Deliverable D200.7

FIspace Software Development Kit (SDK)

WP 200

Project Acronym & Number:	FIspace – 604 123
Project Title:	FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics
Funding Scheme:	Collaborative Project - Large-scale Integrated Project (IP)
Latest version of Annex 1:	2013-10-03
Start date of the project:	01.04.2013
Duration:	24
Status:	Draft
Editor:	Said Rahma (ATOS)
Contributors (to the "R" part of the deliverable[1]; ordered by project partner)	**ATOS**: Said Rahma Rodriguez, Miryam Villegas
Document Identifier:	FIspace-D200.7-FIspace_Integrated_Release_V3-SDK-v0.4.docx
Date:	27.02.2015
Revision:	004
Project website address:	http://www.FIspace.eu

[1] Contributors to FIspace code ("P") include ATB, UDE, IBM, ATOS, KOC, TOG, AST, NKUA, UPM and LimeTri; contributing persons are listed at https://bitbucket.org/fispace/profile/members

The FIspace Project

Leveraging on outcomes of two complementary Phase 1 use case projects (FInest & SmartAgriFood), aim of FIspace is to pioneer towards fundamental changes on how collaborative business networks will work in future. FIspace will develop a multi-domain Business Collaboration Space (short: FIspace) that employs FI technologies for enabling seamless collaboration in open, cross-organizational business networks, establish eight working Experimentation Sites in Europe where Pilot Applications are tested in Early Trials for Agri-Food, Transport & Logistics and prepare for industrial uptake by engaging with players & associations from relevant industry sectors and IT industry.

Project Summary

As a use case project in Phase 2 of the FI PPP, FIspace aims at developing and validating novel Future-Internet-enabled solutions to address the pressing challenges arising in collaborative business networks, focussing on use cases from the Agri-Food, Transport and Logistics industries. FIspace will focus on exploiting, incorporating and validating the Generic Enablers provided by the FI PPP Core Platform with the aim of realising an extensible collaboration service for business networks together with a set of innovative test applications that allow for radical improvements in how networked businesses can work in the future. Those solutions will be demonstrated and tested through early trials on experimentation sites across Europe. The project results will be open to the FI PPP program and the general public, and the pro-active engagement of larger user communities and external solution providers will foster innovation and industrial uptake planned for Phase 3 of the FI PPP.

Project Consortium

- DLO; Netherlands
- ATB Bremen; Germany
- IBM; Israel
- KocSistem; Turkey
- Aston University; United Kingdom
- ENoLL; Belgium
- KTBL; Germany
- NKUA; Greece
- Wageningen University; Netherlands
- PlusFresc; Spain
- FloriCode; Netherlands
- Kverneland; Netherlands
- North Sea Container Line; Norway
- LimeTri; Netherlands
- BO-MO; Slovenia
- MOBICS; Greece
- Fraunhofer IML; Germany
- Q-ray; Netherlands
- FINCONS; Italy
- Kühne + Nagel; Switzerland
- University Duisburg Essen; Germany
- ATOS; Spain
- The Open Group; United Kingdom
- CentMa; Germany
- iMinds; Belgium
- Marintek; Norway
- University Politecnica Madrid; Spain
- Arcelik; Turkey
- EuroPoolSystem; Germany
- GS1 Germany; Germany
- Mieloo & Alexander; Netherlands
- OPEKEPE; Greece
- Innovators; Greece
- CIT; Spain
- SDZ; Germany
- Snoopmedia; Germany
- EECC; Germany
- CBT; Spain

More Information

Harald Sundmaeker (coordinator) e-mail: sundmaeker@atb-bremen.de
Bert Vermeer (deputy coordinator) e-mail: bert.vermeer@wur.nl
Project Website Web link: http://www.fispace.eu/

Dissemination Level

PU	Public	
PP	Restricted to other programme participants (including the Commission Services)	**X**
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Change History

Version	Notes	Date
001	Creation of the document FIspace SDK	09.12.2014
002	Update of the contents, updated text to clarify, improve and narrow the context Update of the number of FIspace plugin (15 to 17) Update of the figures in several sections Update of the list of feature delivered for the current milestone Added the new sections related to new plugin (sketches, import and export plugin) Added the new section RCP app archetype Added the new section Libraries to connect FIspace Minor changes related to rephrasing and typo error Update of the abbreviation table, update of the references table	23.01.2015
003	Internal review process, checking URL links to the FIspace Web online documentation Update of the abbreviation table, update of the references table Final version ready for submission of approved document	06.02.2015
004	Update of the coordinator information in the section "*More Information*" Added FIspace development repository and documentation references, formatting improvement Final version ready for submission to EC	27.02.2015
005		
006		

Abbreviations

AAA	Authentication, Authorisation, and Accounting
ACSI	Artifact-Centric Service Interoperation
AdvB	Advisory Board
AJAX	Asynchronous JavaScript + XML
API	Application Programming Interface
App	Software Application
B2B	Business-to-business
B2C	Business-to-Consumer
BCM	Business Collaboration Module in FIspace
BCO	Business Collaboration Objects in FIspace
BE	Business Entities
BPPC	Business Process Participant Configuration
BSS	Business Support Systems
CDR	Charging Detailed Records
CEP	Complex Event Processing
CSB	Cloud Service Bus
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
D	Deliverable
DAO	Data Access Object
DB	Database
DoW	Description of Work
EC	European Commission
EDI	Electronic Data Interchange
EE	Experimentation Environment
e.g.	Exempli gratia = for example
EPA	Event Processing Agent
EPM	Event Processing Module in FIspace
ESB	Enterprise Service Bus
EU	European Union
FIA	Future Internet Assembly
FI-PPP	Future Internet Public Private Partnership
FP7	Framework Programme 7
GA	Grant Agreement
GE	Generic Enabler
GUI	Graphical User Interface
HTML	HyperText Markup Language
IaaS	Infrastructure as a Service
ICT	Information and Communication Technology
IDE	Integrated Development Environment
IDM	Identity Management
i.e.	id est = that is to say
IE	Integration Environment
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
I/O	Input / Output
IoT	Internet of Things
IP	Intellectual Property
IP (protocol)	Internet Protocol
IPR	Intellectual Property Rights
IPsec	Internet Protocol Security
IT	Information Technology
ITU	International Telecommunication Union
ISO	International Standardization Organisation
J2SE	Java 2 Platform, Standard Edition
JDK	Java Development Kit
JDT	Related to Eclipse Java Development Tools
JMX	Java Management Extensions
JRE	Java Runtime Environment
JS	JavaScript
JSON	JavaScript Object Notation
JSP	Java Server Page
JVM	Java Virtual Machine
KPI	Key Performance Indicator
LPA	Logistics Planning Application
M	Month
MTBF	Mean Time Between Failures
MVC	Model–View–Controller
OASIS	Organization for the Advancement of Structured Information Standards
OAuth	Open standard Authentication protocol
OMG	Object Management Group
OSS	Operational Support Systems
P2P	Peer-to-peer
PaaS	Platform as a Service
PDE	Related to Eclipse Java Development Tools
PE	Production Environment
PIA	Product Information App

PIE	Preliminary Integration Environment
PKI	Public Key Infrastructure
PM	Person Month
POM	Project Object Model (used by maven tools)
Proton	IBM Proactive Technology Online
QoS	Quality of Service
RBAC	Role-Based Access Control
RCP	Rich Client Platform
REST	Representational State Transfer
RFC	Request for Comments
RSS	Revenue Sharing System
RTD	Research and Technological Development
SaaS	Software as a Service
SDI	System and Data Integration layer in FIspace
SDK	Software Development Kit
SME	Small and Medium Sized Enterprise
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOA-RM	(OASIS) Reference Model for Service Oriented Architecture
SPT	Security, Privacy and Trust Framework
SSH	Secure Shell
SSL	Secure Sockets Layer
SSO	Single Sign On
ST	Sub-Task
SWT	Standard Widget Toolkit
T	Task
TCP	Transmission Control Protocol
TIC	Tailored Information for Consumers
TLS	Transport Layer Security
TPM	Transport Planning Module
UAA	User Management, Authentication and Authorisation
UI	User Interface
UML	Unified Modeling Language
URI	Universal Resource Identifier
URL	Universal Resource Locator
USDL	Unified Service Description Language
VM	Virtual Machine
VPN	Virtual Private Network
W3C	World Wide Web Consortium
WADL	Web Application Description Language
WLAN	Wireless Local Area Network
WP	Work Package
WS	Web Service
WSDL	Web Services Description Language
XLS/XLSX	Microsoft Excel file Format
XML	eXtensible Markup Language
XSD	XML Schema Definition

Table of Contents

List of Figures

List of Tables

1 Introduction

This document aims at describing the third release (V3) of the FIspace, encompassing the implementations along with usage guidance and technical documentation of each FIspace component.

It reports on the description concerning the **FIspace Software Development Kit**, the description of the development and implementation of the SDK.

The Software Development Toolkit (SDK) provides tool-support for the development of FIspace Apps. The SDK will ease the work of **App developers** during the implementation of the Apps, providing specific tools and hiding the complexity of the platform.

Particularly, the SDK will include:

- **Tooling** (specifically an **Integrated Development Environment, IDE**), which is built on Eclipse [26] and corresponding IDE. Eclipse is widely adopted by the development community and supported by the Eclipse foundation. The FIspace SDK will offer functionalities such as:
  - Integration of Eclipse JDT (Eclipse Java Development Tools) (e.g., classpath containers) or Eclipse PDE (Eclipse Plug-in Development Environment).
  - Providing access to Javadoc for all referenced elements (FIspace modules) and auto-completion support.
  - Visual management of components and case modelling will be provided.
- **Libraries** to link with the respective modules of the FIspace, such as security, privacy and trust, or the System and Data Integration layer in FIspace (SDI).

Complementing the SDK, there will be a set of tools **targeted to business architects** for customizing and extending the FIspace to the individual needs of Users. This includes tools for authoring of Business Entities and Event Rules, as well as configuring mediators and connectors to backend systems.

Online documentation for SDK: http://dev.fispace.eu/doc/wiki/sdk

1.1 Scope

The aim of this document is mainly to describe and detail the **FIspace SDK** at development and implementation level, giving detailed and technical information related to the design and the implementation as well as information about the related technologies and standard taken as a .reference to build each component.

Along this development activities and tasks, there is a set of resources, online documentation, tutorial and other external resource that refer to the Generic Enablers that can provide more technical information and user guides for the community and people who want to use the FIspace platform for Business collaboration or developers who want to create and develop business application (Apps developer) for a specific domain of application.

Table 1 shows the links to other online resources related to FIspace project and FI-WARE.

Description	Link
FIspace Business collaboration web site	http://www.fispace.eu/
FIspace Developer Documentation web site	http://dev.fispace.eu/doc/wiki/Home
FIspace Deliverables web site	http://www.fispace.eu/deliverable.html
FIspace Tutorial web site	http://www.fispace.eu/tutorials.html
FIWARE web site	http://www.fi-ppp.eu/projects/fi-ware/
FIWARE Catalogue of the Generic Enablers (GEs)	http://catalogue.fi-ware.org/
FIWARE community web site	http://www.fi-ware.org/community/

Table 1: Other FIspace and FIWARE resources

Table 2 shows the links to the Wirecloud online documentation.

Description	Link
FIWARE - Catalogue - Application Mashup - Wirecloud	http://catalogue.fi-ware.org/enablers/application-mashup-wirecloud
FIWARE - Catalogue - Application Mashup - Wirecloud Documentation	http://catalogue.fi-ware.org/enablers/application-mashup-wirecloud/documentation
FIWARE - Application Mashup - Wirecloud - User and Programmer Guide	https://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Application_Mashup_-_Wirecloud_-_User_and_Programmer_Guide
Dashboard - Wirecloud home page	http://conwet.fi.upm.es/wirecloud/
Dashboard - The WireCloud Mashup Platform	http://conwet.fi.upm.es/docs/display/wirecloud/The+WireCloud+Mashup+Platform
Dashboard - Welcome to CoNWeT-Wirecloud Confluence	http://conwet.fi.upm.es/docs/dashboard.action
Dashboard - User Guide	http://conwet.fi.upm.es/docs/display/wirecloud/WireCloud+User%27s+Guide
Dashboard - WireCloud Installation and Administration Guide	http://conwet.fi.upm.es/docs/display/wirecloud/Wire-Cloud+Installation+and+Administration+Guide

Table 2: Wirecloud online documentation

Table 3 shows the links to the WStore online documentation.

Description	Link
FIWARE - Catalogue - Store - WStore	http://catalogue.fi-ware.org/enablers/store-wstore
FIWARE - Catalogue - Store - WStore Documentation	http://catalogue.fi-ware.org/enablers/store-wstore/documentation
FIWARE - Store - W-Store - User and Programmer Guide	https://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Store_-_W-Store_-_User_and_Programmer_Guide
FIWARE - Store - W-Store - Store - W-Store - Installation and Administration Guide	https://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Store_-_W-Store_-_Installation_and_Administration_Guide

Table 3: Store online documentation

Table 4 shows the external development tools references.

Description	Link
Java Environment, JVM, JRE, JDK (Oracle)	http://www.oracle.com/technetwork/java/javase/downloads/index.html
Eclipse IDE (Integrated Development Environment)	https://www.eclipse.org/ , https://www.eclipse.org/downloads/
Maven	http://maven.apache.org/ , http://maven.apache.org/download.cgi

Table 4: External development tools references

Table 5 shows the FIspace development repository and documentation references based on the bitbucket tools for collaborative development.

Bitbucket is a hosting site for the distributed version control systems (DVCS) Git (http://git-scm.com/) and Mercurial (http://mercurial.selenic.com/). The service offering includes an issue tracker and wiki, as well as integration with a number of popular services such as Basecamp, Flowdock, and Twitter.

Description	Link
Bitbucket FIspace repository home page	https://bitbucket.org/fispace

Bitbucket FIspace core component home page	https://bitbucket.org/fispace/core/wiki/Home
Bitbucket FIspace Roadmap page	https://bitbucket.org/fispace/core/wiki/roadmap

Table 5: Bitbucket collaborative environment for FIspace development

1.2 Intended audience

The main interest groups of this deliverable are the participating teams and the responsible partners of FIspace project involved in the development activities, setup and preparation of the development phase. This document is relevant to the software engineer, programmers and developers who are the persons directly involved in the development, participating effectively on the design and implementation of the FIspace platform and the underlying components and sub-systems who want to know more about some technical information intrinsic to the FIspace platform.

At the technical level this document is relevant to: system architects; information systems designers; system developers and application developers; software engineers; other audiences who provide design services and applications using relevant standards and the recommendations of standards bodies like IETF, ITU, ISO, W3C, etc.

Partners involved in the integration tasks include: system integrators; people to test, validate and evaluate the FIspace platform and associated systems; can be also interested.

1.3 General remark

This document follows the ISO/IEC Directives, Part 2: Rules for the structure and drafting of International Standards w.r.t. the usage of the word “shall”. The word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this specification.

This document describes the corresponding core components involved in the FIspace core platform. It presents the development currently done and the corresponding implementation, the main features developed, as well as the related technologies and environment requirements.

In most of the following sections the structure is organized as:

- **Overview**: provides an overall introduction to the component, a description, of the internal architecture and features among other.
- **Interfaces or Application programming interface (API)**: describes the API accessible for the users or entities of the component (typically applications, but a component may also be used by other components).
- **Information model**: describes or specifies the component from an information perspective describing information objects of the component domain.
- **Interaction model**: describes or specifies main usage component “scenarios” associated with the component/GEs, sequence diagrams.
- **High level composite architecture**: describes or shows the main components constituting the set of components (this perspective is optional, since some component consists of only one main component).

Notice that some components only need to describe some of the item above described.

2 FIspace Software Development Kit (SDK)

This chapter describes the developments and implementation currently done in order to create and build the FIspace Software Development Kit. The SDK provides tool-support for the development of FIspace Apps. The SDK will ease the work of App developers during the implementation of the Apps, providing specific tools and hiding the complexity of the platform.

The following sections dedicated to describe each of the developed plugin are structured as the following:

- A short description of the plugin.
- A picture that shows the Eclipse package explorer presenting the Eclipse project created to implement the corresponding plugins.
- A presentation of the corresponding Plugin XML file (Plugin.xml) that define and configure the plugin according to the Eclipse plugin management and specifications.
- A description about the main dependencies.
- A picture of the key classes or the key UML Package Diagram presenting the main packages of classes designed to develop and implement the corresponding plugin.
- A presentation of some Maven command file that generate the plugin (pom.xml).

Notice that this chapter can give useful information on the FIspace SDK plugin implementation to third party plugin developers who want to know more about the plugin developed for FIspace and giving the opportunity to this plugin developer to extend and enhance the FIspace SDK features and functionalities creating additional plugins, for example by Eclipse community developers.

2.1 FIspace SDK developments overview

FIspace Software Development Kit (SDK) is a collection of plugins developed for the Eclipse platform that together make an IDE for the development of applications in FIspace. In this environment it's possible to edit different code, scripts with syntax highlight, code completion, compilation errors, code and comment folding, script execution with a pre-configured interpreter, besides the tools that the eclipse platform provides.

The main goal of the project is that new tools will be developed using the extension architecture that the Eclipse platform provides and that FIspace SDK has available for extension of its capabilities.

Besides the **seventeen FIspace plugins** already implemented, the **FIspaceStudio Tool** is also updated. The binary distribution of FIspace SDK is represented by an Eclipse Rich Client Application. The distribution provides the required user interface for all of the pre-installed FIspace SDK functionalities along with a customised approach of the Eclipse IDE reaching the specific needs of FIspace platform. A set of **libraries** implemented in several programming languages is being developed, in order to provide the FIspace user with the appropriate tool to connect to the platform.

Features delivered for the Milestone Mo21:

- **Project templates**: a new RCP [33] archetype has been implemented and some enhancements have been done in order to connect to FIspace platform (decoupled archetype provides authenticated connection).
- **Preferences window**: archetypes settings have been widely enhanced.
- **Uploader plugin**: new plugin for storing apps (not backend parts) in FIspace. Security feature is being added.
- **Access to authoring tools (BCM editor and EPM editor)**:security issues are being faced.
- **Graphics view plugin**: this plugin is now working. The REST plugin was enhanced, and also the Publisher/Consumer one. The Converters (Transformers) plugin is working with the available formats, and the Widget artifact is ready for apps being uploaded.
- **Import plugin**: new plugin implemented for importing FIspace projects (widget and decoupled ones).
- **Export plugin**: new plugin implemented for exporting FIspace projects.
- **Logging plugin**: this plugin has been extended to provide a back-end to store all generated logs in a database (back-end is developed in Nodejs and database in MongoDB [34]).
- **FIspaceStudio Tool**: some improvements have been implemented in order to make it more useful to the final user (now, you can install more software). Some old bugs have been also fixed.
- **Libraries to access FIspace platform:** several libraries are being implemented in different programming languages (C#, JavaScript and Python libraries are in progress. PHP library is also planned). The aim of these libraries is to provide the corresponding access to FIspace services based on OAuth2 protocol [32].

