~{{/code}}~~
~~)))~~
~~In the code, change the word "$doc.space" with the name of the space where you want you FAQ pages to be created as the commented instructions in the page code suggest.~~**{code:xml}**
~~The line of code should look like this:~~**<!-- replace Main with the Space where you want your documents to be created -->**
<!-- replace the default parent with the one of your choice -->
<!-- Save this template using the ’Save’ button -->
#\set($\class = $\doc.name.substring(0,$\doc.name.indexOf("Class")))
#\set($\defaultparent = "XWiki.$\{class}Class")
#\set($\defaultweb = "Main")
#\include\Form("XWiki.ClassSheet")
{code}
~~{{code language="none"}}~~*** In the code, change the word "Main" with the name of the space where you want you FAQ pages to be created as**

name of the space where you want you FAQ pages to be created as the commented instructions in the page code suggest. Replace the word "Main" with the word "FAQs". The line of code should now look like this:
#set($defaultSpace = 'FAQ')
{{/code}}
You can also change the default parent of the new FAQ documents that are going to be created. To do so, replace the "$defaultParent" variable with the name of your document.{code:xml}
The line of code should look like this:#\set($\defaultweb = "FAQs")
{code}
{{code language="none"}}* Click the "Save" button. The class is now created and you should the be looking at a page titled "Class: FAQClass" that looks like this:
#set($defaultParent = 'FAQ.WebHome')
{{/code}}
Click the "Save & View" button. The class is now created and you should be looking at a page titled "Class: FAQ" that looks like this:
{image:FAQClassDoc.gif}
{{image reference="FAQClass1.PNG"/}}1.1 Add Properties to the Class = Add PropertiesUnder the page title, you should see the words "The first thing to do is to <u>Edit the Class =Class</u> to add properties to it." So, let's just follow those instructions!
Under* Click on the page title, you should see'Edit the words "The class does not have any properties yet. You can use the //__class editor__// to define them." Let's just follow those instructions!Class' link; a blank page will be displayed.
* Note that the link trail in the header is something like "XWiki.XWikiClasses > FAQs.FAQClass". Even though the page is blank, this shows you are indeed on the class page.
* Click on the 'class editor' linkIn our document, we'll store both a ~~question~~ and an ~~answer~~. So we need a property for each.
* Note that the link trail in the header is something like "Wiki Home / FAQ / FAQ Class". This shows you that you are indeed on the FAQ class page in Class edit mode.
In our document, we'll store both a //question// and an //answer//. So we need to create a property for each* Enter the text ~~question~~ in the Property Name field of them.the Class Editor panel.
* Choose a TextArea type for the property and then click the 'Add Property' button. The TextArea will ultimately give us a multi-line text field in our authoring template.
* Configure this property with the following values:
* Enter the text //question// in the "name" field{image:QuestionProperty.gif}
* Choose a TextArea type for the property and then click on "Add". The TextArea will ultimately give us a multi-line text field in our authoring template.(((
{{image reference="AddQuestionProperty.PNG"/}}
)))
* Click on the "+" icon to expand the options for the newly created property
* Change the value of the "Pretty Name" field to "Question"(capital Q):(((
{{image reference="QuestionProperty.PNG"/}}
)))
* Now create another property called //answer// the same way that you did for the previous "question" property (choose TextArea for the property type)
* Expand it from the property list and change the value of the "Pretty Name" field to "Answer"
* When you are done adding and configuring the properties, click the "Save

& View" button
= Create* Now add a property called ~~answer~~ in the Page Design Sheet =same way that you did for the 'question' property (choosing TextArea for the property type).
* Configure this property with the following values:
* After the previous step you are now on the FAQClass page which should look like this:(((**{image:AnswerProperty.gif}**
{{image reference="FAQClass2.PNG"/}}
)))
* Click the first button ("Create the document sheet") to create the document sheet (the Page Design Sheet). This sheet determines how your page's objects will be rendered to the user. The document is automatically created.
* You should see a warning message with the text "The sheet is not bound to the class so it won't be applied automatically when a page that has an object of this class is displayed". Click the "Bind the sheet to the class" link that appears after the text. **What this does is important**:
** It adds an object of type ##XWiki.ClassSheetBinding## to the ##FAQ.FAQClass" document. Basically it ties the FAQ Class to the Sheet.##
** **It's because of this object that users will be sent to form edition mode when editing FAQ entries**
* Now click on "View the sheet document". This takes you to the ##FAQ.FAQSheet## page which you can edit in wiki mode and see its default content:(((
{{code language="none"}}
{{velocity}}
You can modify this page to customize the presentation of your object.
At first you should keep the default presentation and just save the document.
#set($class = $doc.getObject('FAQ.FAQClass').xWikiClass)* When you are done adding and configuring the properties, click the 'Save Class' button.
#foreach($prop in $class.properties)
; $prop.prettyName
: $doc.display($prop.getName())
#end
{{/velocity}}
{{/code}}
)))Let's take a moment now and analyze this code:(((
* The first line retrieves the ##FAQ.FAQClass## from the wiki
* Then we iterate through all its properties and display their values for the current document in a definition list.
As we mentioned, XWiki provides a mechanism that helps us create sheets used for both View and Edit mode.1.1 Create the Page Design Sheet
This is the display function used in the line:
{{code language="none"}}
: $doc.display($prop.getName())
{{/code}}
It detects* Search for the current mode (Edit or View)keywords 'FAQs.FAQClass' and displaysclick on the property referenced by its argument as the mode dictates:document to open it in XWiki.
* The title of the page should be "Class: FAQClass" and it should look like this:
* In view mode it will display the value of the property**{image:FAQClassSheet.gif}**
* In edit mode it will display a form field that will allow the user to edit it

This way we can use a single* Click the first button to create the document sheet (the Page Design SheetSheet). This sheet determines how your page objects will be rendered to both display and edit our FAQ entries. See the [[XWiki API reference>>platform:DevGuide.API]] and [[XWiki Scripting>>platform:DevGuide.Scripting]] pagesuser in ~~read~~ mode. You should see the following page code for more details about this.the document sheet:
* Click "Save & View"{code:xml}
)))#\#\ you can modify this page to customize the presentation of your object
#\#\ at first you should keep the default presentation
#\#\ save this template using the save button at the top left
= Create the Authoring Template =1\ Document $\doc.name
* Navigate back#\#\ Change class name to the ##FAQ.FAQClass## document (you can use the arrows in the breadcrumb to do so). The document should look like this:(((your class name
{{image reference="FAQClass3.PNG"/}}#\set($\class = $\doc.getObject("XWiki.MyClass").xWikiClass)
)))
* Click on the "Create the document template" button. The Authoring Template will be automatically created.
Note that earlier, we changed the space name preceding the page name because we wanted all of our FAQ pages to reside in a space named FAQ. Remember that all our documents will be copies of the Authoring Template used as a prototype so the content will be copied in all our FAQs documents and will execute the Design Sheet code in the context of the current document. See the [[dedicated page>>platform:DevGuide.IncludeInVelocity]] for more information regarding this technique.<table border="1" cellspacing="0" cellpadding="2">
#\foreach($prop in $class.properties)
<tr><td> *$\{prop.prettyName}* </td>
<td>$\doc.display($\prop.getName())</td>
</tr>
#\end
</table>
{code}
Now we need to associate* Change the prototype object with this documentclass name ~~XWiki.MyClass~~ in the code to turn it into a true authoring template:~~FAQs.FAQClass~~ so that the line looks like this:
* If you're on the template page, navigate back to the ##FAQ.FAQClass## document.{code:xml}
* At the bottom of the page, look for the following warning message: "The template does not contain an object of type FAQClass. Add a FAQ object#\#\ Change class name to the template »."your class name
* Click on "Add a FAQ object to the template »":(((#\set($class = $doc.getObject("FAQs.FAQClass").xWikiClass)
{{image reference="FAQObject.png"/}}{code}
)))
Next, we want to remove the title for the newly created template:* Click 'Save'.
* Navigate to1.1 Create the ##FAQ.FAQTemplate## document (you can click on the "View the template page (FAQ / FAQ Template)" link for doing that for example.Authoring Template
* Edit this document in Wiki mode
* Inside the Title field you have "FAQ Template" written -> delete this text
* Save & View

~~Save & View~~
~~This step is needed so that all of our future entries don't have "FAQ Template" as their title.~~*** Search for the keywords 'FAQs.FAQClass' and click on the document to open it in XWiki again. The document should look like this:**
~~Congratulations: you just created an Authoring Template! You're almost done now.~~**{image:PageDesignSheetCreated.gif}**
~~{{image reference="FAQClass4.PNG"/}}~~*** Notice that now, there is a link for the FAQClassSheet in place of the button that was previously there.**
*** Click on the 'Create the document template' button. The template will be created with the following page code:**
~~= Create a home page for the FAQ application =~~**{code:xml}**
#\#\ replace MyName with the real class name
#\#\ save this template using the save button at the top left
#\include\Form("XWiki.MyNameClassSheet")
{code}
~~You want your users to be able to see a list~~*** As the comments instruct, change the words ~~MyName~~ with ~~FAQ~~ so that instead** of ~~all existing questions and answers and to add new questions. The best way to do~~**having ~~XWiki.MyNameClassSheet~~, you will have ~~FAQs.FAQClassSheet~~. In** this ~~is to put~~**case, we changed** the ~~FAQ application in its own~~ space ~~and~~**preceding the page name also because we want all of our FAQ page objects** to ~~use that space's homepage to display existing questions.~~**reside in a unique wiki web space.**
*** Click 'Save' to save the page object.**
*** Now the saved page object (the authoring template) should show the page title "Document FAQClassTemplate". Now, we need to associate the class object with this page object to turn it into a true authoring template.**
*** Click 'More Actions > Edit Objects'.**
*** In the Object Editor panel, select FAQs.FAQClass from the drop-down and then click 'Add Object from this Class'. The ~~Question~~ and ~~Answer~~ form fields appear on the page.**
*** Click 'Save Objects'. Congratulations; You just created an Authoring Template!**
*** Click the FAQs.FAQClass link in the header breadcrumb trail. As you can see, you are almost done!**
~~Thus we now need to create the ##FAQ.WebHome## page~~**{image:MakingProgress.gif}**
~~* Click on "FAQ" in~~**1.1 Test** the ~~breadcrumb to navigate to ##FAQ.WebHome## and notice that the page doesn't exist.~~**Application**
~~* Edit it in wiki mode~~
~~* Type in the title "FAQs"~~
~~== Displaying existing FAQ entries ==~~**Now let's just create a new document in our application to test it out.**
~~You have 2 options when it comes to displaying existing FAQ entries:~~*** Enter a document title in the ~~Document~~ field and click ~~Create~~. For example, enter ~~'What is the meaning of life'~~. Don't use a question mark in the document name.**
*** You can then enter your question in longer form using the ~~question~~ field on the template, like this:**
~~1. Use the livetable component~~**{image:FirstQuestion.gif}**
~~1. Write custom code in order to display them~~
~~=== Using~~*** Click ~~Save~~ and then you will see** the ~~Livetable component ===~~**newly created document, like this:**
~~In this section, we will show how to use the [[Livetable Macro>>extensions:Extension.Livetable Macro]] to display the existing~~

questions and answers.{image:FirstQuestion2.gif}
The livetable component will give users the ability* Search for 'FAQ' to easily listfind the FAQClass document again and filter through existingclick to open it. You can now see that the FAQ documents. The macro is made of 3 parts:documents you are creating will build in a list on the main entry page to the application, like this:
* The list of columns: for each entry, we will display the question, the date when it was created and a special column that lets users quickly modify entries{image:Final.gif}
* The properties of each column: for each column, we will define how it should be displayed, whether it should link to the entry and whether it should be filterable
* The livetable options: those are options related to the display of the livetable (in this case we will to display a tag cloud and 10 rows by default)
Here is1.1 Displaying the resulting code:FAQ Questions as summary {{code language="none"}}If you want to display the FAQ question instead of the FAQ document name in the summary area, you can proceed as follow:
{{velocity}}* Edit the content of the FAQs.FAQClass page
#set($columns = ["question", "doc.creationDate", "_actions"])* Replace the #\includeForm("XWiki.ClassSheet") with the content of the XWiki.ClassSheet (go to the XWiki.ClassSheet, edit the content and past it instead of the includeForm command
#set($columnsProperties = {* After the #\foreach($\item in $\xwiki.searchDocuments($\sql)) command add the following line
 #\set ($\faq=$\xwiki.getDocument($\{item})) and replace the {code: xml}[\$\{item}]{code} with {code: xml}[\$\{faq.display("question")}>$\{item}]{code}
"question" : { "type" : "text", "link" : "view", "html" : "true", "sortable":true },
"_actions" : {"actions": ["edit","delete"]}
})
#set($options = {
"className":"FAQ.FAQClass",
"translationPrefix" : "faq.",
"tagCloud" : true,
"rowCount": 10
})
#livetable("faq" $columns $columnsProperties $options)
{{/velocity}}
{{/code}}
* Copy this code and paste it as Wiki content (inside ##FAQ.WebHome##)1.1 Finally, link to the FAQs.FAQClass page
* Click "Save and View"
* New FAQ entries will now be displayed on the page once you create them
The ##FAQ.WebHome##We don't want our users to have to search for this page should look similarusing the search engine, so finally pick a spot on your Wiki and make a user-friendly link to this:FAQs.FAQClass. For example:
{{image reference="FAQsLivetable.png"/}}{code:xml}
*\ [\Frequently Asked Questions (FAQs) > FAQs.FAQClass]
{code}
Notice how there are some translation keys displayed inside the livetable. Let's create a translations document and change those keys to actual text:1.1 Conclusion
* Create a new page insideThis tutorial has shown how to use the FAQ space called Translations, i.e.Class Wizard app and it has introduced the concepts of classes, objects, properties, authoring templates, and

page design sheets. You may also have learned a little bit about Velocity scripting in documents. You can use these basic concepts to build custom applications at ~~##FAQ.Translations## (using~~ the ~~"+" button)~~**document or presentation layer of XWiki without having to compile or deploy code.**
~~* Edit it in Wiki mode and paste this content inside:(((~~
~~{{code language="none"}}~~
~~faq.question=Question~~
~~faq.doc.creationDate=Creation Date~~
~~faq._actions=Actions~~
~~{{/code}}~~
~~)))~~
~~* Click "Save & View"~~
~~* Edit it again in Object mode and add a new ##XWiki.TranslationDocumentClass## object to mark the page as a page containing translations. Make sure to select the right scope. Using "Global" means the translations would be visible to all wikis (and it requires having Programming Rights!) and "Wiki" means it's visible to the current wiki only. Choose "Wiki" as the scope.~~
~~* Click "Save & View" again~~
~~Now the ##FAQ.WebHome## page should look like this (notice the translations instead of the keys):~~
~~{{image reference="FAQsLivetableWithTranslations.png"/}}~~
~~=== Using custom code ===~~
~~You will need to write the following code:~~
~~* A XWQL query that will find all your FAQ documents. Check the [[Query API documentation>>extensions:Extension.Query Module]] to know more about it.~~
~~** The XWQL query looks for all documents that have a ##FAQ.FAQClass## object other than the template~~
~~** If no document has been created yet, a warning message is displayed~~
~~* A piece of velocity code to display all those documents~~
~~** The velocity code loops in that list~~
~~** For each item, the full document is loaded in memory so that values can be retrieved from it~~
~~** For each document, the question is retrieved and displayed as a link towards the FAQ entry~~
~~Here is the resulting code:~~
~~{{code language="none"}}~~
~~= Existing FAQ entries =~~
~~{{velocity}}~~
~~#set($xwql = "from doc.object(FAQ.FAQClass) as faq where doc.fullName <> 'FAQ.FAQTemplate'")~~
~~#set($results = $services.query.xwql($xwql).execute())~~
~~#if($results.empty)~~
~~No FAQ has been created yet!~~
~~#else~~
~~#foreach ($item in $results)~~
~~#set($faq = $xwiki.getDocument($item))~~
~~* [[${faq.display("question").replace("<p>", "").replace("</p>", "")}>>${item}]]~~
~~#end~~
~~#end~~
~~{{/velocity}}~~
~~{{/code}}~~
~~* Copy this code and paste it as Wiki content inside ##FAQ.WebHome##~~
~~* Click "Save and View"~~
~~* New FAQ entries will now be displayed on the page once you create them~~
~~The ##FAQ.WebHome## page should look similar to this:~~

The ##FAQ.WebHome## page should look similar to this:
{{image reference="FAQsCustomCode.png"/}}
== Creating new FAQ entries ==
There are 2 ways for you to let your users create new FAQ entries:
1. Declare the FAQ as a template
1. Add a custom creation form
=== Using a Template ===
You will have to define a template provider [[as explained on this page>>extensions:Extension.Administration Application||anchor="HCreatethetemplateprovider"]]
Go to your wiki's administration interface, in the "Templates" section (Administration -> Content -> Templates). Create a new template provider in the
##FAQ## space and name it ##FAQTemplateProvider##
You can then use the following values:
* Provider name: ##FAQ Template Provider##
* Template name: ##New FAQ entry##
* Template to use: ##FAQ.FAQTemplate##
If you'd like, you can restrict FAQ entries so that they're created only in the ##FAQ## space. Once you're done, click "Save & View". Your template is now ready to be used! Users who click on the "Add >> Page" button will now have the option to create a new page using the FAQ template.
=== Custom creation form ===
If you choose this option, you will need to write some code to let your users create new FAQ entries. To do this, you will have to create a form in which the user can enter the name of the questions she wants to create. Once typed in, the form calls the same page to trigger the new document creation based on the parameters entered by the user:
* The first part of the code checks whether the page has a parameter. If so:
** The name of the document that will be created is computed
** A check is done to verify the document doesn't exist yet
** If everything's ok, the user is sent to the new document in inline edition mode
* The second part of the code is the actual FAQ creation form
** It builds the name of the document to create it in the ##FAQ## space
** It sets the document parent as being the current document
** It defines the template to use to create the new document
{{code language="none"}}
{{velocity}}
#if("$!request.docName" != '')
Request for creating a new instance
#set($docName = ${request.docName})
#set($targetDocName = "${request.spaceName}.${docName}")
#if(!$xwiki.exists($targetDocName) && $xwiki.hasAccessLevel('edit', $xcontext.user, $targetDocName))
$response.sendRedirect($xwiki.getURL($targetDocName, 'inline', "template=${escapetool.url($request.template)}&parent=${escapetool.url($request.parent)}"))
Stop processing, since we already sent a redirect.
#stop
#end
#end
= Add a new question =
#if("$!targetDocName" != '' && $xwiki.exists($targetDocName))
{{warning}}The target document already exists. Please choose a different name, or [[view the existing document>>$targetDocName]]{{/warning}}
#elseif("$!targetDocName" != '')
{{warning}}You don't have permission to create that document{{/warning}}

#end
{{html}}
<form action="" id="newdoc" method="post">

<input type="hidden" name="parent" value="${doc.fullName}"/>
<input type="hidden" name="template" value="FAQ.FAQTemplate"/>
<input type="hidden" name="sheet" value="1"/>
<input type="hidden" name="spaceName" value="FAQ"/>
Document: <input type="text" name="docName" value="Enter your question here" class="withTip" size="50"/>
<input type="submit" value="Create this FAQ" class="button"/>

</form>
{{/html}}
{{/velocity}}
{{/code}}
* Copy this code and paste it as Wiki content inside ##FAQ.WebHome##, below the code that's already there
* Click "Save and View"
* A form to create new FAQ entries is now available on the page:(((
{{image reference="FAQsWithForm.png"/}}
)))
= Test the Application =
Now let's just create a new document in our application to test it out.
If you previously chose to use a "Custom creation form" for creating new FAQ entries, follow these steps:
* Go to ##FAQ.WebHome##
* Below the "Add a new question" header, enter a question (which will also be used as the document title) in the //Document// field
* Click //Create this FAQ//
* You can then enter your question in longer form using the //Question// field on the template, like this:(((
{{image reference="FAQSheetEdit.PNG"/}}
)))
* Click //Save & View// and then you will see the newly created document, like this:(((
{{image reference="FAQSheetView.PNG"/}}
)))
* Go back to the ##FAQ.WebHome## page (you can use the breadcrumbs) to see the list of existing questions(((
{{image reference="FAQsWithEntry.png"/}}
)))
= Conclusion =
This tutorial has taught you how to use the Class Wizard app and it has detailed the concepts of classes, objects and properties and introduced the authoring templates and page design sheets.
You may also have learned a little bit about Velocity scripting in documents.
You can use these basic concepts to build custom applications at the document or presentation layer of XWiki without having to compile or deploy code.

AddObject.PNG: 1.2 ⇨ ()

AddQuestionProperty.PNG: 1.3 ⇨ ()

ClassWizard.xar: 1.3 ⇨ ()

CreateANewClass1.png: 1.1 ⇨ ()

CreateANewClass2.png: 1.1 ⇨ ()

FAQClass1.PNG: 1.3 ⇨ ()

FAQClass1.png: 1.3 ⇨ ()

FAQClass2.PNG: 1.3 ⇨ ()

FAQClass2.png: 1.3 ⇨ ()

FAQClass3.PNG: 1.3 ⇨ ()

FAQClass3.png: 1.3 ⇨ ()

FAQClass4.PNG: 1.5 ⇨ ()

FAQClass4.png: 1.3 ⇨ ()

FAQClass5.PNG: 1.4 ⇨ ()

FAQObject.png: 1.1 ⇨ ()

FAQSheetEdit.PNG: 1.3 ⇨ ()

FAQSheetEdit.png: 1.3 ⇨ ()

FAQSheetView.PNG: 1.3 ⇨ ()

FAQSheetView.png: 1.3 ⇨ ()

FAQsCustomCode.png: 1.1 ⇨ ()

FAQsLivetable.png: 1.1 ⇨ ()

FAQsLivetableWithTranslations.png: 1.1 ⇨ ()

FAQsSummary.png: 1.4 ⇨ ()

FAQsWithEntry.png: 1.1 ⇨ ()

FAQsWithForm.png: 1.1 ⇨ ()

QuestionProperty.PNG: 1.3 ⇨ ()

QuestionProperty.png: 1.3 ⇨ ()

WebHome.png: 1.1 ⇨ ()

Final.gif: () ⇨ 1.2

XWiki.XWikiComments
comment:

XWiki.XWikiComments
comment:
~~Following line of code will generate wrong links if~~**If you go back and add properties to** a ~~faq entry's question field contain character ">".~~ **class, when editing already existing pages you might be confronted with a lack of inputs for the new fields. I used the Edit Objects command to get around this.**
~~{{code language="none"}}[${faq.display("question")}>${item}]{{/code}}~~
~~This happens because first > character between sqaure brackets [] is treated specially. This can happen if you format a question element. For example if you enter "I am **confused** " in question the resulting text would be something like "I am {{html clean="false" wiki="true"}}~~
~~confused{{/html}}" which contains html tags including the character >.~~
date:
~~2007-02-05 07:17:49.0~~**2006-01-16 06:52:38.0**
author:
~~XWiki.TahirAkhtar~~**XWiki.starwed**

XWiki.XWikiComments
- author:
~~xwiki:XWiki.tapaya~~
- comment:
~~This Tutorial does currently (XWiki 1.9) not work as expected, which is due to the transition from syntax v1.0 to v2.0.~~
~~You can still get it to work when you follow (and maybe extrapolate) the hints Vincent and others give here:~~
~~http://xwiki.markmail.org/message/kl7vi6p2dlc3ouwr?q=FAQTutorial~~
- date:
~~2009-07-21 11:01:55.0~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.MarcLijour~~
- comment:
~~To make the tutorial work in XWiki 1.9.1, use XWiki syntax 2.0 by adding the velocity (around the Velocity code) and html (around the includeForm macro) tags.~~
~~See https://jira.xwiki.org/jira/browse/XWIKI-4032~~
- date:
~~2009-07-22 05:16:37.0~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.GuillaumeLerouge~~
- comment:
~~Hi Silvia, thanks for the update. Could you please also update the code section to reflect the fact that they are now created using XWiki 2.0~~

Syntax?

- date:
2009-09-28 14:42:44.0

XWiki.XWikiComments

- author:
xwiki:XWiki.YangNingYuan

- comment:
Hello xwikier. I downloaded xwiki Enterprise Edition, version 2.1.1. Where installed in tomcat. Using a mysql database. The default language of Chinese or English. The operating system is WIN XP. Because I live in Shanghai, China, so I use Chinese. According to this document, I create a FAQ application, found a small bug. In FAQ.Existing FAQ entries (FAQ.webHome) page, in the name of the docName of <input> years, if the input of Chinese characters (such as '???'), then FAQ.Existing FAQ entries page will prompt "The target document already exists. Please choose a different name ...", fact does not exist "Foxconn jumped" this page. Tracking in FAQ.Existing FAQ entries on the velocity code and found that statement "# set ($ docName = $ util.clearName ($ (request.docName)))" always null value assigned to docName, so to determine whether the presence Logic ($ xwiki.exists ($ targetDocName)) returns true. Then I trace the jar file, suspected com.xpn.xwiki.XWiki's public String clearName method caused problems - the Chinese characters are replaced with a null value. Current temporary solution is to comment out this $ util.clearName code. Hope xwikier to fix this bug, because the use of Chinese to abandon this code, the code will be wasted. And now more and more young people in China using Chinese and English characters mixed. Thank you.
PS: I want to know com.xpn.xwiki.XWiki's public String clearName way in the end is what to do, now I know that this method is used to replace some of the non-Chinese characters. Thank you.

- date:
2010-06-01 05:07:34.0

XWiki.XWikiComments

- author:
xwiki:XWiki.cedric_lamalle

- comment:
Note that if your template is named 'FAQTemplate', the live table component will list it, with all column with empty values. In order to automatically filter the template page from the live table name it 'FAQClassTemplate'.

- date:
2011-04-22 15:20:29.0

XWiki.XWikiComments

- author:
xwiki:XWiki.Sergiu

- comment:
This has been fixed in [[3.0>>https://jira.xwiki.org/jira/browse/XWIKI-5760]]

- date:

~~2011-04-23 00:24:44.0~~
- replyto:
~~12~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.coyote21~~
- comment:
~~Does this tutorial still works in recent versions?~~
- date:
~~2013-01-10 22:23:05.0~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.zzeroo~~
- comment:
~~Yes, thist tut is still working with the 5.1. enterprise version of xwiki. I've finished it right now without any changes.~~
~~Hint! You have to enable hidden documents in your user account to find the "XWikiClasses" doc mentioned at the start.~~
- date:
~~2013-06-02 19:02:54.0~~
- replyto:
~~14~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.cruel_world~~
- comment:
~~This example no longer works correctly in 8.4. However, it will if...~~
~~After you create the WebHome (FAQs) page.~~
~~Delete the FAQClass object from WebHome~~
- date:
~~2016-11-17 00:54:51.0~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.OgisTarkan~~
- comment:
~~Its working on latest 9.3.1~~
- date:
~~2017-05-19 10:30:01.0~~

- title: Creating a FAQ Application ⇨
- parent: xwiki:Documentation.DevGuide.FAQTutorial.WebHome ⇨ DevGuide.Tutorials
- author: XWiki.evalica ⇨ XWiki.VincentMassol
- defaultLanguage: en ⇨
- syntax: XWiki 2.1 ⇨ XWiki 1.0

FAQ Tutorial

Documentation.DevGuide.Tutorials.FAQTutorial.WebHome

On 2017/09/05, the page has been modified by Vincent Massol

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WebHome

On 2017/09/05, the page has been created by Vincent Massol

> ~~These tutorials will show you how to build a Frequently Asked Questions (FAQs)~~**= FAQ** Application ~~much like the one you can find on the [[FAQ page>>xwiki:FAQ.WebHome]].~~**Tutorial Manual =**
> **This tutorial will show you how to [[manually>>DevGuide.FAQTutorialManual]] build a Frequently Asked Questions (FAQs) Application much like the one you can find on the FAQ page. This is a very simple application that makes use of XWiki's classes, properties, and objects.**
> ~~We're providing 2 tutorials:~~**= FAQ Application tutorial using App Within Minutes Application =**
> ~~* [[One~~**This tutorial will show you how to build a Frequently Asked Questions (FAQs) Application much like the one you can find on the FAQ page. [[This tutorial>>DevGuide.FAQTUtorialAWM]] shows the steps to create the application** using App Within ~~Minutes>>FAQTutorialAWM]] which is recommended for newcomers. You won't need any technical knowledge for this one.~~**Minutes Application.**
> ~~* [[Another one using the notion of XWiki Classes/Objects/Templates>>FAQTutorialManual]] which are the technologies used underneath by App Within Minutes. App Within Minutes is a wizard approach that will get you started but at some point if you wish to customize further your application created with it you'll need to [[learn the basics>>DataModel]].~~
> ~~In addition you could also install the [[FAQ Application>>extensions:Extension.FAQ Application]] and check how it was implemented.~~
> ~~There's also an older (but still valid) [[Tutorial for creating a Todo application>>http://www.theserverside.com/news/1363830/XWiki-A-Platform-for-Collaborative-Apps]].~~

- title: FAQ Tutorial ⇨ FAQTutorial
- parent: xwiki:Documentation.DevGuide.WebHome ⇨ DevGuide.WebHome
- author: XWiki.VincentMassol ⇨ XWiki.Sorin

Documentation.DevGuide.Tutorials

Creating a Groovy Class

Documentation.DevGuide.Tutorials.GroovyClassHelloWorldTutorial.WebHome

On 2017/09/05, the page has been modified by Vincent Massol

> XWiki.XWikiComments
> comment:

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WebHome
- author: XWiki.evalica ⇨ XWiki.VincentMassol

On 2017/09/05, the page has been created by Vincent Massol

~~{{box cssClass="floatinginfobox" title="**Contents**"}}~~**#warning("Work in progress, only a snippet for the moment")**
~~{{toc/}}~~
~~{{/box}}~~
~~In general, creating a Groovy class in a wiki page is not really recommended and usually it's nicer to create code in Java and make it available to XWiki pages through [[Script Services>>extensions:Extension.Script Module]].~~**1 My first groovy class**
~~However there might be a few cases when it's handy~~**This tutorial illustrates the xwiki.parseGroovyFromPage API method. This method allow you** to ~~create~~**instantiate** a ~~Groovy~~**groovy** class ~~in a wiki page and reuse it~~ from ~~several other wiki pages.~~**both velocity and groovy code.**
~~= In XWiki Syntax 2.0+ =~~**1.1 Create a groovy class**
* Create a **new** page, for example ~~##Groovy.HelloWorldClass## containing:~~
~~(((~~**"Groovy.HelloWorldClass" containing :**
~~{{info}}~~
~~This page must have been saved by a user with programming [[rights>>platform:Features.RightsManagement]] to be executed.~~
~~{{/info}}~~
~~{{code}}~~**{code}**
~~{{groovy}}~~**public class wizard {**
~~class HelloWorld~~
~~{~~
~~String say()~~
~~{~~
~~return "Hello World"~~
~~}~~
~~}~~
~~{{/groovy}}~~
~~{{/code}}~~
~~)))~~
~~* Use this Groovy class from another wiki page, say from ##Main.HelloWorldFromGroovy##:(((~~
~~{{info}}~~
~~This page must have been saved by a user with programming [[rights>>platform:Features.RightsManagement]] to be executed.~~
~~{{/info}}~~
~~{{code}}~~**def xwiki**
~~{{include document="Groovy.HelloWorldClass"/}}~~**def context**
~~{{groovy}}~~ **public void setObjects(xwiki, context) {**
~~println new HelloWorld().say()~~
~~{{/groovy}}~~
~~{{/code}}~~
~~)))~~
~~Now when you view ##Main.HelloWorldFromVelocity## you'll see:~~
~~{{code}}Hello World{{/code}}~~
~~{{info}}~~
~~* Note that with this strategy it's not possible to call the ##HelloWorld## class from Velocity and you'll need to use a Groovy macro to see it.~~
~~* Several variables such as ##xwiki##, ##doc##, ##xcontext## are already bound to the Groovy execution context and can be used. See the [[Scripting page>>DevGuide.Scripting]] for more details.~~
~~{{/info}}~~
~~= In XWiki Syntax 1.0 =~~
~~You'll need to use the ##XWiki.parseGroovyFromPage## API method. This method allow you to instantiate a Groovy class from both Velocity and

~~Groovy scripts.~~
~~* Create a new page, for example ##Groovy.HelloWorldClass## containing:~~
~~(((~~
~~{{info}}~~
~~* This page must have been saved by a user with programming [[rights>>platform:Features.RightsManagement]] to be executed.~~
~~* When creating a page that has to be accessed via ##parseGroovyFromString## make sure you do not have opening and closing groovy identifiers.~~
~~{{/info}}~~
~~{{code}}~~
~~/* Groovy Class #* */~~
~~class HelloWorldClass~~
~~{~~
~~def xwiki~~
~~def context~~
~~void setObjects(xwiki, context)~~
~~{~~
public void ~~setXWiki(xwiki)~~**setXWiki(xwiki) {**
~~{~~
public void ~~setContext(context)~~**setContext(context) {**
~~{~~
public String ~~say()~~**helloWorld() {**
~~{~~
{/code}
~~/* *# */~~**1.1 Instantiate and use your class**
~~{{/code}}~~
~~{{info}}~~*** Create a new page, for example "Main.HelloWorld" containing :**
~~* Notice the trick of putting a Velocity comment in the Groovy comment so that the code is not parsed by Velocity.~~
~~* As you can see, we can get and store the XWiki and Context objects in the class to be able to use them; their use is not illustrated in this tutorial though.~~
~~{{/info}}~~
~~)))~~
~~== Instantiate and use your class from Velocity ==~~**{code}**
#set ($groovyObject =
$xwiki.parseGroovyFromPage("Groovy.HelloWorldClass"))
$groovyObject.setObjects(xwiki, context)
$groovyObject.helloWorld()
{/code}
~~* Create a new page, for example ##Main.HelloWorldFromVelocity## containing:(((~~
~~{{code}}~~
~~#set($groovyObject =~~
~~$xwiki.parseGroovyFromPage("Groovy.HelloWorldClass"))~~
~~$groovyObject.setObjects($xwiki, $context)~~
~~$groovyObject.say()~~
~~{{/code}}~~
~~)))~~
~~* See the result. Feeling groovy ? ;)~~
~~== Instantiate and use your class from Groovy ==~~
~~* Create a new page, for example ##Main.HelloWorldFromGroovy## containing:(((~~
~~{{info}}~~
~~This page must have been saved by a user with programming [[rights>>platform:Features.RightsManagement]] to be executed.~~

{{/info}}
{{code}}
<%
groovyObject = xwiki.parseGroovyFromPage("Groovy.HelloWorldClass")
groovyObject.setObjects(xwiki, context)
print groovyObject.say()
%>
{{/code}}
)))

XWiki.XWikiComments
- author:
XWiki.cardenizen
- comment:
The velocity snippet works: {{velocity filter="none"}}
{{html clean="false" wiki="true"}}
{{{#set($groovyObject =
$xwiki.parseGroovyFromPage("Groovy.HelloWorldClass"))
$groovyObject.setObjects($xwiki, $context) $groovyObject.helloWorld()}}}
{{/html}}
{{/velocity}}
but the groovy snippet does not.
- date:
2007-06-19 17:37:34.0

XWiki.XWikiComments
- author:
XWiki.DengXun
- comment:
How to add programming rights?
- date:
2007-09-19 02:45:10.0

XWiki.XWikiComments
- author:
xwiki:XWiki.RicardoJafe
- comment:
I can't seem to get this to work. I create the class in the space Tutorials, follow instructions but then I get this error:
org.xwiki.rendering.macro.MacroExecutionException: Failed to evaluate Velocity Macro for content [#set($groovyObject =
$xwiki.parseGroovyFromPage("Tutorials.HelloWorldClass"))
Maybe something in the syntax changed?
- date:
2012-03-29 14:42:05.0

XWiki.XWikiComments
- author:
xwiki:XWiki.RicardoJafe
- comment:
I'm going to alter this tutorial because I believe that the Groovy.HelloWorldClass should be used all the time. Prior to my

modification there are places where Groovy.HelloWorldClass is mentioned, others where Groovy.HelloWorldClass is used and yet another place where Dev.HelloWorldGroovyClass.
Althogh I still didn't get this snippet to work on my instance of Xwiki Enterprise I believe this is one of the problems.

