Reference

Table of contents

1. Extension Declaration and Properties

The extension point `info.smartit.eclipse.spring.pluginContext` defines a plug-in context for the declaring plug-in. By declaring this extension in the plug-in's `plugin.xml` configuration file, the plug-in context is read and published via the `PluginContextService`.

1.1. <location>

The nested `<location>` tag specifies from which file the plug-in context is to be read.

If the extension declaration does not contain any nested `<location>` tag, the plug-in context is read from the file `pluginContext.xml`, located in the plug-in's root directory.

A plug-in context definition which is contained in more than one XML file can be specified by using multiple `<location>` tags.

Attribute	Description
`path` (required)	Path to the location of the XML file which contains the bean definitions. The path is relative to the plug-in's root directory.

Table 1: Attributes of the <location> tag

1.2. Examples

(1) The following extension declaration reads the plug-in context from its default location (`pluginContext.xml` in the plug-in's root directory). The context is identified by the plug-in ID.

```
<extension point="info.smartit.eclipse.spring.pluginContext"/>
```

(2) The next code fragment reads the plug-in context from two XML files. The first file is named `mainContext.xml` and contained in the plug-in's root directory. The second file is named `otherContext.xml` and contained in the subdirectory `contexts` of the plug-in's root directory. The plug-in context is identified by the specified extension point ID, "myPluginContext".

```
<extension id="myPluginContext"
  point="info.smartit.eclipse.spring.pluginContext">
    <location path="mainContext.xml"/>
    <location path="contexts/otherContext.xml"/>
</extension>
```

2. PluginContextBeanProxy

The `info.smartit.eclipse.spring.PluginContextBeanProxy` class allows access to beans from `plugin.xml`.

Whereever you specify a class in `plugin.xml` which is instantiated by another process, you can reference a bean from your plug-in context by specifying `PluginContextBeanProxy` instead.

The bean which is provided by the proxy is identified as follows:

1. If the classname is followed by a colon and a beanname, the bean with the specified name is used.

   Example: `<view class="info.smartit.eclipse.spring.PluginContextBeanProxy:myBeanName">`
2. Otherwise, the tag containing the class specification must have an `id` attribute, the value of which specifies the bean name.

   Example: `<view id="myViewIdIsBeanNameToo" class="info.smartit.eclipse.spring.PluginContextBeanProxy">`.

If the beanname is specified as a suffix to the classname, it can optionally be followed by the fully qualified name of the bean's class in parentheses.

Example: `<view class="info.smartit.eclipse.spring.PluginContextBeanProxy:myBeanName(my.c`

3. PluginContextService

The plug-in context service, which can be obtained from the platform service registry, provides a single method `getRequiredPluginContext(String)` which returns the plug-in context specified by the argument. The method throws a `PluginContextNotFoundException` if there is no plug-in context available for the specified name.