2.2 Related Tools

Eclipse has been designed with modularity in mind, so it offers a suitable environment for developing the kind of applications and functionalities needed inside the project. Plugins are currently being coded in KEPLER release.

Maven 3.0 (http://maven.apache.org/) is also used during the development of plugins.

2.3 FIspace Plugins and Tools

Each plugin has been implemented through three projects:

- **fispace Xplugin**: containing the plugin itself, and shown in the following sections.
- **fispace Xplugin feature**: see 2.3.19.FIspace pluginX feature
- **fispace Xplugin updatesite**: see 2.3.20.

Besides, **FIspaceStudio** Tool is built through its own project (see 2.3.22 FIspace Studio Tool).

The following sections explain the structure and developments of each FIspace SDK developed based on plugins Eclipse technologies.

2.3.1 FIspace-bcoeditor plugin

BCO editor plugin is responsible for providing the ACSI (Artifact-Centric Service Interoperation) editor tool to FIspace developers.

ACSI editor is a web based tool used by BCM module of B2B component for the creation of business entities.

The plugin use an eclipse internal web browser. The browser is automatically directing to the ACSI publicly deployed instance.

Instead of access to that public instance, the *authenticated* access to a *development instance* is being currently developed.

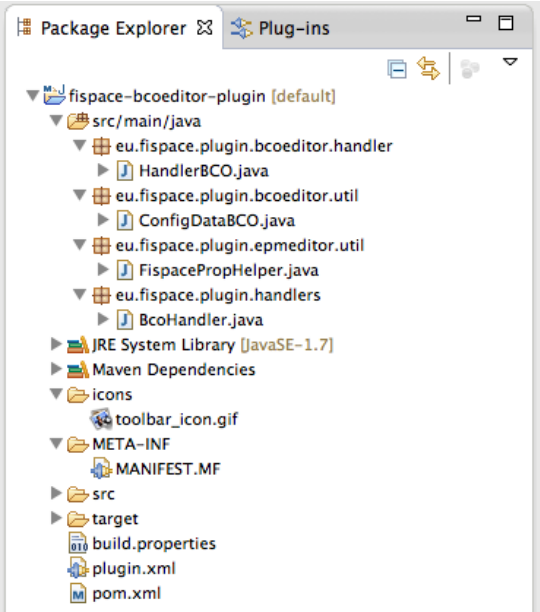


Figure 1: BCO editor project structure

Plugin.xml of EPM editor plugin is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.4"?>
<plugin>

   <extension
         point="org.eclipse.ui.commands">
      <category
            name="FIspace"
            id="eu.fispace.plugin.commands.category">
      </category>
      <command
            name="BCO Editor"
            categoryId="eu.fispace.plugin.commands.category"
            id="eu.fispace.plugin.commands.bcoCommand">
      </command>
   </extension>
   <extension
         point="org.eclipse.ui.handlers">
      <handler
            commandId="eu.fispace.plugin.commands.bcoCommand"
            class="eu.fispace.plugin.handlers.BcoHandler">
      </handler>
```

```
   </extension>
   <extension
         point="org.eclipse.ui.bindings">
      <key
            commandId="eu.fispace.plugin.commands.bcoCommand"
            contextId="org.eclipse.ui.contexts.window"
            sequence="M1+6"
            schemeId="org.eclipse.ui.defaultAcceleratorConfiguration">
      </key>
   </extension>
   <extension
         point="org.eclipse.ui.menus">
      <menuContribution
            locationURI="menu:org.eclipse.ui.main.menu?after=additions">
         <menu
               label="FIspace"
               mnemonic="I"
               id="eu.fispace.plugin.menus.fispaceMenu">
            <command
                  commandId="eu.fispace.plugin.commands.bcoCommand"
                  mnemonic="S"
                   icon="icons/toolbar_icon.gif"
                  id="eu.fispace.plugin.menus.bcoCommand">
            </command>
         </menu>
      </menuContribution>
      <menuContribution
            locationURI="toolbar:org.eclipse.ui.main.toolbar?after=additions">
         <toolbar
               id="eu.fispace.plugin.toolbars.bcoToolbar">
            <command
                  commandId="eu.fispace.plugin.commands.bcoCommand"
                  icon="icons/toolbar_icon.gif"
                  tooltip="BCO Editor"
                  id="eu.fispace.plugin.toolbars.bcoCommand">
            </command>
         </toolbar>
      </menuContribution>
   </extension>


</plugin>
```

2.3.2 FIspace-converter plugin

This plugin is also known like Data Format Transformation plugin. SDK provides with it a set of standard/widely-used data formats that be translated among them.

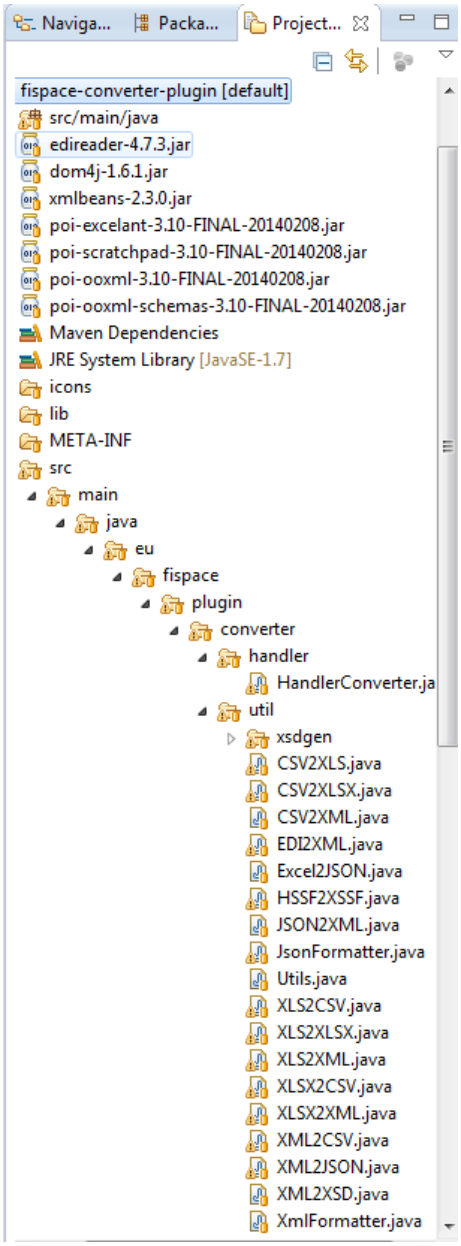


Figure 2: FIspace-converter-plugin project

Handler directory contains the class where are defined the screens of the plugin and the calls to the classes included into the **util** directory.

Into the **util** directory we have a collection of classes needed to do the format conversions.

Extension points declared in plugin.xml:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<?eclipse version=”3.4”?>
<plugin>
```

```
<extension
      point=”org.eclipse.ui.commands”>
   <category
         name=”FIspace”
         id=”eu.fispace.plugin.commands.category”>
   </category>
   <command
         name=”Format Converter”
         categoryId=”eu.fispace.plugin.commands.category”
         id= »eu.fispace.plugin.commands.convCommand »>
   </command>
</extension>
<extension
      point= »org.eclipse.ui.handlers »>
   <handler
         commandId= »eu.fispace.plugin.commands.convCommand »
         class= »eu.fispace.plugin.converter.handler.HandlerConverter »>
   </handler>
</extension>
<extension
      point=”org.eclipse.ui.bindings”>
   <key
         commandId= »eu.fispace.plugin.commands.convCommand »
         contextId= »org.eclipse.ui.contexts.window »
         sequence=”M1+6”
         schemeId=”org.eclipse.ui.defaultAcceleratorConfiguration”>
   </key>
</extension>
<extension
      point= »org.eclipse.ui.menus »>
   <menuContribution
         locationURI= »menu :org.eclipse.ui.main.menu ?after=additions »>
      <menu
            label= »Fispace »
            mnemonic= »I »
            id= »eu.fispace.plugin.menus.fispaceMenu »>
         <command
               commandId=”eu.fispace.plugin.commands.convCommand”
               mnemonic=”S”
                icon=”icons/external_browser.gif”
               id=”eu.fispace.plugin.menus.convCommand”>
         </command>
      </menu>
   </menuContribution>

</extension>
```

Main dependencies declared:

- json-lib: java library for transforming beans, maps, collections, java arrays and XML to JSON
- Apache POI: for read and write MS Excel files using Java
- EdiReader: for EDI to XML conversions
- SWT: open source widget toolkit for Java designed to provide efficient, portable access to the user-interface facilities of the operating systems on which it is implemented.
- Apache commons-io: collection of I/O utilities
- Gson: to convert Java Objects into their JSON representation.
- Apache Xerces: A processor for parsing, validating, serializing and manipulating XML, written in Java.
- XSD-Gen: Generate XML Schema from XML.

- Apache XmlBeans: technology for accessing XML by binding it to Java types.

Key classes:

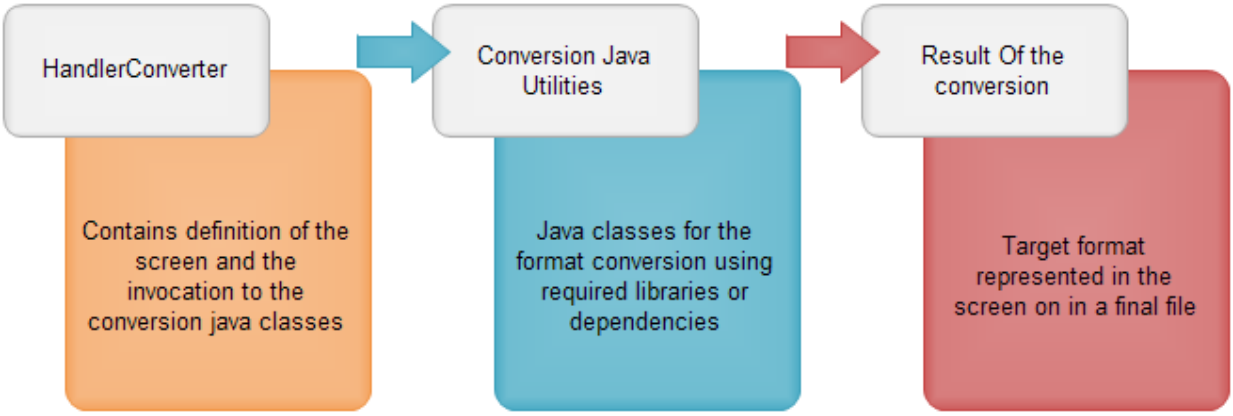


Figure 3: Key classes in Converter plugin

2.3.3 Fispace-doc plugin

The idea with this plugin is to improve the Java doc generation by Fispace using maven capabilities in Eclipse.

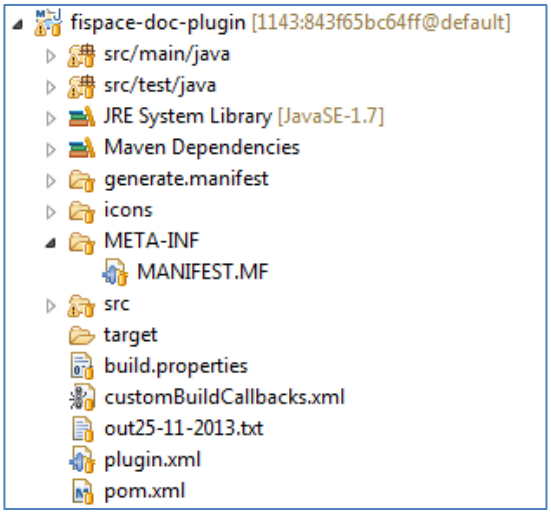


Figure 4: Fispace-doc plugin project

Pom.xml

fispace-doc/pom.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project
3      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"
4      xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
5      <modelVersion>4.0.0</modelVersion>
6
7      <groupId>eu.fispace.plugin.fispace-doc</groupId>
8      <artifactId>fispace-doc</artifactId>
9      <version>0.8.1</version>
10     <name>FIspace-Doc-Plugin-Parent</name>
11     <packaging>pom</packaging>
12     <modules>
13         <module>fispace-doc-plugin</module>
14         <module>fispace-doc-feature</module>
15         <module>fispace-doc-updatesite</module>
16     </modules>
17     <parent>
18         <groupId>eu.fispace.plugin</groupId>
19         <artifactId>plugin-parent</artifactId>
20         <version>0.8.1</version>
21         <relativePath>../pom.xml</relativePath>
22     </parent>
23 </project>
```

Figure 5: Pom.xml file to generate the Plugin FIspace-doc

Main dependencies declared:

- Maven Invoker: A component to programmatically invoke Maven.
- Maven Embedded: This is a maven runtime. Maven runtime allows introspection of maven project metadata at runtime.

Key Package Diagram:

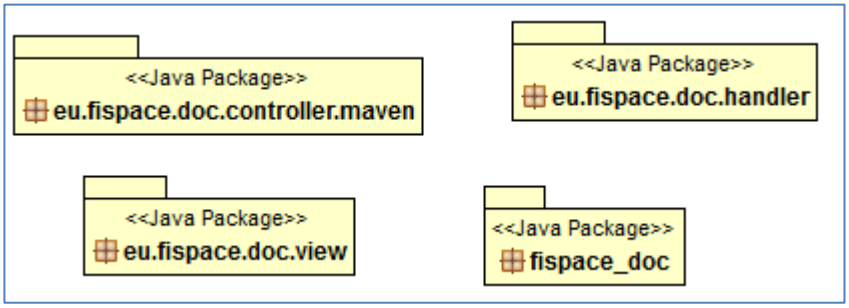
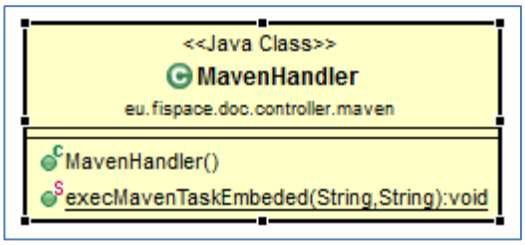


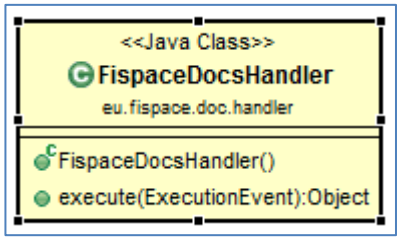
Figure 6: Doc plugin package diagram

Following FIspace doc key classes are shown:

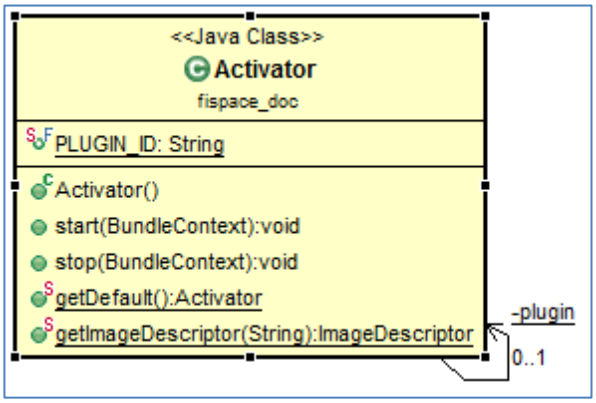
Package eu.fispace.doc.controller.maven key classes:



Package eu.fispace.doc.handler key classes:



Package eu.fispace.doc.view key classes:



Package eu.fispace.doc.view key classes:

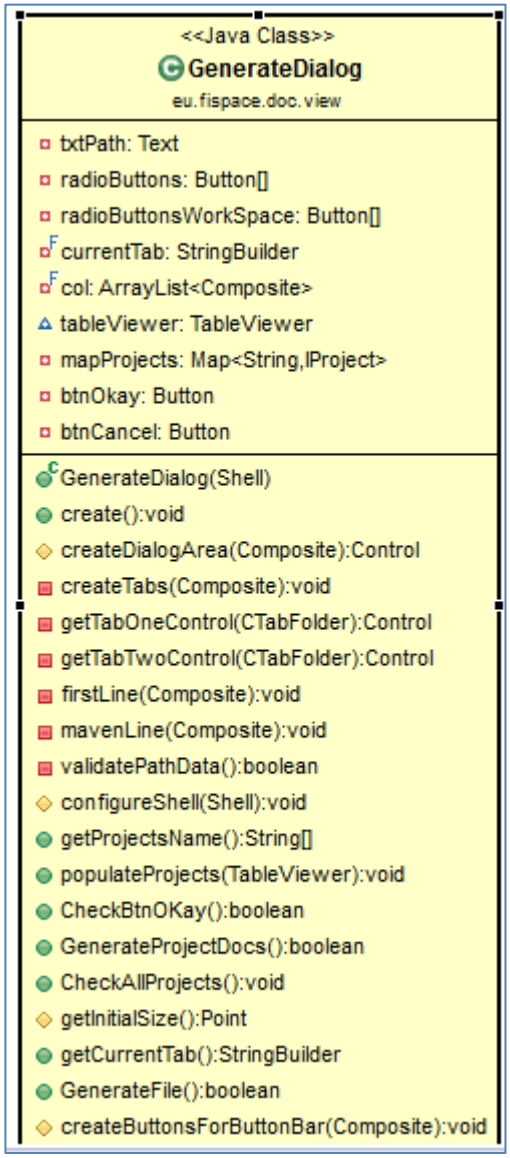


Figure 7: Fispace Doc key classes

2.3.4 Fispace-epmeditor plugin

EPM editor plugin is responsible for providing the EPM authoring tool (Proton).

Proton is a web based tool used by EPM module of B2B component for the creation of partners and the corresponding events.

The plugin use an eclipse internal web browser. The browser is automatically directing to the Proton publicly deployed instance.

Instead of access to that public instance, the *authenticated* access to a *development instance* is being currently developed.

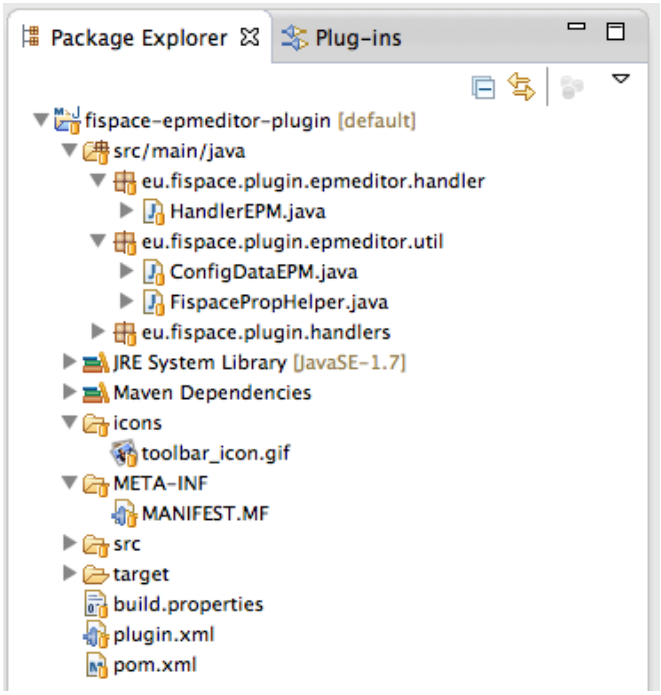


Figure 8 : EPM editor project structure

Plugin.xml of EPM editor plugin is the following:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<?eclipse version=”3.4”?>
<plugin>

   <extension
         point=”org.eclipse.ui.commands”>
      <category
            name=”Fispace”
            id=”eu.fispace.plugin.commands.category”>
      </category>
      <command
            name=”EPM Editor”
            categoryId=”eu.fispace.plugin.commands.category”
            id= »eu.fispace.plugin.commands.epmCommand »>
      </command>
</extension>
   <extension
         point= »org.eclipse.ui.handlers »>
      <handler
            commandId= »eu.fispace.plugin.commands.epmCommand »
            class= »eu.fispace.plugin.handlers.EpmHandler »>
      </handler>
   </extension>
   <extension
         point=”org.eclipse.ui.bindings”>
      <key
```

```
            commandId=”eu.fispace.plugin.commands.epmCommand”
            contextId=”org.eclipse.ui.contexts.window”
            sequence=”M1+6”
            schemeId=”org.eclipse.ui.defaultAcceleratorConfiguration”>
      </key>
   </extension>
   <extension
         point= »org.eclipse.ui.menus »>
      <menuContribution
            locationURI= »menu :org.eclipse.ui.main.menu ?after=additions »>
         <menu
               label= »Fispace »
               mnemonic= »I »
               id= »eu.fispace.plugin.menus.fispaceMenu »>
            <command
                  commandId=”eu.fispace.plugin.commands.epmCommand”
                  mnemonic=”S”
                   icon=”icons/toolbar_icon.gif”
                  id=”eu.fispace.plugin.menus.epmCommand”>
            </command>
         </menu>
      </menuContribution>
      <menuContribution
            locationURI=”toolbar:org.eclipse.ui.main.toolbar?after=additions”>
         <toolbar
               id=”eu.fispace.plugin.toolbars.epmToolbar”>
            <command
                  commandId="eu.fispace.plugin.commands.epmCommand"
                  icon="icons/toolbar_icon.gif"
                  tooltip="EPM Editor"
                  id="eu.fispace.plugin.toolbars.epmCommand">
            </command>
         </toolbar>
      </menuContribution>
   </extension>


</plugin>
```

2.3.5 FIspace-export plugin

FIspace provides a plug-in Export *wizard* to export FIspace projects easily and suitable for deployment.