- date:
2012-03-29 17:10:43.0

XWiki.XWikiComments
- author:
xwiki:XWiki.VincentMassol
- comment:
This is for XWiki Syntax 1.0. We need to rewrite it for XWiki Syntax 2.0
- date:
2012-03-29 17:14:24.0

XWiki.XWikiComments
- author:
xwiki:XWiki.RicardoJafe
- comment:
Vincent this is working in XWiki Syntax 2.1, it's just necessary to omit the {{code}}{{groovy}} and {{/groovy}} {{/code}} tags in the class definition!
I've tested it just now in XWiki Enterprise 3.5
- date:
2012-03-29 17:31:56.0

XWiki.XWikiComments
- author:
xwiki:XWiki.VincentMassol
- comment:
I've rewritten the tutorial
- date:
2012-03-29 17:52:52.0

XWiki.XWikiComments
- author:
xwiki:XWiki.beldaz
- comment:
It's still not clear how one should call a Groovy class from Velocity using 2.1 syntax - the warning at the end is confusing. Do we still need use parseGroovyFromPage as in the 1.0 syntax section?
- date:
2014-07-23 05:49:53.0
- replyto:
6

XWiki.XWikiComments
- author:
xwiki:XWiki.mawen
- comment:

~~The groovy snippet works: Failed to execute the [groovy] macro. Click on this message for details.~~
~~but the groovy snippet works:$groovyObject.say()~~
~~The xwiki version 5.46.~~
~~why?~~

- date:
~~2015-02-07 04:10:47.0~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.VincentMassol~~
- comment:
~~Just retested the script and it works fine. If you have questions please use the [[Mailing Lists>>dev:Community.MailingLists]].~~
- date:
~~2015-02-08 17:22:16.0~~
- replyto:
~~8~~

- title: Creating a Groovy Class ⇨
- parent: xwiki:Documentation.DevGuide.WebHome ⇨
- author: XWiki.VincentMassol ⇨ XWiki.jvdrean
- syntax: XWiki 2.0 ⇨ XWiki 1.0

Documentation.DevGuide.Tutorials

Old Notifications Tutorial

Documentation.DevGuide.Tutorials.GroovyNotificationTutorial.WebHome

On 2017/09/05, the page has been created by Vincent Massol

~~{{box cssClass="floatinginfobox" title="**Contents**"}}{{toc/}}{{/box}}~~**1 Notifications**
~~{{error}}~~**#warning("The notification mechanism is being refactored and will most likely change in XWiki Core 1.4. Thus if you're using the code below you'll have to update your code when you'll upgrade to XWiki Core 1.4.")**
~~You should check the [[New Notification Tutorial>>WritingEventListenerTutorial]].~~
~~{{/error}}~~
~~This tutorial uses the old notification mechanism~~**XWiki supports notifications and it's possible** to ~~listen to events (now deprecated in XWiki 2.0). You should follow it if you're using~~**do some action when** a ~~version of XWiki Enterprise prior to 2.0.~~**document is modified, when a document's objects are modified, etc.**
~~In order~~**This tutorial explains how** to ~~listen~~**implement an action in a XWiki page that responds** to ~~events you need to write 2 pages:~~**document changes. This is done in Groovy. It can also be implemented as a Plugin.**
We need to write 2 pages:
~~=~~**1.1** Groovy Notification ~~Class =~~**Class**
Your Groovy needs to extend the
~~##com.xpn.xwiki.notify.XWikiDocChangeNotificationInterface##~~

<tt>com.xpn.xwiki.notify.XWikiDocChangeNotificationInterface</tt> as shown below.
{{code language="java"}}{code:none}
{{/code}}{code}
In this example we've used a ##DocChangeRule##
<tt>DocChangeRule</tt> rule. There are also {{scm path="xwiki-platform-core/xwiki-platform-legacy/xwiki-platform-legacy-oldcore/src/main/java/com/xpn/xwiki/notify"}}other rules{{/scm}}.[other rules>http://svn.xwiki.org/svnroot/xwiki/xwiki-platform/core/trunk/xwiki-core/src/main/java/com/xpn/xwiki/notify/].
=1.1 Calling the Groovy Class =Class
{{code}}{code:none}
{{/code}}{code}
=1.1 Example: IRC notification on document change =change
{{warning}}#warning("The code below uses the parseGroovyFromPage method which takes 2 parameters. The second one is the name of page containing JARS as attachments. These JARs are put in the classloader used by Groovy when parsing the page. This feature is only working in XWiki Core 1.3 and later.")
The code below uses the parseGroovyFromPage method which takes 2 parameters. The second one is the name of page containing JARS as attachments. These JARs are put in the classloader used by Groovy when parsing the page. This feature is only working in XWiki Core 1.3 and later.
{{/warning}}
* **Step 1**:Step 1: Groovy Class(((Class
{{code language="java"}}{code:none}
{{/code}}{code}
)))* Step 2: Add the PircBot JAR as an attachment to the <tt>MySpace.MyGroovyClass</tt> page created in step 1.
* **Step 2**: Add the PircBot JAR as an attachment to the ##MySpace.MyGroovyClass## page created in step 1.Step 3: Calling page
* **Step 3**: Calling page((({code:none}
{{code}}
{{/code}}{code}
)))
* **Step 4**: Creating a Scheduler job so that the Bot is restarted automatically if the server is restarted for example.(((
Create a Scheduler job, set it to run every 5 minutes for example and use the following Groovy script in the job:
{{code language="java"}}
// Start by looking for a bot in the servlet context
def sc = context.getEngineContext().getServletContext()
// If the bot isn't in the servlet context, start the bot and put in the context
def bot = sc.getAttribute("ircbot")
if (bot == null) {
// Bot is not started, starting it...
bot = xwiki.parseGroovyFromPage("MySpace.MyGroovyClass", "MySpace.MyGroovyClass")
def channel = "#xwiki"
bot.init(xwiki, channel)
bot.connect("irc.freenode.net")
bot.joinChannel(channel)
sc.setAttribute("ircbot", bot)
// Bot started!
}
{{/code}}
)))

~~~~

groovynotifier.txt: 1.3 ⇨ ()

XWiki.XWikiComments
- author:
~~xwiki:XWiki.LSu~~
- comment:
~~which location should should this files be put in and how they corporates?~~
- date:
~~2013-11-27 12:27:03.0~~

- title: Old Notifications Tutorial ⇨
- parent: xwiki:Documentation.DevGuide.WebHome ⇨ DevGuide.Tutorials
- syntax: XWiki 2.0 ⇨ XWiki 1.0

On 2017/09/05, the page has been modified by Vincent Massol

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WebHome
- author: XWiki.evalica ⇨ XWiki.VincentMassol

Documentation.DevGuide.Tutorials

Writing Internationalized XWiki Applications

Documentation.DevGuide.Tutorials.InternationalizingApplications.WebHome

On 2017/09/05, the page has been created by Vincent Massol

~~{{box cssClass="floatinginfobox" title="**Contents**"}}~~**1 Writing Internationalized XWiki Applications**
~~{{toc/}}~~
~~{{/box}}~~
~~= Tutorials =~~**#toc("" "" "")**
~~* [[Tutorial for Internationalizing a Form>>Documentation.DevGuide.TranslationsTutorial]]~~**1.1 Using the static Resource Bundles**
~~= Using properties located in XWiki pages =~~**TODO**
~~* Create a page that will host your key/value pairs~~**1.1 Using properties located in XWiki pages**
~~* Enter all the key/value pairs in that page ({{warning}}use the wiki editor{{/warning}}). For example:(((~~
~~{{code language="none"}}~~
~~greeting=hello~~
~~welcome=Welcome~~
~~withparams=I can {0} pass {1} {2} params too using the Java MessageFormat syntax~~
~~{{/code}}~~
~~)))~~
~~* Do the same for all the translations you want.(((~~
~~{{info}}~~
~~To enable multiple languages you'll need to go to the Administration page and set MultiLingual to true and list the different languages you wish to use in the Languages field.~~
~~{{/info}}~~
~~~~

)))
* Tell XWiki that your page is a Translation page by adding an Object of type ##TranslationDocumentClass## to your page.
* On the page where you want to use use either the ##~{~{translation}}## macro or the Localization Script Service (check the document for the [[Localization module>>extensions:Extension.Localization Module]]). Quick examples:(((
From wiki syntax:
{{code}}**#warning("The features described below are only available in XWiki 1.0 Beta 4 or above")**
{{translation key="key"/}}
{{/code}}
From Script (Velocity in this example):**TODO**
{{code}}
$services.localization.render("key")
$services.localization.render("key", ["param1", "param2", ...])
{{/code}}
where ##key## is the key for the message to retrieve. Parameters can also be passed as is shown in the second syntax above.
)))
== Old way ==
Prior to XWiki 4.3, the way to register a Document Resource Bundle was by going to the Administration page, selecting the "Localization" section and entering it in the "Internationalization Document Bundles" field. You could specify several pages, separated by commas.
{{info}}
When creating "Internationalization Document Bundles", you should avoid naming the pages with names that include spaces. For example, a document name of Main.My Messages could cause issues, instead, use Main.MyMessages
{{/info}}
Alternatively you could also specify the list of Internationalization Document Bundles in ##xwiki.cfg## under the key ##xwiki.documentBundles##.
= I18n of XWiki Objects =
This is currently not implemented (see our logged issue: [[XWIKI-69>>https://jira.xwiki.org/browse/XWIKI-69]]). There are 2 workarounds you can use:
* Have several objects, one for each language, with a language field, and then decide which object to use (for example in a Class Sheet, based on the current language):(((
{{code language="none"}}
$doc.getObject('Your.Class', 'lang', $context.language')
Will select the object whose 'lang' property is set to the current language
{{/code}}
)))
* Use Velocity scripting to do an IF in your object. For example, you could have:(((
{{code language="none"}}
#if ($context.language == "fr")
French texts
#else
Default texts
#end
{{/code}}
)))
= Using the static Resource Bundles =
* Stop your XWiki instance
* Extract the ##ApplicationResources*.properties## file for your language

from the ##WEB-INF/lib/xwiki-*oldcore-*.jar## file and put them in your ##WEB-INF/classes## directory
* Edit this file and add/modify the key/value pairs you need for your application
* In your document, use the following to get the value associated with a key: {{code language="none"}}$services.localization.render("key"){{/code}}
This will get the value from the Resource Bundle corresponding to the current language you are using in XWiki.
If you're writing an extension JAR and wish to have static translation resources, you can put them in a ##ApplicationResources*.properties## file at the root of your extension JAR and they'll get picked up by the Localization module. See the [[Localization module>>extensions:Extension.Localization Module||anchor="HProvidetranslationbundleinajar"]] for more details.

XWiki.XWikiComments
- author:
xwiki:XWiki.evonli
- comment:
How to distinguish diffrent language by "Using properties located in XWiki pages"?
- date:
2008-08-15 09:14:28.0

XWiki.XWikiComments
- author:
xwiki:XWiki.tapaya
- comment:
The procedure for "Using properties located in XWiki pages" lacks information:
* How would I distinguish between different languages? Is there a naming convention for the pages (like "de", "en" ...)?
* Would I create a single page for key/value pairs and translate that to the other languages I want, or are there separate documents for each language?
* How exactly would I access the keys, e.g. {{{$msg.get("<space>.<my language page>.<key>")}}}?
- date:
2009-09-02 09:48:21.0

XWiki.XWikiComments
- author:
xwiki:XWiki.polx
- comment:
Evon Li, Philipp,
properties files that are declared to be part of the i18n bundle are translated using the normal document translation mechanism... i2geo.net translations used this with big success: we moved all translations documents to a single space and invited voluntaries to edit the translations live. They could see their translations automatically having effect.
paul
- date:

~~2013-01-24 22:57:12.0~~

- title: Writing Internationalized XWiki Applications ⇨
- parent: xwiki:Documentation.DevGuide.WebHome ⇨ DevGuide.WebHome
- author: XWiki.evalica ⇨ XWiki.VincentMassol
- syntax: XWiki 2.0 ⇨ XWiki 1.0

On 2017/09/05, the page has been modified by Vincent Massol

XWiki.XWikiComments
comment:

XWiki.XWikiComments
comment:

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WebHome

Documentation.DevGuide.Tutorials

Master Detail Tutorial

Documentation.DevGuide.Tutorials.MasterDetailTutorial.WebHome

On 2017/09/05, the page has been modified by Vincent Massol

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WebHome

On 2017/09/05, the page has been created by Vincent Massol

Explains how to implement a [[master-detail view>>https://en.wikipedia.org/wiki/Master%E2%80%93detail_interface]] in XWiki using [[Applications Within Minutes>>extensions:Extension.App Within Minutes Application]] (AWM) and some custom ~~coding (while waiting for this be [[built in the XWiki product>>https://jira.xwiki.org/browse/XWIKI-12598]]).~~**coding.** More specifically we'd like to have an HTML Form with 2 fields and when selecting a value in one field, the values available in the other field automatically update based on the first selection.
~~{{info}}~~**Note: XWiki forces us to give a name to each entry so we have used ##entry1## till ##entry5##.**
~~Note: XWiki [[currently forces us to give a name to each entry>>https://jira.xwiki.org/browse/XWIKI-7374]] so we have used ##entry1## till ##entry5##.~~
~~{{/info}}~~
Let's now use AWM again to create the State Application which is the application ~~in which~~ we wish to ~~have the master-detail implemented.~~
~~In the design view, start by adding a ##State## field of type ##Database List##:~~**TODO**
~~{{image reference="state-form1.png"/}}~~
~~Note that we link this field with the ##state## field of the first State Data Application we have created through its ##StateDataCode.StateDataClass##.~~
~~Then add a ##City## field of type ##Static List## with no entries:~~
~~{{image reference="state-form2.png"/}}~~
~~= Step 4: Modify the State Class Sheet =~~
~~Let's now modify ##StateCode.StateSheet## to implement dynamically~~

displaying the City HTML Select. Modify the default content to be:
{{code}}
{{velocity}}
{{html wiki="true"}}
#set ($discard = $doc.use('StateCode.StateSheet'))
#set ($discard = $services.localization.use('document',
'StateCode.StateTranslations'))
(% class="xform" %)
(((
<dl>
<dt><label
for="StateCode.StateClass_0_state">$escapetool.xml($doc.displayPrettyN
ame('state', false, false))</label></dt>
<dd>$doc.display('state')</dd>
<dt><label
for="StateCode.StateClass_0_city">$escapetool.xml($doc.displayPrettyNa
me('city', false, false))</label></dt>
<dd>
#if ($xcontext.action == 'edit')
$xwiki.jsx.use('StateCode.StateClass')
<select id="StateCode.StateClass_0_city"
name="StateCode.StateClass_0_city" size="1">
<option></option>
</select>
#else
$!doc.getValue('city')
#end
</dd>
</dl>
)))
{{/html}}
{{/velocity}}
{{/code}}
Note that we're using a [[Javascript Skin
Extension>>platform:DevGuide.SkinExtensionsTutorial]] that we'll be
creating in the current page (i.e. ##StateCode.StateSheet##): {{code}}#set
($discard = $doc.use('StateCode.StateSheet')){{/code}}
= Step 5: Add a Javascript Skin Extension =
Edit ##StateCode.StateSheet## in the Object editor and add a
##XWiki.JavaScriptExtension## Object:
{{image reference="state-jsx.png"/}}
You can copy-paste this code:
{{code}}
require(['jquery'], function ($) {
var stateSelect = $('#StateCode\\.StateClass_0_state');
var citySelect = $('#StateCode\\.StateClass_0_city');
var jsonDocument = new XWiki.Document('WebHome', 'StateCode.JSON');
stateSelect.change(function() {
$.get(jsonDocument.getURL('get', 'outputSyntax=plain&state=' +
stateSelect.val()), function(json) {
var output = '';
$.each(json, function(index, value) {
output += '<option>' + value + '</option>';
});
citySelect.empty().append(output);
});
});

~~});~~
~~{{/code}}~~
~~Note that we expect an XWiki page named ##StateCode.JSON.WebHome## to return some JSON containing a list of City values for the passed State. We'll add it in the next step.~~
~~{{info}}~~
~~Instead of creating our own page that returns JSON we could also reuse the Livetable Results Page created by AWM for our State Application. This would avoid having to create a new page and thus skip Step 6. Here's an example URL of how you'd call it and filter on the ##State## field:~~
~~{{code}}~~
~~var jsonDocument = new XWiki.Document('StateDataLiveTableResults', 'StateDataCode');~~
~~var url = jsonDocument.getURL('get', 'outputSyntax=plain&classname=StateDataCode.StateDataClass&collist=state%2Ccity&state=' + stateSelect.val())~~
~~{{/code}}~~
~~{{/info}}~~
~~= Step 6: Create a JSON Service =~~
~~Create a page named ##StateCode.JSON.WebHome## (just add a page named ##JSON## in the ##StateCode## space in the Create Page UI):~~
~~{{image reference="state-json.png"/}}~~
~~You can copy paste the following as its content:~~
~~{{code}}~~
~~{{velocity}}~~
~~#if($xcontext.action == 'get' && "$!{request.outputSyntax}" == 'plain')~~
~~$response.setContentType('application/json')~~
~~#end~~
~~#set($list = [])~~
~~#set ($state = $request.state)~~
~~#if ("$!state" != '')~~
~~#set ($itemList = $services.query.xwql("where doc.object(StateDataCode.StateDataClass).state like :state").bindValue('state', $state).execute())~~
~~#foreach ($item in $itemList)~~
~~#set ($itemDoc = $xwiki.getDocument($item))~~
~~#set ($discard = $list.add($itemDoc.getValue('city')))~~
~~#end~~
~~$jsontool.serialize($list)~~
~~#end~~
~~{{/velocity}}~~
~~{{/code}}~~
~~You're all set! Let's now try it!~~
~~= Step 7: Create an entry in the State Application =~~
~~Navigate back to the State Application and Create an entry and verify that when you change the State, the list of available Cities is updated :)~~
~~{{image reference="state-result1.png"/}}~~
~~And when viewing the page:~~
~~{{image reference="state-result2.png"/}}~~
~~= Step 8: Wrap it all! =~~
~~Here's in attachment a [[XAR file>>attach:state-1.0.xar]] that you can import in your wiki and that contains all the steps of this tutorial.~~

state-1.0.xar: 1.1 ⇨ ()

state-form1.png: 1.1 ⇨ ()

state-form2.png: 1.1 ⇨ ()

state-json.png: 1.1 ⇨ ()

state-jsx.png: 1.1 ⇨ ()

state-result1.png: 1.1 ⇨ ()

state-result2.png: 1.1 ⇨ ()

statedata-entries.png: 1.1 ⇨ ()

statedata-form.png: 1.1 ⇨ ()

- parent: DevGuide.WebHome ⇨ Drafts.WebHome

Documentation.DevGuide.Tutorials

Performing Asynchronous Tasks

Documentation.DevGuide.Tutorials.PerformingAsynchronousTasks.WebHome

On 2017/09/05, the page has been modified by Vincent Massol

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WebHome

On 2017/09/05, the page has been created by Vincent Massol

{{toc ~~depth="1"/}}**depth="1" /}}**
~~In this tutorial for the [[Job Module>>extensions:Extension.Job Module]] we implement a space rename function~~**Implement space rename** taking into account that:
~~* hence this operation can take a lot of time so we need to display the progress which means we cannot block the HTTP request that triggers the operation; in other words the operation should be asynchronous.~~
This operation can take a lot of time so we need to display the progress. This means we cannot block the HTTP request that triggers the operation. In other words the operation should be asynchronous.

jobFlow.png: 1.1 ⇨ ()

- parent: DevGuide.WebHome ⇨
- author: XWiki.VincentMassol ⇨ XWiki.mflorea

Documentation.DevGuide.Tutorials

Saving Data

Documentation.DevGuide.Tutorials.SavingData.WebHome

On 2017/09/05, the page has been modified by Vincent Massol

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WebHome

On 2017/09/05, the page has been created by Vincent Massol

If the data to be saved should only be saved for the duration of a ~~request~~**Request** (i.e. a call to ~~an~~**a** XWiki ~~URL)~~**URL),** then you should ~~declare a property in~~**use** the **Execution Context**.

// ~~Declaring~~**Saving**

~~ec.newProperty("mykey").initial(initialValue).declare();~~**ec.setProperty("mykey", myvalue);**

~~// Updating~~

~~ec.setProperty("mykey", newValue);~~

~~...~~

~~===~~**From a wiki page (Note that [[we currently cannot access easily the** Execution Context ~~Property Declaration ===~~**from a wiki page but this should be improved>>http://jira.xwiki.org/browse/XCOMMONS-242]] and currently it's not available only from Velocity but it is from Groovy for example):**

~~The declaration is currently optional, the property will be implicitly declared when set the first time. The declaration can be made in an ##ExecutionContextInitializer## component. The initialize method of all such components that are available in the classpath and have been specified in a ##components.txt## file will be called when the execution context is first initialized during the request.~~

~~{{code language="java"}}~~

```
import org.xwiki.context.ExecutionContext;
import org.xwiki.context.ExecutionContextInitializer;
import org.xwiki.context.ExecutionContextException;
import org.xwiki.component.annotation.Component;
@Component
public class DeclareMyPropertyExecutionContextInitializer implements
ExecutionContextInitializer
{
@Override
public void initialize(ExecutionContext executionContext)
{
if (!executionContext.hasProperty("mykey")) {
executionContext.newProperty("mykey").type(String.class).nonNull().initial("
").declare();
}
}
}
```

~~{{/code}}~~

~~In the above example you can see that it is possible to set various attributes on the property when it is beeing declared. The attributes currently supported are listed in the table below.~~

~~During a request cycle, some components may activate a //clean// or a //cloned// execution context by pushing to the execution context stack. You can control how you want your property to be managed in these situations by setting attributes.~~

~~|=(% style="width:25%" %)Attribute |=Parameters |=Description~~

~~|cloneValue()| - |Indicate that you want the value to be cloned when the execution context is cloned or when your property is inherited from a parent execution context~~

~~|makeFinal()| - |Indicate that an exception should be thrown if someone tries to replace the initial value object~~

~~|inherited()| - |Indicate that the property should be inherited by any new execution context that is pushed within the current request~~

~~|nonNull()| - |Indicate that an exception should be thrown if ##null## is set as the property value~~

~~|type(Class<?> type)|((~~

~~|type(Class<?> type)|(((~~
~~; type~~
~~: the class that the value object should be typechecked against~~
~~)))|Set a type for performing typechecking when updating the value object.~~
~~=== Wiki Page Access to Execution Context ===~~
~~{{info}}~~
~~We currently [[cannot access easily the Execution Context from a wiki page>>https://jira.xwiki.org/browse/XCOMMONS-242]] but this should be improved. Currently it's not available from Velocity, but it is from Groovy.~~
~~{{/info}}~~
~~From a wiki page:~~
~~{{info}}~~**Note that from a wiki page it's much easier to use the older **XWiki Context** (which is supposed to be fully replaced by the Execution Context at some point in the future). For example:**
~~From a wiki page it's much easier to use the older **XWiki Context** (which is supposed to be fully replaced by the Execution Context at some point in the future).~~
~~{{/info}}~~
~~For example:~~
~~{{warning}}~~
~~This requires Programming Rights.~~
~~{{/warning}}~~
~~{{info}}~~
~~It's also possible to save data in the HTTP Request even though it's recommended to use the Execution Context instead (since the Execution Context is independent of the execution environment and will work everywhere: Servlets, Portlets, JavaSE, etc).~~
~~{{/info}}~~
~~For example from Velocity in a wiki page:~~
~~{{code}}~~
~~{{velocity}}~~
~~$request.setAttribute("mykey", value)~~
~~{{/velocity}}~~
~~{{/code}}~~
If the data should last a little longer (i.e. for example span more than 1 request), you could save it in the Servlet ~~Session (note that in this case your code will only work in a Servlet Environment).~~**Session.**
~~Example from a wiki page:~~
~~{{code}}~~
~~{{velocity}}~~
~~$request.getSession().setAttribute("mykey", myvalue)~~
~~{{/velocity}}~~
~~{{/code}}~~
If the data should last even a little longer than a session, you could save it in a ~~file~~**File** in the ~~environment's temporary directory.~~**Environment's Temporary Directory.**
If the Data should last as long as the web application is up, you could use the Servlet Context. It's not very easy to access it but it's ~~possible (note that in this case your code will only work in a Servlet Environment).~~**possible.**
The best place to save ~~data~~**Data** is usually in the wiki itself **in an XObject attached to a wiki page** (See the [[Data ~~Model>>DevGuide.DataModel]] and the [[API Guide>>DevGuide.APIGuide]] for some examples). There are several places where you can store data in the model:~~**Model>>DevGuide.DataModel]]).**
~~* In the document content itself~~
~~* In an XObject attached to a document~~
~~* In an attachment~~
Another possibility is to use the ~~environment's permanent~~

~~directory.~~**Environment's Permanent Directory.**

- parent: xwiki:Documentation.DevGuide.WebHome ⇨

Documentation.DevGuide.Tutorials

Skin Extension Tutorial

Documentation.DevGuide.Tutorials.SkinExtensionsTutorial.WebHome

On 2017/09/05, the page has been modified by Vincent Massol

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WebHome
- author: XWiki.evalica ⇨ XWiki.VincentMassol

On 2017/09/05, the page has been created by Vincent Massol

~~{{box cssClass="floatinginfobox" title="**Contents**"}}~~**#startfloatingbox()**
~~{{toc/}}~~***Contents***
~~{{/box}}~~**#toc ("2" "3" "")**
#endfloatingbox()
1 Skin Extension Tutorial
~~{{velocity}}~~**## $xwiki.ssx.use("$doc.fullName") ## Load the SSX object held in this documnet**
~~## $xwiki.ssx.use("$doc.fullName") ## Load~~**#warning("Some parts of this tutorial (the *Used Always* extensions) rely on** the ~~SSX object held~~**version 1.4 of the [Skin Extension plugin>code:Plugins.SkinExtensionsPlugin], which has been released with XWiki Enterprise 1.7RC1. If you are using any version of XWiki Enterprise superior to 1.4, but older than 1.7RC1, you can just upgrade [the 1.4 version of xwiki-plugin-skins>code:Plugins.SkinExtensionsPluginDownloads] (by simply replacing the bundled skinx plugin jar located** in **<tt>webapps/xwiki/WEB-INF/lib/</tt> by 1.4 one), and benefit from everything you need to follow** this ~~document~~**tutorial.")**
~~{{/velocity}}~~
~~= Prerequisites =~~**1.1 Introduction to XWiki Skin Extensions**
~~* Basic knowledge of application development with XWiki. If this~~**XWiki Skins eXtensions (abbreviated as *SX*)** is ~~not the case, we strongly advise you~~**a mechanism available in XWiki that allows** to ~~start with~~**customize** the ~~[[FAQ application tutorial>>platform:DevGuide.FAQTutorialManual]]~~**layout of your wiki,** or **just some pages of your wiki, without** the ~~[[TODO application tutorial>>http://www.theserverside.com/news/1363830/XWiki-A-Platform-for-Collaborative-Apps]].~~**need of changing its skin templates and/or stylesheets. For this, the [SkinExtensions plugin>code:Plugins.SkinExtensionsPlugin] (bundled in all XWiki Enterprise versions superior to 1.5) provides the ability to send to the browser extra JavaScript and CSS files that are not part of the actual skin of the wiki. The code for these ~~extensions~~ is defined in [wiki objects>platform:DevGuide.DataModel].**
~~* Your user must be an [[advanced users>>platform:Features.PageEditing||anchor="HAdvancedMode"]] (The object editor is available only to advanced users).~~
~~= Introduction~~**This tutorial assumes that you already have basic knowledge of application development with XWiki. If this is not the case, we strongly advise you** to ~~XWiki Skin Extensions =~~**start with the**

[FAQ application tutorial>platform:DevGuide.FAQTutorial] or the [TODO application tutorial>http://www.theserverside.com/tt/articles/article.tss?l=XWiki]. To illustrate usage of Skin eXtensions in XWiki, this tutorial will guide you through the creation of minimal JavaScript and StyleSheet working extensions. Then, will push it further to build a fully functional extension based on Natalie Downe's [addSizes.js>http://natbat.net/2008/Aug/27/addSizes/] script.
~~XWiki Skins eXtensions (abbreviated as **SX**)~~**A minimal [JavaScript>http://en.wikipedia.org/wiki/JavaScript] and [CSS>http://en.wikipedia.org/wiki/CSS] knowledge** is ~~a mechanism available in XWiki that allows~~**also needed** to ~~customize the layout~~**take full advantage** of ~~your wiki, or just some pages of your wiki, without the need of changing its skin templates and/or stylesheets. For this, the [[Skin Extension plugin>>extensions:Extension.Skin Extension Plugin]] (bundled in all~~ XWiki ~~Enterprise versions superior~~**Skin eXtensions, although expert knowledge in those fields is not needed** to ~~1.5) provides~~**follow** the ~~ability to send to the browser extra JavaScript and CSS files that are not part of the actual skin of the wiki. The code for these //extensions// is defined in [[wiki objects>>platform:DevGuide.DataModel]].~~**tutorial.**
~~To illustrate usage of~~**#info("If you are interested by the** Skin ~~Extension in XWiki,~~**eXtension mechanism itself and its internals, you should read its [plugin page on code.xwiki.org>code:Plugins.SkinExtensionsPlugin], and its [design page on** this **wiki>Design.SkinExtensions]. This** tutorial ~~will guide you through~~**does not address this topic. Or, since this is an Open Source project, feel free to [browse** the ~~creation of minimal JavaScript~~**code>http://svn.xwiki.org/svnroot/xwiki/platform/xwiki-plugins/trunk/skinx/],** and ~~StyleSheet working extensions. Then, will push it further to build a fully functional extension based on Natalie Downe's ##addSizes.js## script.~~**[propose enhancements or improvements>dev:Community.Contributing].")**
~~A minimal [[JavaScript>>http://en.wikipedia.org/wiki/JavaScript]] and [[CSS>>http://en.wikipedia.org/wiki/CSS]] knowledge is also needed to take full advantage of XWiki~~**1.1 My first** Skin ~~Extensions, although expert knowledge in those fields is not needed to follow the tutorial.~~**eXtension**
~~{{info}}~~**Skin eXtensions are defined as [XWiki Objects>platform:DevGuide.DataModel#HXWikiClasses2CObjects2CandProperties]. As a consequence, you can create them from your browser. Two types of extensions are currently supported: JavaScript eXtensions (incarnated by XWiki objects of class *XWiki.JavaScriptExtension*), and StyleSheet eXtensions (incarnated by XWiki objects of class *XWiki.StyleSheetExtension*). The very first step to create an eXtension is then... to create its object!**
~~If you are interested by the Skin eXtension mechanism itself and its internals, you should read its [[plugin page>>extensions:Extension.Skin Extension Plugin]], and its [[design page on dev.xwiki.org>>dev:Design.Skin Extensions]]. This tutorial does not address this topic. Or, since this is an Open Source project, feel free to {{scm path="xwiki-platform-core/xwiki-platform-skin/xwiki-platform-skin-skinx"}}browse the code{{/scm}}, and [[propose enhancements or improvements>>dev:Community.Contributing]].~~
~~{{/info}}~~
~~= My first Skin eXtension =~~**1.1.1 Minimal JavaScript eXtension**
~~Skin eXtensions are defined as [[XWiki Objects>>platform:DevGuide.DataModel||anchor="HXWikiClasses2CObjects2CandProperties"]]. As a consequence, you can create them from your browser.~~ **1.1.1.1 Creating an eXtension object**

