[EN] 03. Sample and Test Extensions

This chapter is about extensions which have been created to test the extension mechanisms in the Editor and Administrator Application and can be used, by programmers interested in that topic, as examples of the implementation of plugins.

Building Extensions

> **Source Code**
>
> The source code of sample extensions can be downloaded here: dcore-app-extension-tests.zip

Construction of sample extensions from their source files is only possible if the Maven 1.1 tool is installed in the system.

The archive with the source code has to be unpacked. The `dcore-app-extension-tests` directory will contain project directories for particular extensions, with names beginning with `dcore-app-extension-tests`. The directory will also contain the `maven-repo` directory with additional jar files needed for building.

In order to build the selected extension, the user should enter the directory related to it in the console and run the

```
maven clean install
```

One can also build all extensions at once. To do that, the user should run the `maven multiproject:clean multiproject:install` command in the `dcore-app-extension-tests` directory.

Programmers working in the Eclipse environment may find the following command, which creates a ready-to-import Eclipse file in the plugin directory, useful:

```
maven eclipse
```

Once the chosen extension has been built, a `jar` file is placed in the `target` directory; the file can be installed in the dLibra system.

Installing Sample Extensions

Sample extensions are not installed in the Editor and Administrator Application by default. In order to install such a plugin in the system, the user should copy the `jar` file of the plugin in the /WEB-INF/jnlp-jars directory of the reader application and run the Editor and Administrator Application update process in the administration panel. For more information about the administration panel, see here.

Once the Editor Application has been restarted, the plugins should be visible (which can be checked in the “About the program” item of the “Help” menu). The update process in the administration panel must be run after every change of `jar` files.

Configuring Sample Extensions

A sample extension will be displayed correctly in the Editor and Administrator Application if the `src/etc/plugin.xml` contains an appropriate configuration. The construction of that file results from the requirements of the Java Plugin Framework. The content of that file can look as follows:

plugin.xml

```
<?xml version="1.0" ?>
<!DOCTYPE plugin PUBLIC "-//JPF//Java Plug-in Manifest 0.7" "http://jpf.sourceforge.net/plugin_0_7.dtd">
<plugin id="pl.psnc.dlibra.app.extension.op.test" version="${pom.currentVersion}">
<requires>
        <import plugin-id="pl.psnc.dlibra.app.extension"/>
</requires>
        <runtime>
#foreach($dep in ${pom.dependencies})
#if(${dep.type} == "jar" && !${dep.getProperty('dist.skip').equals("true")})
                <library id="${dep.artifact}" path="lib/${dep.artifact}" type="code" />
#end
#end
                <library id="pluginCode" path="${jarName}" type="code" />
        </runtime>
<extension plugin-id="pl.psnc.dlibra.app.extension"
        point-id="objectPanel" id="op-test">
        <doc>
                <doc-text>Polish interface language is provided by Poznan Supercomputing and Networking Center.<
/doc-text>
        </doc>
        <parameter id="class"
                value="pl.psnc.dlibra.app.extension.optest.DummyObjectPanel"/>
</extension>
</plugin>
```

The <runtimes> tag contains the Velocity macro which automatically adds all the necessary jar files (those which are listed as dependencies in the `project.xml` file and which do not have value `<dist.skip>true</dist.skip>` added) to the distribution.

The `<extension>` tag defines the given extension. The `pluginId` attribute must have value "`pl.psnc.dlibra.app.extension`". The point-id attribute must contain the identifier of the extension point to which the plugin is to be connected (corresponding to the names mentioned in the documentation), and the id attribute must be the unique identifier of the extension. The `<extension>` tag may be expanded by the addition of an optional doc tag with the text to be displayed in the information window of the application (in the “About the program” item of the “Help” menu) and by adding `<parameters>` tags which define the parameters mentioned in the description of the extension point.

Note: Each sample extension only uses one extension point, but such a plugin can be connected to many points, by placing more `<extension>` tags in the configuration file.

The List of Sample Extensions

The extensions are listed and described in subchapters:

- [EN] 01. The Sample Tool-Type Extension
- [EN] 02. The Sample eventListener-Type Extension
- [EN] 03. The Sample objectPanel-Type Extension
- [EN] 04. The Sample dataSource-Type Extension

```
maven multiproject:clean multiproject:install
```