Technology Adapter Development : Quick start

Etienne Louboutin

August 28, 2014

Contents

1 Abstract

This document presents technical components of developing a simple Technology Adapter for Openflexo. Some concepts of modelling will be explained.

2 Used technologies

2.1 Versioning : Git

The sources are available to download in https on Github. To develop a technology adapter, only two repositories are necessary to clone, openflexo-packaging and openflexo-technology-adapter. It will be explained in a further section.

2.2 Dependencies management

To manage dependencies, the Apache technology Maven has been chosen. A configuration file to access Openflexo Maven repositories is available in the git repository openflexo-production, named *settings.xml*. On Unix/Linux OSs, this file has to be placed in the .m2 directory in user's home.

3 Installation

3.1 Quick install : using a VM

Some technology are necessary to begin the development, so a virtual machine is available if setting up all the environment is not an option.
This VM is a Debian Jessie, which logging information are simple, there is only one user, named User, which password is user in sudoers. The export was made with Virtualbox using the format OVF2.0. You can get this VM here.

3.2 Custom install : Configuring environment

The main Eclipse plug-in needed is m2e, available in their common repository. This plug-in is used to integrate Maven in Eclipse, which is the nature of Openflexo projects. The installation of eGit is recommended, to manage versioning directly in Eclipse. On windows it is enough. To be able to do artifact generation on command line on Linux (see after) the installation of Maven is needed (available on different distributions' repositories).
To push code, Git option *core.autocrlf* has to be set with the value *input* on Unix/Linux and with the value *true* on Windows.

4 Module generation

4.1 Maven archetype

4.1.1 Introduction

Maven can generate skeleton of projects, and in Openflexo repository, an archetype to generate a technology adapter skeleton is available. This section is about the generation and on the content obtained.

4.1.2 Command-line

To generate in command line such a skeleton, the following command has to be used :

```
mvn archetype:generate
    -DarchetypeGroupId=org.openflexo.archetypes
    -DarchetypeArtifactId=technologyadapter-archetype-simplemodel
    -DarchetypeVersion=1.7.0-SNAPSHOT
```

For consistency purpose, it is recommend to execute that in the root of technology adapters module. It will automatically declare the new module in parent *pom.xml*.
During the execution, further information will be asked, such as module name,

version, file extension of technology files,... There is a default value for most of it, but it is not always relevant to keep the default value.

4.1.3 With IDE

IDE like Eclipse has the tools to reference an archetype catalogue, and generation is possible directly with these tools. No default values are set in Eclipse, each field will have to be filled by hand. The archetype is currently in a SNAPSHOT version, corresponding check-box has to be checked to see it. If it is still not in the list, The catalogue has to be referenced with the following url : https://maven.openflexo.org/artifactory/openflexo-snapshot/archetype-catalog.xml

4.2 Content generated

4.2.1 Modules (or projects)

Three modules have been generated, one containing the two others. and once imported in a IDE, three projects can be seen. On the file system it corresponds to directory named by the technology with a *pom.xml* and two other directories containing each a *pom.xml*. Directories' names are the technology name for the root one and the two others have the root directory name suffixed with connector and connector-ui respectively. The first one is just a Maven aggregation to simplify dependency declaration. In this section, Technology will be used as technology name.

4.2.2 Technologyconnector

Some packages have been generated. In a hierarchical vision, the root one contains four classes, and three other packages : model, rm and fml.

Package model
In this package, there are all concepts needed to describe the model. The usage of Pamela framework is recommended to describe your model. A simple example on how it could be used it in this context is present in section 4.3.

Package rm
The name rm stands for *Resource Management*. In this package, there are two classes and one interface to handle that. The interface use the Pamela framework so its implementation class, referenced with annotation @ImplementationClass, can be abstract.
The class TechnologyResourceRepository doesn't need too much implementations. Only the default base URI to identify models for this technology is defined in this class.
The interface doesn't need any custom content either for file based resource. At the moment, other resource than a file based one is not really supported by the technology-adapter's API.
The last class of the package is the interface's implementation. This class contains all methods that will be used to manage resources, like save, load, create resource and so on. A lot of methods here have default implementation and method that need custom implementation are listed in section 4.3.

Package fml
The name fml stands for *Free Modelling Language*. In this package, everything needed to configure model federation and all primitive of model manipulation have to be put here, which inherit from FlexoRole.

4.2.3 Technologyconnector-ui

There is 3 packages, each containing one class generated. The main class to fill is the ModuleView which is in the view package. This class has to inherit from a JComponent, and implements ModuleView. The default generated one inherit from JPanel, and is empty, but its goal is to provide a representation of a FlexoObject, which will need a specific representation.

4.3 Implementation needed

Setup dependencies
The first thing to do is to add runtime dependencies at project to launch the executable with the newly generated technology adapter. The file *pom.xml*, in flexosemantic project's root, contains all runtime dependencies. There are dependencies to freeplaneconnector and freeplaneconnector-ui declared. These have to be copy-pasted just below it and freeplane has to be replaced by the name of added technology.

Techno-specific implementation
First thing to after that is to reference the external libraries in the classpath, by creating a *lib* directory in project root and put jars in it and referencing it in the build path. The next step is to define what the model is, i.e. which concept are relevant to manipulate and to represent. This will be used in the package model where an interface named TechnologyModel has been previously generated. Like said before, this interface uses Pamela framework, and the generated content shows what to write there. These is by default a property, its getter and setter associated. The cardinality LIST can be added to the getter, then the adder and remover to the interface has to be declared too (annotations are in the same package as @Getter and @Setter). No implementation has to be written if no specific operation has to be done when accessing to referenced property, Pamela handle all getters, setters and common implementations. This is also why the implementation class, which is in the same package, is abstract. The last implementation needed is in the package rm. The class there suffixed with *ResourceImpl* need to have an implementation of *loadResourceData*. Initialization of technology objects (from adapted library) associated with the resources and call the setter on the Openflexo object initialized in the method (this one is generated).