~~Two types~~**Point your wiki on the page you want to create your extension in, and edit it with the object editor. The page itself can be any XWiki page - an existing page or a new page. I use in this example the page *XWiki.MyFirstJavascriptExtension*. From the *Add Object* panel** of ~~extensions are currently supported:~~**the object editor choose *XWiki.JavaScriptExtension* in the Class selector menu. Then, click the "Add Object from this class" button.**
~~* JavaScript eXtensions (incarnated by XWiki objects of class **XWiki.JavaScriptExtension**), also known as **JSX**~~
{image:CreateJSXObject.png}
~~* StyleSheet eXtensions (incarnated by XWiki objects of class **XWiki.StyleSheetExtension**), also known as **SSX**~~**#info("The object editor is available only to [advanced users>platform:Features.PageEditing#HSimpleandAdvancededitionmodes].")**
~~The very first step to create an eXtension is then... to create its object!~~
~~== Minimal JavaScript eXtension ==~~
~~=== Creating an eXtension object ===~~
~~Point your wiki on the page you want to create your extension in, and edit it with the object editor. The page itself can be any XWiki page - an existing page or a new page. I use in this example the page **XWiki.SkinExt**. From the **New Object** drop-down list of the object editor choose **XWiki.JavaScriptExtension**. Then, click the "Add" button.~~
~~{{image reference="CreateJSXObject.png"/}}~~
~~===~~**1.1.1.1** Writing the ~~eXtension ===~~**eXtension**
Now that the object is available, we can just start writing the actual eXtension. For this, we will fill in all the fields of the created object. The first one is the extension name. This is easy! We can just ~~write~~**say** here ~~**Hello World**~~***My First JavaScript eXtension*** (this information is only descriptive, it is not actually used by the SX plugin). The next field name is ~~**code**,~~***code*,** and this is where we will write the javascript code we want our extension to execute. This eXtension is supposed to be minimalist, so let's write something very basic here: a traditional greeting alert **;)**
~~{{code}}~~**{code}**
~~alert("Hello World!");~~**alert("World, Hello from JSX !");**
~~{{/code}}~~**{code}**
Now ~~the~~ next field asks us if we want this extension to be used ~~**Always**~~***Always*** or ~~**On Demand**.~~***On Demand*.** We will explore all the differences between those two modes later in the tutorial, let us for now just precise we want it ~~**On Demand**,~~***On Demand*,** which will force us to ~~call the eXtension explicitly to see it executed.~~
call the eXtension explicitly to see it executed.
Next thing our eXtension wants to know is if we want its content being parsed or not. This option allows to write ~~**[[velocity code>>platform:DevGuide.Scripting]]**,~~***[velocity code>platform:DevGuide.Scripting]*,** for example to dynamically generate the javascript code to be executed. We did not write any velocity, so we can just say ~~**No**.~~***No*.** We will see later on an example of an extension with parsed content.
Finally, we can precise a ~~**caching policy**,~~***caching policy*,** to tune the HTTP headers that will be returned with the generated javascript file. Let's not go wild, and chose the ~~**default**~~***default*** one here **:)**
That's it ! our eXtension is production-ready ! It should by now look like the ~~following:~~**following :**
~~{{image reference="MyFirstJSX.png"/}}~~**{image:MyFirstJSX.png}**
~~//Note:~~**Note:** the "code" area size has been intentionally reduced for this ~~screenshot.//~~**Screenshot.**
~~===~~**1.1.1.1** Testing the actual ~~extension ===~~**extension**

~~~~1.1.1.1 Testing the actual extension ~~~~extension
Let's now test the whole thing! Remember we chose that our extension should be used //on demand// ?~~on demand~~? Well, that's what we are going to do right now. For this we will make a call to the [[Skin[Skin Extension plugin>>extensions:Extension.Skin Extension Plugin]].plugin>code:Plugins.SkinExtensionsPlugin]. We can do it for instance in the wiki content of our extension page, or any other page. For this, we edit the target page in Wiki mode, and write the following:following :
{{code}}{code}
{{velocity}}$xwiki.jsx.use("XWiki.MyFirstJavascriptExtension")
#set ($discard = $xwiki.jsx.use('XWiki.SkinExt.WebHome'))}{code}
{{/velocity}}
{{/code}}
Of course, if you did not use this page name for your extension, you should adapt it. Click "Save and View", et voila!voila ! If everything is fine, you should see the magic:magic :
{{image reference="JSXMagic.png"/}}{image:JSXMagic.png}
You may have noticed that the javascript alert displays before the document is fully loaded in the browser. This is actually expected! If you look close at the generated sources, you will see that your extension has actually been added in the HTML header as any other **.js files***.js files* from the skin: (comments added for this tutorial)
{{code}}{code}
<script type='text/javascript' src='/xwiki/bin/jsx/XWiki/SkinExt?lang=en'> </script>src='/xwiki/bin/jsx/XWiki/MyFirstJavascriptExtension? lang=en'></script>
{{/code}}{code}
You may also note, that the browser is delivered a minified version of the script given in the object's text. This is good practice for the memory of the browser but may make it hard to debug. Using the extra parameter ##debug=true## is a way to prevent it as explained in the [[Debugging Page>>doc:dev:Community.Debugging]].1.1.1 Minimal StyleSheet eXtension
== Minimal StyleSheet eXtension ==Good, we wrote our first javascript extension. But, we see things big and we already are looking forward to modify the graphical appearance of wiki pages using those eXtensions. That's what *StyleSheet eXtensions* are meant for. And the good news is that it just work the same as javascript extensions, the only difference being that the code written is *CSS code*.
Good,Create a new page named *XWiki.MyFirstStylesheetExtension*. In the object editor, we wrote our first javascriptwill now add an object of class *XWiki.StyleSheet* extension. But, we see things big and we already are looking forward to modifyWe will give it the graphical appearance of wiki pages using those eXtensions. That's what **StyleSheet eXtensions** are meant for. And the good news is thatname *My First StyleSheet extension*, give it just worksa *default* cache policy, ask it not to parse the same as javascript extensions,content, and write the only difference being that the code written is **CSS code**. following *code* :
Create a new page named **XWiki.MyFirstStylesheetExtension**. From the **New Object** drop-down list of the object editor choose **XWiki.StyleSheetExtension**. Then, click the "Add" button. We will name it **Blue Background**, give it a **default** cache policy, ask it not to parse the content, and write the following **code**:{code}
{{code}}
{{/code}}{code}
{{info}}Now let's try something new with this eXtension. Instead of
~~~~

loading it "On Demand", we can ask to have it used *"Always"*. For this to happen however, you need to save the extension document with [programming rights>platform:Features.RightsManagement].
If you want to use the colors## TODO change this link once a better description of your active ColorTheme you can check how to [[call those variables>>extensions:Extension.Color Theme Application||anchor="HUsingColorThemesvariables"]].the programming right is written.
{{/info}}
== Put all together ==Your StyleSheet eXtension should now look like the following :
Now let's try something new with this eXtension. Instead of loading it "On Demand", we can ask to have it used **"Always on this wiki"**. For this to happen however, you need to save the extension document with [[programming rights>>platform:AdminGuide.Access Rights||anchor="HSpecialpermissions"]].{image:MyFirstSSX.png}
Your StyleSheet eXtension should now look likeNote: the following:"code" area size has been intentionally reduced for this Screenshot.
{{image reference="MyFirstSSX.png"/}}It's time to test it. No need to call the SkinExtension plugin this time, this is the power of *Use Always* extensions, just click "Save and View" and see the SSX Magic. You can browse your wiki, all pages will be affected by your extension, for example the Main.WebHome page :
//Note: the "code" area size has been intentionally reduced for this Screenshot.//{image:SSXMagic.png|width=800}
It's timeNote: if you want to use StyleSheet extension on demand, the SkinExtension plugin this time, thisprinciple is the power of **Use Always** extensions, just click "Save and View" and see the SSX Magic. You can browse your wiki, all pages will be affected by your extension,same as for examplejavascript, except that the Main.WebHome page:plugin name is *ssx*, not *jsx*. Just make your call like this, and you are done :
{{image reference="SSXMagic.png"/}}{code}
$xwiki.ssx.use("XWiki.MyFirstStylesheetExtension")
{code}
Note: if you want to use StyleSheet extension on demand, the principle is the same as for javascript, except that the plugin name is **ssx**, not **jsx**. Just make your call like this, and you are done:1.1 Real-world eXtension with addSizes.js
{{code}}Let's now go further with this idea, and build a complete extension that will dynamically add the file type and size next to certain links that are present in a wiki document. This extension will make usage of the * [addSizes.js>http://natbat.net/2008/Aug/27/addSizes/]* script published by [Natalie Downe>http://natbat.net/]. This Javascript snippet itself relies on *[json-head>http://simonwillison.net/2008/Jul/29/jsonhead/]*, a Google App Engine application by [Simon Willison>http://simonwillison.net/] which "provides a JSON-P API for running [HEAD requests>http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.4] against an arbitrary URL". addSizes.js consumes this service to dynamically add the file type and size next to links in HTML documents. And this is what we will do in our new eXtension, using the aforementioned script and service.
{{velocity}}
#set ($discard =

$xwiki.ssx.use('XWiki.MyFirstStylesheetExtension.WebHome'))
{{/velocity}}
{{/code}}
{{info}}**Our new skin extension will be composed of a javascript and a stylesheet extension. We will hold the two objects in the same wiki page, namely *XWiki.AddSizesExtension*.**
A document can have as many **ssx** or **jsx** object as it needs, but a skin extension is identified by the name of the document, so in the end an extension is a document. The content of a skin extension is the concatenation of the objects in that document, so it's impossible to write two different extensions in a single document, only different parts of the same extension.
{{/info}}
= Real-world eXtension**The javascript extension will be in charge of, once the document is loaded, finding all the interesting links we want to decorate** with addSizes.js =**sizes and file type icons, actually query for their size on the cloud, and finally inject this information next to each concerned link.**
Let's now go further with this idea, and build a complete extension that will dynamically add the file type and size next to certain links that are present in a wiki document. This extension will make usage of the **addSizes.js** script published by Natalie Downe. This Javascript snippet itself relies on **json-head**, a Google App Engine application by Simon Willison which //"provides a JSON-P API for running [[HEAD requests>>http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.4]] against an arbitrary URL"//. **addSizes.js** consumes this service to dynamically add the file type and size next to links in HTML documents.
And this is what we will do in our new eXtension, using the aforementioned script and service.
Our new skin extension will be composed of a javascript and a stylesheet extension. We will hold the two objects in the same wiki page, namely **XWiki.AddSizesExtension**.
Once the document is loaded the javascript extension will be in charge of finding all the interesting links we want to decorate with sizes and file type icons, actually query for their size on the cloud, and finally inject this information next to each concerned link.
The implementation below looks for the following file formats:**formats :**
Of course, this can adapted to look for other formats that are relevant for your business **:)**
==**1.1.1** Requesting and injecting files size with JSX ==**JSX**
{pre}<style>#parttwo div.code {font-size:0.8em;}</style>{/pre}
** **
AddSizes.js relies on the [[JSON**[JSON** with padding technique>>http://en.wikipedia.org/wiki/JSONP]]**technique>http://en.wikipedia.org/wiki/JSON#JSONP]** to query the **json-head****json-head*** service, which is located on a different domain than the wiki, in a transparent manner. An alternative to this would be to have a similar service on the wiki itself (for example, in the [[groovy language>>platform:DevGuide.Scripting||anchor="HGroovy"]]),**[groovy language>platform:DevGuide.Scripting#HXWiki27sGroovyAPI]),** and query it using a traditional AJAX request.
[[Prototype.js>>http://prototypejs.org/]],
[Prototype.js>http://www.prototypejs.org], the javascript framework bundled with XWiki, does not yet provide support for JSON-P requests. We will use for this a code snippet by Juriy Zaytsev**[Juriy Zaytsev>http://thinkweb2.com/projects/prototype/]** written for this purpose. Let's first paste his code in a new **JSX****JSX*** object, in

~~= XWiki.AddSizesExtension =~~**XWiki.AddSizesExtension**
~~{{code}}~~**{code}**
~~{{/code}}~~**{code}**
With this, we can now have a prototype version of addSizes.js. We can just paste this second snippet under the first one in the ~~**code****~~***code** are of our extension object, or add a new JavaScriptExtension object to the page (as SX combines all the objects of the same page into a single response):
~~{{code}}~~**{code}**
'a[href$=".pdf"], a[href$=".doc"], a[href$=".zip"], a[href$=".xls"], a[href$=".odt"], a[href$=".ods"], a[href$=".odp"],
~~a[href$=".ppt"]'~~**a[href$=".ppt"]]')**
~~var content_length = json.headers['Content-Length'];~~**if(json.ok && json.headers['Content-Length']) {**
~~if(!content_length) {~~ **var length = parseInt(json.headers['Content-Length'], 10);**
~~content_length = json.headers['content-length'];~~
~~}~~
~~if(json.ok && content_length) {~~
~~var length = parseInt(content_length, 10);~~
~~{{/code}}~~**{code}**
We will now make this information ~~look~~**looks** nicer, and add an icon to represent the file type of the link, thanks to a stylesheet extension.
~~==~~**1.1.1** Making it look nice with ~~SSX ==~~**SSX**
This time, we will take advantage of the ~~**Parse****~~***Parse** attribute of extensions that has been evoked upper in this tutorial. This way, we can be lazy and generate the CSS code using the velocity templating language, instead of writing a rule for each file format manually. Thanks to velocity and to the XWiki api, we will also be able to reference images attached to the extension document.
You can ~~[[download here>>attach:addSizesIcons.zip]]~~**download here** an archive with the set of icons used for this tutorial. The icons for MS products and for zip and pdf files are from the ~~**[[Silk~~***[Silk** Icons ~~Set>>http://www.famfamfam.com/lab/icons/silk/]]**~~**Set>http://www.famfamfam.com/lab/icons/silk/]*** by Mark James, under the ~~[[Creative~~**[Creative** Commons Attribution 2.5
~~License>>http://creativecommons.org/licenses/by/2.5/]] license. To add the icons to your extension, just unzip the archive and attach them manually to your **XWiki.AddSizesExtension** document. Of course, you can also use your own set of icons. If you change the name of the files however, keep in mind you will have to adapt the stylesheet extension below.~~**License>http://creativecommons.org/licenses/by/2.5/] licenses. To add the icons to your extension, just unzip the archive and attach them manually to your *XWiki.AddSizesExtension* document. Of course, you can also use your own set of icons. If you change the files name however, keep in mind you will have to adapt the stylesheet extension below.**
Once you have the icons attached, create the stylesheet extension, set its parse attribute to ~~**Yes**,~~***Yes*,** and ~~paste this code in the "Code" section:~~**we go:**
~~{{code language="velocity"}}~~**{code}**
#foreach($ext in ~~['odt', 'ods', 'odp', 'doc', 'xls', 'ppt', 'pdf', 'zip'])~~
["odt","ods","odp","doc","xls","ppt","pdf","zip"])
#foreach($ext in ~~['odt', 'ods', 'odp', 'doc', 'xls', 'ppt', 'pdf', 'zip'])~~
["odt","ods","odp","doc","xls","ppt","pdf","zip"])
~~{{/code}}~~**{code}**
~~==~~**1.1.1** Testing the final ~~eXtension ==~~**eXtension**
Ok, it's time for us to see the whole thing in action! The snippet below is

~~OK, it's time for us to see the whole thing in action! The snippet below is~~ intended to showcase the extension on its own wiki page. It request to the ~~**jsx**~~**jsx*** and ~~**ssx**~~**ssx*** plugins the use of the contained objects, and then give an example of all the supported links.

~~{{code}}~~**{code}**
~~{{velocity}}~~**$xwiki.jsx.use($doc.fullName)**
~~#set($discard =~~
~~$xwiki.jsx.use($doc.fullName))~~**$xwiki.ssx.use($doc.fullName)**
~~#set($discard = $xwiki.ssx.use($doc.fullName))~~
~~{{/velocity}}~~
* ~~[[An~~**<a**
href="http://pt.openoffice.org/coop/ooo2prodspeca4pt.odt">An
OpenOffice.org Writer
~~document>>http://pt.openoffice.org/coop/ooo2prodspeca4pt.odt]]~~**document**
****
* ~~[[A~~**<a**
href="http://www.microsoft.com/hiserver/techinfo/Insurance.doc">A
MS Word
~~document>>http://www.microsoft.com/hiserver/techinfo/Insurance.doc]]~~**doc**
ument
* ~~[[An~~**<a**
href="http://documentation.openoffice.org/Samples_Templates/Sampl
es/Business_planner.ods">An OOo
~~Spreadshet>>http://documentation.openoffice.org/Samples_Templates/Sam~~
~~ples/Business_planner.ods]]~~**Spreadshet**
* ~~[[Link~~**<a**
href="http://www.microsoft.com/MSFT/download/financialhistoryT4Q.
xls">Link to a MS Excel
~~document>>http://www.microsoft.com/MSFT/download/financialhistoryT4Q.~~
~~xls]]~~**document**
* ~~[[An~~**An**
OOo
~~Presentation>>http://pt.openoffice.org/coop/ooo2prodintroen.odp]]~~**Presenta**
tion
* ~~[[Link~~**<a**
href="http://research.microsoft.com/ACM97/nmNoVid.ppt">Link to a
MS Powerpoint
~~file>>http://research.microsoft.com/ACM97/nmNoVid.ppt]]~~**file**
* ~~[[A great~~
~~archive>>http://maven.xwiki.org/releases/com/xpn/xwiki/products/xwiki-~~
~~enterprise-jetty-hsqldb/2.0/xwiki-enterprise-jetty-hsqldb-2.0.zip]]~~**<a**
href="http://maven.xwiki.org/releases/com/xpn/xwiki/products/xwiki-
enterprise-hsqldb/1.6.1/xwiki-enterprise-hsqldb-1.6.1.zip">
~~* [[A PDF file>>http://www.adobe.com/motion/pdfs/sjsu_fumi_ss.pdf]]~~ **A**
great archive
~~{{/code}}~~**<a**
href="http://www.adobe.com/motion/pdfs/sjsu_fumi_ss.pdf">A PDF
file
{code}
* As Natalie Downe ~~wrote,~~
~~//"this~~**[wrote>http://natbat.net/2008/Aug/27/addSizes/], "this** may
not be 100% reliable due to App Engine being occasionally and
unavoidably ~~flakey"//.~~**flakey.".** You may for example experience a
long loading time (but since the extension triggers only once the whole wiki
document is loaded, this will not penalize the wiki users).
In a future extension of this tutorial, we will address those two issues writing
our own version of the json-head service on the wiki itself, using the
~~[[groovy~~**[groovy** programming

language>>platform:DevGuide.Scripting||anchor="HGroovy"]].language>platform:DevGuide.Scripting].
{{image reference="AddSizesMagic.png"/}}{image:AddSizesMagic.png}
==1.1.1 Bonus: links to activate/deactivate the extension ==extension
[[image:bonus.gif]]{image:bonus.gif}
You can add this snippet in the wiki content of the extension document, and users with the programming right granted will be provided a link to activate or not the extension on all pages of the wiki:wiki :
{{code}}{code}
{{velocity}}
{{html}}#if($xwiki.hasAccessLevel("programming",$context.user)) ## Only programmers should be able to change the loading type
#if($xwiki.hasAccessLevel('programming', $context.user)) ## Only programmers should be able to change the loading type
#if($doc.getObject('XWiki.JavaScriptExtension').getProperty('use').value == 'always')#if($doc.getObject("XWiki.JavaScriptExtension").getProperty("use").value=="always")
#info('This#info("This extension is configured to be loaded on all the pages of this wiki.')wiki.")
#info('This#info("This extension is configured to be loaded only on pages that request it.')it")
{{/html}}{{/velocity}}{code}
{{/code}}
= Additional Details =1.1 References
== How to use Velocity in parsed content ==* JSON with Padding : [http://en.wikipedia.org/wiki/JSON#JSONP]
Example for **XWiki.JavaScriptExtension** code:* HTTP HEAD Request : [http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.4]
{{code}}* json-head : [http://simonwillison.net/2008/Jul/29/jsonhead/]
#if (!$xcontext.userReference)* get_json.js : [http://github.com/kangax/protolicious/tree/master/get_json.js]
alert("Hello guest!");* addSizes.js : [http://natbat.net/2008/Aug/27/addSizes/]
#else
alert("Hello user!");
#end
{{/code}}
will show different alerts for different users on page refresh.
Example for **XWiki.StyleSheetExtension** code:
{{code}}
#if (!$xcontext.userReference)
#mainContentArea {
background-color: grey;
}
#else
#mainContentArea {
background-color: blue;
}
#end
{{/code}}
will show different background colors for authenticated and anonymous users.
Velocity doesn't know about CSS. Everything that looks like a Velocity macro or variable is evaluated. So, undefined macros and variables are printed as is. E.g. **#xwikicontent** will be used as CSS ID field, unless **xwikicontent** Velocity macro is defined.
Velocity is a template language, so when the Velocity code is evaluated the

~~velocity variables are simply substituted in the template by their values. So:~~
~~{{code}}~~
~~alert(false);~~
~~{{/code}}~~
~~works because **false*** is a literal boolean value in JavaScript.~~
~~Though:~~
~~{{code}}~~
~~alert($xcontext.user);~~
~~{{/code}}~~
~~will be evaluated e.g. as **alert(XWiki.User);** which will throw an exception unless **User** is a JavaScript variable previously defined, therefor you need to wrap the value in quotes, e.g.~~
~~{{code}}~~
~~alert('$xcontext.user');~~
~~{{/code}}~~
~~Or even more, because the value coming from Velocity can contain characters that are not allowed in a JavaScript string literal. So safest is to write:~~
~~{{code}}~~
~~alert('$escapetool.javascript($xcontext.user)');~~
~~{{/code}}~~
~~{{info}}~~
~~You can use XWiki global velocity variables **$doc**, **$request** and **$xcontext** in parsed content.~~
~~{{/info}}~~
~~= LESS =~~
~~Since XWiki 6.4M2, you can use [[LESS>>http://www.lesscss.org/]] in your Skin Extensions. Get a look to [[extensions:Extension.Skin Extension Plugin||anchor="HUseLESS"]] for more informations.~~
~~= References =~~
~~* [[extensions:Extension.Skin Extension Plugin]]~~
~~* [[JSON with Padding>>http://en.wikipedia.org/wiki/JSONP]]~~
~~* [[HTTP HEAD Request>>http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.4]]~~
~~*~~
~~[[get_json.js>>https://github.com/kangax/protolicious/blob/master/get_json.js]]~~
~~* [[json-head>>http://json-head.appspot.com/]]~~

AddSizesMagic.png: 1.9 ⇨ 1.5

CreateJSXObject.png: 1.6 ⇨ 1.2

JSXMagic.png: 1.6 ⇨ 1.2

MyFirstJSX.png: 1.7 ⇨ 1.3

MyFirstSSX.png: 1.6 ⇨ 1.2

SSXMagic.png: 1.8 ⇨ 1.3

XWiki.StyleSheetExtension
code:
/* Reduce the font size for code in the second part of the tutorial, since its big chunks ~~*/ #parttwo div.code { font-size:0.8em; }~~*/
#parttwo div.code {

font-size:0.8em;
}

XWiki.XWikiComments
- author:
xwiki:XWiki.schaefera
- comment:
For XWiki Syntax 2.0 the import of an on deman SSX (replace ssx with jsx for JSX extensions) file has to be wrapper with this:
~{~{velocity}}~{~{html wiki="true"}}HERE ADD THE IMPORT METNIONED ABOVE~{~{/html}}~{~{/velocity}}
The path to the extends depends on where you created the JSX / SSX file (verified for 1.8.3) and is the path to the page and not the object. For example if a JSX / SSX object is created on the Home page of the Space Test the path is: Test.WebHome.
Enjoy - Andy
- date:
2009-06-08 20:49:44.0

XWiki.XWikiComments
- author:
xwiki:XWiki.jvelociter
- comment:
Thanks Andreas,
I've updated the document for Syntax 2.0. (Note that you don't need the ~{~{html}} for the JSX include calls, as they don't output any HTML - It's all handled by the plugin and injected later in the head tag of the page)
Jerome.
- date:
2009-10-10 12:41:53.0

XWiki.XWikiComments
- author:
xwiki:XWiki.evalica
- comment:
==== If the icon is attached to YOUR page ====
You can explicitly specify the url of your icon:
{{code}}background-image:url('/xwiki/bin/download/_Space_/_YourPage_/_icon_');{{/code}}
OR use Velocity to get it
{{code}}background-image:url("$doc.getAttachmentURL('_icon_')");{{/code}}
==== If the icon is attached to ANOTHER page ====
{{code}}
#set ($adoc = $xwiki.getDocument("_Space_._AnotherPage_"))
#set ($aiconurl = $adoc.getAttachmentURL('_icon_'))
background-image:url("$aiconurl2");
{{/code}}
==== If the icon is attached to your current Skin ====
{{code}}background-image:url($xwiki.getSkinFile("_icon_"));{{/code}}
XWiki is using Silk icons as the default icon set. Use this to get the accept.gif from the set:
{{code}}background-image:url($xwiki.getSkinFile("icons/silk/accept.gif"));{{/code}}

~~In order to find more about XWiki Skins, check the [[Skins Tutorial>>http://platform.xwiki.org/xwiki/bin/view/DevGuide/Skins]].~~

- date:
~~2010-03-16 12:18:11.0~~

XWiki.XWikiComments

- author:
~~xwiki:XWiki.YangNingYuan~~
- comment:
~~?addSize.js? case was not successful. Check the ?get_json.js: http://github.com/kangax/protolicious/tree/master/get_json.js?, with {{code}}script.src = url.replace (/\?(&|$)/, '__jsonp' + id + '$ 1');{{/code}} replaced {{code}}script.src = url.replace (/?(&|$)/, '__jsonp' + id + '$ 1'); {{/code}} run successfully~~
- date:
~~2010-06-04 15:55:30.0~~

XWiki.XWikiComments

- author:
~~xwiki:XWiki.zzeroo~~
- comment:
~~It looks as if the AddSizesExtension will only work with remote resources. {{code}}http://what.ever/file.pdf{{/code}} works, but links to local attachments do not work!~~
- date:
~~2013-08-25 10:39:49.0~~

- title: Skin Extension Tutorial ⇨
- parent: xwiki:Documentation.DevGuide.WebHome ⇨ Drafts.WebHome
- author: XWiki.VincentMassol ⇨ XWiki.jvelociter
- syntax: XWiki 2.1 ⇨ XWiki 1.0

Documentation.DevGuide.Tutorials

Skins
Documentation.DevGuide.Tutorials.Skins.WebHome

On 2017/09/05, the page has been modified by Vincent Massol

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WebHome
- author: XWiki.evalica ⇨ XWiki.VincentMassol

On 2017/09/05, the page has been created by Vincent Massol

~~(% class="floatinginfobox" %)~~**{{velocity}}**
~~(((~~**$xwiki.ssx.use($doc.fullName)**
{{/velocity}}
(% class="floatinginfobox" %)(((
{{warning}}This is a draft about How to create and use Skins on the top of XWiki Enterprise{{/warning}}

= XWiki Skins ={{id name="s1" /}}
==What is a XWiki Skin used for?==
A XWiki Skin is used to customize the interface of a XWiki Enterprise instance.
{{id name="s2" /}}
==What are the components that define a XWiki Skin?==
A XWiki Skin is composed of:
* (% class="sectionblockcodecolored"%)[[Velocity templates>>platform:DevGuide.Scripting#velocity]](%%)
* (% class="sectionblockcodecolored"%)[[{{html}}<acronym title="Cascading Style Sheets">CSS</acronym>{{/html}} stylesheets>>platform:AdminGuide.CSS files]](%%)
* (% class="sectionblockcodecolored"%)[[Javascript files>>platform:DevGuide.JavaScriptAPI]](%%) and
* (% class="sectionblockcodecolored"%)Images(%%)
== How to create a new Skin == All these components are stored on the server that hosts the XWiki Enterprise instance, inside the (% class="sectionblockcodecolored directory"%)webapps/xwiki(%%) folder.
{{id name="s3" /}} ==How to change the Skin== {{id name="3a" /}}===A. The default XWiki Skins===
A XWiki Enterprise instance contains by default a [[list of Skins>>code:Skins.WebHome]] like:
* the (% class="sectionblockcodecolored"%) [[albatross>>code:Skins.AlbatrossSkin]](%%) Skin
* the (% class="sectionblockcodecolored"%) [[toucan>>code:Skins.ToucanSkin]](%%) Skin
* the (% class="sectionblockcodecolored"%) [[colibri>>code:Skins.ColibriSkin]](%%) Skin and
* other skins (the full list of XWiki Skins that are included in your XWiki Enterprise instance can be found on the server filesystem, inside the (% class="sectionblockcodecolored directory"%)webapps/xwiki/skins(%%) folder)
{{id name="3b" /}}===B. The usage level for a XWiki Skin===
A Skin can be used at different levels:
* for the entire XWiki Enterprise (% class="sectionblockcodecolored"%)instance(%%) (if there is no Skin set other levels, the pages will use the global (% class="sectionblockcodecolored"%)instance(%%) Skin)
* for a (% class="sectionblockcodecolored"%)space(%%) inside the XWiki Enterprise instance (it has a higher priority then the (% class="sectionblockcodecolored"%)instance(%%) level Skin)
* for a specific (% class="sectionblockcodecolored"%)user(%%) (it has a higher priority then the (% class="sectionblockcodecolored"%)space(%%) level Skin) and
* for other (% class="sectionblockcodecolored"%)custom(%%) levels that can be created by modifying the Velocity templates from inside the Skin (for example, a Skin can be set for a specific page, for a specific group of users and so on)
{{id name="3c" /}} ===C. Changing the Skin===
To ~~create a new~~change the Skin, ~~you~~the user must have ~~access~~administrating rights. In the case that you have these rights, make sure that you switch to the ~~server side of~~Advanced mode in your profile, to enable the ~~XWiki Enterprise instance.~~advanced options on the top main menu.
{{id name="3c1" /}}====1. How to preview a Skin applied on a page====
To check how a page looks on changing the Skin, you can add a

parameter named (% class="sectionblockcodecolored"%)skin(%%) to the page URL. For example, to preview the wiki main page on using the (% class="sectionblockcodecolored"%)toucan(%%) Skin, use the following URL: http://localhost:8080/xwiki/bin/view/Main/WebHome/?skin=toucan . If you want to preview the page on using a Skin document named (% class="sectionblockcodecolored"%)MySpace.MySkinDoc(%%), use the following URL: http://localhost:8080/xwiki/bin/view/Main/WebHome/?skin=MySpace.MySkinDoc .
{{id name="3c2" /}}====2. Changing the Skin at (% class="sectionblockcodecolored"%)instance(%%) level====
Each XWiki Enterprise instance contains a document ((% class="sectionblockcodecolored"%)Skin Document(%%)) that stores the Skin for the instance. To find out which document is used to store the current Skin, the user has to follow these steps:
|(% style="width:60%;color:#222"%)(((
1. go into the (% class="sectionblockcodecolored"%)Administration(%%) interface
1. go into the (% class="sectionblockcodecolored"%)Presentation(%%) section
1. check the (% class="sectionblockcodecolored"%)Skin(%%) property that stores the full name of the document
)))|((((image:1.png||width="100%" style="border:3px solid #ddd;"]]>>attach:1.png])))
~~To create a new Skin, you have to create a new directory inside~~**{{id name="3c2steps" /}}To change** the ~~webapps/xwiki/skins folder and then add Velocity templates, CSS stylesheets, Javascript files and images inside it. For example, if you want to remove the logo from the header of the pages,~~**current Skin,** follow these steps:
|(% style="width:60%;color:#222"%)(((
(% start="4" %)
1. click on the (% class="sectionblockcodecolored"%)CUSTOMIZE(%%) button right beside the (% class="sectionblockcodecolored"%)Skin(%%) property, to open the document
1. click on the (% class="sectionblockcodecolored"%)Edit this skin(%%) button to edit the document, or go to the top menu and select (% class="sectionblockcodecolored"%)EDIT -> OBJECTS(%%)
1. change the (% class="sectionblockcodecolored"%)Skin(%%) property (the property is set to (% class="sectionblockcodecolored"%)colibri(%%) by default) with the name of another Skin (for example, (% class="sectionblockcodecolored"%)albatross(%%))
1. save the document
)))|((((image:2.png||width="100%" style="border:3px solid #ddd;"]]>>attach:2.png])))
~~* Inside~~ **You can now open any page inside** the ~~webapps/xwiki/skins folder, create a folder named myskin. Copy all files from webapps/xwiki/skins/colibri folder into your new skin directory.~~**wiki instance and you will notice the layout difference.**
~~* Change~~**{{id name="3c3" /}}====3. Changing** the Skin at ~~global or instance level and set it as myskin (see [[this>>platform:AdminGuide.Skins]] page for more informations about how to change the Skin)~~**(% class="sectionblockcodecolored"%)space(%%) level====**
~~* Open any page in view mode and~~ **A XWiki space can use it's own Skin.**

