SMARTIT Eclipse Spring Demos and Examples

Table of contents

1. Preface to Demos and Examples

As of now, there are one demo and six example plug-ins which are distributed as Eclipse projects, ready for import into your Eclipse workspace. They require the Web Tools Platform and the `info.smartit.eclipse.spring` plug-in. (The latter can be obtained from the Downloads area on this site.)

The demo adds a simple XML editor to the workspace when run.

The other examples demonstrate various ways to integrate a plug-in context and specify dependencies between the beans therein (so-called "wiring".) The intention of the examples is to provide documentation "in code" rather than to show the features of the framework in an impressive way.

All example plug-ins add a toolbar button and menu item to the Eclipse workbench. (You can easily recognize the toolbar buttons by their pleasant design.) When selecting either the menu item or the toolbar button, a message window is opened. The example-related sections on this page describe how the displayed message is injected in the respective example plug-in.

The examples can be downloaded as Eclipse plug-in project archives. In order to run one of the plug-ins in the Eclipse IDE, import it as a project, add the SMARTIT Eclipse Spring plug-in (see downloads) and create an appropriate "Run..." configuration for an Eclipse application.

2. Demo: Plug-in with an editor

The demo requires at least version 0.2.1 of SMARTIT Eclipse Spring.

The demo is functionally equivalent to the XML editor which is generated when creating a new plug-in project using the standard "plug-in with an editor" template. See the Eclipse documentation for a feature list of the editor.

The editor in action looks like this (click image to enlarge):

Screenshot of XML editor

The code in this demo was created step by step, first generating a new plug-in from the editor template and then moving all the wiring and most of the object lifecycle management code to Spring XML files.

All editor-dependent beans are defined in a separate file, `editorContext.xml`, which is loaded whenever the editor is opened. When the editor is closed, the editor context is also closed, thereby safely disposing all allocated SWT resources (i.e. colors).

The SMARTIT Eclipse Spring version of the editor plug-in is a striking example of how the Java code is simplified by using the Spring framework. At your option, you can see it as a preview of the forthcoming SMARTIT Eclipse Spring UI framework as well.

Download Eclipse Project (ZIP archive)

3. Simple plug-in context with defaults

This example uses the `PluginContextBeanProxy` with default settings.

When the action is to be executed for the first time, the action bean with the same ID as the proxy is looked up.

Download Eclipse Project (ZIP archive)

4. Access bean with different name

This example uses the `PluginContextBeanProxy` with an explicitly specified bean ID.

When the action is to be executed for the first time, the action bean with the specified ID as the proxy is looked up.

Download Eclipse Project (ZIP archive)

5. Access plug-in context from non-default location

This example loads the plug-in context from two explicitly specified files rather than the default `pluginContext.xml`.

Download Eclipse Project (ZIP archive)

6. Access plug-in context via service

This example plug-in first gets its plug-in context from the plug-in context service, then retrieves the message bean from the context.

Download Eclipse Project (ZIP archive)

7. Awareness of plug-in context

This example plug-in uses the `ApplicationContextAware` interface on the main plug-in class in order to access the plug-in context.

Open Issue

When using this mechanism, the SMARTIT(R) Eclipse Spring plug-in must be active before the plug-in context can be accessed. However, in an ideal world, you shouldn't have to care about this when writing code, so this is still to be improved.

Download Eclipse Project (ZIP archive)

8. Plug-in context with autowire

This example shows that you can use Spring's autowire feature in plug-in contexts, too.

Download Eclipse Project (ZIP archive)