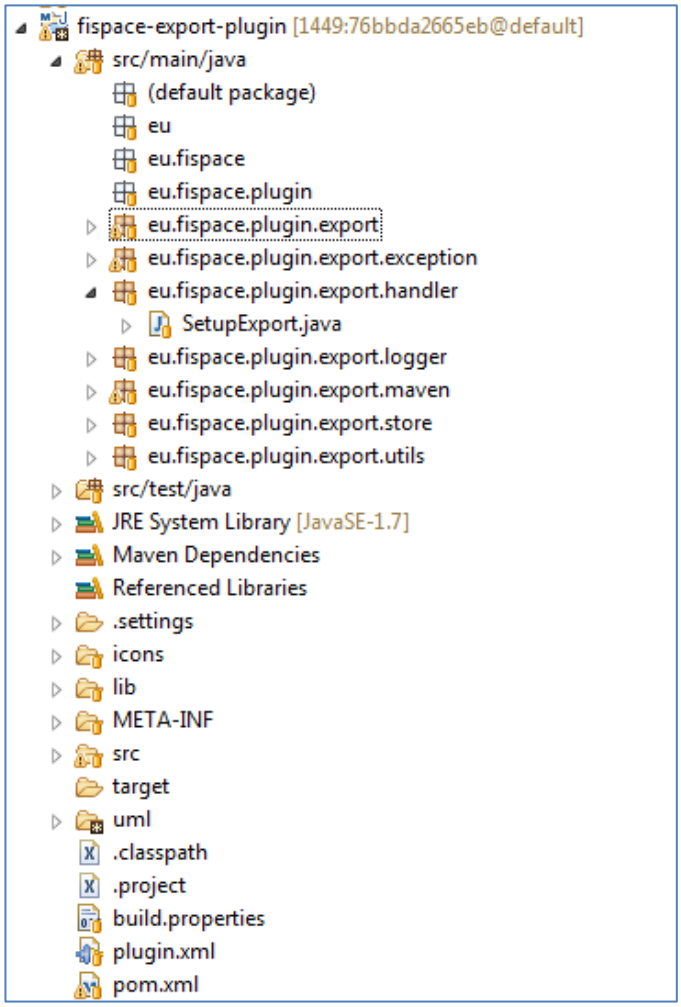


Figure 13: FIspace-export plugin project

Pom.xml

The corresponding pom file is shown in next figure:

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>eu.fispace.plugin.fispace-export</groupId>
        <artifactId>fispace-export</artifactId>
        <version>0.14.0-SNAPSHOT</version>
        <relativePath>../pom.xml</relativePath>
    </parent>

    <groupId>eu.fispace.plugin.fispace-export-plugin</groupId>
    <artifactId>fispace-export-plugin</artifactId>
    <version>0.14.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <name>fispace-export-plugin</name>
    <build>
        <plugins>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.1</version>
                <configuration>
                    <source>1.7</source>
                    <target>1.7</target>
                </configuration>
            </plugin>
        </plugins>
        <pluginManagement>
            <plugins>
                <plugin>
                    <artifactId>maven-eclipse-plugin</artifactId>
                    <version>2.9</version>
                    <configuration>
                        <pde>true</pde>
                    </configuration>
                </plugin>
```

Figure 14: The FIspace-export plugin parent pom.xml

Package Diagram:

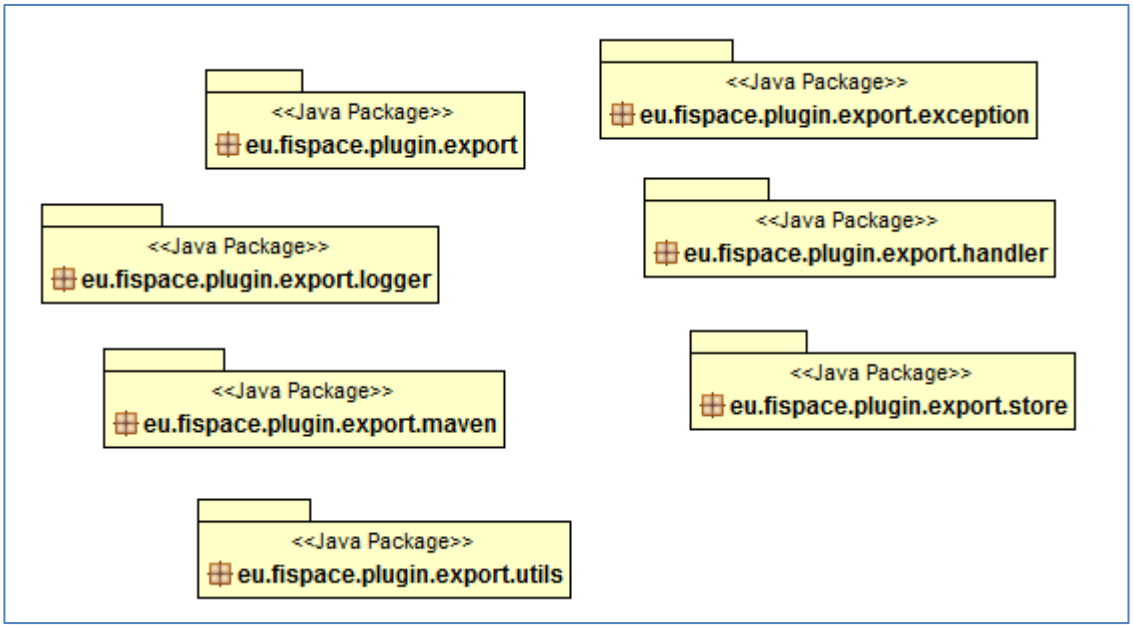


Figure 15: The FIspace-export plugin package diagram.

The FIspace Export key classes are:

Package eu.fispace.plugin.export key classes:

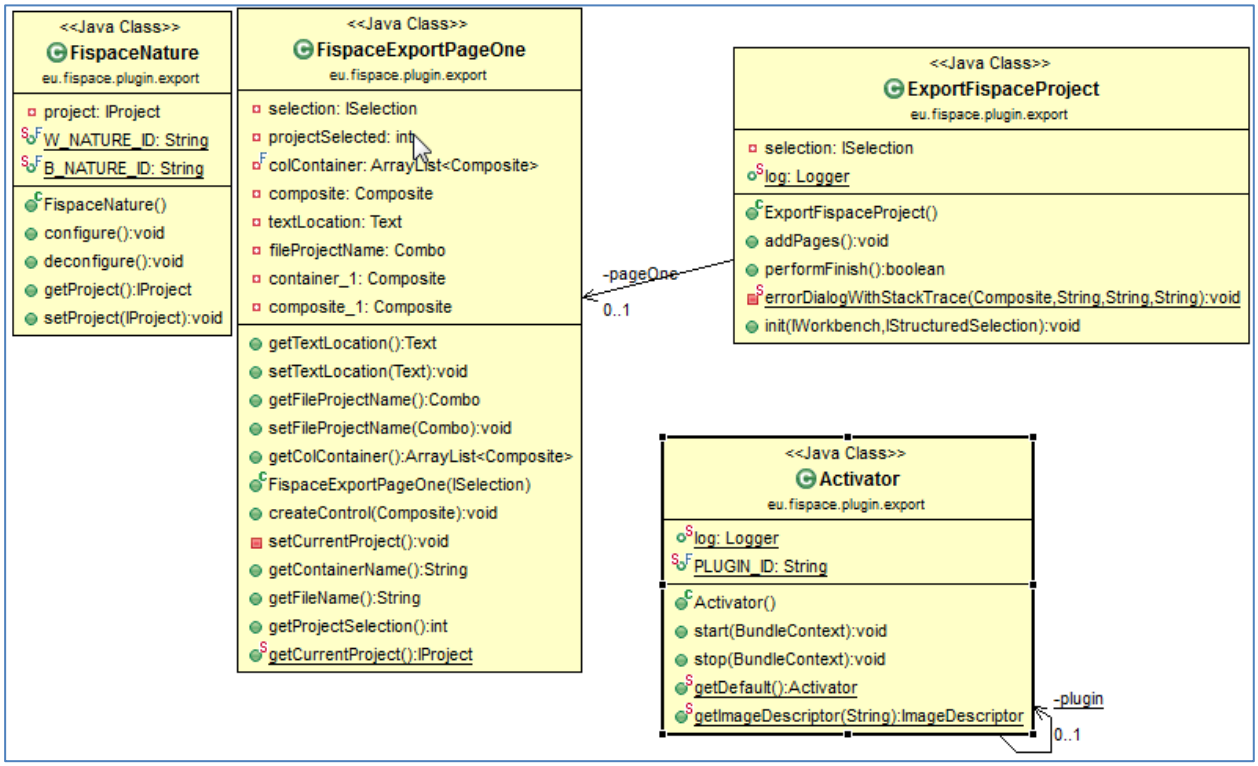


Figure 16: *eu.fispace.plugin.export diagram* classes.

Package eu.fispace.plugin.export.utils key classes:

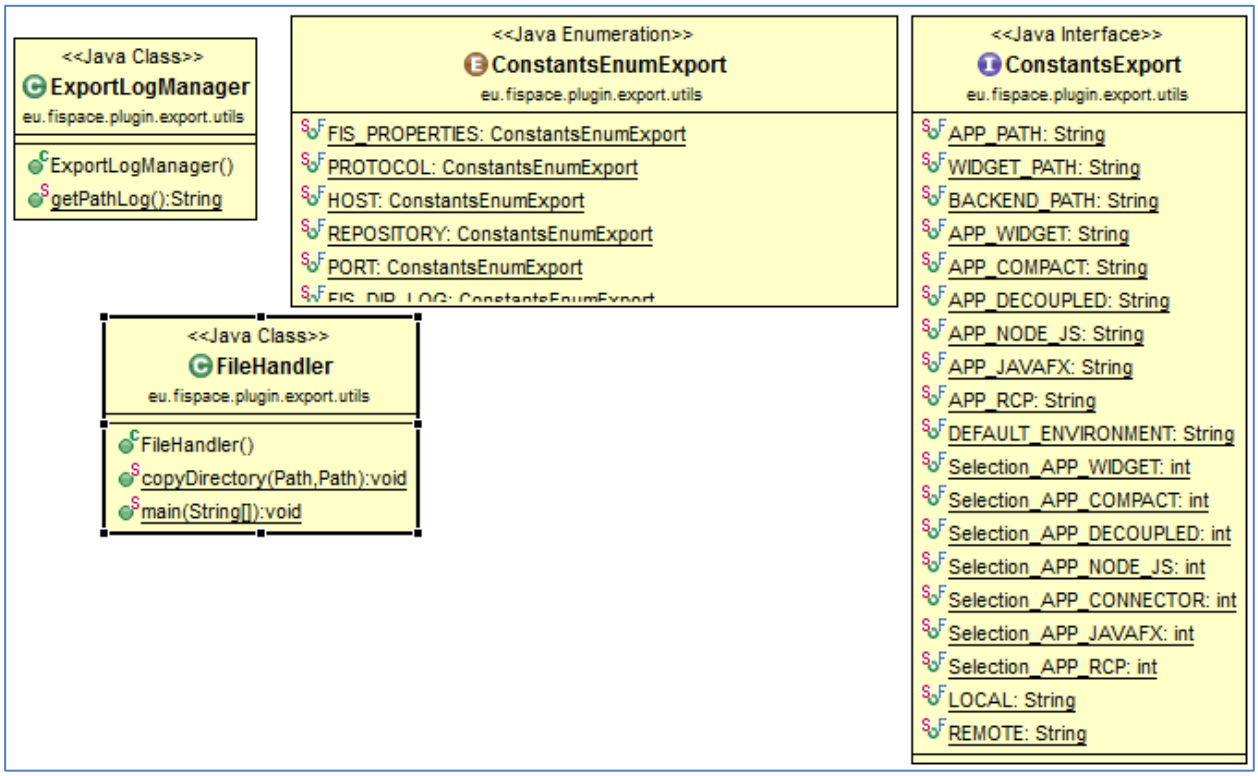


Figure 17: *eu.fispace.plugin.export.utils* classes.

2.3.6 FIspace-graphic plugin

The idea with this plugin is to integrate systems, orchestrate Web services, and smoothly connect anything to anything.

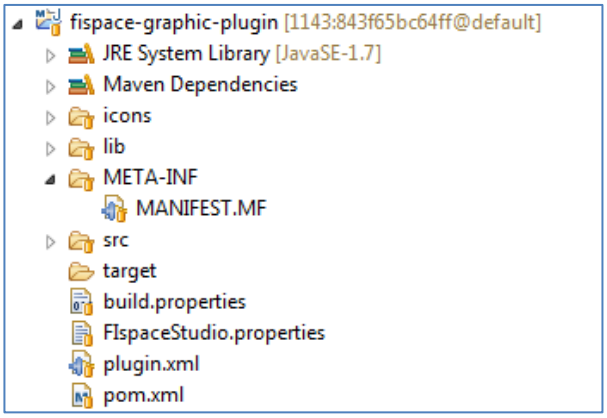


Figure 9: FIspace-graphic plugin project. Part 1.

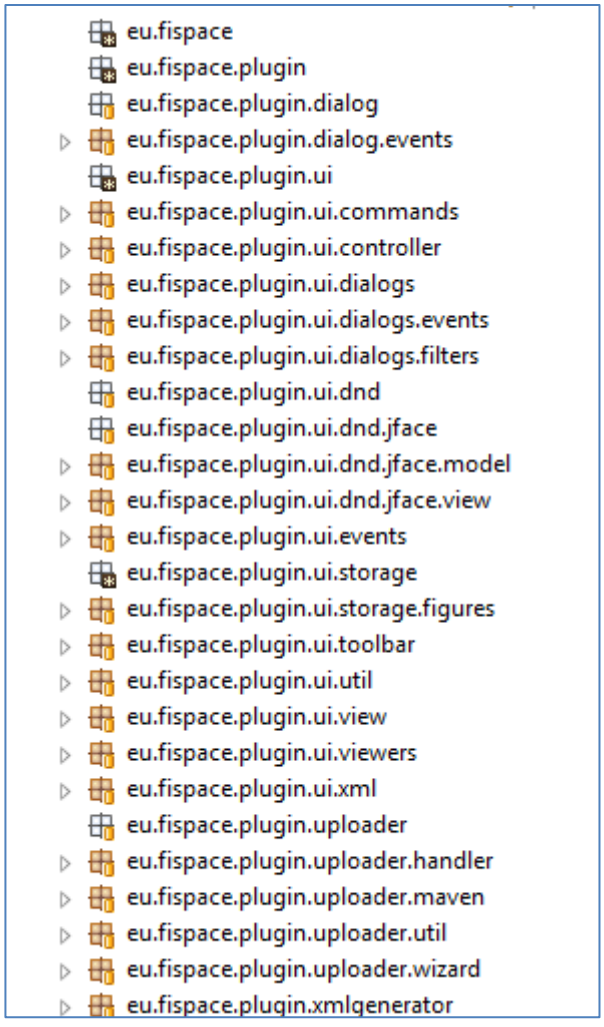


Figure 10: FIspace-graphic plugin project. Part 2.

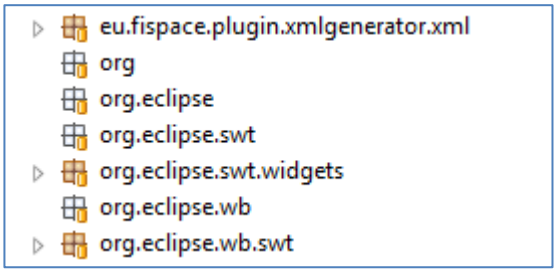


Figure 10: FIspace-graphic plugin project. Part 3.

Pom.xml

fispace-graphic/pom.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project
3      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"
4      xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
5      <modelVersion>4.0.0</modelVersion>
6
7      <groupId>eu.fispace.plugin.fispace-graphic</groupId>
8      <artifactId>fispace-graphic</artifactId>
9      <version>0.8.1</version>
10     <name>FIspace-Graphic-Plugin-Parent</name>
11     <packaging>pom</packaging>
12     <modules>
13         <module>fispace-graphic-plugin</module>
14         <module>fispace-graphic-feature</module>
15         <module>fispace-graphic-updatesite</module>
16     </modules>
17     <parent>
18         <groupId>eu.fispace.plugin</groupId>
19         <artifactId>plugin-parent</artifactId>
20         <version>0.8.1</version>
21         <relativePath>../pom.xml</relativePath>
```

Figure11: FIspace-graphic plugin Pom.xml file.