For example, if you ~~will notice that myskin layout is~~**want to set as Skin for** the ~~same~~**space (% class="sectionblockcodecolored"%)Main(%%)** as **being** the ~~colibri layout (if the skin changes to albatross means that your theme wasn't used. Check the names of your skin directory. Note that skin names are case-sensitive)~~**(% class="sectionblockcodecolored"%)albatross(%%) Skin, follow these steps:**
~~* Change the global template:to remove the logo, modify the global.vm template under the new Skin and delete the line:(((~~**|(% style="width:60%;color:#222"%)(((**
~~{{code language="html"}}~~ **1. go on any page inside the (% class="sectionblockcodecolored"%)Main(%%) space (for example, go on the home page: http://localhost:8080/xwiki/bin/view/Main/)**
~~~~ **1. from the top right menu, choose (% class="sectionblockcodecolored"%)Administration -> Administer space: Main(%%) to open the (% class="sectionblockcodecolored"%)Administration(%%) interface for the space (% class="sectionblockcodecolored"%)Main(%%)**
~~{{/code}}~~ **1. go into the (% class="sectionblockcodecolored"%)Presentation(%%) section**
~~)))~~ **1. follow the steps 4, 5, 6 and 7 from inside the section [[C-2>>#3c2]]**
~~* Save the template~~ **)))|((([[[[image:3.png||width="100%" style="border:3px solid #ddd;"]]>>attach:3.png]])))**
~~Open~~ **You can now open** any page ~~in view mode~~**inside the (% class="sectionblockcodecolored"%)Main(%%) space** and you will notice ~~that~~ the ~~XWiki logo is not displayed anymore. Congratulations! You have created a new minimal XWiki Skin.~~**layout difference.**
)))
{{id name="3c4" /}}====4. Changing the Skin at (% class="sectionblockcodecolored"%)user(%%) level====
For example, if you want to use the (% class="sectionblockcodecolored"%)toucan(%%) Skin for a specific (% class="sectionblockcodecolored"%)user(%%), follow these steps:
1. go to the user's (% class="sectionblockcodecolored"%)profile(%%) page
1. edit the document's objects (if you are in edit mode, use the right panel and choose the (% class="sectionblockcodecolored"%)Objects(%%) editor, or, if you are in view mode, use the top-left menu: (% class="sectionblockcodecolored"%)EDIT -> OBJECTS(%%))
1. set the value of the (% class="sectionblockcodecolored"%)skin(%%) property as being (% class="sectionblockcodecolored"%)toucan(%%)
1. save the document
~~=== skin.properties ===~~ **Now, all the pages inside the instance will have the (% class="sectionblockcodecolored"%)toucan(%%) Skin applied on them for that user.**
~~{{info}}Since 6.4.{{/info}} Inside your new skin, you should have a file called ##skin.properties##. This file contains some properties about the skin and looks like this:~~
~~{{code language="cfg"}}~~**{{id name="s4" /}}==How to create a new Skin==**
~~parent=...~~
~~outputSyntax=...~~
~~{{/code}}~~
~~|=Property|=Description~~
~~|##parent##|Indicate another skin to inherit from. If it is set to an empty~~

~~value (##parent=##) the skin will directly inherit from the WAR. If it is not set at all it will inherit from whatever is configured in ##xwiki.cfg##'s ##xwiki.defaultbaseskin## property.~~
~~|##outputSyntax##|{{info}}Since 7.1M1.{{/info}} Indicate the output syntax the skin is using. XWiki will use the correct renderer to generate valid HTML code. If it is set to an empty value (##outputSyntax=##) or not set at all, this setting will be inherited from the parent skin. If there is no parent, the syntax used is XHTML1.0 (for compatibility with old skins that did not set this property). Supported values are ##html/5.0## and ##xhtml/1.0##.~~
~~== How to override a Skin ==~~
To ~~change the~~**create a new** Skin, ~~the user~~**you** must have ~~administrating rights. In the case that you have these rights, make sure that you switch~~**access** to the ~~Advanced mode in your profile, to enable~~**server side of** the ~~advanced options on the top main menu.~~**XWiki Enterprise instance.**
To create a new Skin, you have to create a new direcory inside the (% class="sectionblockcodecolored directory"%)webapps/xwiki/skins(%%) folder and then add Velocity templates, {{html}}<acronym title="Cascading Style Sheets">CSS</acronym>{{/html}} stylesheets, Javascript files and images inside it. For example, if you want to remove the logo from the header of the pages, follow these steps:
1. inside the (% class="sectionblockcodecolored directory"%)webapps/xwiki/skins(%%) folder, create a folder named (% class="sectionblockcodecolored"%)myskin(%%)
1. change the Skin at (% class="sectionblockcodecolored"%)instance(%%) level and set it as (% class="sectionblockcodecolored"%)myskin(%%) (see the section [[C-2>>#3c2]] for more informations about how to change the Skin)
1. open any page in view mode and you will notice that (% class="sectionblockcodecolored"%)myskin(%%) layout is the same as the (% class="sectionblockcodecolored"%)albatross(%%) layout (this happens because on having no Skin set as the Skin, the default Skin is considered to be (% class="sectionblockcodecolored"%)albatross(%%))
1. change the (% class="sectionblockcodecolored"%)global(%%) template:
1*. copy the (% class="sectionblockcodecolored directory"%)webapps/templates/global.vm(%%) template and paste it inside the (% class="sectionblockcodecolored"%)myskin(%%) folder
1*. to remove the logo, modify the (% class="sectionblockcodecolored"%)global(%%) template under the new Skin and delete the line:(% class="overflow-auto" %)((({{code language="xml"}}{{/code}})))
1. save the template
Open any page in view mode and you will notice that the XWiki logo is not displayed anymore.
Congratulations! You have created a new minimal XWiki Skin.
~~An alternative for creating a new Skin is the possibility~~**{{id name="s5" /}}==How** to override ~~an existing Skin. The advantage of overriding an existing Skin is the fact that you don't need access to the server filesystem in order to modify the files used by the current Skin.~~**a Skin==**
~~To override a Skin, you have to use the same document that is used to store the Skin.\\~~
~~=== A. Creating a Skin document ===~~
~~You need to~~**To change the Skin, the user must** have **administrating rights. In** the ~~**advanced edit mode** activated~~**case that you have these rights, make sure that you switch** to ~~access~~ the ~~object editor. Please~~

~~refer~~Advanced mode in your profile, to ~~[[platform:Features.PageEditing]] for more information.~~enable the advanced options on the top main menu.
An alternative for creating a new Skin is the possibility to override an existing Skin. The advantage of overriding an existing Skin is the fact that you don't need access to the server filesystem in order to modify the files used by the current Skin.
To override a Skin, you have to use the same document that is used to store the Skin.
{{id name="5a" /}}===A. Creating a Skin document===
{{warning}}You need to have the **advance edition mode** activated to access the object editor. Please refer to [[platform:Features.PageEditing]] for more information.{{/warning}}
|(% style="width:60%;color:#222"%)(((
1. create a new page - if there is the option (% class="sectionblockcodecolored"%)CREATE -> PAGE(%%) on the main menu, use it to create a page, or, if there is no such option, add the (% class="sectionblockcodecolored"%)New Page(%%) panel on the interface by following these steps:
1*. go to (% class="sectionblockcodecolored"%)Administration -> Panel Wizard(%%)
1*. select the (% class="sectionblockcodecolored"%)Tools(%%) category
1*. add the (% class="sectionblockcodecolored"%)New Page(%%) panel on the interface by using drag&drop
1*. click on (% class="sectionblockcodecolored"%)Save the new layout(%%) to save your changes
1*. open any page in view mode and
1*. use the (% class="sectionblockcodecolored"%)New Page(%%) panel to create a page
1. edit the document's objects (if you are in (% class="sectionblockcodecolored"%)edit(%%) mode, use the right panel and choose the (% class="sectionblockcodecolored"%)Objects(%%) editor, or, if you are in (% class="sectionblockcodecolored"%)view(%%) mode, use the top-left menu: (% class="sectionblockcodecolored"%)EDIT -> OBJECTS(%%))
1. add an object of type (% class="sectionblockcodecolored"%)XWiki.XWikiSkins(%%) to the document (use the **ADD OBJECT** panel from the right and select the (% class="sectionblockcodecolored"%)XWiki.XWikiSkins(%%) class, then click on (% class="sectionblockcodecolored"%)ADD OBJECT FROM THIS CLASS(%%))
1. save the document
)))|((([[[[image:4.png||width="100%" style="border:3px solid #ddd;"]]>>attach:4.png]])))
~~* Create a new page. Name it MySkinPage(((~~ {{id name="5b" /}}===B. Working with the (% class="sectionblockcodecolored"%)XWiki.XWikiSkins(%%) class===
~~[[image:createSkinPage.png||style="border: 1px solid black; width: 650px;"]]~~
The base element in overriding a Skin is the (% class="sectionblockcodecolored"%)XWiki.XWikiSkins(%%) class (http://localhost:8080/xwiki/bin/view/XWiki/XWikiSkins). In order to modify the templates or the files used by a Skin, you have to work with the properties of this class.
~~)))~~ By default, the Skin class contains a set of properties that allows you to override some of the templates and files used by the Skin, and

it also allows you to set the name of the logo displayed in the header of the interface.
~~* Edit~~ To override other templates or files, you have to add new properties to the ~~document's objects (Edit -> Objects)~~Skin class. For example, if you want to override (% class="sectionblockcodecolored"%)myfile.vm(%%) (or myfile.css, or myfile.js), follow these steps:
~~* Add an object of type XWiki.XWikiSkins to the document~~|(% style="width:60%;color:#222"%)(((
~~* Click on add~~ 1. go to the (% class="sectionblockcodecolored"%)XWiki.XWikiSkins(%%) document
~~* Set~~ 1. click on the ~~base skin property. The base skin property points to the skin you wish to override. You can use either the filesystem skin directory name (for a filesystem skin to override)~~(% class="sectionblockcodecolored"%)add or modify the ~~page name where the skin is defined (for overriding a page based skin). Set it to ##colibri## for using the ##colibri## skin as a base skin. As you can see, no files were copied on the file system, but your skin uses colibri.(((~~class properties(%%) link ~~[[image:addSkinObject.png||style="border: 1px solid black; width: 650px;"]]~~ (or follow this link: http://localhost:8080/xwiki/bin/edit/XWiki/XWikiSkins?editor=class) ~~)))~~ 1. from the **ADD PROPERTY** panel in the right, set the name of the property as being (% class="sectionblockcodecolored"%)myfile.vm(%%), choose the (% class="sectionblockcodecolored"%)TextArea(%%) type, and click on (% class="sectionblockcodecolored"%)Add(%%)
~~* Save~~ 1. save the document
~~)))|((([[[[image:5.png||width="100%" style="border:3px solid #ddd;"]]>>attach:5.png]])))~~
~~=== B. Working with the XWiki.XWikiSkins class ===~~ {{id name="5c" /}}===C. Using a Skin document===
~~{{warning}}~~
~~Since 7.0RC1, this mechanism is deprecated. You don't need to edit the XWiki.XWikiSkins class anymore. You should simply use the [[extensions:Extension.Skin Application]]).~~
~~{{/warning}}~~
~~The base element in overriding a Skin is the XWiki.XWikiSkins class (http:~/~/localhost:8080/xwiki/bin/view/XWiki/XWikiSkins). In order to modify the templates or the files used by a Skin, you have to work with the properties of this class. By default, the Skin class contains a set of properties that allows you to override some of the templates and files used by the Skin, and it also allows you to set the name of the logo displayed in the header of the interface.~~
~~To override other templates or files, you have to add new properties to the Skin class. For example, if you want to override myfile.vm (or myfile.css, or myfile.js), follow these steps:~~
~~* Go to the XWiki.XWikiSkins document~~
~~* Edit the document's class (Edit -> Class) or follow this link: http:~/~/localhost:8080/xwiki/bin/edit/XWiki/XWikiSkins?editor=class~~
~~* Add a new property myfile.vm as TextArea type~~
~~* Click on Add. The myfile.vm property should appear in the list(((~~
~~[[image:editXWikiSkins.png||style="border: 1px solid black; width: 650px;"]]~~
~~)))~~
~~* Save the document~~
~~=== C. Using a Skin document ===~~
1. go to the (% class="sectionblockcodecolored"%)Skin(%%) property

from inside the (% class="sectionblockcodecolored"%)Administration(%%) interface
1. set the value of the property as being the full name of the Skin document
1. save the preferences
{{id name="5d" /}}===D. Overriding the Skin components===
Let's add a "HELLO," before the username on the top right menu, not by modifying the (% class="sectionblockcodecolored"%)global.vm(%%) template on the server, but overriding it inside the Skin document. Follow these steps:
|(% style="width:60%;color:#222;overflow-x:auto;"%)(((
1. add a (% class="sectionblockcodecolored"%)TextArea(%%) property named (% class="sectionblockcodecolored"%)global.vm(%%) to the (% class="sectionblockcodecolored"%)XWiki.XWikiSkins(%%) class
1. edit the Skin document's objects (use the top-left menu: (% class="sectionblockcodecolored"%)EDIT -> OBJECTS(%%)) and open the properties of the (% class="sectionblockcodecolored"%)XWiki.XWikiSkins(%%) object
1. copy the original content of the (% class="sectionblockcodecolored"%)global.vm(%%) template and paste it in the property field named (% class="sectionblockcodecolored"%)global.vm(%%); if you don't have access to the server filesystem, open the template by following one of these links:
1*. http://localhost:8080/xwiki/skins/mybaseskin/global.vm (where mybaseskin is the current Skin used in the Skin document), or
1*. if the previous link is not valid, go to http://localhost:8080/xwiki/templates/global.vm
1. replace the following code: (% style="overflow-x:auto;" %)((({{code language="xml"}}
$!xwiki.getUserName($context.user, false)
{{/code}})))with (% style="overflow-x:auto;" %)((({{code language="xml"}}
HELLO,
$!xwiki.getUserName($context.user, false)
{{/code}})))
1. save the document
)))|((([[[[image:6.png||width="100%" style="border:3px solid #ddd;"]]>>attach:6.png]])))
~~* Go to~~ **Now, open any document in view mode and you will notice** the ~~skin property from inside the Administrationpage, Presentation section~~**nice "HELLO," before your username.**
~~* Set~~ **Follow** the ~~value of the~~**same process in order to override (% class="sectionblockcodecolored"%)Javascript(%%) and (% class="sectionblockcodecolored"%){{html}}<acronym title="Cascading Style Sheets">CSS</acronym>{{/html}}(%%)** files ~~as being the full name of the Skin document. For our skin page created above, we will type Main.MySkinPage~~**(http://localhost:8080/xwiki/skins/colibri/myfile.css).**
~~* Save the preferences~~
~~=== D. Overriding the Skin~~ **{{id name="5e" /}}===E. Adding new** components ~~===~~**to the Skin===**
If you want to add new files to the skin, like (%

~~if you want to add new files to the skin, like (%~~ class="sectionblockcodecolored"%){{html}}<acronym title="Cascading Style Sheets">CSS</acronym>{{/html}} (%%) or (% class="sectionblockcodecolored"%)Javascript(%%) files, you have two options:
* attach the files to the Skin document and reffer them by their URL, or
* add new properties to the (% class="sectionblockcodecolored"%)XWiki.XWikiSkins(%%) object from inside the Skin document, and refer them by their name
~~Let's~~ For example, if you want to add a ~~"Hello" before~~new {{html}}<acronym title="Cascading Style Sheets">CSS</acronym>{{/html}} file to the ~~username on the top right menu, not by modifying the menuview.vm template on the server, but overriding it inside the Skin document.~~
~~Follow~~Skin, named (% class="sectionblockcodecolored"%)mynewfile.css(%%), follow one of these set of steps:
~~==== Method 1: By editing the XWiki.XWikiSkins properties on the Skin document ====~~|(% style="width:60%;color:#222"%)(((
~~{{warning}}~~ 1. attach the {{html}}<acronym title="Cascading Style Sheets">CSS</acronym>{{/html}} file to the Skin document
~~Since 7.0RC1, this method is deprecated. You don't need~~ 1. override the stylesheets.vm template (check the section [[D>>#5d]] to ~~edit~~learn how to override a Skin component), and add the ~~XWiki.XWikiSkins class anymore. You should simply use~~reference to your file in the ~~[[extensions:Extension.Skin Application]]).~~template:(% class="overflow-auto" %)((({{code language="xml"}}<link rel="stylesheet"
~~{{/warning}}~~
~~* Add a TextArea property named menuview.vm to the XWiki.XWikiSkins class(((~~
~~[[image:XWikiSkinsMenuView.png||style="border: 1px solid black; width: 650px;"]]~~
~~)))~~
~~* Edit your skin document's objects (Edit -> Objects) and open the properties of the object~~
~~* Copy the original content of the menuview.vm template and paste it in the property field named menuview.vm. If you don't have access to the server filesystem, open the template by following http:~/~/localhost:8080/xwiki/templates/menuview.vm (to access a template from the skin, use http:~/~/localhost:8080/xwiki/skins/colibri/)~~
~~* Replace the following code:(((~~
~~{{code language="html"}}~~
~~#xwikitopmenuentrystart($xwiki.getURL($xcontext.user, 'view') "$msg.get('core.menu.type.profile')__SEPARATOR__$!xwiki.getUserName($xcontext.user, false)" 'tmUser' 'hasIcon')~~
~~{{/code}}~~
~~with~~
~~{{code language="html"}}~~
~~#xwikitopmenuentrystart($xwiki.getURL($xcontext.user, 'view') "$msg.get('Hello')__SEPARATOR__$!xwiki.getUserName($xcontext.user, false)" 'tmUser' 'hasIcon'){{/code}}~~
~~[[image:MySkinPageWithMenuView.png||style="border: 1px solid black; width: 650px;"]]~~
~~)))~~
~~* Save the document~~
~~==== Method 2: By attaching files to the current Skin document ====~~
~~* Create a file on your computer called menuview.vm. Copy the original content of the menuview.vm into your local file. If you don't have access to the server filesystem, open the template by following~~

http:~/~/localhost:8080/xwiki/skins/colibri/menuview.vm
* Replace the following code:(((
{{code language="html"}}
#xwikitopmenuentrystart($xwiki.getURL($xcontext.user, 'view')
"$msg.get('core.menu.type.profile')__SEPARATOR__$!xwiki.getUserName(
$xcontext.user, false)" 'tmUser' 'hasIcon')
{{/code}}
with
{{code language="html"}}
#xwikitopmenuentrystart($xwiki.getURL($xcontext.user, 'view')
"$msg.get('Hello')__SEPARATOR__$!xwiki.getUserName($xcontext.user,
false)" 'tmUser' 'hasIcon')
{{/code}}
)))
* Attach the copy of menuview.vm template to the current Skin document
page
Now, open any document in view mode and you will notice the nice "Hello"
before your username. Follow the same process in order to override
Javascript and CSS files (
http:~/~/localhost:8080/xwiki/skins/colibri/myfile.css).
==== 3. Overriding a skin resource ====
Just like templates, resource files such as images or css/js files can be
overridden. To do that, the replacement must be attached to the Skin
document, making sure it complies with the naming convention explained
below. To find out the right name for your replacement resource, here is the
trick
* Find the path on the filesystem of the resource to replace (you can easily
obtain that information using development tools such as
[[Firebug>>http://getfirebug.com/]]); for instance, let's say this path is either
{{code}}/xwiki/resources/icons/silk/myicon.gif{{/code}}
for a resource found in the general resource directry, or
{{code}}/xwiki/skins/colibri/myimage.png{{/code}}
if the resource is specific to a certain skin, in this case 'colibri'.
* Strip from this path everything from the beginning until 'resources/' or the
name of the skin, included; in the previous examples, you will obtain in the
first case{{code}}icons/silk/myicon.gif{{/code}}
and in the second case simply
{{code}}myimage.png{{/code}}
* In this path, replace '/' with '.'; after this operation, we will get
{{code}}icons.silk.myicon.gif{{/code}} for the first image example, while the
second one remains unchanged. **This is the name of the file
corresponding to the resource that you need to attach to the Skin document
in order to make the replacement**.
* Create a file named as explained above and attach it to your Skin
{{info}}
Technical note: This trick works for any resource whose URL is
generated in the skin by the {{code}}$xwiki.getSkinFile(){{/code}},
{{code}}$xwiki.jsfx(){{/code}} or {{code}}$xwiki.ssfx(){{/code}} methods, for
instance like this:
{{code language="none"}}
.infomessage {
background-image: url("$xwiki.getSkinFile('icons/silk/information.gif')");
}
{{/code}}
In this case, {{code}}$xwiki.getSkinFile('icons/silk/information.gif'){{/code}}
will first search for an image called 'icons.silk.information.gif' attached to the
configured Skin document, if any. If that image exists, it will generate its

URL. Otherwise, it will continue to search for 'icons/silk/information.gif' in the base skins and finally in the general 'resources/' directory.
{{/info}}
=== E. Adding new components to the Skin ===
If you want to add new files to the skin, like CSS or Javascript files, you have two options:
* attach the files to the Skin document and reffer them by their URL, or
* add new properties to the (% class="sectionblockcodecolored" %)XWiki.XWikiSkins(%%) object from inside the Skin document, and refer them by their name
For example, if you want to add a new {{html}}<acronym title="Cascading Style Sheets">CSS</acronym>{{/html}} file to the Skin, named (% class="sectionblockcodecolored" %)mynewfile.css(%%), follow one of these set of steps:
* Attach the CSS file to the Skin document
* Override the stylesheets.vm template and add the reference to your file in the template:(((
{{code language="html"}}
<link rel="stylesheet"
/>/>{{/code}})))
{{/code}} 1. save the document
)))))))|((([[[[image:7.png||width="100%" style="border:3px solid #ddd;"]]>>attach:7.png]])))
* Save the document
(((OR
OR|(% style="width:60%;color:#222"%)(((
))) 1. add a new property to the (% class="sectionblockcodecolored"%)XWiki.XWikiSkins(%%), name it (% class="sectionblockcodecolored"%)mynewfile.css(%%) (check the section [[B>>#5b]] for more information about how to work with this class), edit the Skin document's objects (use the top-left menu: (% class="sectionblockcodecolored"%)EDIT -> OBJECTS(%%)), and paste the {{html}}<acronym title="Cascading Style Sheets">CSS</acronym>{{/html}} code into the (% class="sectionblockcodecolored"%)mynewfile.css(%%) field
1. override the (% class="sectionblockcodecolored"%)style.css(%%) component (see the section [[D>>#5d]] for more information about how to override a Skin compoment) and add the following line to the content of the (% class="sectionblockcodecolored"%)style.css(%%) property:(% class="overflow-auto" %)((({{code language="css"}}@import "mynewfile.css";{{/code}}))) the important thing is to add a reference to the new component from a existing one; you must know most of the components content, so that you will take the best decision about which existing component will contain a reference to the new one
1. save the document
))))|((([[[[image:8.png||width="100%" style="border:3px solid #ddd;"]]>>attach:8.png]])))
(((Open any document in view mode and you will notice that the stylesheet is now applied.
* Add a new property to the XWiki.XWikiSkins, name it mynewfile.css, edit the Skin document's objects (Edit -> Objects), and paste the CSS code into the mynewfile.css field
* Override the Style component and add the following line to the content of the Style property:(((
{{code language="css"}}
@import "mynewfile.css";
{{/code}}

{{/code}}
The important thing is to add a reference to the new component from a existing one; you must know most of the components content, so that you will take the best decision about which existing component will contain a reference to the new one .
[[image:importCSS.png||style="border: 1px solid black; width: 650px;"]]
)))
* Save the document
Open any document in view mode and you will notice that the stylesheet is now applied.
{{id name="5f" /}}===F. Using (% class="sectionblockcodecolored"%)images(%%) in a Skin document===
{{id name="5f1" /}}====1. Changing the (% class="sectionblockcodecolored"%)logo(%%)====
For example, if you want to add an image named (% class="sectionblockcodecolored"%)myimage.jpg(%%) as being the logo for the current Skin, follow these steps:
1. attach the image to the Skin document
1. edit the document's objects (use the top-left menu: (% class="sectionblockcodecolored"%)EDIT -> OBJECTS(%%))
1. use the (% class="sectionblockcodecolored"%)XWiki.XWikiSkins(%%) object and set the (% class="sectionblockcodecolored"%)Logo(%%) property as being the name of the attached image: (% class="sectionblockcodecolored"%)myimage.jpg(%%)
1. save the document
=== F. Using images **Open any page** in a Skin document ===**view mode and you will notice that the logo is now changed.**
==== 1. Changing the logo ==== **{{id name="5f2" /}}**
For example, if you want **====2. Adding images** to add an image named myimage.jpg as being the logo for the current Skin, follow these steps:**Skin interface====**
* Attach **To add images to** the image**interface, attach them** to the Skin document**document and use them in the {{html}}<acronym title="Cascading Style Sheets">CSS</acronym>{{/html}} code. For example, if you want to use (% class="sectionblockcodecolored"%)myimage.jpg(%%) as background image for the pages inside your XWiki Enterprise instance, follow these steps:**
* Edit **1. attach** the document's objects**image to the Skin document**
* Use the XWiki.XWikiSkins object and set the Logo property as being the name **1. choose one** of the attached image myimage.jpg**these actions:**
* Save **1*. add a new {{html}}<acronym title="Cascading Style Sheets">CSS</acronym>{{/html}} component to** the document**Skin, name it (% class="sectionblockcodecolored"%)pagebackground.css(%%) and add a reference to it from the (% class="sectionblockcodecolored"%)style.css(%%) component (see the section [[E>>#5e]] for more information about how to add new components to the Skin)**
Open any page in view mode and you will notice that **1*. override** the logo**(% class="sectionblockcodecolored"%)mybaseskin.css(%%) component (where mybaseskin** is now changed.**the name of the current Skin - for example: (% class="sectionblockcodecolored"%)albatross.css(%%), (% class="sectionblockcodecolored"%)toucan.css(%%) or (% class="sectionblockcodecolored"%)colibri.css(%%))**
==== 2. Adding images to **1. paste** the Skin interface ====**following code**

inside the {{html}}<acronym title="Cascading Style Sheets">CSS</acronym>{{/html}} component:(% class="codeblock overflow-auto addcode" %)((({{code language="css"}}body{
To add images to the interface, attach them to the Skin document and use them in the CSS code. For example, if you want to use myimage.jpg as background image for the pages inside your XWiki Enterprise instance, follow these steps:
* Attach the image to the Skin document
* Choose one of these actions:
** Add a new CSS component to the Skin, name it pagebackground.css and add a reference to it from the style.css component
** Override the mybaseskin.css component (where mybaseskin is the name of the current Skin - for example: albatross.css, toucan.css or colibri.css)
* Paste the following code inside the CSS component:(((
{{code language="css"}}
body{
}{{/code}})))
{{/code}}
)))
Open **Open** any page in view mode and you will notice that the the background of the page contains your image. Have fun taking advantage of the power of XWiki Skins!
Have fun taking advantage of the power of XWiki Skins!
== More about XWiki Skins ==**==Further reading==**
In order to find more about XWiki Skins, check the [[Skins Tutorial>>platform:AdminGuide.Skins]] from inside the Admin Guide and the [[Skins Tutorial>>platform:Features.Skins]] from inside the User Guide * **the [[XWiki Data Model>>http://platform.xwiki.org/xwiki/bin/view/DevGuide/DataModel]]**
== Further reading == * **using the [[Skin Editor Application>>http://code.xwiki.org/xwiki/bin/view/Applications/SkinEditorApplication]]**
* the [[XWiki Data Model>>http://platform.xwiki.org/xwiki/bin/view/DevGuide/DataModel]] * **[[edit only one object>>http://dev.xwiki.org/xwiki/bin/view/Drafts/EditOneObject]]**
* using the [[Skin Editor Application>>extensions:Extension.Skin Editor Application]]
* [[edit only one object>>http://dev.xwiki.org/xwiki/bin/view/Drafts/EditOneObject]]
)))

1.png: 1.1 ⇨ ()

2.png: 1.1 ⇨ ()

3.png: 1.1 ⇨ ()

4.png: 1.1 ⇨ ()

5.png: 1.1 ⇨ ()

6.png: 1.1 ⇨ ()

7.png: 1.1 ⇨ ()

8.png: 1.1 ⇨ ()

MySkinPageWithMenuView.png: 1.4 ⇨ ()

XWikiSkinsMenuView.png: 1.2 ⇨ ()

addSkinObject.png: 1.2 ⇨ ()

createSkinPage.png: 1.1 ⇨ ()

editXWikiSkins.png: 1.4 ⇨ ()

importCSS.png: 1.3 ⇨ ()

XWiki.XWikiComments
- author:
~~xwiki:XWiki.Supreet~~
- comment:
~~The images of the menus need to be updated as the new UI is different.~~
- date:
~~2011-03-28 21:22:02.0~~

- parent: xwiki:Documentation.DevGuide.WebHome ⇨
- author: XWiki.VincentMassol ⇨ XWiki.RalucaMorosan

Documentation.DevGuide.Tutorials

Tutorial for Internationalizing a Form
Documentation.DevGuide.Tutorials.TranslationsTutorial.WebHome

On 2017/09/05, the page has been created by Vincent Massol

~~{{box cssClass="floatinginfobox" title="**Contents**"}}~~ **There are multiple ways you can integrate translation in XWiki. If you are a web developer which builds custom applications on top of XWiki, you certainly tried to create pages that contain multiple translations.**
~~{{toc/}}~~
~~{{/box}}~~
~~There are multiple ways you can integrate translation in XWiki. If you are~~**In order to facilitate these tasks, here is** a ~~web developer~~**tutorial** which ~~builds custom applications on top of XWiki, you certainly tried to create pages that contain multiple translations.~~**uses the best practices for achieving your goal.**
~~In order to facilitate these tasks, here is a tutorial which uses the best practices for achieving the following goals:~~**{{box}}**
This tutorial is divided in two sections. First section shows users how to create properly a small inline form.
The second section shows you how to add translations in Romanian language to that form in the proper and recommended way. If you are interested only on the translations tutorial, you can skip the first section.
{{/box}}
~~* The first section shows how to create properly a small inline form.~~
~~* The second section shows you how to add translations in Romanian~~

~~The second section shows you how to add translations in Romanian language to that form in the proper and recommended way. If you are interested only on the translations tutorial, you can skip the first section.~~
= Step ~~1: Creating the XWiki Class~~**1** =
* Create a page called ~~**MyCoolClass.** This~~**MyCoolClass.**(% **style="font-weight: bold;" %) (%%)This** will hold the definition of the class.
* Add the following ~~fields:~~**fields :**
* edit the document in wiki mode and add the following ~~line:(((~~**line:**
~~)))~~
~~* Click on the "Save & View" button~~
~~= Step 2: Creating~~*** Click on** the ~~XWiki Sheet =~~**Save & View Button**
= Step 2 =
* Edit the page in wiki mode and add the ~~following(((~~**following**
~~)))~~
~~* Add a new class of type ##XWiki.SheetClass## and set its only property to "inline".~~
~~* Click on the "Save & View" button~~
~~= Step 3: Creating the XWiki Template =~~*** Add a new class of type XWiki.SheetClass and set its only proprety to "inline".**
*** Click on the Save & View Button**
= Step 3 =
* Edit the page in wiki mode and add the following ~~line:(((~~**line:**
~~)))~~
~~* Click on the "Save & View" button~~
~~= Step 4: Internationalizing~~*** Click on** the ~~display =~~**Save & View Button**
~~Before being able to add translations, XWiki must be set to support multiple languages:~~**= Step 4 =**
~~* Go~~**Before** to **be able to add translations,** the ~~administration section of the wiki~~**XWiki must be set to support multiple languages.**
~~** Click on "Localization"~~*** Go to administration section of the wiki,**
** Set ~~Multilingual~~**Multi Lingual** to yes
** in the ~~"Supported Languages" field~~**Languages field,** add the languages you want your wiki to ~~support in this format: "fr, en, ro" (without the quotes)~~**support**
~~**~~***** Click the ~~"Save"~~**Save** button
* Create a page called ~~**MyCoolTranslations**;~~
~~this~~**MyCoolTranslations**. This** page will hold the translations to other languages other than the default one.
* Open the page in Wiki mode, and on the Document Translations panel from the right side of the page, click the ~~"ro" button; now~~**translate this document in Ro button. Now** we will add the translation for the Romanian language.
* Insert this text into the ~~content:(((~~**content**
~~{{code}}~~**(% class="macro-output" style="display: block;" %)**
(((
(% class="box code" %)
(((
~~Main.MyCoolClass_favColor_red=Rosu~~**\\Main.MyCoolClass_favColor_red=Rosu**
~~{{/code}}~~
~~* The rule for creating translation keys is the following:~~
~~** For class proprieties : spacename.classname + _ + property name~~
~~** For static lists and boolean properties : spacename.classname + _ + property name + _ + property value(((~~
~~Note that for boolean properties the possible values are 0 (=false), 1 (=true) and 2 (=undecided). E.g. SomeSpace.SomeClass_enabled_0=Disabled~~
~~* Click on the "Save & View" button~~

~~* To test your translation, append **?language=ro** to your URL. For a better comparison you can see the English and the Romanian versions below.~~
~~* For example, if your are using your xwiki on localhost use the following URLs:~~
~~** English version: http://localhost:8080/xwiki/bin/view/Main/MyCoolTemplate or http://localhost:8080/xwiki/bin/view/Main/MyCoolTemplate?language=en~~
~~** Romanian version: http://localhost:8080/xwiki/bin/view/Main/MyCoolTemplate?language=ro~~
* ~~English Version:~~**The rule for creating translations key is the following**
**** For class proprieties : spacename.classname + _ + property name**
**** For static lists : spacename.classname + _ + property name + _ + property value**
**** Click on the Save & View Button**
**** To test your translation, append to your URL **?language=ro**. For a better comparison you can see below the English version, and the Romanian version.**
**** For example, if your are using your xwiki on localhost use the following URL's:**
**** English Version: http:~/~/localhost:8080/xwiki/bin/view/Main/MyCoolTemplate or http:~/~/localhost:8080/xwiki/bin/view/Main/MyCoolTemplate?language=en**
**** Romanian versions: http:~/~/localhost:8080/xwiki/bin/view/Main/MyCoolTemplate?language=ro**
**** English Version.**
* Romanian ~~Version:~~**Version.**
~~[[image:MyCoolTemplateROView.png]]~~
~~(((~~**[[image:MyCoolTemplateROView.png]]**
~~Only~~**Note that only** the Translation document needs to have translations. The class, sheet, template and instance pages don't need to be ~~translated;~~**translated,** the localized messages will automatically be displayed in the selected language.
~~)))~~

MyCoolTemplateENView.png: 1.2 ⇨ ()

MyCoolTemplateROView.png: 1.2 ⇨ ()

MyCoolTranslationsAddTranslations.png: 1.2 ⇨ ()

addFavColorProprety.png: 1.2 ⇨ ()

addNameProprety.png: 1.2 ⇨ ()

addStreetProprety.png: 1.2 ⇨ ()

- title: Tutorial for Internationalizing a Form ⇨ TranslationsTutorial
- author: XWiki.VincentMassol ⇨ XWiki.Sorin

On 2017/09/05, the page has been modified by Vincent Massol

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WebHome

Documentation.DevGuide.Tutorials

Extension Point Tutorial

Documentation.DevGuide.Tutorials.UIXTutorial.WebHome

On 2017/09/05, the page has been modified by Vincent Massol

parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WebHome

On 2017/09/05, the page has been created by Vincent Massol

~~{{box cssClass="floatinginfobox" title="**Contents**"}}~~
~~{{toc start='2'/}}~~
~~{{/box}}~~
~~== Introduction to Interface Extensions and Extension Points ==~~
~~User Interface Extensions (abbreviated as UIX) are used in order to provide a way to alter the content of existing interface elements. This functionality was added in version 4.2 and is documented in the [[UI Extension Module>>extensions:Extension.UIExtension Module]].~~
~~The main use case Interface Extensions try to fix is the need for applications (like Blog, Watchlist, etc.) to insert custom content in already existing interface components (like panels, menus, layout, etc.).~~
~~Let's take an example: We developed an application called 'Hello World' and we want to provide a link to it inside the 'Applications' panel.~~
~~There are two questions that need to be answered:~~
~~* where we insert? - this is the Extension Point (UIXP)~~
~~* what we insert? - this is the UI Extension (UIX)~~
~~=== About Extension Points ===~~
~~Extensions Points (UIXP) specify where the new content is about to be inserted. Is like a hoof defined in the interface where Interface Extensions are gathered.~~
~~For our example, we need to know the extension point ID for the 'Applications' panel. That is '##org.xwiki.platform.panels.Applications##'.~~
~~There is a list of [[available extension points>>platform:ExtensionPoint.WebHome]] where we can add functionality, but we can also manually define new extension points.~~
~~=== About Interface Extensions ===~~
~~All the UIXs provided for a given Extension Point are displayed in that location.~~
~~For our example, if we want to add a new link in the 'Applications' panel, we need to create a new UIX that uses the '##org.xwiki.platform.panels.Applications##' extension point. UIXs are stored as standard XObjects, instances of ##XWiki.UIExtensionClass##.~~
