Escenic Widget Framework

Upgrade Guide

2.1.0.139063


vizrt

Disclaimer

Vizrt provides this publication “as is” without warranty of any kind, either expressed or implied.

This publication may contain technical inaccuracies or typographical errors. While every precaution has been taken in the preparation of this document to ensure that it contains accurate and up-to-date information, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained in this document.

Vizrt’s policy is one of continual development, so the content of this document is periodically subject to be modified without notice. These changes will be incorporated in new editions of the publication. Vizrt may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

Vizrt may have patents or pending patent applications covering subject matters in this document. The furnishing of this document does not give you any license to these patents.

Technical Support

For technical support and the latest news of upgrades, documentation, and related products, visit the Vizrt web site at www.vizrt.com.

Last Updated

23.04.2013


vizrt

Table of Contents


vizrt

1 Introduction

One goal for WF-2.x is that it is possible to upgrade the framework from older versions. In WF-2.x lots of changes and framework clean ups has been done. At the same time backward compatibility issue has been considered also. This guide will describe how to upgrade from Widget Framework 1.x to 2.x. **Escenic Content Engine 5.4 version is the prerequisite of Widget Framework 2.x.**

This manual is a user guide for:

- Template developers who wants to extends/modify the Widget Framework to fit their needs
- System administrators who installs the Widget Framework
- Publication designers who want to use the Widget Framework to design Escenic publications

The prerequisites for using this manual are:

- You have the knowledge how to create new publication
- You know how to develop template using Widget Framework
- You have the general design skills needed to work with publication layouts
- You are familiar with the general structure of Escenic publications
- You already know how to use Content Studio for editorial purposes

---

In this guide we will talk about two directories. One is **maven** which contains the widgets and framework modules and **publications** which contains a maven project as demo publication. In WF 2.x corresponding directories for these two modules are **misc/widgets** and **misc/demo**. These directories are available with Widget Framework distributions.

---

2 Change required in the build process

This chapter will describe the maven changes required to build Widget Framework 2.x

2.1 Changes required in maven directory

Required changes in the root pom.xml in the **maven** directory.

- Change all engine dependencies to 5.4
- Remove widget-framework-core-tags,widget-framework-community-tags,widget-framework-common, widget-framework-build-tools from the module list
- If you have any maven repository, plugin repository settings in your pom file we highly recommends to clear those from pom file and move them in settings.xml file of apache maven
- In WF 2.x all dependencies are managed in the parent pom. Benefit is that we don't have to manage version, scope in child poms'. So, it is a recommendation that you manage dependency in this pom for all of its child. Just wrap inside **<dependencyManagement>**

To check what dependencies should be used in **maven/pom.xml** please use **misc/widgets/pom.xml** of WF 2.x distribution as a reference. If some widget has **`widget-framework-common`** as a dependency then replace it by following dependency

```
<dependency>
  <groupId>com.escenic.widget-framework</groupId>
  <artifactId>wf-engine</artifactId>
  <version>2.1.0.139063</version>
  <scope>provided</scope>
</dependency>
```

In the previous version all framework related code was inside the following three modules

- widget-framework-core
- widget-framework-community (required for community widgets)
- widget-framework-mobile (required for mobile widgets)

Now these has been separated to widget framework plugins. If you have widget framework common as a plugin or module you must remove this. In version 2.x this plugin is required no more. Java codes from **`widget-framework-core`** and **`widget-framework-community`** has been moved to widget framework core plugin and widget framework community plugin. Also some jsp, javascript files has been removed. So, you should replace widget-framework-core,widget-framework-community (if you have community widgets) and widget-framework-mobile (if you have mobile widgets) module from the version 2.x.

If you have any changes in the wireframe jsp, template group jsp, core content type, layout-group ensure that they are added in the corresponding places of widget-framework-core, widget-framework-community module or widget-framework-mobile module. Add your javascript and css files in the head.jsp.

In widget framework 2.x you don't need to compile any framework code except jsp and your custom defined classes or components. So, you may remove any compile time dependencies of previous version of Widget Framework. Of course you should keep compile time dependencies for your custom jsps, classes. All required third party jars will be distributed from the WF plugins. Be sure that you are using same version of those jars in your pom.