Package Diagram:

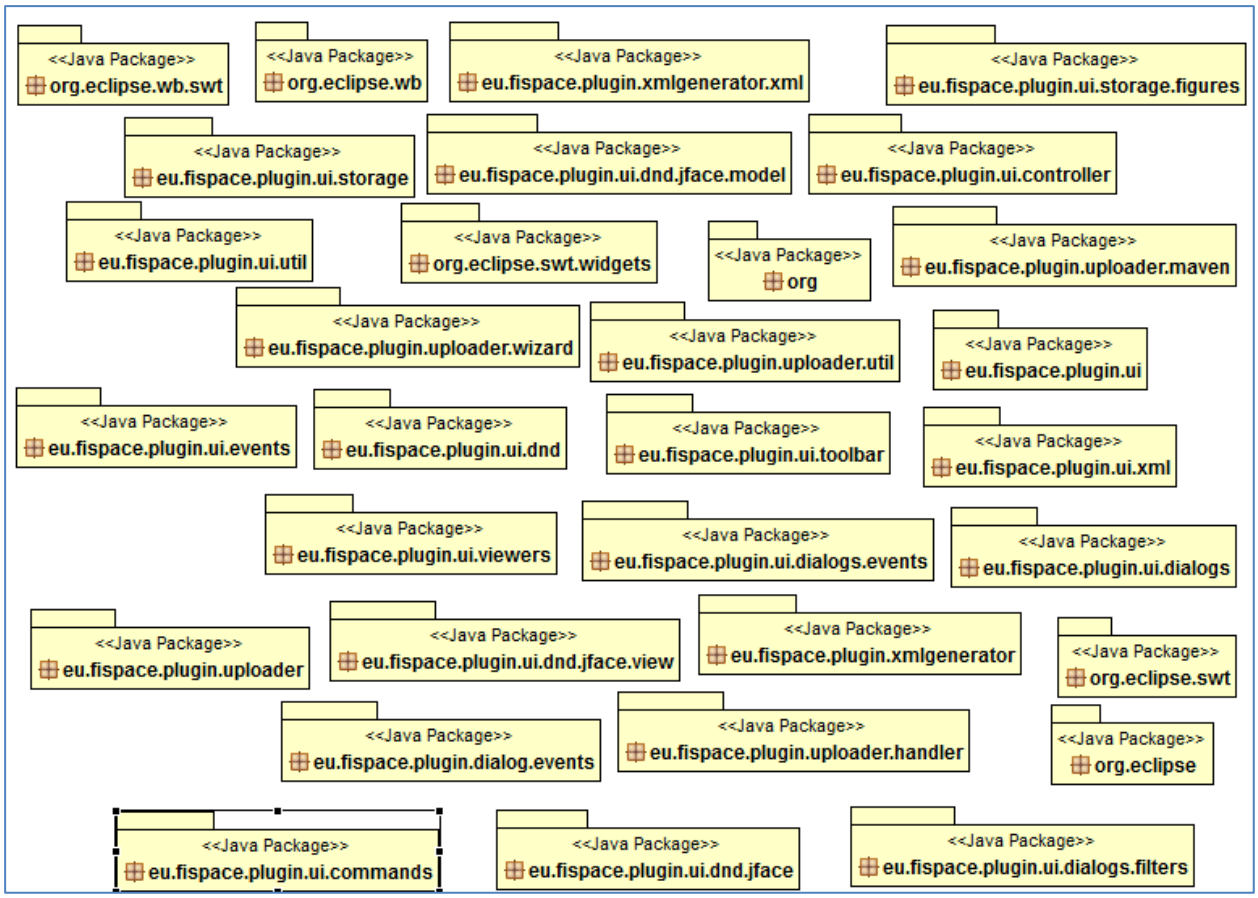


Figure 12: FIspace Graphic Package Diagram

Plugin.xml of Graphic plugin is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.4"?>
<plugin>

   <extension
         id="application"
         point="org.eclipse.core.runtime.applications">
      <application>
         <run
               class="de.vogella.dnd.jface.Application">
         </run>
      </application>
   </extension>
   <extension
         point="org.eclipse.ui.perspectives">
      <perspective
            name="FIspace (Graphics)"
            class="de.vogella.dnd.jface.Perspective"
            fixed="true"
            icon="icons/fispace_icon.gif"
            id="de.vogella.dnd.jface.perspective">
      </perspective>
   </extension>
   <extension
         point="org.eclipse.ui.views">
       <category
          name="FIspace"
          id="fispace-tool" parentCategory="fispace-tool">
      </category>
```

```
        <view
        category="fispace-tool"
            class="eu.fispace.plugin.ui.dnd.jface.view.CanvasView"
            id="eu.fispace.plugin.ui.dnd.jface.view.canvasView"
             icon="icons/CollaborationDiagram.gif"
            name="FIspace Visual Components"
            restorable="true">
      </view>
        <view
        category="fispace-tool"
              class="eu.fispace.plugin.ui.dnd.jface.view.TreeView"
              icon="icons/alt_window_16.gif"
              id="eu.fispace.plugin.ui.dnd.jface.view.treeview"
              name="FIspace Graphic Explorer">
        </view>
   </extension>
   <extension
         point="org.eclipse.ui.perspectiveExtensions">
      <perspectiveExtension
            targetID="*">
         <view
               id="eu.fispace.plugin.ui.dnd.jface.view.treeview"
               minimized="false"
               relationship="left"
               relative="org.eclipse.ui.editorss">
         </view>
         <view
               id="eu.fispace.plugin.ui.dnd.jface.view.canvasView"
               minimized="false"
               relationship="right"
               relative="org.eclipse.ui.editorss">
         </view>
      </perspectiveExtension>
   </extension>
   <extension
         point="org.eclipse.ui.commands">
      <command
            defaultHandler="eu.fispace.plugin.ui.commands.ExitHandler"
            id="fispace-graphic-plugin.command1"
            name="Exist">
      </command>
   </extension>

</plugin>
```

2.3.7 FIspace-import plugin

An import wizard is provided for selecting existing FIspace projects to be added on the eclipse workspace.

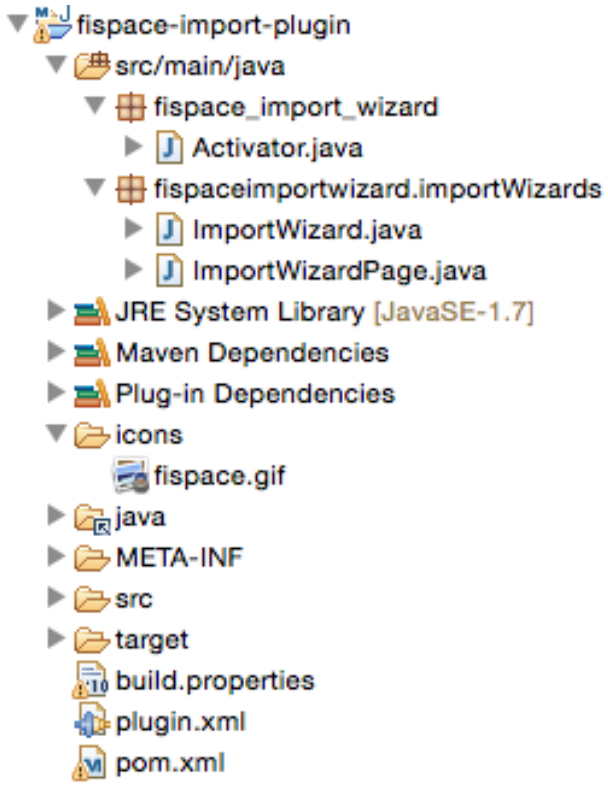


Figure 10: FIspace Import project structure

The plugin.xml file is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.4"?>
<plugin>

   <extension
         point="org.eclipse.ui.importWizards">
      <category
            name="FIspace"
            id="fispaceimportwizard.importWizards.sampleCategory">
      </category>
      <wizard
            name="Existing FIspace Project"
            icon="icons/fispace.gif"
            category="fispaceimportwizard.importWizards.sampleCategory"
            class="fispaceimportwizard.importWizards.ImportWizard"
            id="fispaceimportwizard.importWizards.ImportWizard">
         <description>
                        Import Existing FIspace Projects
         </description>
      </wizard>
   </extension>

</plugin>
```

The following class diagram shows the most important classes defined in this plugin:

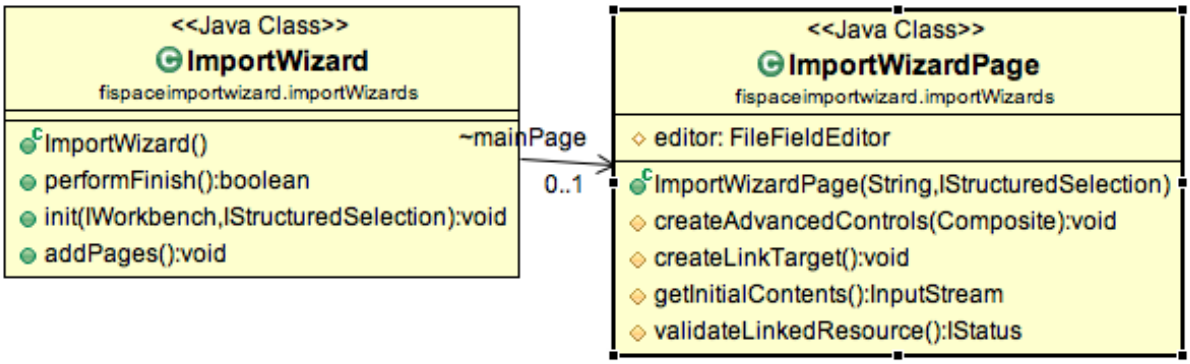


Figure 11: FIspace Import plugin class diagram

2.3.8 FIspace-perspective plugin

Perspective plugin provides a custom eclipse perspective view which includes a custom FIspace project explorer as well as some shortcuts for the rest of FIspace SDK tools.

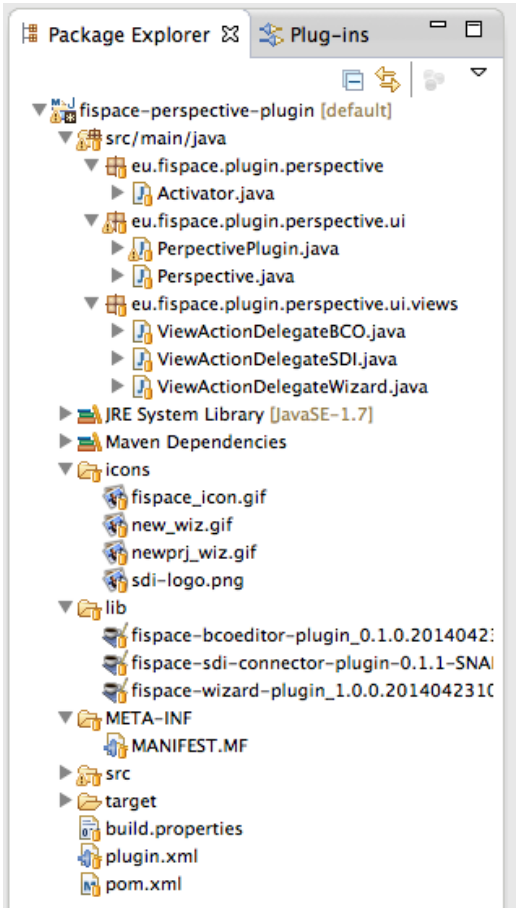


Figure 12: FIspace perspective project structure

The plugin.xml file is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.2"?>
<plugin>
      <extension
         point="org.eclipse.ui.perspectives">
      <perspective
            class="eu.fispace.plugin.perspective.ui.PerpectivePlugin"
            fixed="true"
            icon="icons/fispace_icon.gif"
            id="Perspective.perspective2"
            name="FIspace Studio">
      </perspective>
   </extension>
         <!--extension
    point = "org.eclipse.ui.viewActions">
      <viewContribution
         id="eu.fispace.plugin.perspective.ui.views.vc1"
         targetID="org.eclipse.ui.navigator.ProjectExplorer">
           <action id="eu.fispace.plugin.perspective.ui.views.va1"
              label="SDI"
              menubarPath="additions"
              toolbarPath="additions"
              icon="icons/sdi-logo.png"
              style="push"
```

```
            tooltip="SDI"
            helpContextId="eu.fispace.plugin.perspective.ui.views.view_action_context"
             class="eu.fispace.plugin.perspective.ui.views.ViewActionDelegateSDI">
          </action>
     </viewContribution>
 </extension-->

  <extension
    point = "org.eclipse.ui.viewActions">
      <viewContribution
         id="eu.fispace.plugin.perspective.ui.views.vc2"
         targetID="org.eclipse.ui.navigator.ProjectExplorer">
           <action id="eu.fispace.plugin.perspective.ui.views.va2"
              label="BCO Editor"
              menubarPath="additions"
              toolbarPath="additions"
              icon="icons/fispace_icon.gif"
              style="push"
              tooltip="BCO Editor"
              helpContextId="eu.fispace.plugin.perspective.ui.views.view_action_context"
              class="eu.fispace.plugin.perspective.ui.views.ViewActionDelegateBCO">
           </action>
     </viewContribution>
 </extension>

  <extension
    point = "org.eclipse.ui.viewActions">
      <viewContribution
         id="eu.fispace.plugin.perspective.ui.views.vc3"
         targetID="org.eclipse.ui.navigator.ProjectExplorer">
           <action id="eu.fispace.plugin.perspective.ui.views.va3"
              label="FIstudio Wizard"
              menubarPath="additions"
              toolbarPath="additions"
              icon="icons/newprj_wiz.gif"
              style="push"
              tooltip="FIstudio Wizard"
              helpContextId="eu.fispace.plugin.perspective.ui.views.view_action_context"
              class="eu.fispace.plugin.perspective.ui.views.ViewActionDelegateWizard">
           </action>
     </viewContribution>
 </extension>
  <extension
         point="org.eclipse.ui.views">
         <category
            name="FIspace"
            id="fispace-tool">
     </category>
     <view
           category="fispace-tool"
           class="org.eclipse.ui.navigator.resources.ProjectExplorer"
           id="org.eclipse.ui.navigator.ProjectExplorer"
           icon="icons/fispace_icon.gif"
           name="FIspace Explorer"/>
   </extension>
</plugin>
```

2.3.9 FIspace-preferences plugin

Each FIspace plug-in has a local workspace preference store for saving arbitrary primitive data types and strings. FIspace SDK Preferences displayed in the Workbench Preferences dialog are typically stored in these local plug-in preference Fispace stores.

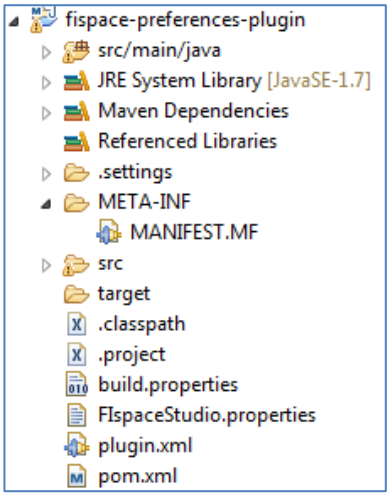


Figure 13: FIspace-preferences plugin project

Pom.xml

fispace-preferences/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"
    xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <modelVersion>4.0.0</modelVersion>

    <groupId>eu.fispace.plugin.fispace-preferences</groupId>
    <artifactId>fispace-preferences</artifactId>
    <version>0.8.1</version>
    <name>FIspace-Preferences-Plugin-Parent</name>
    <packaging>pom</packaging>
    <modules>
        <module>fispace-preferences-plugin</module>
        <module>fispace-preferences-feature</module>
        <module>fispace-preferences-updatesite</module>
    </modules>

    <repositories>
        <repository>
            <id>eclipse keppler</id>
            <layout>p2</layout>
            <url>http://download.eclipse.org/releases/kepler</url>
        </repository>
    </repositories>

    <parent>
        <groupId>eu.fispace.plugin</groupId>
        <artifactId>plugin-parent</artifactId>
        <version>0.8.1</version>
        <relativePath>../pom.xml</relativePath>
    </parent>
</project>
```

Key Package Diagram:

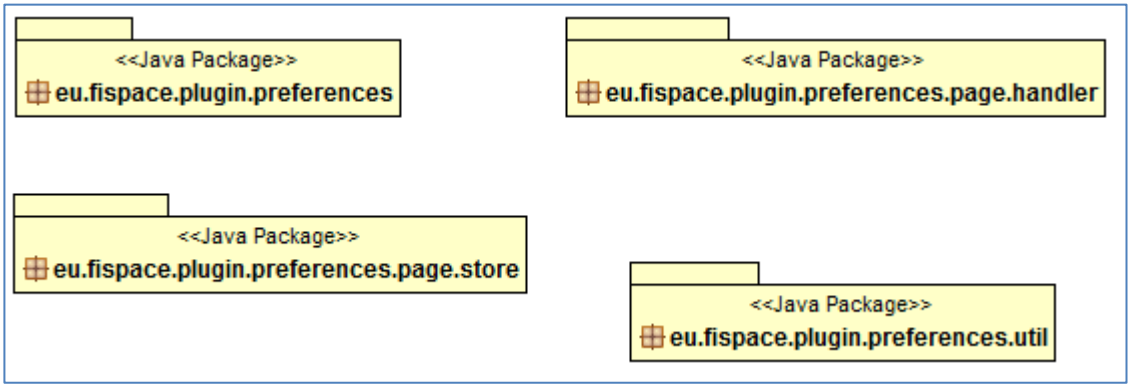


Figure 14: FIspace- preferences package diagram

2.3.10 FIspace-pubsub plugin

This plugin shows the capabilities of information transmission from different layers and systems through *RabbitMQ* (open source message broker software that implements the Advanced Message Queuing Protocol – AMQP – http://www.rabbitmq.com/).

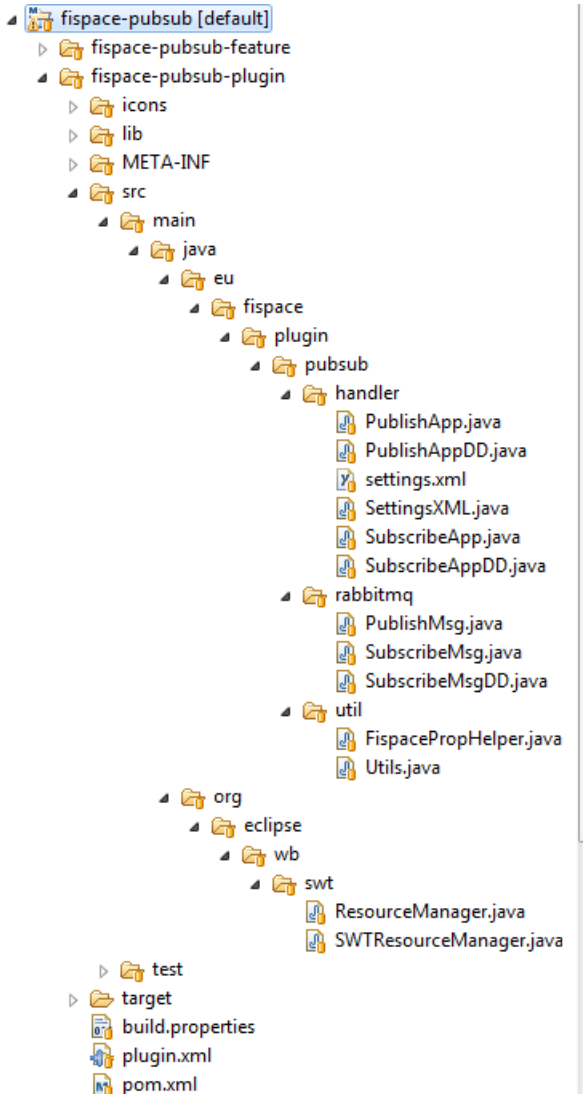


Figure 15: Fispace-pubsub plugin project

Handler directory contains the class where are defined the screens of the plugin and the calls to the classes included into the **rabbitmq** directory.