~~For our UIX we will need to provide the label, target and icon parameters, in order to be properly displayed in the panel.~~
~~Read the documentation on [[how to add an UIX>>extensions:Extension.UIExtension Module||anchor="HWritingasimpleUIExtension"]] inside the 'Applications' panel.~~
~~== Adding your own Extension Point ==~~
~~Your UIExtension can define its own extension points, where other extensions can plug in. Here is an example of velocity code to include in your extension in order to make it an entry point for others:~~
~~* Without parameters:(((~~
~~{{code language="velocity"}}~~
~~{{velocity}}~~

~~{{velocity}}~~
~~#foreach ($extension in~~
~~$services.uix.getExtensions("my.new.extension.point"))~~
~~$services.rendering.render($extension.execute(), 'xhtml/1.0')~~
~~#end~~
~~{{/velocity}}~~
~~{{/code}}~~
~~)))~~
~~* Display the parameters:(((~~
~~{{code language="velocity"}}~~
~~{{velocity}}~~
~~#set ($extensions = $services.uix.getExtensions('my.new.extension.point',~~
~~{'sortById' : ''}))~~
~~#foreach ($extension in $extensions)~~
~~$extension.parameters.label~~
~~#end~~
~~{{/velocity}}~~
~~{{/code}}~~
~~)))~~
~~* Test the parameter for a particular value:(((~~
~~{{code language="velocity"}}~~
~~{{velocity}}~~
~~#foreach ($extension in~~
~~$services.uix.getExtensions("my.new.extension.point"))~~
~~#if ($extension.getParameters().get('parameter_name') ==~~
~~'expected_value')~~
~~{{html clean=false}}$services.rendering.render($extension.execute(),~~
~~'xhtml/1.0'){{/html}}~~
~~#end~~
~~#end~~
~~{{/velocity}}~~
~~{{/code}}~~
~~)))~~

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ DevGuide.WebHome

Documentation.DevGuide

Tutorials & Guides

Documentation.DevGuide.Tutorials.WebHome

Between 2017/09/05 and 2017/09/05, the page has been modified 2 times, by 1 user: Vincent Massol

{{documents
~~location='DevGuide.Tutorials'/}}~~**location='Documentation.DevGuide.Tutorials.'/}}**

- title: Tutorials ⇨ Tutorials & Guides

On 2017/09/05, the page has been created by Vincent Massol

Here's a list of tutorials for an advanced usage of XWiki.
{{documents location='DevGuide.Tutorials'/}}

- title: ⇨ Tutorials
- author: XWiki.XWikiGuest ⇨ XWiki.VincentMassol
- defaultLanguage: ⇨ en

Documentation.DevGuide.Tutorials

Creating XWiki Components

Documentation.DevGuide.Tutorials.WritingComponents.WebHome

Between 2017/09/05 and 2017/09/05, the page has been modified 2 times, by 1 user: Vincent Massol

XWiki.XWikiComments
comment:

- title: Writing XWiki Components ⇨ Creating XWiki Components
- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WebHome
- author: XWiki.evalica ⇨ XWiki.VincentMassol

On 2017/09/05, the page has been created by Vincent Massol

~~{{box cssClass="floatinginfobox" title="**Contents**"}}~~**1 Writing XWiki components**
~~{{toc/}}~~
~~{{/box}}~~
~~This~~**~~#warning("This~~** tutorial ~~guides you through the creation of a XWiki component, which~~ is ~~a way to extend or customize the XWiki platform. Indeed the XWiki platform is composed of components and it's possible to replace the default implementations with your own implementations. It's also possible to add new component implementations to extend the platform such as by implementing new [[Rendering Macros>>platform:DevGuide.RenderingMacroTutorial]].~~**work in progress.")**
~~{{info}}~~**#toc("", "", "")**
~~Components replace the older plugin architecture which has been deprecated a while ago.~~
~~{{/info}}~~
~~You~~**This tutorial will guide you through the creation of an XWiki component that will soon replace the plugins architecture and which is now the recommended way of writing XWiki modules. They** should ~~start~~**be able to execute any java code and communicate with XWiki** by ~~reading~~**using** the ~~[[Reference document on~~**existing** XWiki ~~Components>>extensions:Extension.Component Module]].~~**(core) components, as well as being exposed to the xwiki documents scripting environment (velocity and groovy).**
~~= Let's get started! =~~**1.1 What are Components?**
~~Enough talking, let's see some code!~~**First, you should know a few things about [Plexus > http://plexus.codehaus.org/] engine and [component based architecture > http://plexus.codehaus.org/ref/why-use-components.html]. In very few words, component based architecture is a design principle based on low-coupling / high-cohesion pattern, with components (modules) focused on a single "job" in the system, that describe themselves through interfaces (services) and communicate with other components through these "contracts", without any concern about the implementation. The key is in the components engine (or manager) that handles instantiation and components dependency, based on each component's declaration of these (in its description file), automatically injecting the**

~~implementations at runtime.~~
~~In the following tutorial~~**So, what** we ~~will guide you through writing a simple component, helping you to quickly get oriented~~**should do** in ~~the XWiki components world and explaining how it works.~~**order to write a component is:**
*** define the service the component "provides" (the interface). This interface is all that the component will expose to the other components. Any function you should need to ask to your component goes here.**
*** create one or more implementations for this service. Of course, you must ~~actually provide~~ the services you described earlier**
*** register the implementation and the interface in the <tt>components.xml</tt> file (<tt>META-INF/plexus/components.xml</tt>), as well as the dependencies on other components, requirements and other configurations for your component.**
~~== Creating a XWiki component using Maven ==~~
~~As you've read in the [[XWiki Component Reference>>extensions:Extension.Component Module]] writing a component is a three-steps process (component interface, component implementation and registration of component).~~
~~To make it easier for you to get started, we have created a [[Maven Archetype>>http://maven.apache.org/archetype/maven-archetype-plugin/]] to help create a simple component module with a single command.~~
~~After you've [[installed Maven>>http://maven.apache.org/]], open a shell prompt an type: {{code language="none"}}mvn archetype:generate{{/code}}.~~
~~This will list all archetypes available on Maven Central. If instead you wish to directly use the~~**1.1** XWiki ~~Component Archetype, you can directly type (update the version to use the version you wish to use):~~**Specifics** ~~{{code language="none"}}~~**In XWiki, although we use the [Plexus>http://plexus.codehaus.org/] engine for the moment, we don't want to depend on the Plexus implementation (because we should be able to change components engine anytime, if we wanted to) so we have our own set of lifecycle interfaces to communicate with the manager:**
~~mvn archetype:generate \~~*
[org.xwiki.component.phase.Initializable>http://maven.xwiki.org/site/xwiki-core-parent/xwiki-core-component/apidocs/org/xwiki/component/phase/Initializable.html]: allows the <tt>initialize</tt> method of a component to be called when it's first initialized
~~-DarchetypeArtifactId=xwiki-commons-component-archetype \~~*
[org.xwiki.component.phase.Composable>http://maven.xwiki.org/site/xwiki-core-parent/xwiki-core-component/apidocs/org/xwiki/component/phase/Composable.html]: allows injecting the component manager in a component. Note that component requirements should usually be defined as private field members and defined in <tt>components.xml</tt>. The need for <tt>Composable</tt> is very specific, usually when you need to iterate over some component list, or query various implementations for a component at runtime.
~~-DarchetypeGroupId=org.xwiki.commons \~~*
[org.xwiki.component.phase.LogEnabled>http://maven.xwiki.org/site/xwiki-core-parent/xwiki-core-component/apidocs/org/xwiki/component/phase/LogEnabled.html]: for getting a logger. Usually you should extend the [AbstractLogEnabled>http://maven.xwiki.org/site/xwiki-core-parent/xwiki-core-

component/apidocs/org/xwiki/component/logging/AbstractLogEnabled.html] class to hide the logging setup code.
~~-DarchetypeVersion=5.4.4~~
~~{{/code}}~~
~~Then follow~~**All these are defined in** the ~~instructions. For example:~~**[xwiki-component module in platform/core>https://svn.xwiki.org/svnroot/xwiki/platform/core/trunk/xwiki-component/].**
~~{{code language="none"}}~~
~~[INFO] Scanning for projects...~~
~~[INFO]~~
~~[INFO] --~~
~~[INFO] Building Maven Stub Project (No POM) 1~~
~~[INFO] --~~
~~[INFO]~~
~~[INFO] >>> maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom >>>~~
~~[INFO]~~
~~[INFO] <<< maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom <<<~~
~~[INFO]~~
~~[INFO] --- maven-archetype-plugin:2.2:generate (default-cli) @ standalone-pom ---~~
~~[INFO] Generating project in Interactive mode~~
~~[INFO] Archetype repository missing. Using the one from [org.xwiki.commons:xwiki-commons-component-archetype:6.1-milestone-1] found in catalog remote~~
~~Define value for property 'groupId': : com.acme~~
~~Define value for property 'artifactId': : example~~
~~Define value for property 'version': 1.0-SNAPSHOT: :~~
~~Define value for property 'package': com.acme: :~~
~~Confirm properties configuration:~~
~~groupId: com.acme~~
~~artifactId: example~~
~~version: 1.0-SNAPSHOT~~
~~package: com.acme~~
~~Y: : Y~~
~~[INFO] --~~
~~[INFO] Using following parameters for creating project from Archetype: xwiki-commons-component-archetype:5.4.4~~
~~[INFO] --~~
~~[INFO] Parameter: groupId, Value: com.acme~~
~~[INFO] Parameter: artifactId, Value: example~~
~~[INFO] Parameter: version, Value: 1.0-SNAPSHOT~~
~~[INFO] Parameter: package, Value: com.acme~~
~~[INFO] Parameter: packageInPathFormat, Value: com/acme~~
~~[INFO] Parameter: package, Value: com.acme~~
~~[INFO] Parameter: version, Value: 1.0-SNAPSHOT~~
~~[INFO] Parameter: groupId, Value: com.acme~~
~~[INFO] Parameter: artifactId, Value: example~~
~~[INFO] project created from Archetype in dir: /private/tmp/example~~
~~[INFO] --~~
~~[INFO] BUILD SUCCESS~~
~~[INFO] --~~
~~[INFO] Total time: 25.019s~~
~~[INFO] Finished at: Thu May 29 18:49:46 CEST 2014~~
~~[INFO] Final Memory: 11M/26M~~

~~[INFO] --~~
~~{{/code}}~~
~~Then go in the created directory (##example## in our example above) and run ##mvn install## to build your component.~~
~~== The Component explained ==~~**1.1 Let's get started!**
~~Assume, for the following explanations, that the package you used is ##com.acme##~~**Enough talking, let's see some code!**
~~Navigating in~~**In** the ~~component project folder, you~~**followings we** will ~~see the following standard Maven project structure:~~**guide you through writing a simple component, helping you to quickly get oriented in XWiki components world and explaining how it works.**
~~{{code language="none"}}~~**1.1.1 Creating a XWiki component using maven**
~~pom.xml~~
~~src/main/java/com/acme/HelloWorld.java~~
~~src/main/java/com/acme/internal/DefaultHelloWorld.java~~
~~src/main/java/com/acme/internal/HelloWorldScriptService.java~~
~~src/main/resources/META-INF/components.txt~~
~~src/test/java/com/acme/HelloWorldTest.java~~
~~{{/code}}~~
~~which corresponds to~~**To simplify** the ~~default files created: the ##HelloWorld## interface (a.k.a~~**three steps process of** component ~~role), its implementation ##DefaultHelloWorld## (component implementation), a test class for this component ##HelloWorldTest##, the component declaration file ##components.txt###~~**creation in XWiki,** and **since** the ~~Maven project ##pom.xml## file. The ##HelloWorldScriptService## file~~**XWiki code lifecycle** is ~~described below when~~**based on [maven>http://maven.apache.org/],** we ~~explain how~~**have created a maven archetype** to ~~make the component's API available~~**help create a simple component module with a single command, with respect** to ~~wiki pages.~~**the XWiki architecture and components specific requirements.**
*** download the archetype from here: {attach:xwiki-archetype-component-1.0-SNAPSHOT.jar} (it will soon be uploaded on our maven repository).**
*** use maven to install this file on your local repository by executing (make sure you replace <tt>path-to-jar-file</tt> with your own path):**
{code}
mvn install:install-file -Dfile=<path-to-jar-file> -DartifactId=xwiki-archetype-component -DgroupId=com.xpn.xwiki.platform.tools -Dversion=1.0-SNAPSHOT -Dpackaging=jar
{code}
*** now you're ready to use maven to generate the xwiki component based on this archetype. Navigate to the directory where you want your component to be located and type:**
{code}
mvn archetype:generate -DarchetypeGroupId=com.xpn.xwiki.platform.tools -DarchetypeArtifactId=xwiki-archetype-component -DarchetypeVersion=1.0-SNAPSHOT -DgroupId=<component-group-id> -DartifactId=<component-artifact-id> -Dpackage=<component-package> -Dversion=<component-version> -Dpackaging=jar
{code}
where you replace <tt>component-group-id</tt>, <tt>component-artifact-id</tt>, <tt>component-package</tt>, <tt>component-version</tt> with the corresponding values for your component. To create a server XWiki Watch component, for example, we used <tt>-DgroupId=com.xpn.xwiki.products -DartifactId=xwiki-watch-

component -Dpackage=org.xwiki.watch.component -Dversion=1.1-SNAPSHOT</tt>. Don't forget to follow the [xwiki package names guidelines > http://dev.xwiki.org/xwiki/bin/view/Community/CodeStyle#HPackagenames-1].
~~If you have~~Now this will create a ~~look~~new maven module in ~~##pom.xml## you'll notice the following dependencies:~~a folder named <tt>component-artifact-id</tt> in your folder, with a default xwiki component inside.
#info("Note that if your parent (current, from where you are executing maven) folder is the folder of a maven module (contains a <tt>pom.xml</tt> file), then the command above will fail unless the module is packaged as <tt>pom</tt>. If the project is packaged as <tt>pom</tt>, then the newly created module will be added in its modules list, and the parent of the newly created component module will be set to this project's <tt>pom</tt>.")
~~{{code language="xml"}}~~1.1.1 The component explained
Assume, for the following explanations, that the package you used is <tt>org.xwiki.component</tt>
Navigating in the component project folder, you will see standard maven project structure like this:
{code}
pom.xml
src/main/java/org/xwiki/component/HelloWorld.java
src/main/java/org/xwiki/component/internal/DefaultHelloWorld.java
src/main/resources/META-INF/plexus/components.xml
src/test/java/org/xwiki/component/HelloWorldTest.java
{code}
which corresponds to the default files created: the <tt>HelloWorld</tt> interface (service), its implementation <tt>DefaultHelloWorld</tt>, a test class for this component <tt>HelloWorldTest</tt>, the component description file <tt>components.xml</tt> and the maven project <tt>pom</tt> file.
If we have a look in the <tt>pom</tt>, we see something like this:
{code}
<groupId>your-group-id</groupId>
<artifactId>your-artifact-id</artifactId>
<version>your-version</version>
{code}
which are the group, artifact and version you used when you created your component
{code}
<properties>
<!-- TODO: remove this if you inherit a project that has the core version set -->
<platform.core.version>1.5-SNAPSHOT</platform.core.version>
</properties>
{code}
used to define the core version for the xwiki-component dependency.
If your component is created as part of a project already depending on the core (and which most probably already has a property for the core version), use the inherited property value instead of redefining it here.
Or, if the component should have the same version as the rest of the XWiki modules, you can use <tt>${pom.version}</tt> as the version for other XWiki dependencies.
{code}
~~<groupId>org.xwiki.commons</groupId>~~
<groupId>org.xwiki.platform</groupId>

<groupId>org.xwiki.platform</groupId>
<artifactId>xwiki-commons-component-api</artifactId><artifactId>xwiki-core-component</artifactId>
<version>${commons.version}</version>
<version>${platform.core.version}</version>
<!-- Testing dependencies -->
<groupId>org.xwiki.commons</groupId><groupId>junit</groupId>
<artifactId>xwiki-commons-test</artifactId><artifactId>junit</artifactId>
<version>${commons.version}</version><version>3.8.1</version>
</dependency></dependency>
<!-- Add here all your other dependencies -->
{{/code}}{code}
which define the dependency on the xwiki-component module in the core and the junit for the testing phase and, of course, marks the spot for you to add all your other components, modules, and libraries dependencies for maven to know and put them on your classpath when compiling, etc.
The code above definesinterface file (<tt>HelloWorld.java</tt>) contains the dependency on the ##xwiki-core-component-api## in the core which is where XWiki Component notions are defined. There's alsodefinition of a dependency on ##xwiki-core-shared-tests## which provides helper classes to easily test components.regular Java interface, with a special
<tt>ROLE</tt> field
{code}
/** The role associated with the component. */
String ROLE = HelloWorld.class.getName();
{code}
which associates a name (that would identify a functionality, concern) to this service. Keep in mind that only this interface specifies the functions the other components will use to communicate with our component, no other functions besides the ones defined in this interface will be accessible to the "outside world". In our case, we'll build a polite component that can only <tt>sayHello()</tt>.
The interface file (##HelloWorld.java##) contains the definition of a regular Java interface and looks like this: You can find more information about components identity in plexus at
[http://plexus.codehaus.org/guides/developer-guide/building-components/component-identity.html].
{{code language="java"}}Then we have the implementation of the interface, the <tt>DefaltHelloWorld</tt> class.
@Role /* annotation used for declaring the service our component provides */{code}
public interface HelloWorld class DefaultHelloWorld extends AbstractLogEnabled implements HelloWorld, Initializable
{{code}
String sayHello(); it extends <tt>AbstractLogEnabled</tt> to be able to use the logging system, implements the component interface, and also the <tt>Initializable</tt> interface which allows it to hook initialization code upon its instantiation by the component manager, in the <tt>initialize()</tt> function:
}{code}
{{/code}} public void initialize() throws InitializationException
{
//TODO: initialize component
// getLogger is inherited from AbstractLogEnabled
getLogger().debug("DefaultHelloWorld initialized");
}
{code}

~~Keep in mind that this interface specifies the API that other components can use on your component. In our case, we'll build~~**We also find** a ~~polite component~~**field** that ~~can ##sayHello()##.~~**contains the greeting to be used and the implementation of the <tt>sayHello()</tt> function in the interface:**

{code}
/ The greeting that was specified in the configuration. */**
private String greeting;
~~Then we have the implementation of the interface, the ##DefaultHelloWorld## class.~~ **/**
*** Says hello by returning a greeting to the caller.**
*****
*** @return A greeting.**
***/**
public String sayHello()
{
return greeting;
}
{code}

~~{{code language="java"}}~~**The very interesting part of all the project, which gives all the salt and pepper to our component, is the description file, <tt>components.xml</tt>:**

~~@Component /* annotation used for declaring a component implementation */~~**{code}**
~~@Singleton /* annotation used for defining the component as a singleton */~~**<component-set>**
~~public class DefaultHelloWorld implements HelloWorld~~ **<components>**
~~{{/code}}~~ **<component>**
<role>org.xwiki.component.HelloWorld</role>
<implementation>org.xwiki.component.internal.DefaultHelloWorld</implementation>
<configuration>
<!-- injected in the greeting field in the implementation -->
<greeting>Hello World!</greeting>
</configuration>
</component>
</components>
</component-set>
{code}

which basically tells Plexus to use the implementation in the <tt>DefaultHelloWorld</tt> class to provide the service defined by the <tt>HelloWorld</tt> interface. Note that the role can have any value as long as it uniquely identifies a role, we use as convention the name of the service interface. The <tt>configuration</tt> element of the component allows us to specify the set of configurations to be injected by the component manager at runtime, for the implementation: in our case, the greeting to be used, which will be mapped on the <tt>greeting</tt> field in the class.

~~Note that optionally, there is a ##@Named## annotation to specify a component //hint//. This is useful especially when we want to distinguish between several implementations for~~**A more detailed guide through** the ~~same type of component. Imagine we had a special HelloWorld implementation taking the greeting message from a database; it could look like:~~**<tt>components.xml</tt> file can be found at http://plexus.codehaus.org/guides/developer-guide/configuration/component-descriptor.html and http://plexus.codehaus.org/ref/component-configuration.html**

{{code language="java"}}
@Component
@Named("database")
public class DatabaseHelloWorld implements HelloWorld
{{/code}}
Then the ##sayHello## in ##DefaultHelloWorld## is basic in this example:**1.1 How to find my component and use it?**
{{code language="java"}}**1.1.1 From other components**
/**
* Says hello by returning a greeting to the caller.
*
* @return A greeting.
*/
public String sayHello()
{
return "Hello world!";
}
{{/code}}
And now,**To access your component from another component we use** the ##components.txt## file, in which**components engine, and specify the dependencies declarative, leaving instantiation and** component implementations present in this jar are specified for**injection to** the ##ComponentManager##**be handled by the component manager. The most straightforward way is the use of the requirements mechanism of plexus, specifying that our component is required by the component that needs** to register them.**access it.**
{{code language="none"}}**Don't forget that any code that uses the component we wrote needs to have the component interface accessible in its classpath. Even if instantiation and dependency is handled by the engine at runtime, the code still needs to compile.** com.acme.internal.DefaultHelloWorld**If the two components are not in the same module (the same .jar), don't forget to add the module of the greeter component as a dependency of the module of any component that uses it.**
{{/code}}
= How**Then,** to find my**effectively use the <tt>HelloWorld</tt> component, we need a reference to it in the the** component and**that uses it. For this, we** use it? =**a member variable in the implementation of the using component, for example, a <tt>Socializer</tt> component will need to be able to say hello to the world:**
== From other components =={code}
To access your component from another component we use the components engine, and specify the dependencies, leaving instantiation and component injection to be handled by the component manager.**public class DefaultSocializer extends AbstractLogEnabled implements Socializer, Initializable**
In order to use the ##HelloWorld## component, you need a reference to it in the component that uses it. For this, you should use a member variable in the implementation of the using component, for example, a ##Socializer## component will need to be able to say hello to the world:
{{code}}
@Component
@Singleton
public class DefaultSocializer implements Socializer
@Inject
/** Will be injected by the component manager */
@Inject
@Named("database")

~~@Named("database")~~
~~private HelloWorld databaseWorld;~~
~~{{/code}}~~**{code}**
~~Note~~**Then, we need to tell Plexus about this dependency so that it assignes a <tt>HelloWorld</tt> instance to** the ~~##@Inject## annotation, which instructs~~**<tt>helloWorld</tt> variable, at runtime. For this, we use** the ~~component manager to inject~~**components specification file of** the ~~required component where needed.~~**<tt>Socializer</tt> component, which would look something like this:**
{code}
<component-set>
<components>
<component>
<role>org.xwiki.component.Socializer</role>
<implementation>org.xwiki.component.internal.DefaultSocializer</implementation>
<requirements>
<requirement>
<role>org.xwiki.component.HelloWorld</role>
</requirement>
</requirements>
</component>
</components>
</component-set>
{code}
~~And that's it, you can now use~~**Note** the ~~##helloWorld## member anywhere in~~**<tt>requirements</tt> element, which contains a requirement for** the ~~##DefaultSocializer## class freely, without further concerns, it will be assigned by the component manager provided~~**<tt>HelloWorld</tt> role,** that the ~~##HelloWorld##~~**<tt>HelloWorld</tt>** component ~~is on the classpath at runtime when the ##Socializer## is used. Such as:~~**provides.**
~~{{code}}~~**And that's it, you can now use the <tt>helloWorld</tt> member anywhere in the <tt>DefaultSocializer</tt> class freely, without further concerns, it will be assigned by the component manager provided that the <tt>HelloWorld</tt> component is on the classpath at runtime when the <tt>Socializer</tt> is used. Such as:**
~~public class DefaultSocializer implements Socializer~~ **{code}**
public class DefaultSocializer extends AbstractLogEnabled implements Socializer, Initializable
~~{{/code}}~~**{code}**
More, note that all through the process of defining a communication path between two components, we never referred components implementations, all specifications being done through ~~//roles//~~**~~roles~~** and ~~//interfaces//:~~**~~interfaces~~:** the implementation of a service is completely hidden from any code external to the component.
~~== From non-components java code (e.g. older plugins) ==~~**TODO: refer to the other ways of implementing dependencies but requirements mechanism. Details, explanations, links.**
~~For this kind of usages, since we cannot use the component based architecture advantages and the "magic" of the component manager, the XWiki team has created a helper method that acts like a bridge between component~~**1.1.1 From non-components java** code ~~and non-component code, the ##com.xpn.xwiki.web.Utils.getComponent(String role, String hint)## that gets the specified component instance from the component manager and returns it. As seen in the previous sections, the hint is an optional identifier, additional to ##role##, used to differentiate between implementations of the same interface: the //roles// identify services while the hints help differentiate between implementations. The~~

~~##getComponent## function also has a signature without the ##hint## parameter, that uses the default hint.~~**(e.g. older plugins)**
~~To use our greetings provider component,~~**For this kind of usages, since** we ~~would simply invoke:~~**cannot use the component-based architecture advantages and the "magic" of the component manager, the XWiki team has created a helper method that acts like a bridge between component code and non-component code, the <tt>com.xpn.xwiki.web.Utils.getComponent(String role, String hint) </tt> that gets the specified component instance from the component manager and returns it. The hint is an optional identifier, additional to <tt>role</tt>, used to differentiate between implementations of the same interface: the ~~roles~~ identify services while the hints help differentiate between implementations (see more at [http://plexus.codehaus.org/guides/developer-guide/building-components/component-identity.html > http://plexus.codehaus.org/guides/developer-guide/building-components/component-identity.html]). The <tt>getComponent</tt> function also has a version without the <tt>hint</tt> parameter, that uses the default hint.**
~~{{code}}~~**To use our greetings provider component, we simply invoke:**
~~HelloWorld greeter = Utils.getComponent(HelloWorld.class);~~**{code}**
HelloWorld greeter = (HelloWorld)
Utils.getComponent(HelloWorld.ROLE);
//use the HelloWorld service
{code}
~~HelloWorld databaseGreeter = Utils.getComponent(HelloWorld.class, "database");~~**Note that, even if, in fact, the object returned by this function is an instance of the DefaultHelloWorld, you should *never declare your object of the implementation type nor cast to implementation instead of interface*. A component is represented by its interface, the implementation for such a service can be provided by any code, any class so relying on the implementation type is nor good practice (since the interface contract should be enough for a component), nor safe. In the future, a maven enforcer plugin will be setup in the build lifecycle, so that any reference to component implementations (located in an "internal" subpackage) will cause build errors.**
~~greeter.sayHello();~~
~~{{/code}}~~
~~{{warning}}~~**1.1.1 From wiki pages**
~~Even if the object returned by this function is an instance of the ##DefaultHelloWorld##, you should never declare your object of the implementation type nor cast to implementation instead of interface.~~
~~{{/warning}}~~
~~A component is represented by its interface, the implementation for such a service can be provided by any code, any class so relying on the implementation type is neither good practice (since the interface contract should be enough for a component), nor safe. In the future, a maven enforcer plugin will be setup in the build lifecycle, so that any reference~~**In order** to **use a** component ~~implementations (located~~ in ~~an "internal" subpackage) will cause build errors.~~**wiki pages, we need to expose it to the scripting environments: groovy and velocity.**
~~{{info}}~~**1.1.1.1 Accessing a component from groovy**
~~The usage of ##Utils.getComponent()## functions is highly discouraged, reserved for this type of situations, when you need to access a component from non-componentized code. For the componentized code, you should use either dependency declaration at 'compile-time' (as shown before with~~

~~annotations) or, if you need to resolve components dependencies at runtime, use the ##ComponentManager##, which you can access by implementing the Composable interface as described in the [[Component Module Reference>>extensions:Extension.Component Module]].~~
~~{{/info}}~~
~~== From wiki pages ==~~**Since, in groovy, we have access to all classes and functions in XWiki (all this protected by the requirement for programming rights), it means that we can use the same method as in [the previous section > CreatingComponents#HFromnoncomponentsjavacode28egolderplugins29], using the <tt>Utils</tt> class. A simple page that would print the greeting from the <tt>HelloWorld</tt> component would look like this (of course, with your own package for the HelloWorld interface):**
~~Components can be made accessible to wiki pages by writing a ##ScriptService## implementation. They can then be accessed using any provided scripting language (velocity, groovy, python, ruby, php, etc).~~**{code}**
<%
def greeter = com.xpn.xwiki.web.Utils.getComponent(org.xwiki.component.HelloWorld.ROLE);
println greeter.sayHello();
%>
{code}
~~Let's~~**TODO: talk about the future plans (?) to** make ~~our ##sayHello## method accessible:~~**a component accessible in the groovy context through a groovy bridge.**
~~{{code language="java"}}~~**1.1.1.1 Accessing a component from velocity**
~~@Component~~**XWiki dev team is currently working on the design of a VelocityBridge interface that will handle components access from velocity. Until this specification is ready and its first implementation done, we can do it as follows:**
~~@Named("hello")~~**We write another component in our package, that implements the <tt>VelocityContextInitializer</tt>, which is responsible for the initialization of the velocity context in XWiki through its method <tt>initialize(VelocityContext context)</tt>, called automatically when a new velocity context is created. As you probably guessed, we will use this function to add our <tt>HelloWorld</tt> component to the velocity context.**
~~@Singleton~~**{code}**
public class ~~HelloWorldScriptService~~**HelloWorldVelocityContextInitializer** implements ~~ScriptService~~**VelocityContextInitializer**
~~@Inject~~**/** The key to add to the velocity context */**
public static final String VELOCITY_CONTEXT_KEY = "greeter";
/ The component instance to add to the velocity context, injected by the component manager */**
~~public String greet()~~**/**
**** Add the component instance to the velocity context received as parameter.**
/
public void initialize(VelocityContext context)
~~return this.helloWorld.sayHello();~~**context.put(VELOCITY_CONTEXT_KEY, this.helloWorld);**
~~{{/code}}~~**{code}**
~~Note: We could have also injected~~**The result of this will be** the ~~Named~~

Note: We could have also injected The result of this will be the Named component instead, "database" which would look like:availability of the <tt>HelloWorld</tt> instance in velocity through the key ~~greeter~~ (as in <tt>$greeter.sayHello()</tt>).
{{code language="java"}}Of course, we need to have our <tt>HelloWorld</tt> component reference when we execute this code so we add it as a dependency to this velocity context initializer component, as described in [the section above>#HFromothercomponents]:
// Or inject a Named Component {code}
@Inject<component>
@Named("database")
<role>org.xwiki.velocity.VelocityContextInitializer</role>
private HelloWorld databaseWorld; <role-hint>greeter</role-hint>
{{/code}}
<implementation>org.xwiki.component.internal.vcinitializer.HelloWorldVelocityContextInitializer</implementation>
<instantiation-strategy>singleton</instantiation-strategy>
<requirements>
<requirement>
<role>org.xwiki.component.HelloWorld</role>
</requirement>
</requirements>
</component>
{code}
{{info}}This code goes in the <tt>components.xml</tt> file of the package where the velocity context initializer is located. In our case, in the same file as the description for the <tt>HelloWorld</tt> is.
The component hint used (the ##hello## part in the ##@Component##) is the name under which the script service will be accessible from scripting languages.
{{/info}}
For exampleNote that this time, we also use roles for component identification, because we need to accessdifferentiate this implementation of the <tt>VelocityContextInitializer</tt> from the other implementations, as it is not the only component with this role in Velocity you'd write:XWiki. Also note an <tt>instantion-strategy</tt> that tells Plexus to instantiate this component only once.
{{code language="none"}}{{velocity}}Of course, in order to for all this to compile, we need to have the <tt>VelocityContextInitializer</tt> interface available on the classpath so we have this new dependency in the component module's pom:
$services.hello.greet()}{code}
{{/velocity}}{{/code}}. <dependency>
<groupId>org.xwiki.platform</groupId>
<artifactId>xwiki-core-velocity</artifactId>
<version>${platform.core.version}</version>
</dependency>
{code}
From Groovy:And that's it, you have made your <tt>HelloWorld</tt> component velocity-accessible! Just recompile your package, copy it in the WEB-INF/lib folder of your xwiki webbapp container, and restart the server. You'll be able to get a greeting in velocity through:
{code}
$greeter.sayHello()
{code}
{{code language="none"}}{{groovy}}For the automatic creation of a velocity accessible xwiki component through this method, we have

also created a maven archetype for this purpose too, the {attach:xwiki-archetype-velocity-component-1.0-SNAPSHOT.jar}. Download it and use it as described in [the first part of this tutorial > CreatingComponents#HCreatingaXWikicomponentusingmaven].
print services.hello.greet()
{{/groovy}}{{/code}}.
Now for our script service to work we need to register it as a component and thus add it to the ##META-INF/components.txt## file:
{{code language="none"}}
...
com.acme.internal.HelloWorldScriptService
{{/code}}
We also need to make the Script Service infrastructure available in our classpath. This is done by adding the following in your ##pom.xml## file:
{{code language="xml"}}
<dependency>
<groupId>org.xwiki.commons</groupId>
<artifactId>xwiki-commons-script</artifactId>
<version>${commons.version}</version>
</dependency>
{{/code}}
= Accessing Legacy code =
By legacy we mean old XWiki code that hasn't been moved to components yet.
== The XWiki data model ==
Since the XWiki data model (documents, objects, attachments, etc.) reside in the big, old ##xwiki-core## module, and since we don't want to add the whole core and all its dependencies as a dependency of a simple lightweight component (this would eventually lead to a circular dependency, which is not allowed by maven), the current strategy, until the data model is completely turned into a component, is to use a //bridge// between the new component architecture and the old ##xwiki-core##.**1.1 How do I find other code?**
In short, the way this works is based on the fact that implementations for a component don't have to be in the same ##.jar## as the interface, and there is no dependency //from// the component interface //to// the actual implementation, only the other way around. So, we made a few simple components that offer basic access to XWiki documents, and declare the classes in ##xwiki-core## as the default implementation for those components.**1.1.1 Other components**