Widget framework tag library dependencies has been changed. You can change these dependencies in all widgets. The new dependencies are

```
<dependency>
  <groupId>com.escenic.widget-framework</groupId>
  <artifactId>wf-taglib</artifactId>
  <version>2.1.0.139063</version>
  <scope>provided</scope>
</dependency>

<dependency>
  <groupId>com.escenic.widget-framework</groupId>
  <artifactId>wf-community-taglib</artifactId>
  <version>2.1.0.139063</version>
  <scope>provided</scope>
</dependency>

<dependency>
  <groupId>com.escenic.widget-framework</groupId>
  <artifactId>wf-mobile-taglib</artifactId>
  <version>2.1.0.139063</version>
  <scope>provided</scope>
</dependency>
```

2.2 Changes required in publications/demo/pom.xml

In older version there was a separate module (**`widget-framework-build-tools`**) for build tools. This is not required any more. Because build tools now is a part of Widget Framework Core plugin. So, you have to fix the dependency in the demo publication pom file. The new dependency is

```
<dependency>
    <groupId>com.escenic.widget-framework</groupId>
    <artifactId>wf-build-tools</artifactId>
    <version>2.1.0.139063</version>
    <scope>provided</scope>
</dependency>
```

Build tools used to merge and replaces content types and publication resources from various widgets. So, you have to edit the **`merge-content-type, replace-content-type`** steps in the publication root pom file. The correct set up for **merge-content-type** is

```
      <plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>exec-maven-plugin</artifactId>
        <version>1.1</version>
        <executions>
          <execution>
            <id>merge-content-type</id>
            <phase>package</phase>
            <goals>
              <goal>java</goal>
            </goals>
          </execution>
        </executions>
        <configuration>
          <mainClass>com.escenic.widgetframework.DemoWebappResourceModifier</mainClass>
          <arguments>
         <argument>${basedir}/target/demo-core-2.1.0.139063.war</argument>
            <argument>${basedir}/src/main/webapp/template/widgets</argument>
            <argument>${basedir}/src/main/webapp/META-INF/escenic/publication-resources/escenic/
content-type</argument>
            <argument>${basedir}/src/main/resources</argument>
            <argument>/com/escenic/framework/ApplicationResources.properties</argument>
          </arguments>
        </configuration>
      </plugin>
```

Correct set up for **replace-content-type** step is

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-antrun-plugin</artifactId>
  <executions>
    <execution>
      <id>replace-content-type</id>
      <phase>install</phase>
      <goals>
        <goal>run</goal>
      </goals>
      <configuration>
        <tasks>
          <ant dir="${basedir}" antfile="${basedir}/src/main/ant/build.xml"
            target="extractAndModifyArchive">
            <property name="base.dir" value="${basedir}"/>
            <property name="project.name" value="demo-core"/>
            <property name="project.version" value="${project.version}"/>
            <property name="resource.bundle.extract.dir.name" value="resourceBundles"/>
          </ant>
        </tasks>
      </configuration>
    </execution>
  </executions>
</plugin>
```

Modify dependency version of widget-framework-core to 2.x. If you have widget-framework-common dependency replace it with wf-engine.

To build the demo publication you don't require any escenic dependencies. So, you should remove all escenic dependencies from the pom. Widget Framework dependencies (widget-framework-core/ widget-framework-community/widget-framework-mobile) and widget dependencies should be present in the pom file. You can remove widget framework core, community taglib dependencies from the pom or you can use WF 2.x taglib dependencies. If you use a different version of core, community tags there might be problem. You should also remove all third party dependencies except`jdom`. Widget Framework Core plugin is shipped with all third party libraries required for itself. If you need

any other third party libraries except those you should manage those yourself. You can check **misc/demo/pom.xml** from WF 2.x distribution as a reference.

---

3 The New Framework

This chapter will address changes required to use the new framework.

3.1 Install Widget Framework plugins

Widget Framework now has three plugins. All framework related code has been moved to these plugins. You have to install them like other Escenic plugins. The plugins are -

- WF Core plugin (this must be installed)
- WF Community plugin (install if you have community features and widgets)
- WF Mobile plugin (install if you have mobile widgets, having the mobile-expansion plugin is the pre-requisite for mobile widgets and plugin)

3.2 How to handle the new framework

The most significant change in the widget framework 2.x is that most of framework related code which were in jsp has been moved to java space. Three widget framework plugin is introduced.**To use WF 2.x widget framework core plugin must be installed**.

Widget Framework 2.x has a Controller framework. This controller framework reads all fields of widget and put them in a map. A **DefaultMapController** is provided in the WF 2.x. All widgets in WF 2.x used this default controller. Still you can write your own controller. To check how to write a new controller and use that for any widget please check WF developer guide. In jsp controller you can override any value set by the java controller.

3.3 Controller for 1.x widgets

The 1.x widgets that used **`java.util.HashMap`** can still be used with widget framework 2.1 or higher. For this there is a controller called **LegacyMapController**. This can be used for all widgets or for some particular widget. To use this for all widgets, the configuration in **`ControllerFactory.properties`** file will be