Into the **rabbitmq** directory we have a collection of classes needed to manage the connections with the rabbitmq server.

Extension points declared in plugin.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.4"?>
<plugin>
   <extension
         point="org.eclipse.ui.menus">

      <menuContribution
            locationURI="menu:org.eclipse.ui.main.menu?after=additions">
```

```
         <menu
          label="FIspace"
                   mnemonic="I"
                id="eu.fispace.plugin.menus.fispaceMenu">
                <menu
                           label="Publish/Subscribe"
                                    mnemonic="S"
                                    id="eu.fispace.plugin.menus.pubsubMenu"
icon="icons/allversions_rep.gif">
                                  <command
                                       commandId="eu.fispace.plugin.publish"
                                       label="Publish Message"
                                       icon="icons/mail_send.png">
                                  </command>
                               <command
                                    commandId="eu.fispace.plugin.subscribe"
                                    label="Subscribe Message"
                                    icon="icons/mail_receive.png">
                               </command>
                          </menu>
                 </menu>
      </menuContribution>
    </extension>

   <extension
         point="org.eclipse.ui.commands">
      <command
            id="eu.fispace.plugin.publish"
            name="Publisher">
      </command>
      <command
            id="eu.fispace.plugin.subscribe"
            name="Subscriber">
      </command>
   </extension>

   <extension
          point="org.eclipse.ui.handlers">
        <handler
                          class="eu.fispace.plugin.pubsub.handler.PublishApp"
                          commandId="eu.fispace.plugin.publish">
               </handler>
               <handler
                          class="eu.fispace.plugin.pubsub.handler.SubscribeApp"
                          commandId="eu.fispace.plugin.subscribe">
               </handler>
    </extension>

</plugin>
```

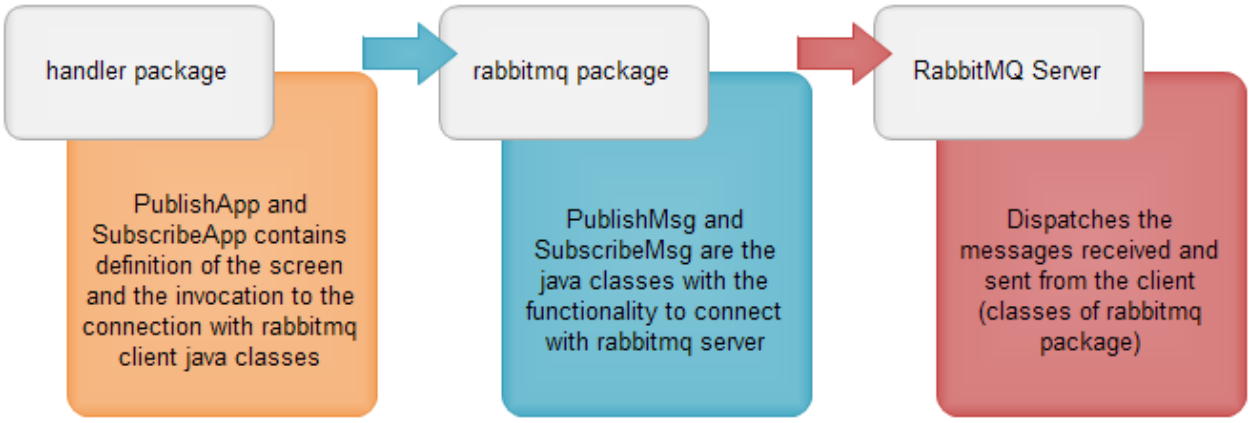
Main dependencies declared:

- rabbitmq-client: it allows the connection with rabbitmq server to publish and subscribe messages.

Key classes:

This plugin works in two separate ways:

- Called from the handler, with a screen for publish or subscribe a message connecting with the rabbitmq server.
- From another plugin using the properties file of fispace.
    - Called from the handler:



    - Called from another plugin:

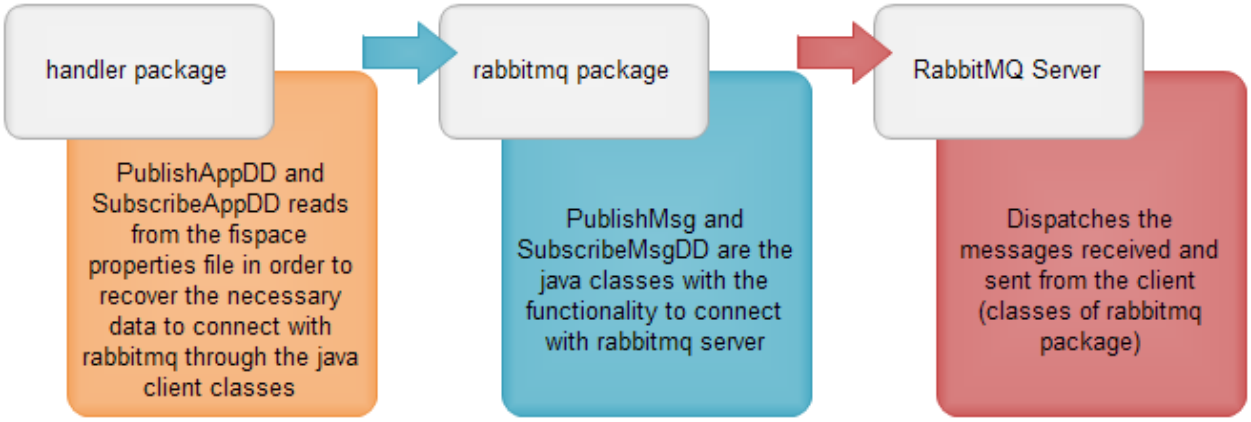


Figure 16: Key classes in Publish/Subscription plugin

2.3.11 FIspace-REST plugin

Rest Client Generator GE (plugin version) has been used to generate Rest Client applications.

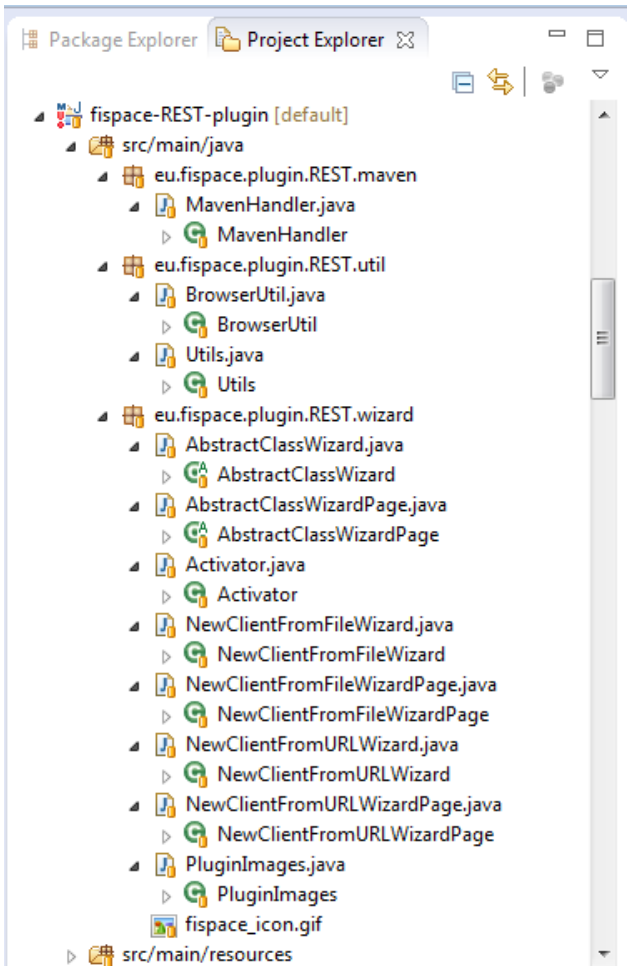


Figure 17: FIspace REST plugin project

Package model is shown here:

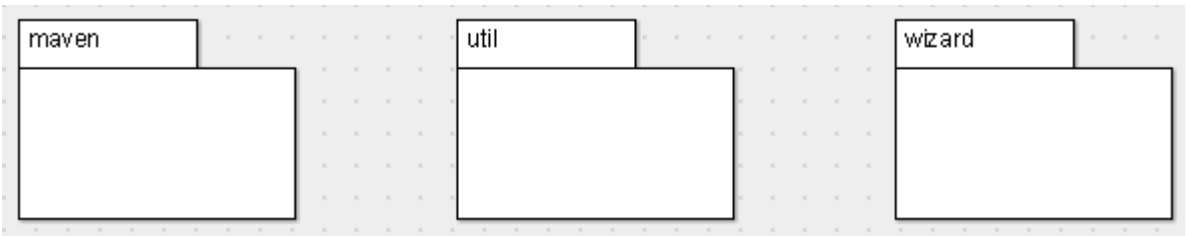


Figure 18: REST plugin package diagram

Some class diagrams are included now:

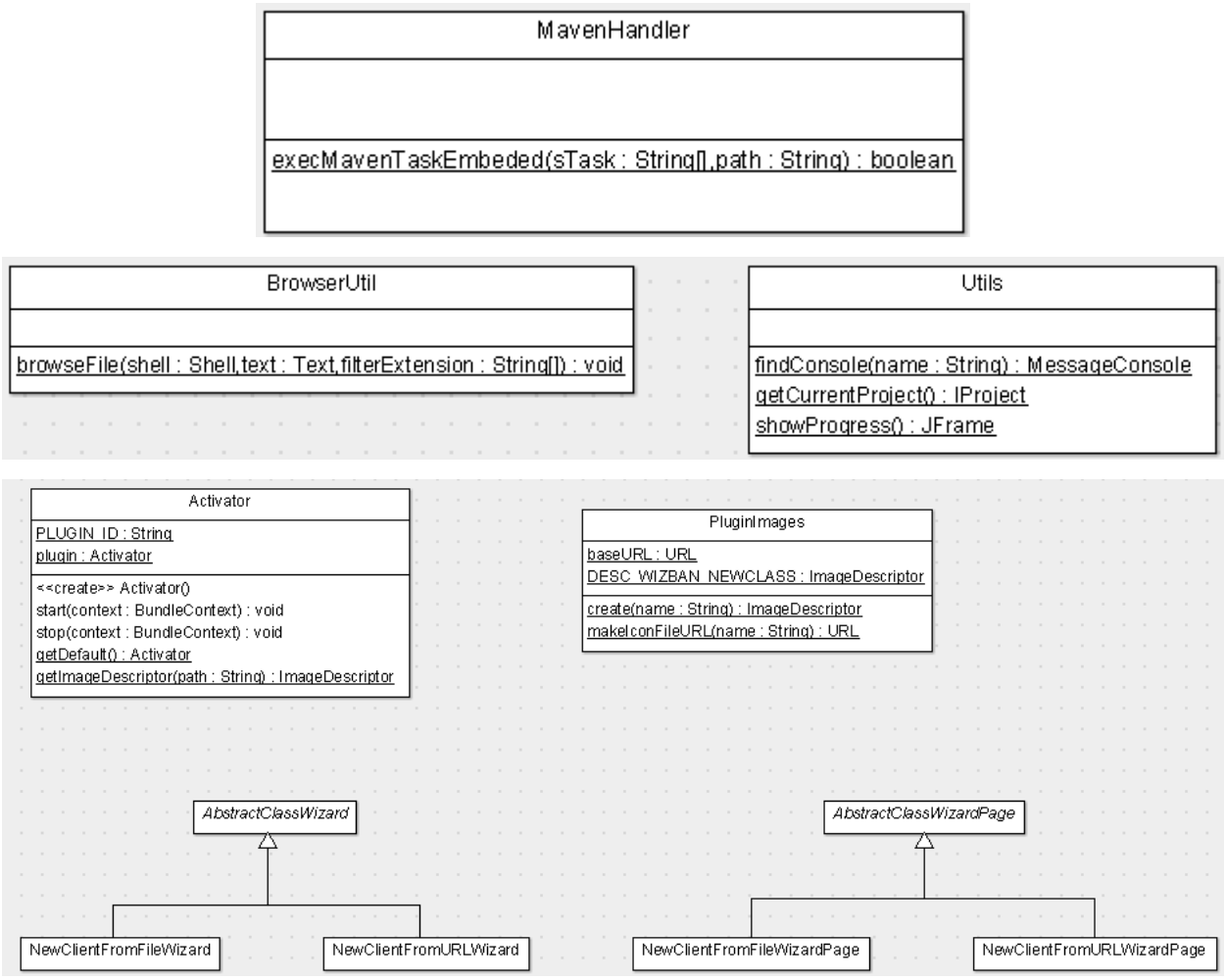


Figure 19: Classes in REST wizard (I)

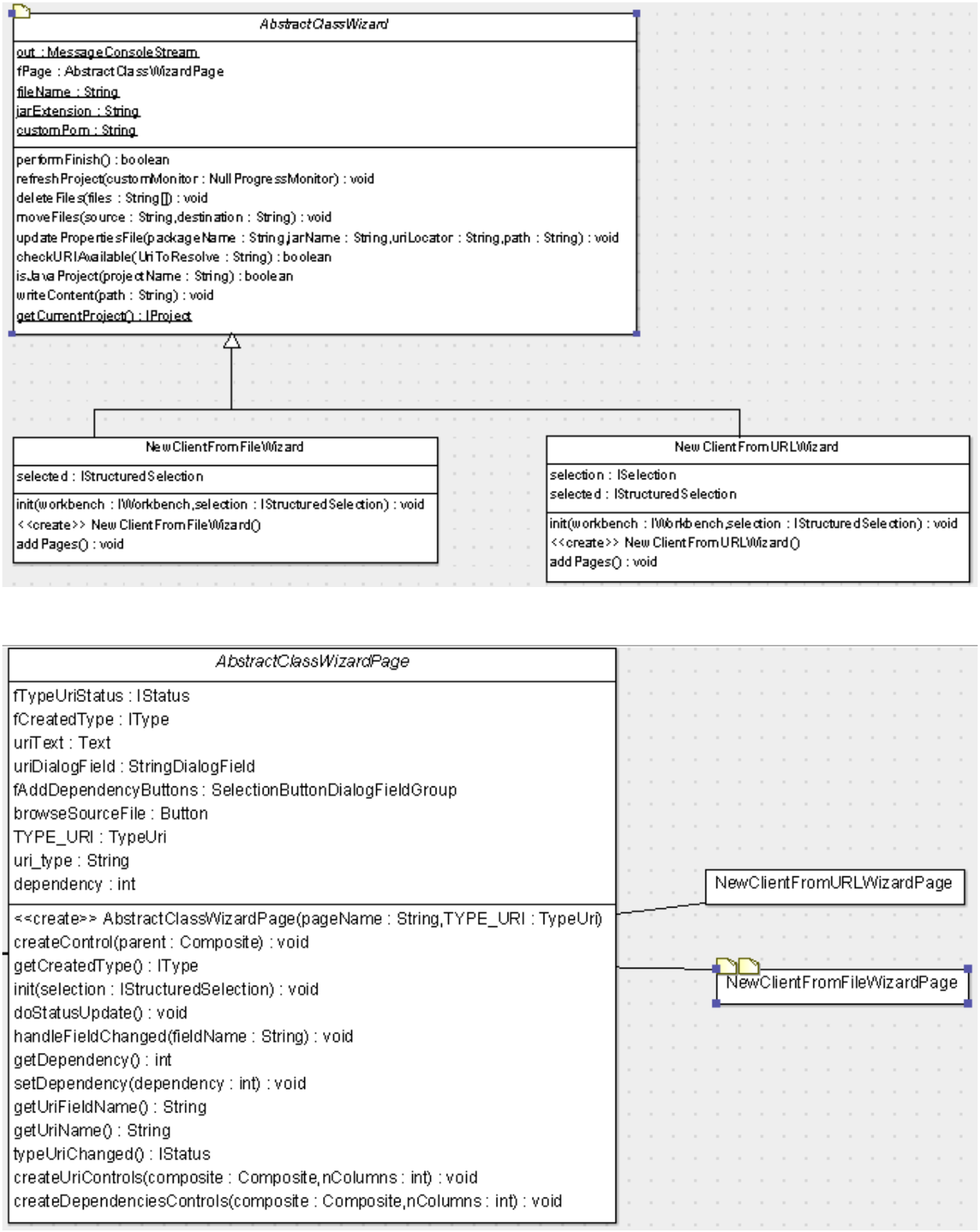


Figure 20: Classes in REST wizard (II)

The plugin.xml is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.2"?>
<plugin>
    <extension point="org.eclipse.ui.startup">
       <startup class="eu.fispace.plugin.REST.wizard.Activator"/>
    </extension>
```

```
<extension
        point="org.eclipse.ui.newWizards">
     <category
           id="create.rest.client"
           name="REST Client Generator"/>
      <wizard
           category="create.rest.client"
           hasPages="true"
           icon="icons/newclass_wiz_url.gif"
           id="eu.fispace.plugin.REST.wizard.Activator"
           name="Generate client from URL"
           project="true"
                      class="eu.fispace.plugin.REST.wizard.NewClientFromURLWizard"/>
           <wizard
           category="create.rest.client"
           hasPages="true"
           icon="icons/newclass_wiz_file.gif"
           id="eu.fispace.plugin.REST.wizard.NewClientFromFileWizard"
           name="Generate client from file"
           project="true"
                      class="eu.fispace.plugin.REST.wizard.NewClientFromFileWizard"/>
       </extension>


</plugin>
```

2.3.12 FIspace-sdi-connector plugin

It allows the final user to experiment with SDI module *locally*. Using an easy-to-use user interface, the user will be able to perform a complete set of tests on the SDI. In this functionality, it is included a Rest Client to execute the available instructions. For this purpose, this feature is provided as an eclipse view, allowing end-users to test different services in different machines, not only focusing on SDI connector plugin.