If your component needs to access the XWiki data model, it will**To** use **other components, you follow** the components from**same paradigm explained in** the ##xwiki-platform-bridge## module for that. Note that these interfaces are rather small, so you can't do everything that you could with**[previous section > CreatingComponents#HFromothercomponents] but this time adding** the old model. If you need to add some methods to**other components as a dependency of** the bridge, feel free to propose**<tt>HelloWorld</tt> component, in its <tt>components.xml</tt> file and having** it on**injected in a member field of** the [[mailing list>>dev:Community.MailingLists]].**<tt>HelloWorld</tt> implementation. For example, we could have such a dependency:**
For example:**{code}**
{{code}}**public class DefaultHelloWorld extends AbstractLogEnabled implements HelloWorld, Initializable**
@Component
@Singleton

~~public class DefaultHelloWorld implements HelloWorld~~
/** ~~Provides access to documents. Injected~~**Will be injected** by the ~~Component Manager.~~**component manager** */
~~@Inject~~**private Localizer localizer;**
~~private DocumentAccessBridge documentAccessBridge;~~
~~[...]~~**/** The greeting that was specified in the configuration. */**
private String greeting;
~~private String getConfiguredGreeting()~~**/****
*** Says hello by returning a greeting to the caller.**
*****
*** @return A greeting.**
***/**
public String sayHello()
~~return documentAccessBridge.getProperty("XWiki.XWikiPreferences", "greeting_text");~~**// Get the localized version for the needed greeting**
return localizer.get(this.greeting);
~~{{/code}}~~**}**
{code}
~~=== Querying the data model ===~~**{code}**
<component-set>
<components>
<component>
<role>org.xwiki.component.HelloWorld</role>
<implementation>org.xwiki.component.internal.DefaultHelloWorld</implementation>
<configuration>
<!-- bundle key for the greeting to provide -->
<greeting>greetings.hello</greeting>
</configuration>
<!-- the greeter requires a localization component -->
<requirements>
<requirement>
<role>org.xwiki.component.Localizer</role>
</requirement>
</requirements>
</component>
</components>
</component-set>
{code}
if we needed to implement a polyglot greeter component.
~~Queries can be performed by using an instance of a QueryManager, which can be obtained and used as follows :~~**1.1.1 The XWiki data model**
~~{{code}}~~**Since the XWiki data model (documents, objects, attachments, etc.) reside in the big, old <tt>xwiki-core</tt> module, and since we don't want to add the whole core and all its dependencies as a dependency of a simple lightweight component (this would eventually lead to a circular dependency, which is not allowed by maven), the current strategy, until the data model is completely turned into a component, is to use a ~~bridge~~ between the new component architecture and the old <tt>xwiki-core</tt>.**
~~QueryManager queryManager = (QueryManager) componentManager.getInstance(QueryManager.class);~~
~~Query query = queryManager.createQuery(xwqlstatement,Query.HQL);~~
~~List<Object> results = query.execute();~~
~~{{/code}}~~
~~{{info}}~~**In short, the way this works is based on the fact that implementations for a component don't have to be in the same <tt>jar</tt> as the interface, and there is no dependency [illegible] the**

<tt>.jar</tt> as the interface, and there is no dependency ~~from~~ the component interface ~~to~~ the actual implementation, only the other way around. So, we made a few simple components that offer basic access to XWiki documents, and declared the classes in <tt>xwiki-core</tt> as the default implementation for those components.
A reference to a ComponentManager can be obtained through injection, as explained on the [[Component module extension page>>extensions:Extension.Component Module#HGettingaccesstotheComponentManager]].
{{/info}}
== TheIf your component needs to access the XWiki context ==data model, it will use the components from the <tt>xwiki-core-bridge</tt> module for that. Note that these interfaces are rather small, so you can't do everything that you could with the old model. If you need to add some methods to the bridge, feel free to propose it on the [mailing list>Community.MailingLists].
Note that the XWiki context is deprecated. It was an older way of keeping track of the current request, which had to be passed around from method to method, looking like a [[ball and chain>>http://en.wikipedia.org/wiki/Ball_and_chain]] present everywhere in the code.For example:
{code}
public class DefaultHelloWorld implements HelloWorld
{
/** Provides access to documents. Injected by the Component Manager. */
private DocumentAccessBridge documentAccessBridge;
In the component world, the current request information is held in an **{{scm project="xwiki-commons" path="xwiki-commons-core/xwiki-commons-context/src/main/java/org/xwiki/context/ExecutionContext.java"}}ExecutionContext{{/scm}}**. This is actually more powerful than the old XWikiContext, as it is a generic execution context, hold in a ThreadLocal variable, and you can create one anytime you want and use it anyway you want. And you don't have to manually pass it around with all method calls, as execution contexts are managed by the **{{scm project="xwiki-commons" path="xwiki-commons-core/xwiki-commons-context/src/main/java/org/xwiki/context/Execution.java"}}Execution{{/scm}}** component, which you can use just like any other XWiki component. [...]
In short, if you want to get access to the execution context (which holds context information inserted by the new components), you must declare an injection point on the ##Execution## component (located in the ##xwiki-commons-context## module), and then you can write: private String getConfiguredGreeting()
{
return documentAccessBridge.getProperty("XWiki.XWikiPreferences", "greeting_text");
}
{code}
{{code language="java"}}1.1.1 The XWiki context
/** Provides access to the request context. Injected by the Component Manager. */
@Inject
private Execution execution;
...
private void workWithTheContext()
{
ExecutionContext context = execution.getContext();

~~// Do something with the execution context~~
~~}~~
~~{{/code}}~~
~~All~~**Note** that ~~said, we're still in a transition phase and a lot of information is still available only through~~ the ~~old XWikiContext and has not yet been moved to the ExecutioncContext (the current user for example just to mention one). Thus you may still need to access the old~~ XWiki ~~Context. You can get a reference to it from the Execution Context. If you can you should try to not cast it to~~**context is deprecated. It was** an ~~##XWikiContext##, which would pull~~**older way of keeping track of** the ~~whole ##xwiki-platform-oldcore## as a dependency, but~~**current request, which had** to ~~a ##Map##. Doing it this way, you won't~~ be ~~able~~**passed around from method** to ~~access all the properties,~~**method, looking** like ~~the current user name or the URL factory, but you can access anything placed~~**a [ball and chain>http://en.wikipedia.org/wiki/Ball_and_chain] present everywhere** in the ~~internal map of the XWikiContext.~~**code.**
~~{{code language="java"}}~~**In the component world, the current request information is held in an *[execution context>http://maven.xwiki.org/site/xwiki-core-parent/xwiki-core-context/apidocs/org/xwiki/context/ExecutionContext.html]*. This is actually more powerful than the old XWiki context, as it is a generic execution context, and you can create one anytime you want and use it anyway you want. And you don't have to manually pass it around with all method calls, as execution contexts are managed by the *[Execution component>http://maven.xwiki.org/site/xwiki-core-parent/xwiki-core-context/apidocs/org/xwiki/context/Execution.html]*, which you can use just like any other XWiki component.**
~~private void workWithTheContext()~~
~~{~~
~~ExecutionContext context = execution.getContext();~~
~~Map<Object, Object> xwikiContext = (Map<Object, Object>) context.getProperty("xwikicontext");~~
~~// Do something with the XWiki context~~
~~}~~
~~{{/code}}~~
~~If~~**In short, if** you ~~need~~**want** to **get** access ~~typed~~**to the execution context (which holds context** information ~~then~~**inserted by** the ~~easiest is to use~~**new components), you must declare** a ~~Provider~~**requirement on** the ~~following way:~~**<tt>Execution</tt> component (located in the <tt>xwiki-core-context</tt> module), and then you can write:**
{code}
/ Provides access to the request context. Injected by the Component Manager. */**
private Execution execution;
~~{{code language="java"}}~~ **[...]**
~~@Inject~~
~~private Provider<XWikiContext> xwikiContextProvider;~~
~~...~~
~~XWikiContext xcontext = this.xwikiContextProvider.get();~~
~~{{/code}}~~
~~which will always provide a usuable XWikiContext (if there is none in the current ExecutionContext it will create one).~~ **private void workWithTheContext()**
{
ExecutionContext context = execution.getContext();
// Do something with the execution context
}

{code}
~~Since 7.2 if~~If you ~~just want~~still need to access the old XWiki context, then you can get ~~one if there is one (i.e.~~a reference to it from the execution context, but you ~~don't need~~should not cast it to ~~automatically create~~an <tt>XWikiContext</tt>, which would pull the whole xwiki-core as a ~~new one)~~dependency, but to a <tt>Map</tt>. You won't be able to access all the properties, like the current user name or the URL factory, but you can ~~use~~access anything placed in the ~~"readonly" XWikiContext Provider as in:~~internal map of the XWikiContext.
{code}
private void workWithTheContext()
{
ExecutionContext context = execution.getContext();
Map<Object, Object> xwikiContext = (Map<Object, Object>) context.getProperty("xwikicontext");
// Do something with the XWiki context
}
{code}
~~{{code language="java"}}~~If you want not just to use the execution context, but to make something available in every execution context, you can create an implementation of the [ExecutionContextInitializer>http://maven.xwiki.org/site/xwiki-core-parent/xwiki-core-context/apidocs/org/xwiki/context/ExecutionContextInitializer.html] component, and populate newly created execution contexts, just like with [velocity contexts>#HAccessingacomponentfromvelocity].
~~@Inject~~
~~@Named("readonly")~~
~~private Provider<XWikiContext> xwikiContextProvider;~~
~~...~~
~~XWikiContext xcontext = this.xwikiContextProvider.get();~~
~~if (xcontext != null) {~~
~~...~~
~~}~~
~~{{/code}}~~
~~If you want not just to use the Execution Context, but to make something available in every execution context, you can create an implementation of the **{{scm project="xwiki-commons" path="xwiki-commons-core/xwiki-commons-context/src/main/java/org/xwiki/context/ExecutionContextInitializer.java"}}ExecutionContextInitializer{{/scm}}** component, and populate newly created execution contexts, just like with [[velocity contexts>>platform:DevGuide.WritingComponents#HFromwikipages]].~~1.1.1 Code outside components
~~== Code outside components ==~~You can use external libraries as in any other maven module, just declare the right dependencies in your module's <tt>pom.xml</tt>.
~~You can use external libraries as in~~As a general rule, you should *not* work with any ~~other maven~~non-componentized XWiki code, as the way the old code was designed leads to an eventual dependency on the whole <tt>xwiki-core</tt> module, ~~just declare~~which we are trying to avoid. If the ~~right dependencies in your module's ##pom.xml##.~~component you are writing is needed by other modules (which is the case with most components, since a component which isn't providing any usable/used services is kind of useless), then this will likely lead to an eventual cyclic dependency, which will break the whole build.
~~As a general rule, if you should **not** work with any non-componentized~~

As a general rule,**If** you ~~should "not" work with any non-componentized XWiki code, as~~**need some functionality from** the ~~way the~~ old ~~code was designed leads to an eventual dependency on the whole ##xwiki-core## module, which we are trying to avoid. If the component you are writing is needed by other modules (which is the case with most components, since~~**core, consider rewriting that part as** a **new** component ~~which isn't providing any usable/used services is kind of useless),~~**first, and** then ~~this will likely lead to an eventual cyclic dependency, which will break~~**use that new component from your code. You should ask first on** the ~~whole build.~~**[devs mailing list>Community.MailingLists], so that we can design and implement it collaboratively.**

~~If you need some functionality from the old core, consider rewriting that part as a new component first, and then use that new component from your code. You should ask first on the [[devs mailing list>>dev:Community.MailingLists]], so that we can design and implement it collaboratively.~~

~~= Deploying the Component =~~

~~Now that we have a functioning Component let's build it and deploy it to an XWiki instance. There are 2 ways.~~

~~== Manually ==~~

~~* To build the component, issue ##mvn install##. This generates a JAR in the ##target## directory of your project.~~

~~* To install it into a XWiki Enterprise instance, just copy that JAR file in ##XE_WAR_HOME/WEB-INF/lib## where ##XE_WAR_HOME## is where the XWiki Enterprise WAR is deployed.~~

~~== Using the Extension Manager ==~~

~~The advantage over the Manual way is that you don't need to regularly start/stop your XWiki instance and thus you don't incur the start wait times.~~

~~* Have a running XWiki instance configured with a local Extension Repository pointing to your Maven local Repository. Edit ##xwiki.properties## and make sure you have the following set:(((~~

~~{{code language="none"}}~~

~~extension.repositories=local:maven:file://${sys:user.home}/.m2/repository~~

~~extension.repositories=maven-xwiki:maven:http://nexus.xwiki.org/nexus/content/groups/public~~

~~extension.repositories=extensions.xwiki.org:xwiki:http://extensions.xwiki.org/xwiki/rest/~~

~~{{/code}}~~

~~)))~~

~~* Build your component and deploy it in your local Maven repository with ##mvn install##~~

~~* Inside your running XWiki instance, go to the Extension Manager in the Admin UI (e.g. ##{{{http://localhost:8080/xwiki/bin/admin/XWiki/XWikiPreferences?editor=globaladmin&section=XWiki.AddExtensions}}}##) and click on Advanced Search and enter your extension's id and version and follow the instructions~~

~~{{warning}}~~

~~If you want to redeploy an extension and it's already installed with the same version, the Extension Manager won't let you do so. Thus you'll need to uninstall it first using the Extension Manager. You'll also need to remove metadata in memory using the [[Extension Tweak>>extensions:Extension.Extension Tweak]].~~

~~{{/warning}}~~

~~Your component is now ready for service.~~

~~Enjoy!~~

~~= See also =~~

~~* [[extensions:Extension Component Module]]~~

~~* If you are experiencing problems trying to install the component using Maven, check the [[Building Guide>>dev:Community.Building]].~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.OlivierBerger~~
- comment:
~~Note that the use of plexus is now deprecated and another Component system is used for XWiki~~
- date:
~~2010-01-26 10:49:13.0~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.YangNingYuan~~
- comment:
~~Other components of the system is used xwiki. Which component system is used? Is Guice ?~~
- date:
~~2010-01-30 18:03:36.0~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.Sergiu~~
- comment:
~~Currently it's our own implementation, no external library.~~
- date:
~~2010-01-30 19:05:58.0~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.YangNingYuan~~
- comment:
~~3ks Sergiu Dumitriu :?~~
- date:
~~2010-02-12 02:22:24.0~~
- replyto:
~~2~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.MirkoMittmann~~
- comment:
~~In the pom.xml must always be the actual xwiki-core-version.~~
- date:
~~2010-03-09 09:03:50.0~~

XWiki.XWikiComments
- author:

xwiki:XWiki.AreyouKiddingme

- comment:

The command
{{code}}
mvn archetype:generate -DarchetypeCatalog=https://github.com/xwiki/xwiki-platform/blob/master/xwiki-platform-tools/xwiki-platform-tool-archetypes/xwiki-platform-tool-archetype-component/archetype-catalog.xml
{{/code}}
fails since the URL links to the web page describing archetype-catalog.xml, not the file itself. Does anybody know the correct link?

- date:

2011-07-25 16:42:34.0

XWiki.XWikiComments

- author:

xwiki:XWiki.VincentMassol

- comment:

Fixed thanks

- date:

2011-07-25 16:51:16.0

- replyto:

5

XWiki.XWikiComments

- author:

xwiki:XWiki.AreyouKiddingme

- comment:

The corrected URL does link to a file but maven mysteriously doesn't see the catalog :(

- date:

2011-07-25 17:02:22.0

- replyto:

6

XWiki.XWikiComments

- author:

xwiki:XWiki.VincentMassol

- comment:

Thanks for reporting that. The documentation is now fixed.

- date:

2011-07-26 09:11:11.0

- replyto:

7

XWiki.XWikiComments

- author:

xwiki:XWiki.Milton

- comment:

~~How can I get the archetype of the web component by maven (m2e)? The maven repository is broken. org.xwiki.commons:xwiki-commons-component-archetype thanks!~~

- date: ~~2013-11-20 00:38:51.0~~

XWiki.XWikiComments

- author: ~~xwiki:XWiki.beldaz~~
- comment: ~~Documentation links for Execution (http://maven.xwiki.org/site/xwiki-core-parent/xwiki-core-context/apidocs/org/xwiki/context/Execution.html) are broken.~~
- date: ~~2014-11-27 06:58:03.0~~

XWiki.XWikiComments

- author: ~~xwiki:XWiki.beldaz~~
- comment: ~~The ##xwiki-core-bridge## module doesn't appear to be available through the Maven Central archetype, so to use the ##DocumentAccessBridge## you probably need to add custom repositories as explained in [[dev:Community.Building]]~~
- date: ~~2014-11-28 00:02:37.0~~

XWiki.XWikiComments

- author: ~~xwiki:XWiki.VincentMassol~~
- comment: ~~Fixed, thanks~~
- date: ~~2014-11-28 09:59:26.0~~
- replyto: ~~10~~

XWiki.XWikiComments

- author: ~~xwiki:XWiki.VincentMassol~~
- comment: ~~Only code located in the [[XWiki Commons>>https://github.com/xwiki/xwiki-commons]] and [[XWiki Rendering>>https://github.com/xwiki/xwiki-rendering]] are available through Maven Central. For the rest you need to add the XWiki Remote Maven repo to your ##settings.xml## indeed.~~
- date: ~~2014-11-28 10:00:42.0~~
- replyto:

~~11~~

- title: Writing XWiki Components ⇨
- parent: xwiki:Documentation.DevGuide.WebHome ⇨
- author: XWiki.VincentMassol ⇨ XWiki.lucaa
- syntax: XWiki 2.0 ⇨ XWiki 1.0

Documentation.DevGuide.Tutorials

Writing a custom SOLR search for an AppWithinMinutes application

Documentation.DevGuide.Tutorials.WritingCustomSolrSearchForAWM.WebHome

On 2017/09/05, the page has been created by Vincent Massol

~~This tutorial shows how to write a customized SOLR search screen in XWiki for an AWM application. For this example we have used the [["Expense Report" application>>https://goo.gl/pMWRAb]].~~
This tutorial shows how to write a customized SOLR search screen in XWiki for an AWM application. For this example we have used the [["Expense Report" application>>http://extensions.xwiki.org/xwiki/bin/view/Extension/Expense+Report+Application]].
~~Create the ##ExpenseReport.Search## page with the following content:~~
{{code}}
~~{{code language="none"}}~~
'filterQuery': ~~[~~**["class:ERCode.ERCodeClass", "type:DOCUMENT"],**
~~'type:DOCUMENT',~~**'facetFields':**
['property.ERCode.ERCodeClass.status_string','property.ERCode.ERCodeClass.organisation_string',
'property.ERCode.ERCodeClass.currency_string',
'property.ERCode.ERCodeClass.status_client',
'property.ERCode.ERCodeClass.user_string', 'author', 'creator', 'date',
~~'class:ERCode.ERCodeClass'~~**'creationdate'],**
~~],~~**'facetDisplayers': {**
~~'facetFields': [~~
~~'property.ERCode.ERCodeClass.status_string',~~
~~'property.ERCode.ERCodeClass.organisation_string',~~
~~'property.ERCode.ERCodeClass.currency_string',~~
~~'property.ERCode.ERCodeClass.user_string',~~
~~'author',~~
~~'creator',~~
~~'date',~~
~~'creationdate'~~
~~],~~
~~'facetDisplayers': {~~
We need to provide a code for the facet for list fields, as it is not provided by default in ~~XWiki. Put the following code in the content of ##ExpenseReport.ListFacet##:~~**XWiki.**
Here is the result of our customized SOLR ~~search (from the ##ExpenseReport.Search## page):~~**search:**

customsolrsearch.png: 1.1 ⇨ ()

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ DevGuide.WebHome

- author: XWiki.evalica ⇨ XWiki.ludovic
- syntax: XWiki 2.0 ⇨ XWiki 2.1

On 2017/09/05, the page has been modified by Vincent Massol

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WebHome
- author: XWiki.evalica ⇨ XWiki.VincentMassol

Documentation.DevGuide.Tutorials

Writing an Event Listener

Documentation.DevGuide.Tutorials.WritingEventListenerTutorial.WebHome

On 2017/09/05, the page has been created by Vincent Massol

{{box cssClass="floatinginfobox" ~~title="**Contents**"}}**title="**Contents**"}}{{toc/}}{{/box}}~~
~~{{toc/}}~~
~~{{/box}}~~
This example uses the ~~[[~~{~~{groovy}}~~**[[~{~{groovy}}** macro>>extensions:Extension.Groovy Macro]] and thus requires having Programming Rights. If you're in a wiki in a farm this means having Programming Rights on the main wiki.
~~{{/groovy}}~~**{{/groovy}}{{/code}}**
~~{{/code}}~~
~~=~~**===** Log when a document is modified ~~=~~**===**
This example uses the ~~[[~~{~~{groovy}}~~**[[~{~{groovy}}** macro>>extensions:Extension.Groovy Macro]] and thus requires having Programming Rights. If you're in a wiki in a farm this means having Programming Rights on the main wiki.
~~import org.xwiki.model.reference.*~~
~~import org.xwiki.model.*~~
def ~~logReference = new EntityReference("Logs", EntityType.DOCUMENT,~~ **xwiki**
~~new EntityReference("Main", EntityType.SPACE))~~**def context**
LoggingEventListener(xwiki, context)
{
this.xwiki = xwiki
this.context = context
}
void onEvent(Event event, Object source, Object ~~context)~~**data)**
def ~~xwiki~~**document** = ~~context.getWiki()~~**xwiki.getDocument("Main.Logs")**
~~def document = xwiki.getDocument(logReference, context)~~
~~xwiki.saveDocument(document, "Logging~~**document.save("Logging** event", ~~true, context)~~**true)**
def listener = new ~~LoggingEventListener()~~**LoggingEventListener(xwiki, xcontext)**
return ~~[new~~**Arrays.asList(new** DocumentUpdateEvent(), new DocumentSaveEvent(), new ~~DocumentDeleteEvent()]~~**DocumentDeleteEvent())**
~~= Send~~**And here's the [[full code with nice pretty-printing and** a ~~mail whenever a comment is added =~~**register button>>attach:groovynotifier.txt]].**
== Send a mail whenever a comment is added ==
~~{{code language="xml"}}~~
~~{{/code}}~~

~~{{info}}~~
~~If you are experiencing problems trying to build it, see:~~
~~http://dev.xwiki.org/xwiki/bin/view/Community/Building~~
~~{{/info}}~~
~~= Old Notifications Tutorial =~~
~~If you're using an old version of XWiki (prior to 2.0) you should check this [[old Notifications Tutorial>>GroovyNotificationTutorial]].~~

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ DevGuide.WebHome

On 2017/09/05, the page has been modified by Vincent Massol

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WebHome

Documentation.DevGuide.Tutorials

Writing XWiki Rendering Macros

Documentation.DevGuide.Tutorials.WritingMacros.WebHome

Between 2017/09/05 and 2017/09/05, the page has been modified 2 times, by 1 user: Vincent Massol

* [[XWiki Rendering Macro in Wiki page ~~Tutorial>>Documentation.DevGuide.WikiMacroTutorial]]~~**Tutorial>>Documentation.DevGuide.Tutorials.WritingMacros.WikiMacroTutorial.WebHome]]**

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WebHome
- author: XWiki.evalica ⇨ XWiki.VincentMassol

On 2017/09/05, the page has been created by Vincent Massol

~~These tutorials explain how to implement~~**1 Writing An** XWiki ~~Macros using XWiki's [[Rendering Architecture>>extensions:Extension.Rendering Module]].~~**Macro**
~~XWiki Rendering macros (available in XWiki Syntax 2.0 only) can be written either in Java or directly in wiki pages:~~
~~* [[XWiki Rendering Macro in Java Tutorial>>platform:DevGuide.RenderingMacroTutorial]]~~
~~* [[XWiki Rendering Macro in Wiki page Tutorial>>platform:DevGuide.WikiMacroTutorial]]~~

XWiki.XWikiComments

- author: ~~xwiki:XWiki.simba~~
- comment: ~~Is it possible to support other parameter types other than string? For example, boolean, enumeration, even file, color or image. It's even better to have a image picker, or color picker.~~
- date: ~~2009-11-14 08:37:27.0~~

- title: Writing XWiki Rendering Macros ⇨ Writing An XWiki Macro
- parent: xwiki:Documentation.DevGuide.WebHome ⇨ DevGuide.Tutorials

- author: XWiki.VincentMassol ⇨ XWiki.asiri
- syntax: XWiki 2.0 ⇨ XWiki 1.0

Documentation.DevGuide.Tutorials.WritingMacros

Writing XWiki Rendering Macros in wiki pages

Documentation.DevGuide.Tutorials.WritingMacros.WikiMacroTutorial.WebHome

On 2017/09/05, the page has been modified by Vincent Massol

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:Documentation.DevGuide.Tutorials.WritingMacros.WebHome

On 2017/09/05, the page has been created by Vincent Massol

~~{{box cssClass="floatinginfobox" title="**Contents**"}}~~**1 Wiki Macro Tutorial**
~~{{toc/}}~~
~~{{/box}}~~
~~Wiki macros allow macro authors to develop reusable and distributable macro modules. There is no java code involved; hence no compiling or packaging. The macro author simply needs to create a wiki page according to a particular specification and that's all!~~**#error("Under development")**
~~This page is a tutorial but you can also access the [[reference documentation for the Wiki Macro feature>>extensions:Extension.WikiMacroBridge Application]].~~
~~= Macro Visibility and Rights =~~
~~There are 3 levels of visibility for a macro:~~
~~* farm (if we're in a multiwiki environment), meaning that the macro will be available in all the wikis of the farm~~
~~* wiki, which means that the macro will be available in its wiki~~
~~* user, which means that the macro will only be available to the user which is its author~~
~~The rights required to create macros are different depending on the visibility we want for our macro:~~
~~* the macro author needs to have **programming** rights for a macro available in the whole **farm**~~
~~* the macro author needs to have **admin** rights for a macro available in its **wiki**~~
~~* no special rights besides the obvious right to edit the page are needed for a macro available only to its author.~~
~~== Using protected API in wiki macros ==~~
~~Also, if the macro needs to use [[protected API>>Scripting||anchor="HXWikiCoreAccess"]], the author of the macro will need to have programming rights. Note that the macro will always be executed with the rights of its author, and not with the rights of the author of the calling document (the document using the macro). Specifically, if the macro uses protected API, only the macro author needs to have programming rights, not all the authors of the documents that call this macro.~~
~~= Hello Macro =~~
~~We are going to start with a very simple xwiki/2.0 wiki macro which prints a greeting message to the document content. It isn't a very useful macro but the idea is to get you familiarised with the wiki macro creation process.~~
~~== Definition ==~~
~~Wiki macros are defined using objects of type ##XWiki.WikiMacroClass##. You define a wiki macro by creating a new wiki page and attaching to it an~~

~~object of type ##XWiki.WikiMacroClass##.~~
~~{{warning}}~~
~~There can be only one object of type ##XWiki.WikiMacroClass## per wiki page (if you add more only the first will be used).~~
~~{{/warning}}~~
~~This class contains the following fields:~~
~~* Macro id: Id of the macro to be used by users when invoking your macro from wiki code~~
~~* Macro name: Name of the macro to be displayed on the wysiwyg editor~~
~~* Macro description: A short description of the macro to be displayed on the WYSIWYG editor~~
~~* Default category: Default category under which this macro should be listed~~
~~* Supports inline mode: Whether the macro can be used in an inline context or not~~
~~* Macro content type: Whether this macro should support a body or not~~
~~* Content description: A short description about the macro's content to be displayed on the WYSIWYG editor~~
~~* Macro code: The actual wiki code that will be evaluated when the macro is executed, can be any xwiki content (should be in the same syntax as the document)~~
~~Now we can define our hello macro as shown below:~~
~~[[image:macro1.png]]~~
~~== Invocation ==~~
~~A wiki macro can be invoked just like any other macro is invoked. Since we are writing a xwiki/2.0 wiki macro, we can invoke our **hello macro** as below:~~
~~{{code}}~~
~~{{hello/}}~~
~~{{/code}}~~
~~And if you view the result it would say "Hello World!" (of course).~~
~~== Content ==~~
~~If macro content is used, it can be shown by executing the following velocity code in the macro body:~~
~~{{code language="none"}}~~
~~{{velocity}}$xcontext.macro.content{{/velocity}}~~
~~{{/code}}~~
~~For more details, see the [[Scripting Tips section below>>||anchor="HScriptingTips"]].~~
~~== Parameters ==~~
~~Introducing a parameter to a wiki macro is pretty straight forward; you simply need to add an object of type ##XWiki.WikiMacroParameterClass## into your wiki macro document (one object per parameter). This class contains several fields that allow you to define your parameter clearly:~~
~~* Parameter name: Name of the parameter, users will refer this name when invoking your macro with parameters~~
~~* Parameter description: A short description of the parameter, this description will be made available on the WYSIWYG editor~~
~~* Parameter mandatory: Indicates if this particular parameter is mandatory, wiki macro will fail to execute if a mandatory parameter is missing~~
~~Now we're going to extend our **hello macro** with a parameter. We will introduce a parameter named //greetUser// that will indicate if the greeting message should be tailored for the current user viewing the page. The definition of the parameter is shown below:~~
~~[[image:macro3.png]]~~
~~A macro parameter defined this way can be accessed from any scripting language within the macro code. For example, we are going to utilize our~~

//greetUser// parameter within **hello macro** as shown below:
[[image:macro4.png]]
As you might have realized already, direct binding of parameters is not supported at the moment. That is, you cannot access //greetUser// parameter with **$greetUser**. Instead you must use **$xcontext.macro.params.greetUser**. We plan to introduce some form of direct parameter binding in near future.
Finally, we can test our new version of **hello macro** with the following invocation:
{{code language="none"}}
{{hello greetUser="true"/}}
{{/code}}
== Translations ==
When your macro is ready, you might want to provide the description of the macro and its parameters in different languages. For that, you need to create a set of translation keys and values (as described [[here>>DevGuide.InternationalizingApplications]]) and then just use the following convention for the keys you add in this storage (no modification is needed on the macro itself, the association of the translations to the macro is done based on a convention of the form of the translation keys):
{{code}}
rendering.macro.<macro id>.name=Name of the macro, displayed in the macros list in the macros wizard
rendering.macro.<macro id>.description=Description of the macro, displayed as a help in the macros list in the macros wizard
rendering.macro.<macro id>.parameter.<parameter name>.name=Name of the macro parameter, to be displayed in the form for the macro settings in the macros wizard
rendering.macro.<macro id>.parameter.<parameter name>.description=Description of the macro parameter, to be displayed as a help in the form for the macro settings in the macros wizard
{{/code}}
Don't forget to make sure that the visibility of the translations is the same as the visibility of the macro, so that anywhere you use the macro you also have the translations.
In our example, french translations would be something like this:
{{code}}
rendering.macro.hello.name=Macro pour dire bonjour
rendering.macro.hello.description=Ceci est une macro qui va dire "Bonjour" a l'utilisateur
rendering.macro.hello.parameter.greetUser.name=Personnaliser le message
rendering.macro.hello.parameter.greetUser.description=Personnaliser le message pour l'utilisateur courant en train de visualiser la page. Les valeurs possibles sont "true" (oui) et "false" (non).
{{/code}}
= WYSIWYG Access =
A wiki macros is treated just like any other rendering macro in the system. As such, the moment you save your wiki macro it will be available to the users through the WYSIWYG editor's **Insert Macro** dialog box:
[[image:macro2.png]]
[[image:macro5.png]]
== Special code for WYSIWYG edit mode ==
Even in edit mode, the WYSIWYG editor will execute the macro and feed the result back into the document. If your macro use some JSX, these will not be loaded. But, if your macro produce some Javascript that use those JSX or manipulate the document's DOM (injecting new elements, moving existing elements, removing elements, etc.), you may want to protect the

existing elements, removing elements, etc.), you may want to protect the content in WYSIWYG edit mode in order to prevent the performed transformation to get saved. Here is how you can prevent this behavior:
{{code language="velocity"}}
{{velocity}}
#if("$xcontext.action" != "edit")
{{html}}
<script type="text/javascript">
//<![CDATA[
... some javascript ...
//]]>
</script>
{{/html}}
#end
##
Rest of the code.
{{/velocity}}
{{/code}}
= Scripting Tips =
Following are a few useful hints if you plan to do advanced scripting inside your wiki macros:
* Access parameters: Use the context object (Ex. ##$xcontext.macro.params.param1##)
* Access macro body (if your macro defines one): Use the context object (Ex. ##$xcontext.macro.content##)
* Access [[MacroTransformationContext>>http://svn.xwiki.org/svnroot/xwiki/platform/core/trunk/xwiki-rendering/xwiki-rendering-api/src/main/java/org/xwiki/rendering/transformation/MacroTransformationContext.java]]: Use the context object (Ex. ##$xcontext.macro.context##)
* Since 2.4M1, it's possible to directly return the desired list of rendering blocks without having to render them first to let them be parsed back by the macro transformation. The benefits are that it could be a lots quicker and most of all it means supporting syntax which does not provide any renderer. It also makes it possible to generate some XDOM which is impossible to write in any some syntax. For example the following wiki macro is generating a LinkBlock targeting a relative URL:(((
{{code language="groovy"}}
{{groovy}}
import java.util.Collections;
import org.xwiki.rendering.listener.Link;
import org.xwiki.rendering.block.WordBlock;
import org.xwiki.rendering.block.LinkBlock;
ref link = new Link();
link.setReference("/xwiki/edit/Main/WebHome");
link.setType(LinkType.URI);
ref linkBlock = new LinkBlock(Collections.singletonList(new WordBlock("Edit home page"))), link, false);
xcontext.macro.result = Collections.singletonList(linkBlock)
{{/groovy}}
This text will not appear in the result.
{{/code}}
)))
* If you are using ##$xcontext.macro.content## in your velocity macro, that content will not be able to support scripting, since nested scripting is not supported. To workaround that limitation, thanks to the above, you may do the parsing yourself using the rendering service. Here is a small sample:(((
{{code languege="velocity"}}

~~{{velocity output="no"}}~~
~~## get the macro content in a velocity string~~
~~#set($wikiresult = $xcontext.macro.content)~~
~~## Add a wrapping div as a sample of the action of this macro~~
~~#set($wikiresult = "(% class='newstyle' %)(((${wikiresult})))")~~
~~## parse the string and return the resulting blocks~~
~~#set($xcontext.macro.result = $services.rendering.parse($wikiresult, $xwiki.getCurrentContentSyntaxId()).getChildren())~~
~~{{/velocity}}~~
~~{{/code}}~~
~~)))~~
~~* Since 9.1RC1 you can access the macro descriptor using ##$xcontext.macro.descriptor## binding. It returns a ##org.xwiki.rendering.macro.descriptor.MacroDescriptor## Java object.~~
~~= Troubleshooting =~~
~~== A Pitfall of Optional Parameters ==~~
~~{{info}}~~
~~This pitfall has been fixed in XWiki 2.2~~
~~{{/info}}~~
~~There is a common pitfall for using optional paramters. The following macro code contains a not so obvious bug:~~
~~{{code languege="velocity"}}~~
~~{{velocity}}~~
~~#set($greetUser=$xcontext.macro.params.greetUser)~~
~~#if ("true" == $greetUser && "XWiki.XWikiGuest" != "$xcontext.user")~~
~~Hello $xwiki.user.email!~~
~~#else~~
~~Hello world!~~
~~#end~~
~~~~
~~{{/code}}~~
~~If we invoke it twice in a row:~~
~~{{code}}~~
~~{{hello greetUser="true" /}}~~
~~{{hello /}}~~
~~{{/code}}~~
~~The second invocation will not print "Hello World!" as we'd expect. But it will print the same result as the first invocation. The reasons are:~~
~~* Macro parameters are implemented as global parameters. So, they remain the same across multiple macro invocations.~~
~~* If ##$xcontext.macro.params.greetUser## contains "null", it will not be assigned to ##$greetUser##. This is different from C/C++ or Java.~~
~~So in order to get around it, you can use:~~
~~{{code}}~~
~~#set($greetUser="$!xcontext.macro.params.greetUser")~~
~~{{/code}}~~