```
$class=com.escenic.framework.controller.factory.ControllerFactory
defaultController=/com/escenic/framework/controller/impl/LegacyMapController
```

To use this for a particular widget, the configuration in **`ControllerFactory.properties`** file will be

```
$class=com.escenic.framework.controller.factory.ControllerFactory
controller.stories=/com/escenic/framework/controller/impl/LegacyMapController
```

4 Required changes in content-type and layout-group

This chapter will describe changes required in the content type and layout group.

4.1 Changes required in content type

Widget framework 2.x allows filtering of duplicate stories. For that you have to add **CssClassAdderDecorator** decorator in the content type for which you want this feature. Say, you want this feature for **`news`** content type. Then you have to add the following decorator in news content type

```
<ui:decorator class="com.escenic.framework.decorator.CssClassAdderDecorator"/>
```

To check details about filtering of duplicate stories please check **Widget Framework Developers Guide**.

4.2 Changes in layout-group

To make the inheritance working you must add the request pool decorator in all root group of your layout-group file. The decorator that has to be added is -

```
<ui:decorator name="wfItemsResolver"/>
```

5 Changes in jsps, tags, javascripts

In WF-2.x some jsp file has been removed, some tags has been deprecated, some tags has been moved to custom jstl function and some javascript function has been changed. If you don't want to replace **`widget-framework-core, widget-framework-community, widget-framework-mobile`** from WF 2.x rather than you want to change jsp, javascripts manually this chapter will help you.

5.1 JSP Changes

`index.jsp` file has been removed. This file is no longer required in the new framework. But if you have this file controller will call this file. You may have some special logic in this file such as caching of widget. **`controller.jsp`** file is also not required. If you need this file you can keep. Framework will still call this for each widget if this file exist. Controller jsp is no longer creates the map that contains all fields. Rather it just use the map that exist in request scope and set by **DefaultMapController**. Its type has been changed also. You have to change this in all widgets. As for example for SEO widget it will be like

```
<jsp:useBean id="seo" type="java.util.Map" scope="request"/>
```

Alternatively you can use **LegacyMapController** (see **section 3.3**). Otherwise widgets will not work.

`common.jsp` has been simplified. Wireframe and config section related code has been moved to separate tags. If you don’t have any change in your old common.jsp you should use the new one. If you have changes in this file then you have to use the following tags in the common.jsp file and config section related code should be removed.

```
<wf-core:resolveWireframe var="wireframe" scope="request"/>
<wf-core:resolveConfigSections/>
```

`showitems.jsp` file has been moved to a tag called **`showItems.tag`**. Adding a custom group like tabbing group, carousel group is easy now. You just have to add a separate jsp file for the rendering logic and add this file in the **`template/framework/group`** directory of **`widget-framework-core`** module.

Logic of **`getitems.jsp`** has been moved to a request pool decorator. This file is no longer required. **`wfItemsResolver`** decorator will do this work. Profiling related logic has been moved to a processor called**`ProfilingProcessor`**. All newspaper related jsp files has been removed.

5.2 Changes in the tag file

Following tags has been converted to jstl function and deprecated

- convertFahrenheitToCelsius.tag
- getCaptchaHTML.tag
- getDate.tag
- getDateObject.tag
- getDateString.tag
- getGeoFieldList.tag
- getImageDimension.tag
- getRssSource.tag
- getTermFromTagIdentifier.tag
- getDateDifference.tag
- getImageRepresentation.tag
- handleLineBreaks.tag
- isContainedInList.tag
- isFieldAccessible.tag
- isValidLocation.tag
- parseFeedForImage.tag

Following tags has been deprecated

- getEnumFieldValue.tag
- getStyleClassName.tag

We highly recommend to modify your widgets to use those jstl functions instead of jsp tags. In any future version jsp tags may be removed.

5.3 Javascript changes

Several third party libraries has been removed and their functionalities has been implemented using jquery, jquery ui. The third party plugins that has been removed are

- jscal2.js
- en.js
- jquery.form.js
- jquery.jqpopup.min.js
- jquery.charcounter.js
- captify.js

If you don't update widgets then you must add these files in the corresponding location and updates the **`head.jsp`** file. If you upgrade all widgets to WF 2.x then you don't need to these files.