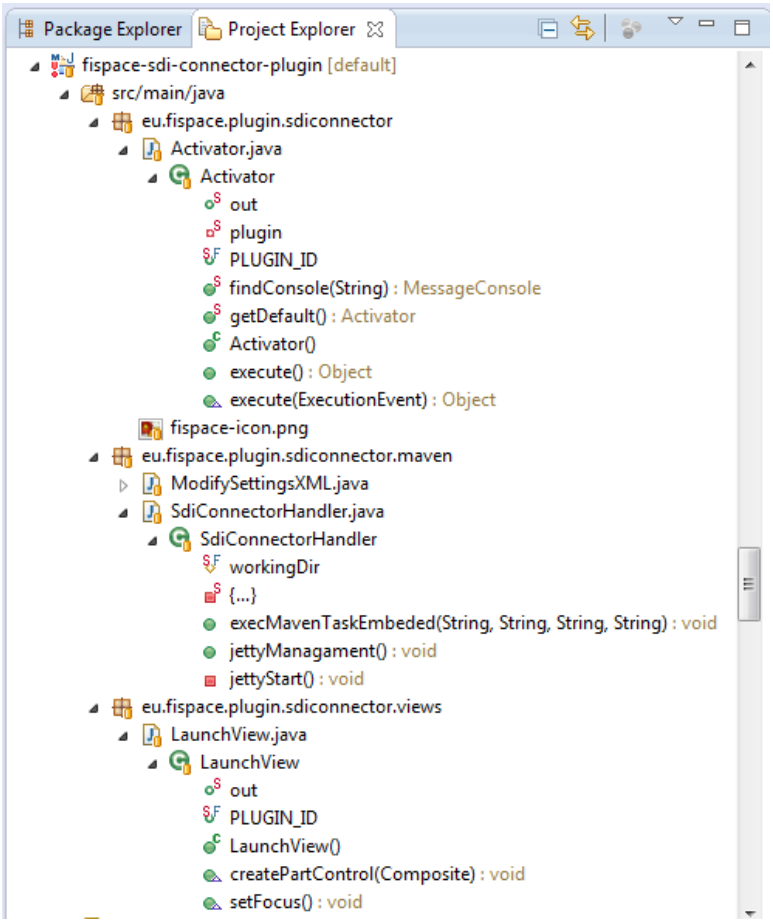


Figure 21: FIspace SDI connector plugin project

Key classes are the following:

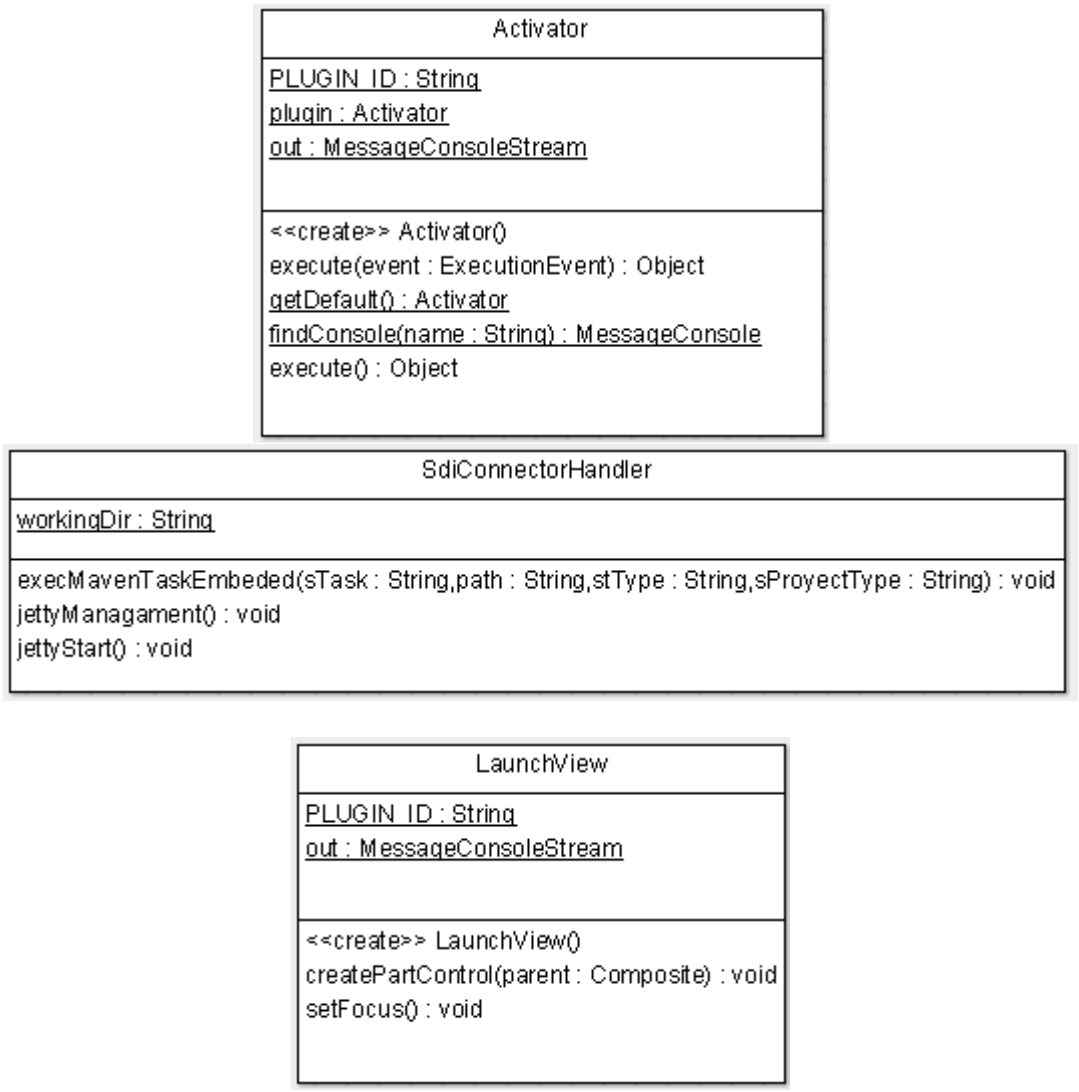


Figure 22: SDI connector key classes

The plugin.xml is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.4"?>
<plugin>

      <extension
         point="org.eclipse.ui.perspectiveExtensions">
      <perspectiveExtension
            targetID="org.eclipse.jdt.ui.JavaPerspective">
         <view
               ratio="0.5"
               relative="org.eclipse.ui.views.TaskList"
               relationship="right"
               id="eu.fispace.plugin.sdiconnector.views.LaunchView">
         </view>
      </perspectiveExtension>
   </extension>

   <extension
         point="org.eclipse.ui.views">
      <category
            name="FIspace"
            id="fispace-tool">
      </category>
      <view
            name="SDI Rest Tool"
            icon="icons/sdi-logo.png"
            category="fispace-tool"
            class="eu.fispace.plugin.sdiconnector.views.LaunchView"
            id="eu.fispace.plugin.sdiconnector.views.LaunchView">
      </view>
   </extension>

   <extension
         point="org.eclipse.ui.handlers">
      <handler
            commandId="eu.fispace.plugin.commands.sdiCommand"
            class="eu.fispace.plugin.sdiconnector.Activator">
      </handler>
   </extension>

   <extension
         point="org.eclipse.ui.commands">
         <category
            name="SDI Plugin"
            id="rc-tool">
      </category>
      <category
            name="FIspace"
            id="eu.fispace.plugin.commands.category">
      </category>
      <command
            name="SDI Connector"
            categoryId="eu.fispace.plugin.commands.category"
            id="eu.fispace.plugin.commands.sdiCommand">
      </command>
   </extension>
   <extension
         point="org.eclipse.ui.perspectiveExtensions">
      <perspectiveExtension
            targetID="org.eclipse.jdt.ui.JavaPerspective">
         <view
               ratio="0.5"
               relative="org.eclipse.ui.views.TaskList"
               relationship="right"
               id="code.google.restclient.eclipse.views.LaunchView">
         </view>
      </perspectiveExtension>
   </extension>
   <extension
         point="org.eclipse.help.contexts">
      <contexts
            file="contexts.xml">
      </contexts>
   </extension>
```

```
   <extension
         point="org.eclipse.ui.bindings">
      <key
            commandId="eu.fispace.plugin.commands.sdiCommand"
            contextId="org.eclipse.ui.contexts.window"
            sequence="M1+6"
            schemeId="org.eclipse.ui.defaultAcceleratorConfiguration">
      </key>
   </extension>
   <extension
         point="org.eclipse.ui.menus">
      <menuContribution
            locationURI="menu:org.eclipse.ui.main.menu?after=additions">
         <menu
               label="FIspace"
               mnemonic="M"
               id="eu.fispace.plugin.menus.fispaceMenu">
            <command
                  commandId="eu.fispace.plugin.commands.sdiCommand"
                  mnemonic="S"
                  icon="icons/sdi-logo.png"
                  id="eu.fispace.plugin.menus.sdiCommand">
            </command>
         </menu>
      </menuContribution>
      <menuContribution
            locationURI="toolbar:org.eclipse.ui.main.toolbar?after=additions">
         <toolbar
               id="eu.fispace.plugin.toolbars.sdiToolbar">
            <command
                  commandId="eu.fispace.plugin.commands.sdiCommand"
                  icon="icons/sdi-logo.png"
                  tooltip="SDI Connector"
                  id="eu.fispace.plugin.toolbars.sdiCommand">
            </command>
         </toolbar>
      </menuContribution>
   </extension>
   <extension
         point="org.eclipse.core.resources.natures">
   </extension>


</plugin>
```

2.3.13 FIspace-sketches plugin

This plugin allows the user to draw and make some simple diagrams that can be useful to depict icons, flow diagrams, component analysis, etc.

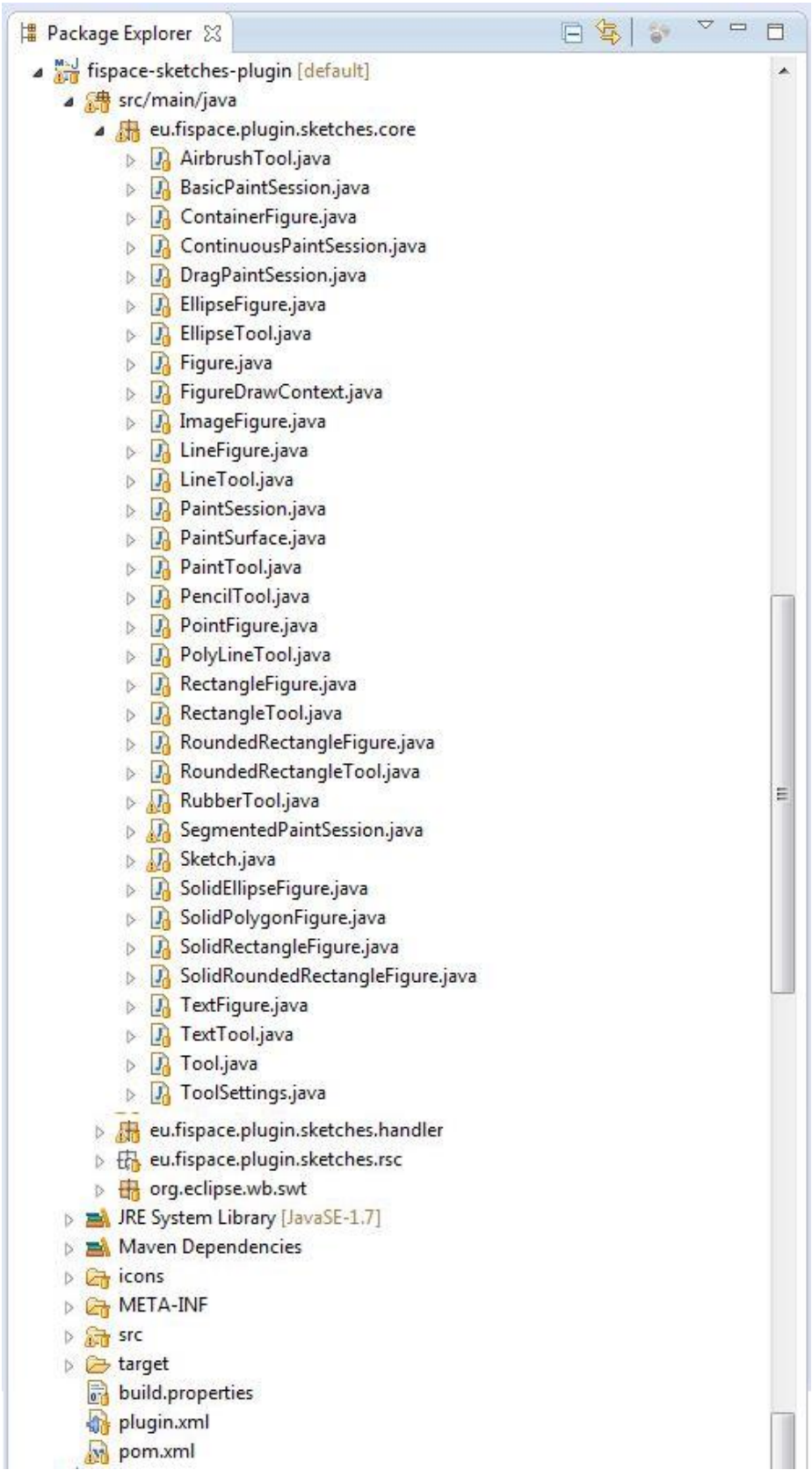


Figure 23: FIspace sketches plugin project

The plugin.xml is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.4"?>
<plugin>

   <extension
         point="org.eclipse.ui.commands">
      <category
            name="FIspace"
            id="eu.fispace.plugin.commands.category">
      </category>
      <command
            name="My Sketches"
            categoryId="eu.fispace.plugin.commands.category"
            id="eu.fispace.plugin.commands.sketchCommand">
      </command>
   </extension>
   <extension
         point="org.eclipse.ui.handlers">
      <handler
            commandId="eu.fispace.plugin.commands.sketchCommand"
            class="eu.fispace.plugin.sketches.handler.HandlerSketches">
      </handler>
   </extension>
   <extension
         point="org.eclipse.ui.bindings">
      <key
            commandId="eu.fispace.plugin.commands.sketchCommand"
            contextId="org.eclipse.ui.contexts.window"
            sequence="M1+6"
            schemeId="org.eclipse.ui.defaultAcceleratorConfiguration">
      </key>
   </extension>
   <extension
         point="org.eclipse.ui.menus">
      <menuContribution
            locationURI="menu:org.eclipse.ui.main.menu?after=additions">
         <menu
               label="FIspace"
               mnemonic="I"
               id="eu.fispace.plugin.menus.fispaceMenu">
            <command
                  commandId="eu.fispace.plugin.commands.sketchCommand"
                  mnemonic="S"
                   icon="icons/draw_icon.gif"
                  id="eu.fispace.plugin.menus.sketchCommand">
            </command>
         </menu>
      </menuContribution>
   </extension>


</plugin>
```

2.3.14 FIspace-SOAP plugin

Following a similar approach that in 2.3.11 FIspace-REST plugin, and to have to have a final product as much open as possible, this functionality generates SOAP clients automated and transparent to the end user.

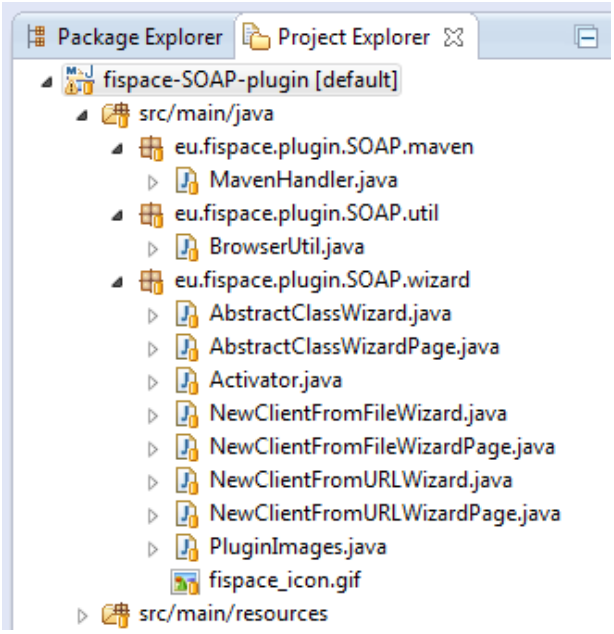


Figure 24: FIspace SOAP plugin project

The package diagram is shown below:

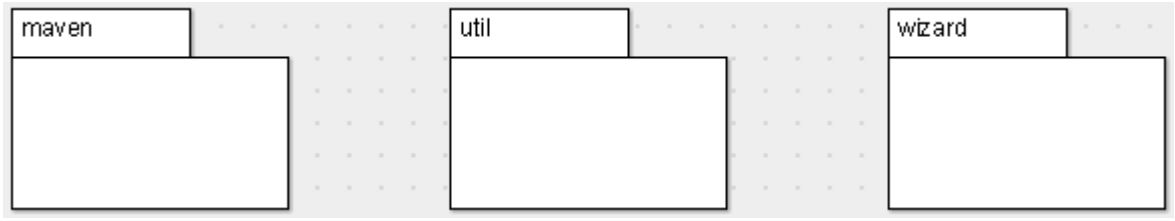
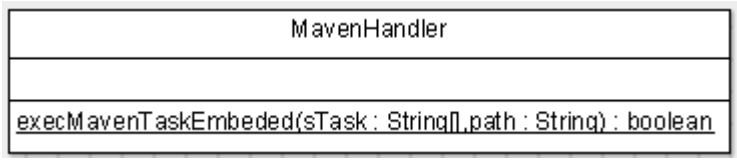
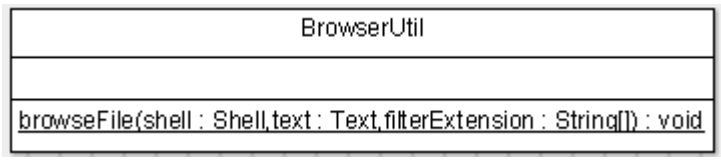


Figure 25: SOAP wizard package diagram

The maven classes:



The util classes:



The wizard classes:

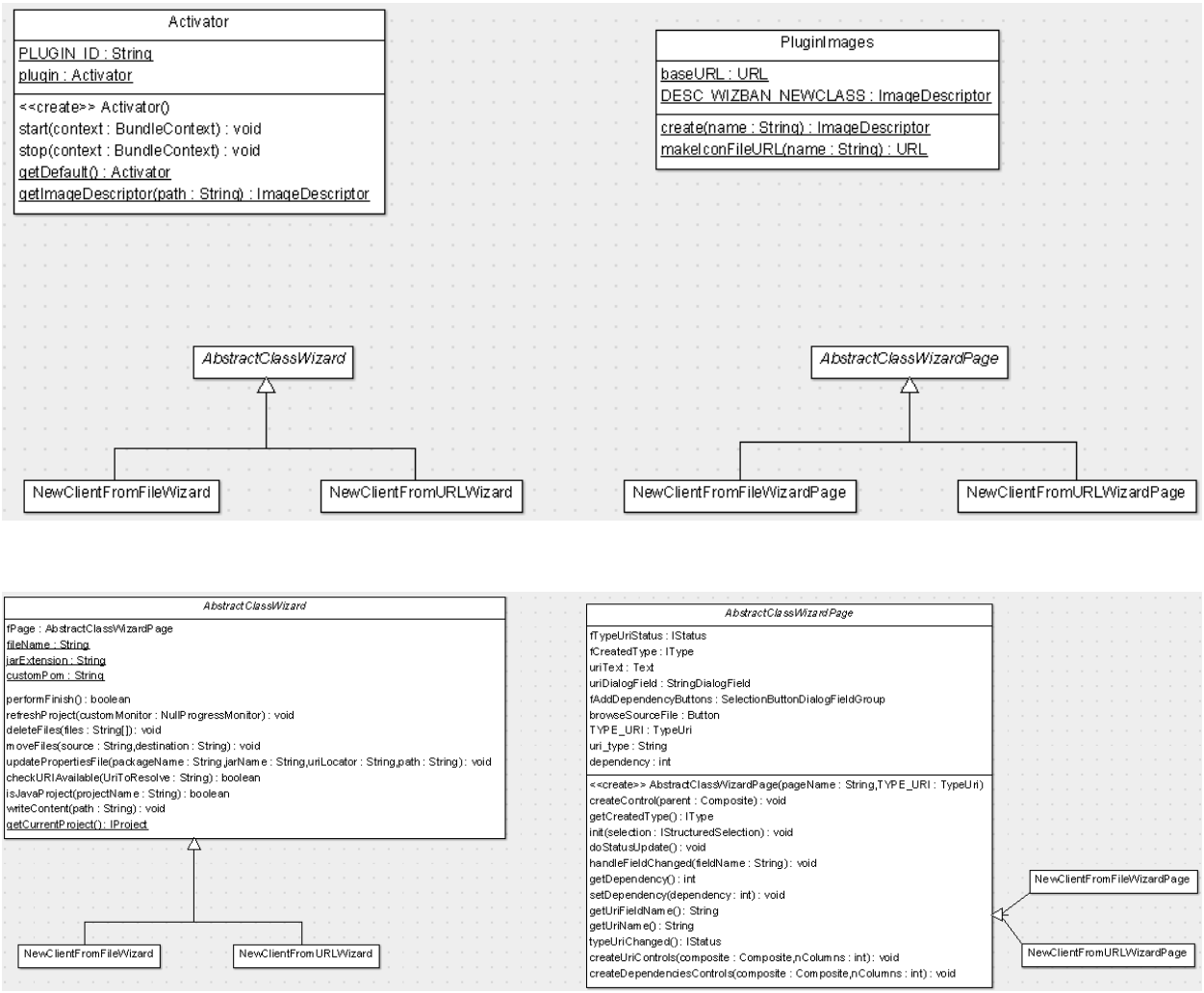
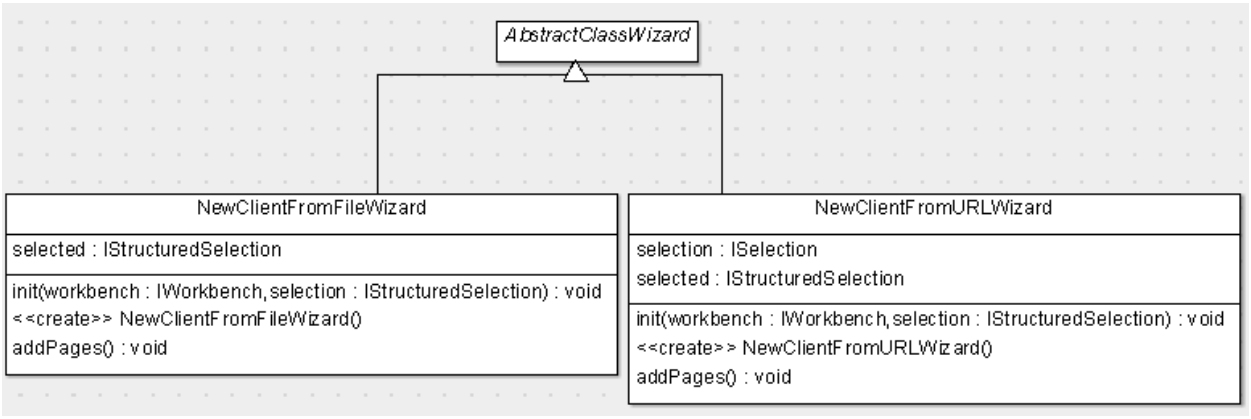


Figure 26: SOAP wizard key classes (I)

More detailed class diagrams follow:



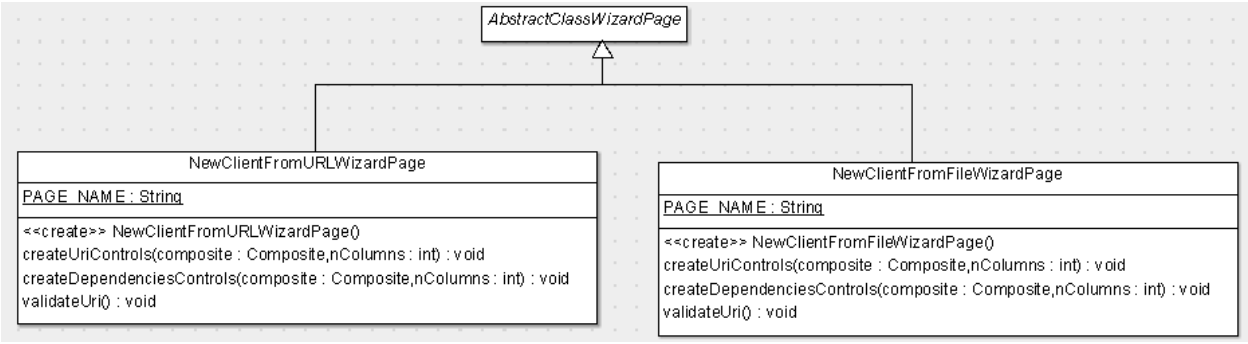


Figure 27: SOAP wizard key classes (II)

The plugin.xml is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.2"?>
<plugin>
    <extension point="org.eclipse.ui.startup">
       <startup class="eu.fispace.plugin.SOAP.wizard.Activator"/>
    </extension>

 <extension
         point="org.eclipse.ui.newWizards">
      <category
            id="create.SOAP.client"
            name="SOAP Client Generator"/>
       <wizard
            category="create.SOAP.client"
            hasPages="true"
            icon="icons/newclass_wiz_url.gif"
            id="eu.fispace.plugin.SOAP.wizard.Activator"
            name="Generate client from URL"
            project="true"
                        class="eu.fispace.plugin.SOAP.wizard.NewClientFromURLWizard"/>
            <wizard
            category="create.SOAP.client"
            hasPages="true"
            icon="icons/newclass_wiz_file.gif"
            id="eu.fispace.plugin.SOAP.wizard.NewClientFromFileWizard"
            name="Generate client from file"
            project="true"
                        class="eu.fispace.plugin.SOAP.wizard.NewClientFromFileWizard"/>
       </extension>


</plugin>
```

2.3.15 FIspace-uploader plugin

Its functionality is to upload widgets to the FIspace Store. It is exclusively for projects based on the Widget Archetype or on the Decoupled (Widget part) archetype. Once the development of the widget is completed, the user will be able to validate the project and then to upload it to a FIspace internal repository.

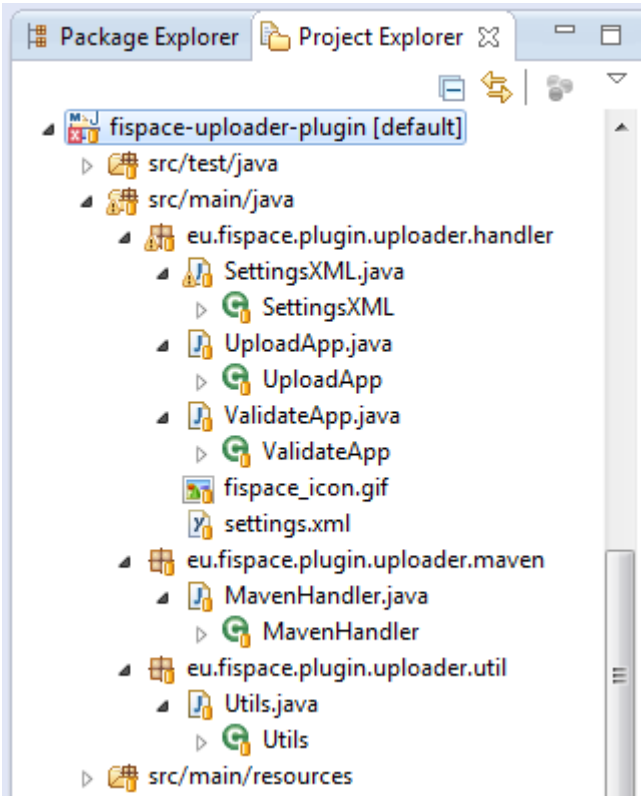


Figure 28: FIspace Uploader plugin project

Package diagram follows:

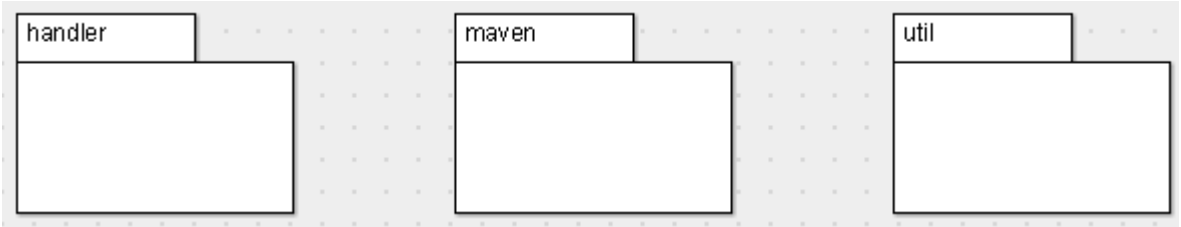


Figure 29: Uploader package diagram

Classes in handler package are :

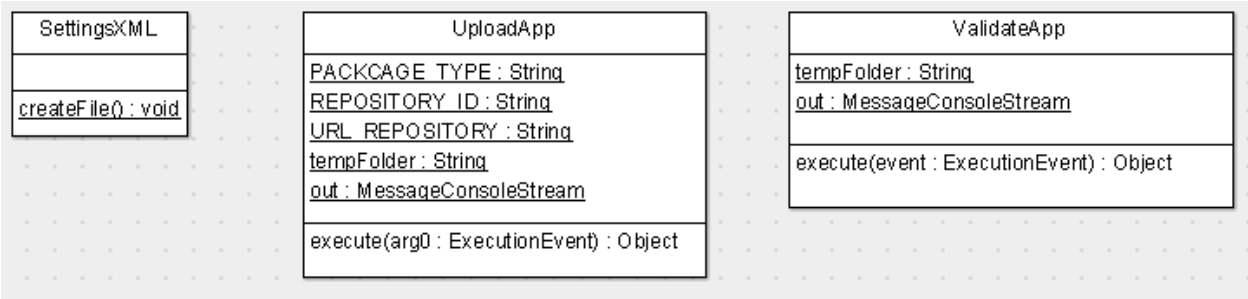
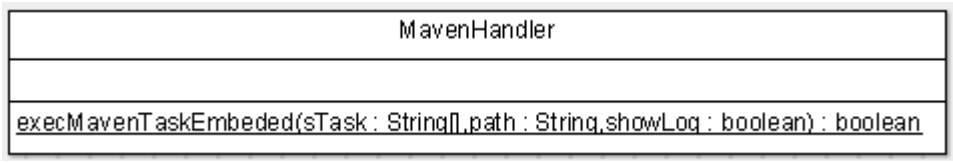
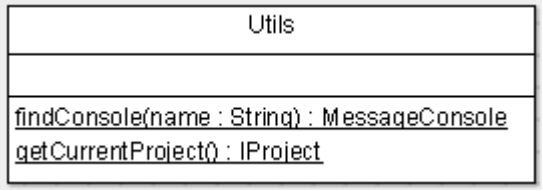


Figure 30: Uploader key classes

Class in maven package:



Class in util package:



The plugin.xml is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.4"?>
<plugin>
   <extension
         point="org.eclipse.ui.menus">

      <menuContribution
            locationURI="popup:org.eclipse.ui.popup.any?after=additions">
         <menu
          label="FIspace"
               id="eu.fispace.plugin.menus.popupMenu">
                     <command
                           commandId="eu.fispace.plugin.validate"
                           label="Generate Widget">
                           <visibleWhen>
                                 <iterate operator="and">
                                  <test property="org.eclipse.core.resources.projectNature"
                                       value="eu.fispace.plugin.wizards.widgetdNature">
                                  </test>
                                 </iterate>
                           </visibleWhen>
                     </command>
                     <!--command
                           commandId="eu.fispace.plugin.upload"
                           label="Upload Application">
                           <visibleWhen>
                                 <iterate operator="and">
                                  <test property="org.eclipse.core.resources.projectNature"
                                       value="eu.fispace.plugin.wizards.widgetdNature">
                                  </test>
```

```
                                        </iterate>
                                </visibleWhen>
                        </command-->
            </menu>
        </menuContribution>
    </extension>
    <extension
          point="org.eclipse.ui.commands">
        <command
              id="eu.fispace.plugin.upload"
              name="Upload">
        </command>
        <command
              id="eu.fispace.plugin.validate"
              name="Validate">
        </command>
    </extension>
     <extension
           point="org.eclipse.ui.handlers">

          <handler
                        class="eu.fispace.plugin.uploader.handler.ValidateApp"
                        commandId="eu.fispace.plugin.validate">
                </handler>
                <handler
                        class="eu.fispace.plugin.uploader.handler.UploadApp"
                        commandId="eu.fispace.plugin.upload">
                </handler>
     </extension>

</plugin>
```

2.3.16 FIspace-wizard plugin

The wizard will provide the creation of FIspace applications.

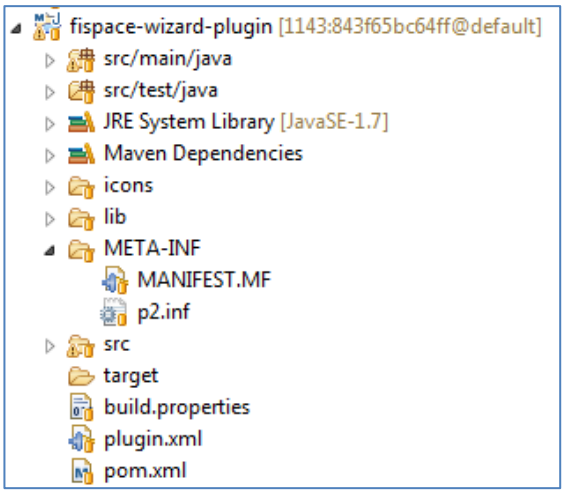


Figure 31: FIspace Wizard plugin project

Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <parent>
                <groupId>eu.fispace.plugin.fispace-wizard</groupId>
                <artifactId>fispace-wizard</artifactId>
                <version>0.8.1</version>
                <relativePath>../pom.xml</relativePath>
        </parent>

        <groupId>eu.fispace.plugin.fispace-wizard-plugin</groupId>
        <artifactId>fispace-wizard-plugin</artifactId>
        <version>1.0.0.qualifier</version>
        <packaging>jar</packaging>

        <name>fispace-wizard-plugin</name>
        <build>

                <plugins>
                        <plugin>
                                <artifactId>maven-compiler-plugin</artifactId>
                                <version>3.1</version>
                                <configuration>
                                        <source>1.7</source>
                                        <target>1.7</target>
                                </configuration>
                        </plugin>
                </plugins>

                <pluginManagement>
                        <plugins>
                                <plugin>
                                        <artifactId>maven-eclipse-plugin</artifactId>
                                        <version>2.9</version>
                                        <configuration>
                                                <pde>true</pde>
                                        </configuration>
                                </plugin>
                                <plugin>
                                        <artifactId>maven-dependency-plugin</artifactId>
                                        <version>2.8</version>
                                </plugin>
                                <plugin>
```

```
                                        <artifactId>maven-clean-plugin</artifactId>
                                        <version>2.3</version>
                                        <configuration>
                                                <filesets>
                                                        <fileset>
                                                                <directory>${basedir}</directory>
                                                                <includes>
                                                                        <include>*.jar</include>
                                                                </includes>
                                                                <followSymlinks>false</followSymlinks>
                                                        </fileset>
                                                </filesets>
                                        </configuration>
                                </plugin>
                        </plugins>
                </pluginManagement>

        </build>

        <dependencies>

                <!-- maven api -->
                <dependency>
                        <groupId>org.apache.maven</groupId>
                        <artifactId>maven-embedder</artifactId>
                        <version>3.1.1</version>
                </dependency>
                <dependency>
                        <groupId>org.eclipse.core</groupId>
                        <artifactId>org.eclipse.core.commands</artifactId>
                        <version>3.6.0</version>
                </dependency>
                <dependency>
                        <groupId>org.eclipse.core</groupId>
                        <artifactId>org.eclipse.core.resources</artifactId>
                        <version>3.7.100</version>
                </dependency>

                <dependency>
                        <groupId>org.eclipse.ui</groupId>
                        <artifactId>console</artifactId>
                        <version>3.2.0-v20070530</version>
                </dependency>
                <dependency>
                        <groupId>org.eclipse.ui</groupId>
                        <artifactId>workbench</artifactId>
                        <version>3.3.0-I20070608-1100</version>
                </dependency>

                <dependency>
                        <groupId>org.eclipse.swt.wpf.win32</groupId>
                        <artifactId>x86</artifactId>
                        <version>3.3.0-v3346</version>
                </dependency>

                <dependency>
                        <groupId>org.apache.maven</groupId>
                        <artifactId>maven-core</artifactId>
                        <version>3.1.1</version>
                </dependency>
                <dependency>
                        <groupId>org.eclipse.equinox</groupId>
                        <artifactId>app</artifactId>
                        <version>1.0.0-v20070606</version>
                </dependency>

                <dependency>
                        <groupId>eu.medsea.mimeutil</groupId>
                        <artifactId>mime-util</artifactId>
                        <version>2.1.3</version>
                </dependency>

                <dependency>
                        <groupId>commons-logging</groupId>
                        <artifactId>commons-logging</artifactId>
                        <version>1.1.3</version>
```

```
                </dependency>


                <dependency>
                        <groupId>org.eclipse.jdt</groupId>
                        <artifactId>core</artifactId>
                        <version>3.2.0.666</version>
                </dependency>

                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <version>4.11</version>
                </dependency>

                <dependency>
                        <groupId>org.apache.commons</groupId>
                        <artifactId>commons-io</artifactId>
                        <version>1.3.2</version>
                </dependency>

                <dependency>
           <groupId>eu.fispace.helper</groupId>
           <artifactId>xml-helper</artifactId>
           <version>1.0</version>
           <scope>system</scope>
           <systemPath>${basedir}/lib/FIspaceUtils.jar</systemPath>
        </dependency>