macro1.png: 1.9 ⇨ ()

macro2.png: 1.4 ⇨ ()

macro3.png: 1.4 ⇨ ()

macro4.png: 1.6 ⇨ ()

macro5.png: 1.3 ⇨ ()

XWiki.XWikiComments
- author:
~~xwiki:XWiki.StefanBachert~~
- comment:
~~Note parameters are of type string. passing other types like a list of string will fail.~~
- date:
~~2010-02-23 10:46:55.0~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.VincentMassol~~
- comment:
~~Stefan, what you say is not correct. You just cannot pass any type other than String so there's no chance of making a mistake. I believe you haven't read the XWiki Syntax Reference page well enough. Please check it again.~~
- date:
~~2010-02-24 08:25:25.0~~

XWiki.XWikiComments
- author:
~~xwiki:XWiki.JorisDirks~~
- comment:
~~I'd love to have a drop-down selection for parameters in the WYSIWYG editor, like the toc macro has.~~
- date:
~~2010-07-28 17:25:34.0~~

- title: Writing XWiki Rendering Macros in wiki pages ⇨ Wiki Macro Tutorial
- parent: xwiki:Documentation.DevGuide.WebHome ⇨
- author: XWiki.VincentMassol ⇨ XWiki.asiri
- syntax: XWiki 2.1 ⇨ XWiki 1.0

Documentation.DevGuide

UIXTutorial
Documentation.DevGuide.UIXTutorial

On 2017/09/05, the page has been deleted by Vincent Massol

URL Architecture
Documentation.DevGuide.URL Architecture

On 2017/09/05, the page has been deleted by Vincent Massol

Documentation.DevGuide.URL Architecture

Standard URL Format

Documentation.DevGuide.URL Architecture.Standard URL Format.WebHome

On 2017/09/05, the page has been deleted by Vincent Massol
On 2017/09/05, the page has been created by Vincent Massol

XWiki.RedirectClass
location:
xwiki:Documentation.DevGuide.Architecture.URL Architecture.Standard URL Format.WebHome

- hidden: false ⇨ true

Documentation.DevGuide

URL Architecture

Documentation.DevGuide.URL Architecture.WebHome

On 2017/09/05, the page has been deleted by Vincent Massol
On 2017/09/05, the page has been created by Vincent Massol

XWiki.RedirectClass
location:
xwiki:Documentation.DevGuide.Architecture.URL Architecture.WebHome

- hidden: false ⇨ true

Documentation.DevGuide

UsingJQuery

Documentation.DevGuide.UsingJQuery

On 2017/09/05, the page has been deleted by Vincent Massol

Developer Guide

Documentation.DevGuide.WebHome

Between 2017/09/05 and 2017/09/05, the page has been modified 33 times, by 2 users: Vincent Massol, Ecaterina Moraru (Valica)

==== (% class="card-title" ~~%)[[XWiki API Guide>>platform:DevGuide.APIGuide]](%%)~~%)**[[API Guide>>Documentation.DevGuide.Scripting.APIGuide.WebHome]](%%)** ====
==== (% class="card-title" ~~%)[[XWiki API Reference>>platform:DevGuide.API]](%%)~~%)**[[API Reference>>Documentation.DevGuide.API.WebHome]](%%)** ====
==== (% class="card-title" ~~%)[[XWiki Scripting>>platform:DevGuide.Scripting]](%%)~~%) **[[Scripting>>Documentation.DevGuide.Scripting.WebHome]](%%)** ====
===== (% class="card-title" ~~%)[[XWiki Architecture>>platform:DevGuide.Architecture]](%%)~~%) **[[Architecture>>Documentation.DevGuide.Architecture.WebHome]](%%)** =====

~~===== (% class="card-title" %)[[XWiki Data Model>>platform:DevGuide.DataModel]](%%)~~**%)[[Data Model>>Documentation.DevGuide.DataModel]](%%) =====**
~~===== (% class="card-title" %)[[Database Schema>>platform:DevGuide.DatabaseSchema]] (%%)~~**Schema>>Documentation.DevGuide.DatabaseSchema]](%%) =====**
~~[[Subscribe to~~**[[Post on** the ~~mailing lists!>>dev:Community.MailingLists]]~~**forum!>>dev:Community.Discuss]]**
; **[[Front-end ~~Resources>>platform:DevGuide.FrontendResources]]~~****Resources>>Documentation.DevGuide.FrontendResources]]****
; **[[Saving ~~Data>>platform:DevGuide.SavingData]]~~****Data>>Documentation.DevGuide.Tutorials.SavingData.WebHome]]****
; **[[URL ~~Architecture>>platform:DevGuide.URL Architecture]]~~****Architecture>>Documentation.DevGuide.Architecture.URL Architecture.WebHome]]****
: Explains how XWiki URLs work, the format of the [[Standard URL ~~Scheme>>platform:DevGuide.Standard~~**Scheme>>Documentation.DevGuide.Architecture.URL Architecture.Standard** URL ~~Format]]~~**Format.WebHome]]** and how to configure them.
; ~~**[[XWiki Module~~****[[Module** Reference>>extensions:Main.WebHome]]**
== Tutorials **& Guides** ==
~~===== (% class="card-title" %)[[Creating a FAQ application>>FAQTutorial]] (%%)~~**application>>Documentation.DevGuide.Tutorials.FAQTutorial.WebHome]](%%) =====**
~~===== (% class="card-title" %)[[Creating a TODO application in XWiki>>http://www.theserverside.com/tt/articles/article.tss?l=XWiki]] (%%)~~**%) [[Skin eXtensions>>Documentation.DevGuide.Tutorials.SkinExtensionsTutorial.WebHome]] (%%) =====**
~~====== Another helpful tutorial to using~~**Illustrates** XWiki's ~~classes, velocity scripts, hibernate queries~~**[[Skin eXtensions>>platform:Features.Skins||anchor="HSkinseXtensions"]] mechanism that allows developers to pull extra Javascript and StyleSheet files on demand or** for **all wiki pages without modifying the wiki templates and skin. Creation minimal Stylesheet and Javascript extensions, and continue with** a ~~dashboard.~~**complete example of a real-world extension. ======**
~~===== (% class="card-title" %)[[Creating~~**%) [[Creating** an ~~Extension>>Documentation.DevGuide.CreatingExtensions.WebHome]] (%%)~~**XWiki Component>>Documentation.DevGuide.WritingComponents]] (%%) =====**
~~====== Explains how~~**Getting started with the new XWiki components architecture: introduction** to ~~create~~**this new design, a step-by-step guide to writing** an XWiki ~~Extension~~**component (the recommended way of creating XWiki plugins) and some guidelines and development good practices in this area. ======**
~~--~~(% class="row cards" %)
(((
(% class="col-xs-12 col-sm-6 col-md-4 cards-container" %)
(((
(% class="card card2" %)
(((
(% class="card-body" %)
(((

(((
===== (% class="card-title" %)[[Creating Rendering Macros>>Documentation.DevGuide.WritingMacros]](%%) =====
; **[[Creating an XWiki Rendering Macros>>platform:DevGuide.WritingMacros]]**(% class="card-subtitle mb-2" %)
: How to develop XWiki Rendering macros (available in XWiki Syntax 2.0 only)(((
====== How to develop XWiki Rendering macros (available in XWiki Syntax 2.0+ only). ======
)))
)))
)))
)))
; **[[Creating New XClass Property Types>>platform:DevGuide.CreatingNewXClassPropertyTypes]]**(% class="col-xs-12 col-sm-6 col-md-4 cards-container" %)
(((
(% class="card card2" %)
(((
(% class="card-body" %)
(((
===== (% class="card-title" %)[[Creating an Extension>>Documentation.DevGuide.Tutorials.CreatingExtensions.WebHome]](%%) =====
(% class="card-subtitle mb-2" %)
(((
====== Explains how to create an XWiki Extension. ======
)))
)))
)))
)))
(% class="col-xs-12 col-sm-6 col-md-4 cards-container" %)
(((
(% class="card card2" %)
(((
(% class="card-body" %)
(((
===== (% class="card-title" %) [[Creating a Groovy Class>>Documentation.DevGuide.GroovyClassHelloWorldTutorial]] (%%) =====
(% class="card-subtitle mb-2" %)
(((
====== Demonstrates how to put Groovy code in one page and then reuse it by importing it from another page. ======
)))
)))
)))
)))
)))

; **[[Creating New XClass Property Types>>Documentation.DevGuide.CreatingNewXClassPropertyTypes]]**
; **[[Creating a form with validation and tooltips>>platform:DevGuide.Creating a form with validation and tooltips]]**using UI Extension Points>>doc:Documentation.DevGuide.Tutorials.UIXTutorial.WebHom

e]]**
: UI Extension Points are ways to modify the UI without having to write a full skin. Learn how to create new Extension Points and how to use existing ones.
; **[[Creating a form with validation and tooltips>>Documentation.DevGuide.Creating a form with validation and tooltips]]**
; **[[Creating a Tree ~~View>>platform:DevGuide.CreatingATreeView]]**~~View>>Documentation.DevGuide.CreatingATreeView]]**
; **[[Creating a custom SOLR search for an AppWithinMinutes ~~application>>platform:DevGuide.WritingCustomSolrSearchForAWM]]**~~application>>Documentation.DevGuide.WritingCustomSolrSearchForAWM]]**
; **[[Creating a Master-Detail ~~view>>platform:DevGuide.MasterDetailTutorial]]**~~view>>Documentation.DevGuide.MasterDetailTutorial]]**
; **[[Creating an Event ~~Listener>>platform:DevGuide.WritingEventListenerTutorial]]**~~Listener>>Documentation.DevGuide.WritingEventListenerTutorial]]**
; **[[Creating a ~~Groovy Class>>platform:DevGuide.GroovyClassHelloWorldTutorial]]**~~Plugin>>Documentation.DevGuide.CreatingPlugins]]**
~~: Demonstrates how to put Groovy code in one page and then reuse it by importing it from another page.~~
~~; **[[Creating an XWiki Component>>platform:DevGuide.WritingComponents]]**~~
~~: Getting started with the new XWiki components architecture: introduction to this new design, a step-by-step guide to writing an XWiki component (the recommended way of creating XWiki plugins) and some guidelines and development good practices in this area.~~
~~; **[[Creating a Plugin>>platform:DevGuide.CreatingPlugins]]**~~
~~(% class="row cards" %)~~; **[[Creating a TODO application in XWiki>>http://www.theserverside.com/tt/articles/article.tss?l=XWiki]]**
~~(((~~: Another helpful tutorial to using XWiki's classes, velocity scripts, hibernate queries for a dashboard.
~~)))~~; **[[How to use Velocity in wiki pages>>Documentation.DevGuide.XWikiVelocityTraining]]**
~~== Guides ==~~
~~; **[[How to use Velocity in XWiki pages>>platform:DevGuide.XWikiVelocityTraining]]**~~
; ~~**[[Skin eXtensions>>platform:DevGuide.SkinExtensionsTutorial]]**~~**[[Internationalizing Applications>>Documentation.DevGuide.Tutorials.InternationalizingApplications.WebHome]]**
~~: Illustrate XWiki's [[Skin eXtensions>>platform:Features.Skins||anchor="HSkinseXtensions"]] mechanism that allows developers to pull extra Javascript and stylesheet files on demand or for all wiki pages without modifying the wiki templates and skin. The tutorial guides through the creation of minimal stylesheet and Javascript extensions, and continues with a complete exemple of a real-world extension to demonstrate the power of XWiki's skin eXtensions!~~
~~; **[[Internationalizing Applications>>platform:DevGuide.InternationalizingApplications]]**~~
; **[[Custom ~~Mapping>>platform:DevGuide.CustomMapping]]**~~Mapping>>Documentation.DevGuide.Tutorials.CustomMapping.WebHome]]**

; **[[Integrating JavaScript ~~Libraries>>platform:DevGuide.IntegratingJavaScriptLibraries]]~~**Libraries>>Documentation.DevGuide.IntegratingJavaScriptLibraries]]**
; **[[Performing Asynchronous ~~Tasks>>platform:DevGuide.PerformingAsynchronousTasks]]~~**Tasks>>Documentation.DevGuide.Tutorials.PerformingAsynchronousTasks.WebHome]]**
; **[[Best ~~Practices>>platform:DevGuide.BestPractices]]~~**Practices>>Documentation.DevGuide.BestPractices.WebHome]]**
See the ~~[[Drafts>>dev:Drafts.WebHome]]~~**[[Drafts page>>dev:Drafts.WebHome]]** for other topics that are still under review.

- author: XWiki.evalica ⇨ XWiki.VincentMassol

WikiMacroTutorial
Documentation.DevGuide.WikiMacroTutorial

On 2017/09/05, the page has been deleted by Vincent Massol

WritingComponents
Documentation.DevGuide.WritingComponents

On 2017/09/05, the page has been deleted by Vincent Massol

WritingCustomSolrSearchForAWM
Documentation.DevGuide.WritingCustomSolrSearchForAWM

On 2017/09/05, the page has been deleted by Vincent Massol

WritingEventListenerTutorial
Documentation.DevGuide.WritingEventListenerTutorial

On 2017/09/05, the page has been deleted by Vincent Massol

WritingMacros
Documentation.DevGuide.WritingMacros

On 2017/09/05, the page has been deleted by Vincent Massol

XWikiLinks
Documentation.DevGuide.XWikiLinks

On 2017/09/05, the page has been deleted by Vincent Massol

XWikiVelocityTraining
Documentation.DevGuide.XWikiVelocityTraining

On 2017/09/05, the page has been deleted by Vincent Massol

Documentation.DevGuide

XWikiVelocityTraining
Documentation.DevGuide.XWikiVelocityTraining.WebHome

On 2017/09/05, the page has been deleted by Vincent Massol
On 2017/09/05, the page has been created by Vincent Massol

XWiki.RedirectClass
location:
xwiki:Documentation.DevGuide.Scripting.XWikiVelocityTraining.WebHome

- hidden: false ⇨ true

XWikiVelocityTrainingCodeMacro

Documentation.DevGuide.XWikiVelocityTraining.XWikiVelocityTrainingCodeMacro

Between 2017/09/05 and 2017/09/05, the page has been modified 2 times, by 1 user: Vincent Massol

XWiki.RedirectClass
location:
xwiki:Documentation.DevGuide.Scripting.XWikiVelocityTraining.XWikiVelocityTrainingCodeMacro

- hidden: false ⇨ true

On 2017/09/05, the page has been created by Vincent Massol

XWiki.RedirectClass
location:
xwiki:Documentation.DevGuide.Scripting.XWikiVelocityTraining.XWikiVelocityTrainingCodeMacro

- hidden: false ⇨ true

XWikiVelocityTrainingImageMacro

Documentation.DevGuide.XWikiVelocityTraining.XWikiVelocityTrainingImageMacro

Between 2017/09/05 and 2017/09/05, the page has been modified 3 times, by 1 user: Vincent Massol

XWiki.RedirectClass
location:
xwiki:Documentation.DevGuide.Scripting.XWikiVelocityTraining.XWikiVelocityTrainingImageMacro

- hidden: false ⇨ true

On 2017/09/05, the page has been created by Vincent Massol

XWiki.RedirectClass
location:
xwiki:Documentation.DevGuide.Scripting.XWikiVelocityTraining.XWikiVelocityTrainingImageMacro

- hidden: false ⇨ true

Documentation.DevGuide

XWikiVelocityTrainingCodeMacro
Documentation.DevGuide.XWikiVelocityTrainingCodeMacro

On 2017/09/05, the page has been deleted by Vincent Massol

XWikiVelocityTrainingImageMacro
Documentation.DevGuide.XWikiVelocityTrainingImageMacro

On 2017/09/05, the page has been deleted by Vincent Massol

velocityHqlExamples
Documentation.DevGuide.velocityHqlExamples

On 2017/09/05, the page has been deleted by Vincent Massol

FAQ

How to include a velocity page into another page?
FAQ.IncludeInVelocity

Between 2017/09/05 and 2017/09/05, the page has been modified 5 times, by 1 user: Vincent Massol

~~{{box cssClass="floatinginfobox" title="**Contents**"}}~~
~~{{toc/}}~~
~~{{/box}}~~
~~Here are several solutions to include a page containing Velocity script into another page.~~
~~= Use the ~{~{include}} Macro =~~
~~That's the recommended way when using XWiki Syntax 2.0+. See the [[Include Macro>>doc:extensions:Extension.Include Macro]].~~
~~Example:~~
~~{{code}}~~
~~{{include reference="some.page.containing.velocity"/}}~~
~~{{/code}}~~
~~= Include a *.vm script =~~
~~These macros take as parameter a relative URL to a {{code}}*.vm{{/code}} script.~~
~~== template() ==~~
~~{{info}}~~
~~Specification: [[extensions:Extension.Template Macro]]~~
~~{{/info}}~~
~~This is the equivalent of the C preprocessor include directive: the result is a copy of the script passed as argument, into your script.~~
~~= Using the old #include* Velocity macros =~~
~~These macros take as parameter the XWiki name of the page, ##Space.Page##.~~
~~== includeInContext() ==~~
~~{{info}}~~
~~Specification: [[extensions:Extension.Include In Context Macro]]~~
~~{{/info}}~~
~~If you have a script stored as an XWiki page you can include its code into another script by using includeInContext:~~
~~{{code}}~~
~~#includeInContext("mySpace.myPage")~~
~~{{/code}}~~

~~== includeForm() ==~~
~~{{info}}~~
~~Specification: [[extensions:Extension.Include Form Macro]]~~
~~{{/info}}~~
~~Similar to includeInContext(), except that using includeForm() in a page will set the default edit mode for that page as "inline". For example, the script you want to include uses the method~~
~~##com.xpn.xwiki.doc.XWikiDocument.display()## to access a field of an object attached to the including page, like a typical class sheet:~~
~~{{code language="velocity"}}~~
~~## Change class name to your class name~~
~~#set($class = $doc.getObject("CompanionTemplateSystem.CompanionDocumentMetaDataClass").xWikiClass)~~
~~<dl>~~
~~#foreach($prop in $class.properties)~~
~~#if(($prop.getName() != "Copyright") && ($prop.getName() != "TargetGroup"))~~
~~<dt> *${prop.prettyName}* </dt>~~
~~<dd>$doc.display($prop.getName())</dd>~~
~~#end~~
~~#end~~
~~</dl>~~
~~{{/code}}~~
~~which is stored as a regular XWiki page (let's say myPage in mySpace). You will include the page using includeForm():~~
~~{{code}}~~
~~#includeForm("mySpace.myPage")~~
~~{{/code}}~~
~~This way, the including page will always be edited in inline mode by default. Thus in the example of the class sheet, the including page will display input fields mapped to the respective object fields.~~
~~If you prefer to keep the default edit as set in your XWiki.XWikiPreferences class, you can use the other macros, and still edit a page in inline mode: choose the "Inline form" option in the "Edit" menu, provided your user type is advanced.~~
~~== includeTopic() ==~~
~~{{info}}~~
~~Specification: [[extensions:Extension.Include Topic Macro]]~~
~~{{/info}}~~
~~Same syntax as includeInContext(). Contrary to includeInContext(), the included XWiki page is interpreted in its own context. For example, you would include a page using includeTopic() if the included page had to access its own objects in order for the including page to be displayed properly.~~

FAQCode.FAQClass
answer:
{{box cssClass="floatinginfobox" title="Contents**"}}**
{{toc/}}
{{/box}}
Here are several solutions to include a page containing Velocity script into another page.
= Use the ~{~{include}} Macro =
That's the recommended way when using XWiki Syntax 2.0+. See the [[Include Macro>>doc:extensions:Extension.Include Macro]].
Example:
{{code}}

{{code}}
{{include reference="some.page.containing.velocity"/}}
{{/code}}
= Include a *.vm script =
These macros take as parameter a relative URL to a {{code}}*.vm{{/code}} script.
== template() ==
{{info}}
Specification: [[extensions:Extension.Template Macro]]
{{/info}}
This is the equivalent of the C preprocessor include directive: the result is a copy of the script passed as argument, into your script.
= Using the old #include* Velocity macros =
These macros take as parameter the XWiki name of the page, ##Space.Page##.
== includeInContext() ==
{{info}}
Specification: [[extensions:Extension.Include In Context Macro]]
{{/info}}
If you have a script stored as an XWiki page you can include its code into another script by using includeInContext:
{{code}}
#includeInContext("mySpace.myPage")
{{/code}}
== includeForm() ==
{{info}}
Specification: [[extensions:Extension.Include Form Macro]]
{{/info}}
Similar to includeInContext(), except that using includeForm() in a page will set the default edit mode for that page as "inline". For example, the script you want to include uses the method ##com.xpn.xwiki.doc.XWikiDocument.display()## to access a field of an object attached to the including page, like a typical class sheet:
{{code language="velocity"}}
Change class name to your class name
#set($class = $doc.getObject("CompanionTemplateSystem.CompanionDocumentMetaDataClass").xWikiClass)
<dl>
#foreach($prop in $class.properties)
#if(($prop.getName() != "Copyright") && ($prop.getName() != "TargetGroup"))
<dt> *${prop.prettyName}* </dt>
<dd>$doc.display($prop.getName())</dd>
#end
#end
</dl>
{{/code}}
which is stored as a regular XWiki page (let's say myPage in mySpace).
You will include the page using includeForm():
{{code}}
#includeForm("mySpace.myPage")
{{/code}}
This way, the including page will always be edited in inline mode by default. Thus in the example of the class sheet, the including page will display input fields mapped to the respective object fields.
If you prefer to keep the default edit as set in your

XWiki.XWikiPreferences class, you can use the other macros, and still edit a page in inline mode: choose the "Inline form" option in the "Edit" menu, provided your user type is advanced.
== includeTopic() ==
{{info}}
Specification: [[extensions:Extension.Include Topic Macro]]
{{/info}}
Same syntax as includeInContext(). Contrary to includeInContext(), the included XWiki page is interpreted in its own context. For example, you would include a page using includeTopic() if the included page had to access its own objects in order for the including page to be displayed properly.

- title: How to include a velocity page into another page ? ⇨ How to include a velocity page into another page
- parent: xwiki:Documentation.DevGuide.WebHome ⇨ xwiki:FAQ.WebHome

On 2017/09/05, the page has been created by Vincent Massol

~~{{box cssClass="floatinginfobox" title="**Contents**"}}~~**1 How to include a velocity page into another page ?**
~~{{toc/}}~~
~~{{/box}}~~
~~In order to include a velocity page into another page you have to use velocity macros. Here are the possible macros you can use for this purpose:~~**#toc("" "")**
~~= Include a *.vm script =~~**1.1 <code>template()</code>**
~~These macros take as parameter~~**This is the equivalent of the C preprocessor <code>include</code> directive : the result is** a ~~relative URL to a {{code}}*.vm{{/code}}~~**copy of the script passed as argument, into your** script.
The parameter is the path to the script you want to include. The path is relative to the directory of your skin. For example, if you are using the albatross skin, and have installed XWiki using the installer with default settings, the skin path will be :
{code}
C:\Program Files\XWiki Enterprise\webapps\xwiki\skins\albatross
{code}
So if you have a skin script called <code>myscript.vm</code> you want to include, and that script is located in a folder <code>mydir</code> inside the skin folder,
{code}
C:\Program Files\XWiki Enterprise\webapps\xwiki\skins\albatross\mydir\myscript.vm
{code}
you will use the following Velocity instruction in your scripts :
{code}
#template("mydir/myscript.vm")
{code}
~~== template() ==~~**1.1 <code>includeForm()</code>**
~~{{info}}~~**If the script you want to include uses the method <code>com.xpn.xwiki.doc.XWikiDocument.display()</code> to access a field of a <code>com.xpn.xwiki.objects.BaseObject</code>, for example like the default class sheet :**
~~Specification: [[extensions:Extension.Template Macro]]~~**{code:velocity}**
~~{{/info}}~~
~~This is the equivalent of the C preprocessor include directive: the result is a copy of the script passed as argument, into your script.~~

= Include the Velocity code of another XWiki page =
These macros take as parameter the XWiki name of the page, Space.Page.
== includeInContext() ==
{{info}}
Specification: [[extensions:Extension.Include In Context Macro]]
{{/info}}
If you have a script stored as an XWiki page you can include its code into another script by using includeInContext:
{{code}}
#includeInContext("mySpace.myPage")
{{/code}}
== includeForm() ==
{{info}}
Specification: [[extensions:Extension.Include Form Macro]]
{{/info}}
Similar to includeInContext(), except that using includeForm() in a page will set the default edit mode for that page as "inline". For example, the script you want to include uses the method
##com.xpn.xwiki.doc.XWikiDocument.display()## to access a field of an object attached to the including page, like a typical class sheet:
{{code language="velocity"}}
{{/code}}**{code}**
which is storedthen you have to store the script as a regular XWiki page (let's say myPage**<code>myPage</code>** in mySpace).
<code>mySpace</code>) and include the page with (using XWiki link syntax for the parameter) :
You will include the page using includeForm():**{code}**
{{code}}
{{/code}}**{code}**
This way,**way** the including page**include** will always be **done properly. When** edited in inline mode by default. Thus in the example of the class sheet, the including**<code>inline</code> mode, your** page will display **the** input fields mapped to the respective object fields.
<code>BaseObject</code> fields. To edit a page in inline mode, choose the "Inline form" option in the "Edit" menu, provided you have enabled "Advanced Edit" in your user profile.
If you prefer to keep the default edit as set in your XWiki.XWikiPreferences class, you can use the other macros, and still edit a page in inline mode: choose the "Inline form" option in the "Edit" menu, provided your user type is advanced.**1.1 <code>includeInContext</code>**
== includeTopic() ==**If you have a script stored as an XWiki page, you can include its code into another script by using <code>includeInContext</code> :**
{{info}}**{code}**
Specification: [[extensions:Extension.Include Topic Macro]]**#includeInContext("mySpace.myPage")**
{{/info}}**{code}**
Same syntax as includeInContext(). Contrary to includeInContext(), the included XWiki page is interpreted in its own context. For example, you would include a page using includeTopic() if the included page had to access its own objects in order for the including page to be displayed properly.

XWiki.XWikiComments

- author:
XWiki.VincentMassol
- comment:

~~Jean-Vivien, you should reference the page describing these macros in the Code zone. Actually it would be nice to move all your content there and only link from here. Thanks!~~

- date:
~~2007-11-12 18:27:46.0~~

XWiki.XWikiComments
- author:
~~XWiki.jeanvivienmaurice~~
- comment:
~~Point taken.~~
- date:
~~2007-11-13 10:48:49.0~~

XWiki.XWikiComments
- author:
~~XWiki.jeanvivienmaurice~~
- comment:
~~Which section shall I put it in ? Snippet, Macro ? Thanks~~
- date:
~~2007-11-13 10:54:32.0~~

XWiki.XWikiComments
- author:
~~XWiki.VincentMassol~~
- comment:
~~There are already existing documentation for the includeContext and includeForm macros.~~
- date:
~~2007-11-13 15:07:51.0~~

XWiki.XWikiComments
- author:
~~XWiki.VincentMassol~~
- comment:
~~Forgot to mention: you're doing a great job! Thanks a lot for doing all of these improvements :) It really helps.~~
- date:
~~2007-11-13 15:10:53.0~~

XWiki.XWikiComments
- author:
~~XWiki.jeanvivienmaurice~~
- comment:
~~Shall I delete even more text in this page ? Thanks.~~
- date:
~~2007-11-15 08:29:59.0~~

- title: How to include a velocity page into another page ? ⇨

- parent: xwiki:Documentation.DevGuide.WebHome ⇨ DevGuide.Scripting
- author: XWiki.VincentMassol ⇨ XWiki.jeanvivienmaurice
- syntax: XWiki 2.1 ⇨ XWiki 1.0

XWiki

Profile of Vincent Massol

XWiki.VincentMassol

Between 2017/09/05 and 2017/09/05, the page has been modified 2 times, by 2 users: Vincent Massol, Marius Dumitru Florea

XWiki.XWikiUsers
active:
~~No~~**Yes**
comment:
displayHiddenDocuments:
~~No~~**Yes**

Profile of Ecaterina Moraru (Valica)

XWiki.evalica

Between 2017/09/05 and 2017/09/05, the page has been modified 3 times, by 1 user: Ecaterina Moraru (Valica)

XWiki.XWikiUsers
comment:
displayHiddenDocuments:
Yes

Profile of Ilie Andriuta

XWiki.iandriuta

Between 2017/09/05 and 2017/09/05, the page has been modified 13 times, by 1 user: Ilie Andriuta

XWiki.WatchListClass
documents:
xwiki:XWiki.iandriuta test:QA.XWiki 9\.5 RC1 test:QA.Google Chrome 59 test:Administration Tests.Annotations In Old Syntax test:TestReportingExecutions.Administration Tests\.Create And Delete Group9 test:TestReportingExecutions.Administration Tests\.Create And Delete User9 test:TestReportingExecutions.Administration Tests\.Create Page From Template9 test:TestReportingExecutions.Administration Tests\.Register From Administration9 test:TestReportingExecutions.Administration Tests\.Image Selection Limited To Current Page9 test:Administration Tests.Space Rights Show Users test:TestReportingExecutions.Nested Spaces\.Copy a space9 test:TestReportingExecutions.Nested Spaces\.Copy a WebHome page from one space to another9 test:Nested Spaces.Copy and Paste the link of a document test:TestReportingExecutions.Nested Spaces\.Create a child document of any current document9

test:TestReportingExecutions.Nested Spaces\.Create a child document to store information about sub aspects of documents9 test:TestReportingExecutions.Nested Spaces\.Create a L1 Nested Document9 test:TestReportingExecutions.Nested Spaces\.Create a L4 Nested Document9 test:TestReportingExecutions.Nested Spaces\.Create terminal page9 test:TestReportingExecutions.Nested Spaces\.Delete non terminal page9 test:TestReportingExecutions.Nested Spaces\.Expand tree structure9 test:TestReportingExecutions.Nested Spaces\.Export Import a document with defined permissions9 test:TestReportingExecutions.Nested Spaces\.Export Import a document with history9 test:TestReportingExecutions.Nested Spaces\.Grant view rights on a document for a group9 test:TestReportingExecutions.Nested Spaces\.Inherit permissions to any child document (with no overridden permissions on the child)9 test:TestReportingExecutions.Nested Spaces\.Move document to top-level path non-existing parents9 test:TestReportingExecutions.Nested Spaces\.Move only parent9 test:TestReportingExecutions.Nested Spaces\.Navigate through the parents using the breadcrumb9 test:TestReportingExecutions.Nested Spaces\.Preserve document permissions after move9 test:TestReportingExecutions.Nested Spaces\.Rename Document9 test:TestReportingExecutions.Nested Spaces\.Share the link containing a nested space to another person9 test:TestReportingExecutions.Nested Spaces\.The breadcrumb displays the correct path9 test:Security Tests.Deny Comment Right For A Group test:TestReportingExecutions.Security Tests\.Deny Comment Right For A Group9 test:Security Tests.Deny Comment Rights For An User test:Security Tests.Deny Edit and Comment right on a page for Admin test:Security Tests.Deny edit right for a group and grant edit right for a user test:Security Tests.Deny Edit Rights For A Group test:Security Tests.Deny Edit Rights For An User test:Security Tests.Deny the rights for a user at document level and give the rights to a group test:Security Tests.Deny the rights for a user at space level and give the rights to a group at document level test:Security Tests.Deny View Rights For A Group test:Security Tests.Deny View Rights For An User test:TestReportingExecutions.Security Tests\.Deny Comment Rights For An User9 test:TestReportingExecutions.Security Tests\.Deny Edit and Comment right on a page for Admin9 test:TestReportingExecutions.Security Tests\.Deny edit right for a group and grant edit right for a user9 test:TestReportingExecutions.Security Tests\.Deny Edit Rights For A Group9 test:TestReportingExecutions.Security Tests\.Deny Edit Rights For An User9 test:TestReportingExecutions.Security Tests\.Deny the rights for a user at document level and give the rights to a group9 test:TestReportingExecutions.Security Tests\.Deny the rights for a user at space level and give the rights to a group at document level9 test:TestReportingExecutions.Security Tests\.Deny the rights for a user at wiki level and give rights to a group at space level9 test:TestReportingExecutions.Security Tests\.Deny View Rights For A Group9 test:TestReportingExecutions.Security Tests\.Deny View Rights For An User9 test:TestReportingExecutions.Security Tests\.Grant Create Wiki Right For An User9 test:Security Tests.Grant edit right for a user on a page test:TestReportingExecutions.Security Tests\.Grant edit right for a user on a page9 test:Security Tests.Group rights on a space test:Security Tests.Group rights on a space with a Subgroup included test:Security Tests.Subwiki: Group rights on a space test:Security Tests.Subwiki: Group rights on a space with a Subgroup included test:TestReportingExecutions.Security Tests\.Group rights on a space9 test:TestReportingExecutions.Security Tests\.Group rights on a space

test:TestReportingExecutions.Security Tests\.Group rights on a space with a Subgroup included9 test:QA.XWiki 9\.5\.1 test:TestReportingExecutions.Security Tests\.Deny Create Wiki Right For A Group9 test:Security Tests.Subwiki: Deny the rights for a user at document level and give the rights to a group test:TestReportingExecutions.Security Tests\.Subwiki: Deny the rights for a user at document level and give the rights to a group9 test:Security Tests.Subwiki: Deny the rights for a user at space level and give the rights to a group at document level test:TestReportingExecutions.Security Tests\.Subwiki: Deny the rights for a user at space level and give the rights to a group at document level9 test:TestReportingExecutions.Security Tests\.Subwiki: Deny the rights for a user at wiki level and give rights to a group at space level9 test:TestReportingExecutions.Security Tests\.Subwiki: Group rights on a space9 test:TestReportingExecutions.Security Tests\.Subwiki: Group rights on a space with a Subgroup included9 test:TestReportingExecutions.Search Tests\.Modify Page And Search9 test:TestReportingExecutions.Search Tests\.Search Location Facet9 test:CKEditor.Div select test:TestReportingExecutions.CKEditor\.Div select9 test:TestReportingExecutions.CKEditor\.Remove a link9 test:TestReportingExecutions.CKEditor\.Text styles9 test:TestReportingExecutions.CKEditor\.Use XWiki link syntax in CKEditor9 test:TestReportingExecutions.WYSIWYG Tests\.Add Italic Text In WYSIWYG Mode9 test:WYSIWYG Tests.Insert Translation Macro test:WYSIWYG Tests.Insert Velocity Macro test:TestReportingExecutions.WYSIWYG Tests\.Add Subscript Text In WYSIWYG Mode9 test:TestReportingExecutions.WYSIWYG Tests\.Add Superscript Text In WYSIWYG Mode9 test:TestReportingExecutions.WYSIWYG Tests\.Add Title Formatting To Selected Text9 test:TestReportingExecutions.WYSIWYG Tests\.Add Underline Text In WYSIWYG Mode9 test:TestReportingExecutions.WYSIWYG Tests\.Apply And Remove Numbering Styling In WYSIWYG Mode9 test:TestReportingExecutions.WYSIWYG Tests\.Apply Bold Styling in WYSIWYG Mode9 test:TestReportingExecutions.WYSIWYG Tests\.Apply Bullets Styling In WYSIWYG Mode9 test:TestReportingExecutions.WYSIWYG Tests\.Apply Plain Text Styling9 test:TestReportingExecutions.WYSIWYG Tests\.Attached Image Wysiwyg Center Alignment9 test:TestReportingExecutions.WYSIWYG Tests\.Attached Image Wysiwyg Left Alignment9 test:TestReportingExecutions.WYSIWYG Tests\.Attached Image Wysiwyg Right Alignment9 test:TestReportingExecutions.WYSIWYG Tests\.Attached Image Wysiwyg Top Alignment9 test:TestReportingExecutions.WYSIWYG Tests\.Cancel Fullscreen Wysiwyg9 test:TestReportingExecutions.WYSIWYG Tests\.Cancel Wysiwyg9 test:TestReportingExecutions.WYSIWYG Tests\.Changing an existing link9 test:TestReportingExecutions.WYSIWYG Tests\.Collapse And Expand Selected Macros9 test:TestReportingExecutions.WYSIWYG Tests\.Create Link To Email Address9 test:TestReportingExecutions.WYSIWYG Tests\.Create Link To Existing Page9 test:TestReportingExecutions.WYSIWYG Tests\.Create Link To Attachment9 test:TestReportingExecutions.WYSIWYG Tests\.Create Link To Web Page With Changed Label9 test:TestReportingExecutions.WYSIWYG Tests\.Create Link To Web Page With Tooltip That Opens In New Window9 test:TestReportingExecutions.WYSIWYG Tests\.Delete Macro9 test:TestReportingExecutions.WYSIWYG Tests\.Delete Table9 test:TestReportingExecutions.WYSIWYG Tests\.Delete Table Row9

test:TestReportingExecutions.WYSIWYG Tests\.Edit HTML Macro9
test:TestReportingExecutions.WYSIWYG Tests\.Edit In Full Screen
WYSIWYG Mode9 test:TestReportingExecutions.WYSIWYG Tests\.Edit
Section In WYSIWYG Mode9 test:TestReportingExecutions.WYSIWYG
Tests\.Edit Source Cancel9 test:TestReportingExecutions.WYSIWYG
Tests\.Edit Source Save Continue9
test:TestReportingExecutions.WYSIWYG Tests\.Edit Source Save View9
test:WYSIWYG Tests.Edit Title Wysiwyg
test:TestReportingExecutions.WYSIWYG Tests\.Edit Title Wysiwyg9
test:TestReportingExecutions.WYSIWYG Tests\.Insert A Horizontal Ruler
In WYSIWYG Mode9 test:TestReportingExecutions.WYSIWYG
Tests\.Insert Activity Macro9 test:TestReportingExecutions.WYSIWYG
Tests\.Insert Attachment Selector Macro9
test:TestReportingExecutions.WYSIWYG Tests\.Insert Box Macro9
test:TestReportingExecutions.WYSIWYG Tests\.Insert Cache Macro9
test:TestReportingExecutions.WYSIWYG Tests\.Insert Chart Macro9
test:TestReportingExecutions.WYSIWYG Tests\.Insert Container Macro9
test:TestReportingExecutions.WYSIWYG Tests\.Insert Content Macro
With Syntax 1x9 test:TestReportingExecutions.WYSIWYG Tests\.Insert
Context Macro9 test:TestReportingExecutions.WYSIWYG Tests\.Insert
Display Macro9 test:TestReportingExecutions.WYSIWYG Tests\.Insert
Error Message Macro9 test:TestReportingExecutions.WYSIWYG
Tests\.Insert External Image9 test:TestReportingExecutions.WYSIWYG
Tests\.Insert External Link9 test:TestReportingExecutions.WYSIWYG
Tests\.Insert Footnote Macro9 test:TestReportingExecutions.WYSIWYG
Tests\.Insert Formula Macro9 test:TestReportingExecutions.WYSIWYG
Tests\.Insert Code Macro9 test:TestReportingExecutions.WYSIWYG
Tests\.Insert Groovy Macro9 test:TestReportingExecutions.WYSIWYG
Tests\.Insert Image From All Pages9
test:TestReportingExecutions.WYSIWYG Tests\.Insert Include Macro9
test:TestReportingExecutions.WYSIWYG Tests\.Insert Documents
Macro9 test:WYSIWYG Tests.Insert Id Macro
test:TestReportingExecutions.WYSIWYG Tests\.Insert Id Macro9
test:TestReportingExecutions.WYSIWYG Tests\.Insert PDF Macro9
test:TestReportingExecutions.WYSIWYG Tests\.Insert Put Footnote
Macro9 test:TestReportingExecutions.WYSIWYG Tests\.Insert Rss
Macro9 test:TestReportingExecutions.WYSIWYG Tests\.Insert Script
Macro9 test:TestReportingExecutions.WYSIWYG Tests\.Insert Space
Index Macro9 test:TestReportingExecutions.WYSIWYG Tests\.Insert
Success Message Macro9 test:TestReportingExecutions.WYSIWYG
Tests\.Insert Symbol9 test:TestReportingExecutions.WYSIWYG
Tests\.Insert Table9 test:TestReportingExecutions.WYSIWYG
Tests\.Insert Table Uncheck First Row Is A Header Row9 test:WYSIWYG
Tests.Insert Tag Cloud Macro test:TestReportingExecutions.WYSIWYG
Tests\.Insert TOC Macro9 test:TestReportingExecutions.WYSIWYG
Tests\.Insert Translation Macro9 test:TestReportingExecutions.WYSIWYG
Tests\.Insert Velocity Macro9 test:TestReportingExecutions.WYSIWYG
Tests\.Insert Warning Message Macro9
test:TestReportingExecutions.WYSIWYG Tests\.Insert Watchlist Macro9
test:TestReportingExecutions.WYSIWYG Tests\.Insert Wikis Macro9
test:TestReportingExecutions.WYSIWYG Tests\.Link Attached File
Wysiwyg9 test:TestReportingExecutions.WYSIWYG Tests\.Link Wiki
Page Wysiwyg9 test:TestReportingExecutions.WYSIWYG Tests\.Paste
From Clipboard With Filter Styles9
test:TestReportingExecutions.WYSIWYG Tests\.Paste From Clipboard
Without Filter Styles9 test:TestReportingExecutions.WYSIWYG
Tests\.Paste some content in the WYSIWYG editor9

test:TestReportingExecutions.WYSIWYG Tests\.Preview A Page In Fullscreen WYSIWYG Mode9 test:TestReportingExecutions.WYSIWYG Tests\.Preview A Page In WYSIWYG Mode9 test:TestReportingExecutions.WYSIWYG Tests\.Save Continue Minor Wysiwyg9 test:TestReportingExecutions.WYSIWYG Tests\.Save Continue Version Summary Wysiwyg9 test:TestReportingExecutions.WYSIWYG Tests\.Save Continue Wysiwyg9 test:TestReportingExecutions.WYSIWYG Tests\.Save View Minor Wysiwyg9 test:TestReportingExecutions.WYSIWYG Tests\.Save View Version Summary Wysiwyg9 test:TestReportingExecutions.WYSIWYG Tests\.Save View Wysiwyg9 test:TestReportingExecutions.WYSIWYG Tests\.Script Macro9 test:TestReportingExecutions.WYSIWYG Tests\.Table Delete Column9 test:TestReportingExecutions.WYSIWYG Tests\.Table Insert Column After9 test:TestReportingExecutions.WYSIWYG Tests\.Table Insert Column Before9 test:TestReportingExecutions.WYSIWYG Tests\.Table Insert Row After9 test:TestReportingExecutions.WYSIWYG Tests\.Table Insert Row Before9 test:TestReportingExecutions.WYSIWYG Tests\.Undo And Redo in WYSIWYG Mode9 test:TestReportingExecutions.WYSIWYG Tests\.Insert Message Sender Macro9 test:TestReportingExecutions.WYSIWYG Tests\.Insert Office Document Viewer9 test:TestReportingExecutions.WYSIWYG Tests\.Insert Python Macro9 test:TestReportingExecutions.WYSIWYG Tests\.Insert User Avatar Macro9 test:Wiki Tests.Cancel the deleting process of a wiki test:Wiki Tests.Delete Wiki test:Wiki Tests.Edit a translation test:Wiki Tests.Install extensions only on subwikis test:TestReportingExecutions.Web Tests\.Add A Comment9 test:TestReportingExecutions.Web Tests\.Add A Heading2 In Wiki Mode9 test:TestReportingExecutions.Web Tests\.Add A Heading4 In Wiki Mode9 test:TestReportingExecutions.Web Tests\.Delete an attachment from More Actions Menu9 test:TestReportingExecutions.Web Tests\.Add Italic Text In Wiki Mode9 test:TestReportingExecutions.Web Tests\.Add Property In Class Edit Mode9 test:TestReportingExecutions.Web Tests\.Add Strikethrough Text In Wiki Mode9 test:TestReportingExecutions.Web Tests\.Add Underline Text In Wiki Mode9 test:TestReportingExecutions.Web Tests\.Add Velocity Code In Wiki Mode9 test:TestReportingExecutions.Web Tests\.Apply Bullets Styling In Wiki Mode9 test:TestReportingExecutions.Web Tests\.Apply HTML To Selected Text9 test:TestReportingExecutions.Web Tests\.Attach A File Twice from More Actions Menu9 test:Web Tests.Attach a office document from More Actions Menu test:TestReportingExecutions.Web Tests\.Attach Two Files9 test:TestReportingExecutions.Web Tests\.Cancel A Comment9 test:TestReportingExecutions.Web Tests\.Cancel a comment from More Actions Menu9 test:TestReportingExecutions.Web Tests\.Cancel Editing In Fullscreen Wiki Mode9 test:TestReportingExecutions.Web Tests\.Cancel Editing In Wiki Mode9 test:TestReportingExecutions.Web Tests\.Compare Two Minor Versions9 test:Web Tests.Copy A Page test:TestReportingExecutions.Web Tests\.Copy A Page9 test:TestReportingExecutions.Web Tests\.Correct Url Is Accessed After Login9 test:Web Tests.Create External Link In Wiki Mode test:Web Tests.Create Internal Link In Wiki Mode test:TestReportingExecutions.Web Tests\.Create Internal Link In Wiki Mode9 test:TestReportingExecutions.Web Tests\.Create Page9 test:TestReportingExecutions.Web Tests\.Default Attachment Size Limit of 32MB9 test:TestReportingExecutions.Web Tests\.Delete a comment from More Actions Menu9 test:TestReportingExecutions.Web Tests\.Delete A Page9 test:TestReportingExecutions.Web Tests\.Delete An Attachment9 test:TestReportingExecutions.Web Tests\.Delete Comment9

test:TestReportingExecutions.Web Tests\.Delete Comment9
test:TestReportingExecutions.Web Tests\.Delete version range from More Actions Menu9 test:Web Tests.Create Space
test:TestReportingExecutions.Web Tests\.Edit A Page In Object Mode9
test:TestReportingExecutions.Web Tests\.Edit A Page Inline9 test:Web Tests.Edit Title In Wiki Mode test:Web Tests.Exit Fullscreen In Wiki Mode
test:TestReportingExecutions.Web Tests\.Edit Title In Wiki Mode9
test:TestReportingExecutions.Web Tests\.Forgot your username9
test:TestReportingExecutions.Web Tests\.Information from More Actions Menu9 test:Web Tests.Insert A Horizontal Ruler In Wiki Mode
test:TestReportingExecutions.Web Tests\.Insert A Horizontal Ruler In Wiki Mode9 test:Web Tests.Insert HTML Code In Wiki Mode test:Web Tests.Preview A Page In Fullscreen Wiki Mode test:Web Tests.Preview A Page In Wiki Mode test:TestReportingExecutions.Web Tests\.Jump To A Page Using Ctrl-G9 test:TestReportingExecutions.Web Tests\.Login And Logout9 test:TestReportingExecutions.Web Tests\.Login With Invalid Credentials9 test:TestReportingExecutions.Web Tests\.Login With Invalid Username9 test:TestReportingExecutions.Web Tests\.Preview A Comment9 test:TestReportingExecutions.Web Tests\.Preview a comment from More Actions Menu9 test:TestReportingExecutions.Web Tests\.Preview A Page In Fullscreen Wiki Mode9
test:TestReportingExecutions.Web Tests\.Preview A Page In Wiki Mode9
test:Web Tests.Quick Links Jump to a Page
test:TestReportingExecutions.Web Tests\.Quick Links Jump to a Page9
test:TestReportingExecutions.Web Tests\.Redirect To The Reset Password Page9 test:TestReportingExecutions.Web Tests\.Register John Smith9 test:TestReportingExecutions.Web Tests\.Rename Page9
test:TestReportingExecutions.Web Tests\.Reply To A Comment9
test:TestReportingExecutions.Web Tests\.Import Office Document9
test:TestReportingExecutions.Web Tests\.Rollback To A Minor Version9
test:TestReportingExecutions.Web Tests\.Save A Page As A Minor Version9 test:TestReportingExecutions.Web Tests\.Save And Continue A Page9 test:Web Tests.Save And Continue In Fullscreen Wiki Mode
test:Web Tests.Save And Continue With A Version Summary In Wiki Mode test:Web Tests.Save And View In Fullscreen Wiki Mode test:Web Tests.Save View Version Summary Wiki
test:TestReportingExecutions.Web Tests\.Save And Continue In Fullscreen Wiki Mode9 test:TestReportingExecutions.Web Tests\.Save And Continue With A Version Summary In Wiki Mode9
test:TestReportingExecutions.Web Tests\.Save And View In Fullscreen Wiki Mode9 test:TestReportingExecutions.Web Tests\.Save View Version Summary Wiki9 test:TestReportingExecutions.Web Tests\.See A Specific Version Of A Page9 test:TestReportingExecutions.Web Tests\.See Information About A Page9 test:Web Tests.Select Text And Create External Link In Wiki Mode test:Web Tests.Select Text And Create Internal Link In Wiki Mode test:Web Tests.Select An Image With The Attached Image Button test:TestReportingExecutions.Web Tests\.Select An Image With The Attached Image Button9
test:TestReportingExecutions.Web Tests\.Select Text And Create External Link In Wiki Mode9 test:TestReportingExecutions.Web Tests\.Select Text And Create Internal Link In Wiki Mode9
test:TestReportingExecutions.Web Tests\.Share Email9
test:TestReportingExecutions.Web Tests\.Share page by email9
test:TestReportingExecutions.Web Tests\.Show And Hide Minor Edits In History Tab9 test:TestReportingExecutions.Web Tests\.View An Office Attachment9 test:TestReportingExecutions.Web Tests\.View Source9
test:User Tests.Change Avatar Image test:User Tests.Change Avatar Image Select Another Image test:TestReportingExecutions.User

Tests\.Edit Profile9 test:User Tests.Watch Page And Delete It From Watchlist test:Notification Tests.Clear All test:Notification Tests.Load older notifications test:Notification Tests.Notification Filters (Notification Menu) test:TestReportingExecutions.Notification Tests\.Notification Filters (Notification Menu)9 test:Notification Tests.Notification Filters (E-Mail) test:TestReportingExecutions.Notification Tests\.Notification Filters (E-Mail)9 test:Installers.Test EXE installer on Windows 10 test:Installers.Test EXE Installer on Windows 7 test:Installers.Test EXE Installer on Windows 8\.1 test:Installers.WebHome test:XPoll Application Tests.Add characters into title test:XPoll Application Tests.Add proposals test:XPoll Application Tests.Change the description test:XPoll Application Tests.Change the name test:TestReportingExecutions.XPoll Application Tests\.Change the name9 test:XPoll Application Tests.Change the proposals test:XPoll Application Tests.Change the Status test:XPoll Application Tests.Change the Status to Active test:XPoll Application Tests.Change the Status to Finished test:TestReportingExecutions.XPoll Application Tests\.Change the Status9 test:TestReportingExecutions.XPoll Application Tests\.Change the Status to Active9 test:TestReportingExecutions.XPoll Application Tests\.Create a new poll9 test:XPoll Application Tests.Delete a poll test:TestReportingExecutions.XPoll Application Tests\.Delete a poll9 test:XPoll Application Tests.Deny view rights for a user test:XPoll Application Tests.Filter the polls test:TestReportingExecutions.XPoll Application Tests\.Deny view rights for a user9 test:TestReportingExecutions.XPoll Application Tests\.Filter the polls9 test:TestReportingExecutions.XPoll Application Tests\.Multiple clicks on Save button9 test:XPoll Application Tests.Sort the polls test:XPoll Application Tests.Vote a poll test:XPoll Application Tests.Winners are accurate in the livetable test:XPoll Application Tests.Remove proposals test:TestReportingExecutions.XPoll Application Tests\.Sort the polls9 test:XPoll Application Tests.Special characters from pool title into Recent pools panel test:TestReportingExecutions.XPoll Application Tests\.Special characters from pool title into Recent pools panel9 test:TestReportingExecutions.XPoll Application Tests\.Vote a poll9 test:TestReportingExecutions.XPoll Application Tests\.Winners are accurate in the livetable9 test:TestReportingExecutions.Flash Messages Application Tests\.Change the group9 test:TestReportingExecutions.Flash Messages Application Tests\.Change the message9 test:TestReportingExecutions.Flash Messages Application Tests\.Create new Flash entry9 test:TestReportingExecutions.Flash Messages Application Tests\.Delete an entry9 test:TestReportingExecutions.Flash Messages Application Tests\.Deny view rights for a user9 test:Flash Messages Application Tests.Filter the entries test:TestReportingExecutions.Flash Messages Application Tests\.Filter the entries9 test:TestReportingExecutions.Flash Messages Application Tests\.Modify the Navigation bar height9 test:Flash Messages Application Tests.Switch between entries test:TestReportingExecutions.Flash Messages Application Tests\.Switch between entries9 test:Flash Messages Application Tests.Uninstall Flash Application test:TestReportingExecutions.Flash Messages Application Tests\.Check the installation logs9 test:TestReportingExecutions.Flash Messages Application Tests\.Add an image to the Flash message9 test:Forum Application Tests.Add a comment test:Forum Application Tests.Add a comment to a flag test:Forum Application Tests.Delete a comment test:Forum Application Tests.Delete a topic test:Forum Application Tests.Add forum test:TestReportingExecutions.Forum Application Tests\.Add an answer9 test:Forum Application Tests.Change the reason to a flag test:Forum Application Tests.Change the status to a flag

test:TestReportingExecutions.Forum Application Tests\.Check the breadcrumb9 test:TestReportingExecutions.Forum Application Tests\.Delete a comment9 test:TestReportingExecutions.Forum Application Tests\.Delete Forums9 test:TestReportingExecutions.Forum Application Tests\.Deny the view Rights for a user on a Topic page9 test:TestReportingExecutions.Forum Application Tests\.Deny View rights for a group on a Topic page9 test:TestReportingExecutions.Forum Application Tests\.Edit a comment9 test:TestReportingExecutions.Forum Application Tests\.Edit an answer9 test:TestReportingExecutions.Forum Application Tests\.Flag an answer9 test:TestReportingExecutions.Forum Application Tests\.Permalinks for answers9 test:TestReportingExecutions.Forum Application Tests\.Permalinks for topics9 test:TestReportingExecutions.Forum Application Tests\.Rename the forum9 test:TestReportingExecutions.Forum Application Tests\.Total topics counter9 test:TestReportingExecutions.Forum Application Tests\.Sorting by Dates, Votes, Comments9 test:TestReportingExecutions.Forum Application Tests\.Uninstall the Forum Application9 test:TestReportingExecutions.Mocca Calendar Application Tests\.Calendar view month, week, day9 test:TestReportingExecutions.Mocca Calendar Application Tests\.Change the calendar's description9 test:TestReportingExecutions.Mocca Calendar Application Tests\.Change the calendar's title9 test:TestReportingExecutions.Mocca Calendar Application Tests\.Change the event's calendar9 test:Mocca Calendar Application Tests.Change the event's description test:Mocca Calendar Application Tests.Change the event's end date test:TestReportingExecutions.Mocca Calendar Application Tests\.Change the event's end date9 test:Mocca Calendar Application Tests.Change the event's start date test:TestReportingExecutions.Mocca Calendar Application Tests\.Change the event's start date9 test:Mocca Calendar Application Tests.Change the event's title test:TestReportingExecutions.Mocca Calendar Application Tests\.Change the event's title9 test:TestReportingExecutions.Mocca Calendar Application Tests\.Create a new event9 test:TestReportingExecutions.Mocca Calendar Application Tests\.Delete a calendar9 test:Mocca Calendar Application Tests.Deny view rights for a group test:TestReportingExecutions.Mocca Calendar Application Tests\.Deny view rights for a group9 test:Mocca Calendar Application Tests.Deny view rights for a user test:TestReportingExecutions.Mocca Calendar Application Tests\.Deny view rights for a user9 test:TestReportingExecutions.Mocca Calendar Application Tests\.Drag and drop an event in month view9 test:Mocca Calendar Application Tests.Expand an event in month view test:TestReportingExecutions.Mocca Calendar Application Tests\.Drag and drop an event in week view9 test:Mocca Calendar Application Tests.Expand an event in day view test:TestReportingExecutions.Mocca Calendar Application Tests\.Expand an event in day view9 test:Mocca Calendar Application Tests.Expand an event in week view test:TestReportingExecutions.Mocca Calendar Application Tests\.Expand an event in week view9 test:TestReportingExecutions.Mocca Calendar Application Tests\.Filter the calendars9 test:Mocca Calendar Application Tests.Mocca Calendar App accessible through Applications panel test:TestReportingExecutions.Mocca Calendar Application Tests\.Mocca Calendar Macro9 test:TestReportingExecutions.Mocca Calendar Application Tests\.Multiple events with the same name in a calendar9 test:TestReportingExecutions.Mocca Calendar Application Tests\.Navigate through calendar's days9 test:TestReportingExecutions.Mocca Calendar Application Tests\.Navigate through calendar's months9

Tests\.Navigate through calendar's months9 test:TestReportingExecutions.Mocca Calendar Application Tests\.Navigate through calendar's weeks9 test:Mocca Calendar Application Tests.Shrink an event in month view test:TestReportingExecutions.Mocca Calendar Application Tests\.Shrink an event in day view9 test:TestReportingExecutions.Mocca Calendar Application Tests\.Shrink an event in week view9 test:TestReportingExecutions.Mocca Calendar Application Tests\.Sort the calendars9 test:TestReportingExecutions.Mocca Calendar Application Tests\.Start Page9 test:Statistics Application Tests.Statistics in Apps Panel test:CKEditor.Apply Bold Styling test:CKEditor.Apply Italic Styling test:CKEditor.Add Underline Styling test:CKEditor.Add Strikethrough Styling test:CKEditor.Add Subscript text test:CKEditor.Add Superscript text test:CKEditor.Remove text format test:CKEditor.Insert%2FRemove Numbered List test:CKEditor.Insert%2FRemove Bulleted List test:CKEditor.Decrease Indent test:CKEditor.Increase Indent test:CKEditor.Block Quote test:CKEditor.Cut text test:CKEditor.Copy text test:CKEditor.Undo and Redo actions test:CKEditor.Find and Replace text test:CKEditor.Insert Horizontal Line test:CKEditor.Delete a macro test:CKEditor.Add a simple macro in CKEditor test:CKEditor.Insert HTML Macro test:CKEditor.Insert Activity Macro test:CKEditor.Insert Attachment Selector Macro test:CKEditor.Insert Box Macro test:CKEditor.Insert Cache Macro test:CKEditor.Insert Chart Macro test:CKEditor.Insert Code Macro test:CKEditor.Insert Comment Macro test:CKEditor.Insert Container Macro test:WYSIWYG Tests.Insert Content Macro With Syntax 1x test:CKEditor.Insert Context Macro test:CKEditor.Insert Display Macro test:CKEditor.Insert Error Message Macro test:CKEditor.Insert Footnote Macro test:CKEditor.Insert Formula Macro test:CKEditor.Insert Gallery Macro test:CKEditor.Insert Groovy Macro test:CKEditor.Insert Id Macro test:CKEditor.Insert Include Macro test:CKEditor.Insert Message Sender Macro test:CKEditor.Insert Pages Macro test:CKEditor.Insert PDF Macro test:CKEditor.Insert Put Footnote Macro test:CKEditor.Insert Python Macro test:CKEditor.Insert RSS Macro test:CKEditor.Insert Script Macro test:CKEditor.Insert Space Index Macro test:CKEditor.Insert Spaces Macro test:CKEditor.Insert Success Message Macro test:CKEditor.Insert Tag Cloud Macro test:CKEditor.Insert "Table of Contents" Macro test:CKEditor.Insert Table of Contents Macro test:CKEditor.Insert Translations Macro test:CKEditor.Insert User Avatar Macro test:CKEditor.Insert Velocity Macro test:CKEditor.Insert Warning Message Macro test:CKEditor.Insert Watchlist Macro test:CKEditor.Insert Wikis Macro test:CKEditor.Script Macro test:CKEditor.Text styles test:CKEditor.Paste as plain text test:CKEditor.Paste from Word test:CKEditor.Add a link using the CKEditor menu test:CKEditor.Add a link to an Email address test:CKEditor.Add link to searched page test:CKEditor.Delete a table test:CKEditor.Insert%2FDelete a table cell%2Frow%2Fcolumn test:CKEditor.Cell operations test:CKEditor.Edit source cancel test:CKEditor.Edit source preview test:CKEditor.Edit source Save & Continue test:CKEditor.Edit source Save and View test:CKEditor.Edit title test:CKEditor.Insert Office Document Viewer Macro test:CKEditor.Insert Special Character test:CKEditor.Save and Continue Minor Edit test:CKEditor.Save and Continue Version Summary test:CKEditor.Preview A Page In Fullscreen Mode test:CKEditor.Preview A Page test:CKEditor.Save and Continue test:CKEditor.Save and Continue Fullscreen test:CKEditor.Save and View test:CKEditor.Save and View Fullscreen test:CKEditor.Save and View Minor Edit test:CKEditor.Save and View Version Summary test:CKEditor.Text Alignment test:CKEditor.Font and text size test:CKEditor.Text color and background color test:CKEditor.Insert Children Macro test:CKEditor.Insert

background color test:CKEditor.Insert Children Macro test:CKEditor.Insert Dashboard Macro test:CKEditor.Insert Diagram Macro test:CKEditor.Insert Example Macro test:CKEditor.Insert Icon Picker Macro test:CKEditor.Insert Menu Macro test:CKEditor.Insert Page Tree Macro test:CKEditor.Insert Publication Workflow Macro test:CKEditor.Insert Tree Macro test:CKEditor.Insert Template Macro test:Wiki Tests.Update a subwiki using Distribution Wizard test:QA.XWiki 9\.6 test:QA.Google Chrome 60 test:QA.PostgreSQL 9\.6\.3 test:QA.Mozilla Firefox 55 test:Admin Tools Application.Make a Sanity Check test:Admin Tools Application.Run Query test:Admin Tools Application.Show Large History test:Admin Tools Application.Show spammed pages test:Admin Tools Application.Access Space Used test:Ideas Application Tests.WebHome test:TestReportingExecutions.Menu Application\.Global Menu display Inside the left or right panel8 test:TestReportingExecutions.Invitation\.Show Invitation in the Applications panel8 test:Menu Application.Add an image with a link reference in the menu test:TestReportingExecutions.Invitation\.Type the message on multiple lines8 test:TestReportingExecutions.Flash Messages Application Tests\.Repeat the flash message9 test:Flash Messages Application Tests.Deny view rights for a user test:Flash Messages Application Tests.Deny view rights for a group test:TestReportingExecutions.Mocca Calendar Application Tests\.Calendar view month, week, day5 test:Publication Workflow Tests.Access the Overview Page test:Mocca Calendar Application Tests.Old events in a renamed calendar test:TestReportingExecutions.Mocca Calendar Application Tests\.Uninstall Mocca Calendar using Extension Manager8 test:TestReportingExecutions.Publication Workflow Tests\.Access the Overview Page8 test:Forum Application Tests.Uninstall the Forum Application test:Totem.Uninstall Totem Application test:TestReportingExecutions.Totem\.Uninstall Totem Application8 test:Statistics Application Tests.Uninstall Statistics Application test:TestReportingExecutions.Statistics Application Tests\.Uninstall Statistics Application8 test:TestReportingExecutions.Holiday Request Application Tests\.Install Holiday Request Application with Extension Manager8 test:TestReportingExecutions.Holiday Request Application Tests\.Holiday Request Application in Apps Panel8 test:TestReportingExecutions.Holiday Request Application Tests\.Create a new Holiday Request8 test:TestReportingExecutions.Holiday Request Application Tests\.Manualy change the number of days8 test:TestReportingExecutions.Holiday Request Application Tests\.Overview option8 test:TestReportingExecutions.Holiday Request Application Tests\.Reject a holiday request8 test:TestReportingExecutions.Holiday Request Application Tests\.Reject a holiday request without a comment8 test:TestReportingExecutions.Holiday Request Application Tests\.Calendar option8 test:Holiday Request Application Tests.Change the manager test:TestReportingExecutions.Holiday Request Application Tests\.Change the manager8 test:TestReportingExecutions.Holiday Request Application Tests\.Change the Status8 test:TestReportingExecutions.Holiday Request Application Tests\.Change the Type of holidays8 test:TestReportingExecutions.Holiday Request Application Tests\.Edit a holiday request with a different user than employee and manager8 test:TestReportingExecutions.Holiday Request Application Tests\.Filter the entries8 test:TestReportingExecutions.Holiday Request Application Tests\.Hover over a username from the livetable8 test:QA.XWiki 9\.7 test:TestReportingExecutions.Administration

> Tests\.Change Invitation Application Settings9 test:Administration Tests.Create a New Template Provider test:Administration Tests.Make Version Summaries Mandatory test:Administration Tests.WYSIWYG Editor - Modify the Color settings test:Administration Tests.WYSIWYG Editor - Modify the Font settings test:Administration Tests.WYSIWYG Editor - Modify the Menu Bar test:Administration Tests.WYSIWYG Editor - Modify the Plugins list test:Administration Tests.WYSIWYG Editor - Modify the Style settings test:Administration Tests.WYSIWYG Editor - Modify the Tool Bar test:Administration Tests.Source Editor And External Images Disabled In Wysiwyg test:Administration Tests.Image Selection Limited To Current Page test:Administration Tests.Do Not Clean Paste Content Automatically test:Administration Tests.Attachment Selection Limited To Current Page test:Nested Spaces.Export Import a document with history test:Security Tests.Deny Create Wiki Right For An User test:Security Tests.Deny Create Wiki Right For A ~~Group~~**Group test:Syntax.Link to an anchor in a page test:TestReportingExecutions.CKEditor\.Add a link to an Email address9 test:TestReportingExecutions.CKEditor\.Add link to searched page9 test:TestReportingExecutions.CKEditor\.Add Strikethrough Styling9 test:TestReportingExecutions.CKEditor\.Add Subscript text9 test:TestReportingExecutions.CKEditor\.Add Superscript text9 test:TestReportingExecutions.CKEditor\.Apply Italic Styling9 test:TestReportingExecutions.CKEditor\.Apply Bold Styling9 test:TestReportingExecutions.CKEditor\.Block Quote9 test:TestReportingExecutions.CKEditor\.Cell operations9 test:TestReportingExecutions.CKEditor\.Copy text9 test:TestReportingExecutions.CKEditor\.Cut text9 test:TestReportingExecutions.CKEditor\.Decrease Indent9**