        </dependencies>

</project>
```

Main dependencies declared:

- Maven Invoker: A component to programmatically invoke Maven.
- Maven Embedded: This is a maven runtime. Maven runtime allows introspection of maven project metadata at runtime.

Key Package Diagram:

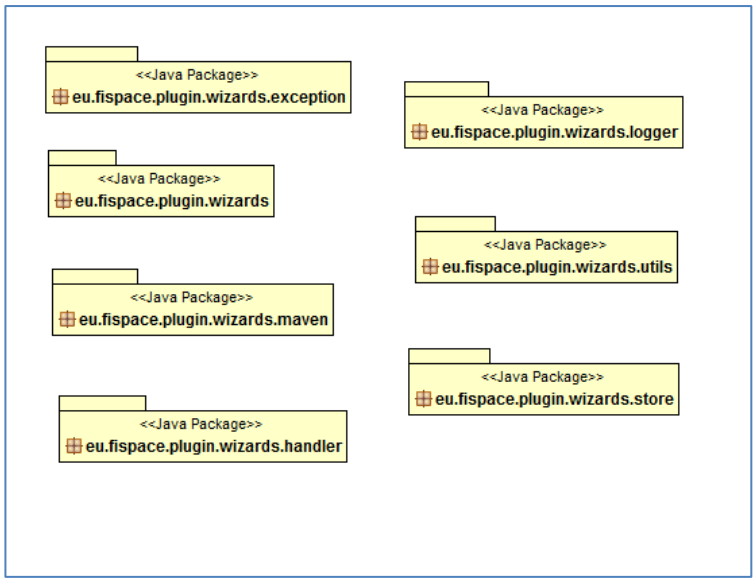


Figure 32: Wizard package diagram

Package eu.fispace.plugin.wizards key classes:

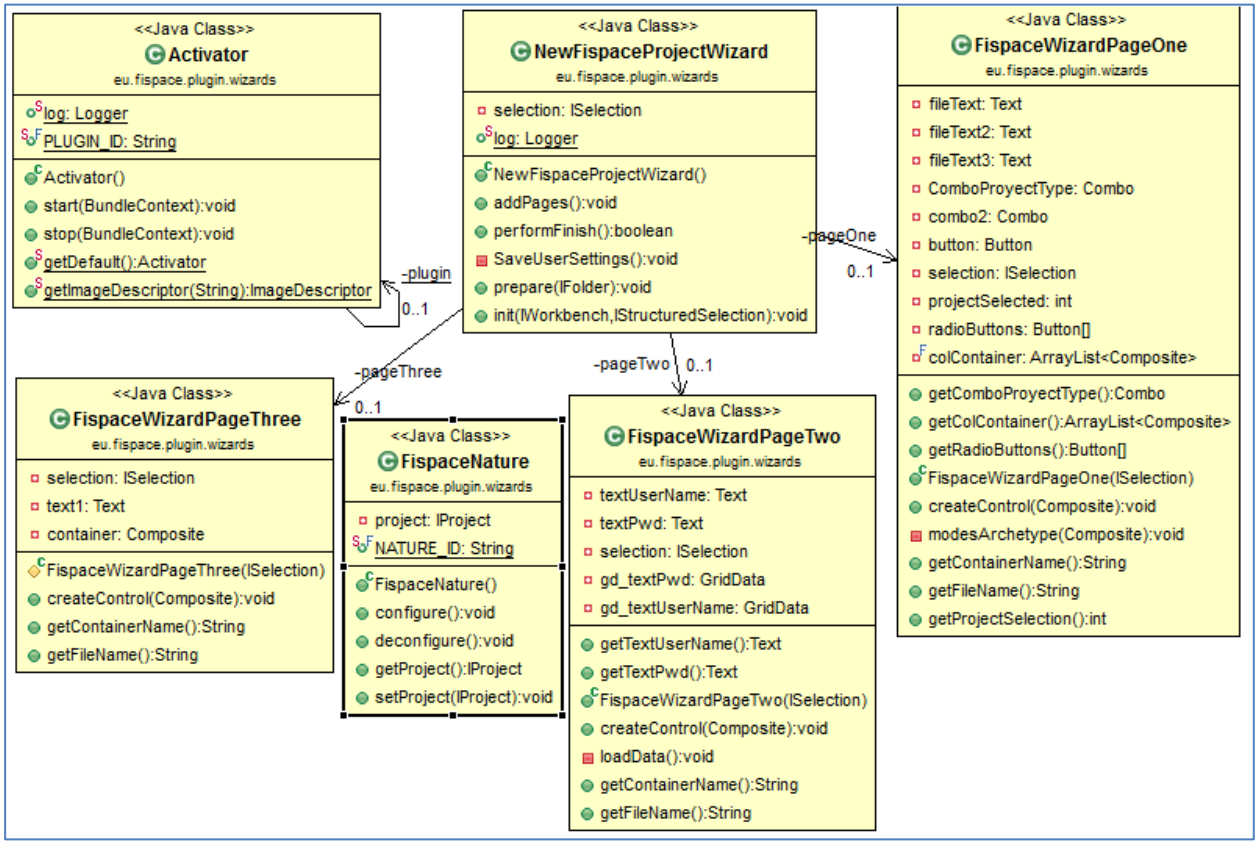
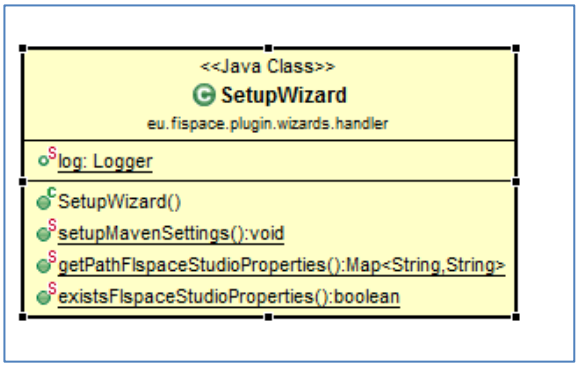


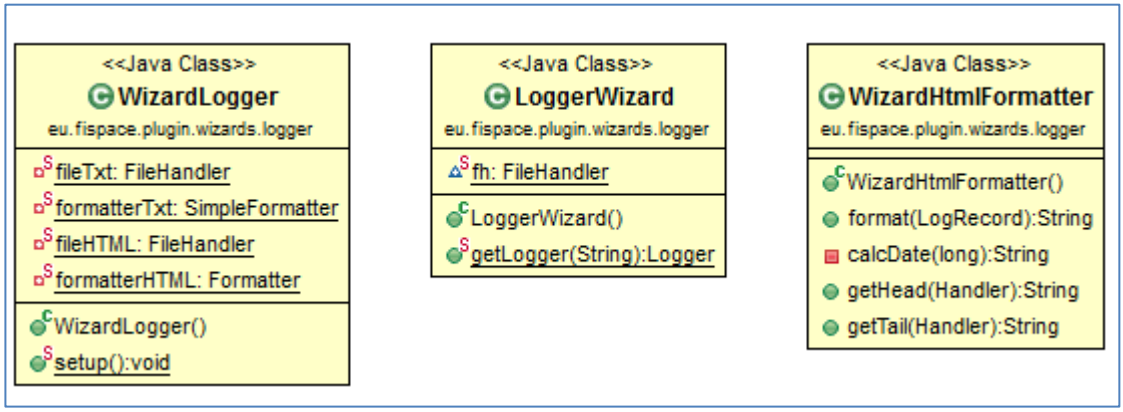
Figure 33: Wizard key classes

Package eu.fispace.plugin.wizards.exception key classes:

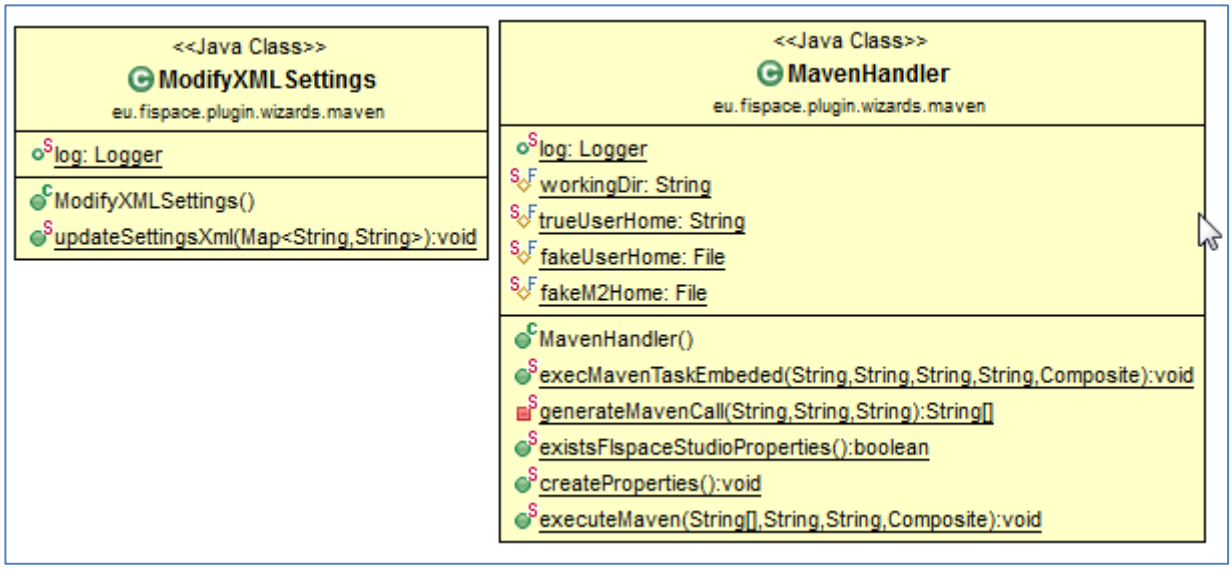
Package eu.fispace.plugin.wizards.hander key classes:



Package eu.fispace.plugin.wizards.logger key classes:

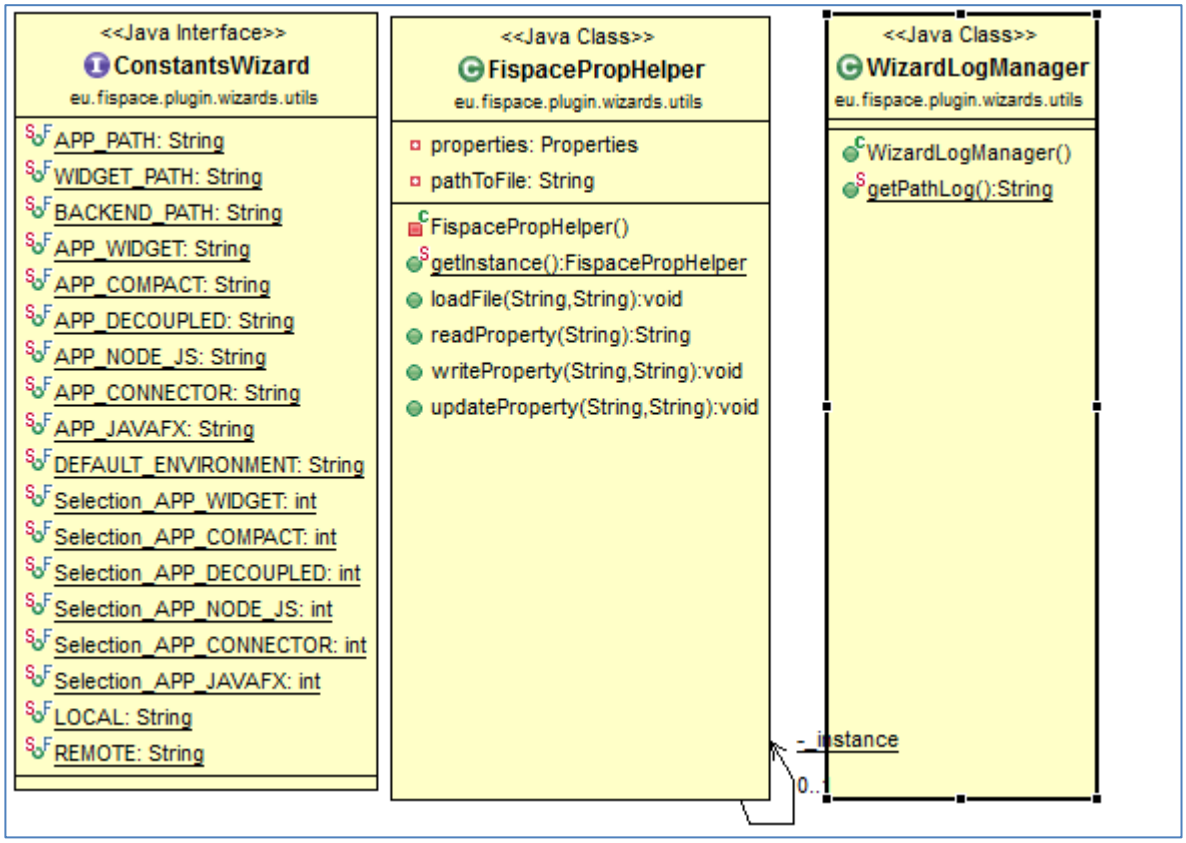


Package eu.fispace.plugin.wizards.maven key classes:



Package eu.fispace.plugin.wizards.store key classes:

Package eu.fispace.plugin.wizards.utils key classes:


<<Java Interface>>
ConstantsWizard
eu.fispace.plugin.wizards.utils
APP_PATH: String
WIDGET_PATH: String
BACKEND_PATH: String
APP_WIDGET: String
APP_COMPACT: String
APP_DECOUPLED: String
APP_NODE_JS: String
APP_CONNECTOR: String
APP_JAVAFX: String
DEFAULT_ENVIRONMENT: String
Selection_APP_WIDGET: int
Selection_APP_COMPACT: int
Selection_APP_DECOUPLED: int
Selection_APP_NODE_JS: int
Selection_APP_CONNECTOR: int
Selection_APP_JAVAFX: int
LOCAL: String
REMOTE: String
<<Java Class>>
FispacePropHelper
eu.fispace.plugin.wizards.utils
properties: Properties
pathToFile: String
FispacePropHelper()
getInstance():FispacePropHelper
loadFile(String,String):void
readProperty(String):String
writeProperty(String,String):void
updateProperty(String,String):void
<<Java Class>>
WizardLogManager
eu.fispace.plugin.wizards.utils
WizardLogManager()
getPathLog():String
- instance
0..1

2.3.16.1 FIspace-archetypes

Archetype is a Maven project toolkit for templates. An archetype is defined as an original pattern or model from which all other things of the same kind are made. The names fit as we are trying to provide a system that provides a consistent means of generating Maven projects. Archetype will help authors create Maven project templates for users, and provides users with the means to generate parameterized versions of those project templates.

Archetypes are located in https://bitbucket.org/fispace/archetypes

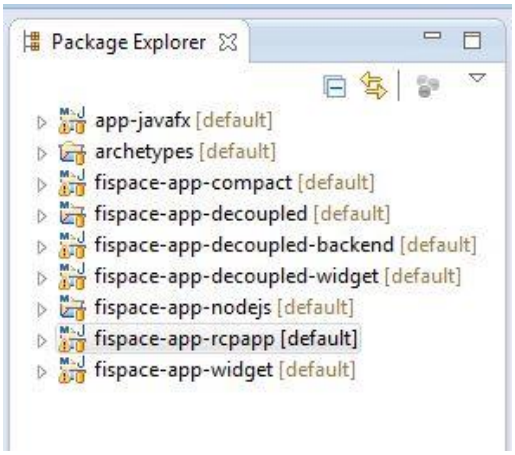


Figure 34: Archetype repository in Eclipse

In FIspace we have defined six archetypes that are described in the next subsections.

2.3.16.2 Widget archetype

This archetype is used to create a widget project.

Widget refers to a small application with limited functionality that can be installed and executed within a web page by an end-user. The Widget archetype enables the development of *WireCloud* projects, which is the platform in which these Widgets can be installed and executed.

This archetype creates a simple project containing a single .html file with a "*Hello World*". It is also necessary to provide a config.xml file containing instructions of how the WireCloud should install and execute the widget. The widget project was designed to work *out of the box*, that is, once the project is created it can be packaged as a Widget and published into *Wirecloud* directly, without needing to configure the index.html and config.xml files.

This archetype has the structure represented in next figure:

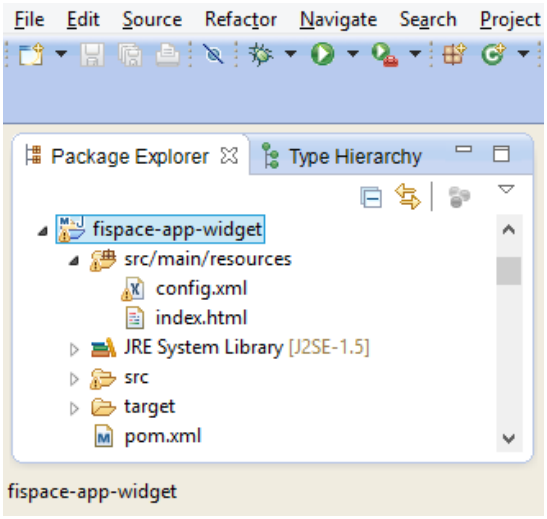


Figure 35: Widget archetype structure

2.3.16.3 Compact archetype

This archetype is used to create a compact project.

The compact archetype also contains a Widget for *WireCloud*, but, unlike the Widget archetype, this one is more complex, with HTML5 features. The HTML template Boilerplate[2] was used to implement this Widget. According to the authors, this template helps you build fast, robust and adaptable web apps or sites.

This archetype has the structure represented in next figure:

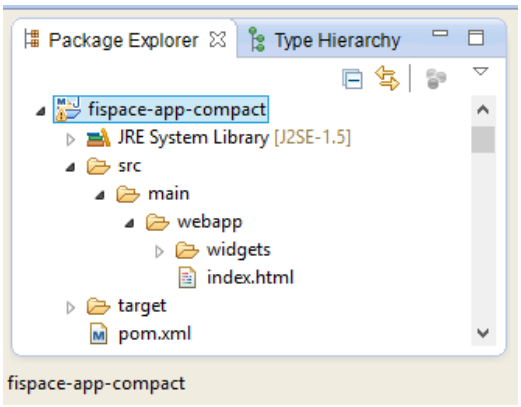


Figure 36: Compact archetype structure

2.3.16.4 Decoupled archetype

This archetype is used to create a decoupled project.

The decoupled archetype consists in two parts, the widget part and the backend part. The widget part is the same as the one provided by the Widget archetype. The backend part provides support to the widget part providing some java classes that can be used to connect both sides.

[2] http://html5boilerplate.com/

So that the backend part can connect to the Widget, the backend must be deployed by using a java-based application container, such as Jetty.

The backend side contains a class called *FIspaceClient.java*, which prepares the backend to connect to FIspace through the SDI, importing the libraries eu.fispace.api. This class provides some examples of requests and responses to FIspace, following the scenarios *GreenHouse advice*, and *ShipmentInfo*, form the domains *Agriculture* and *Retail* respectively.

The *KeycloakConnector.java* implements the OAuth client [32] to connect to FIspace SPT module, in order to make use of the SDI REST services.

A manual test is also included, just to check the Keycloak connection.

This archetype has the structure represented in next figure.

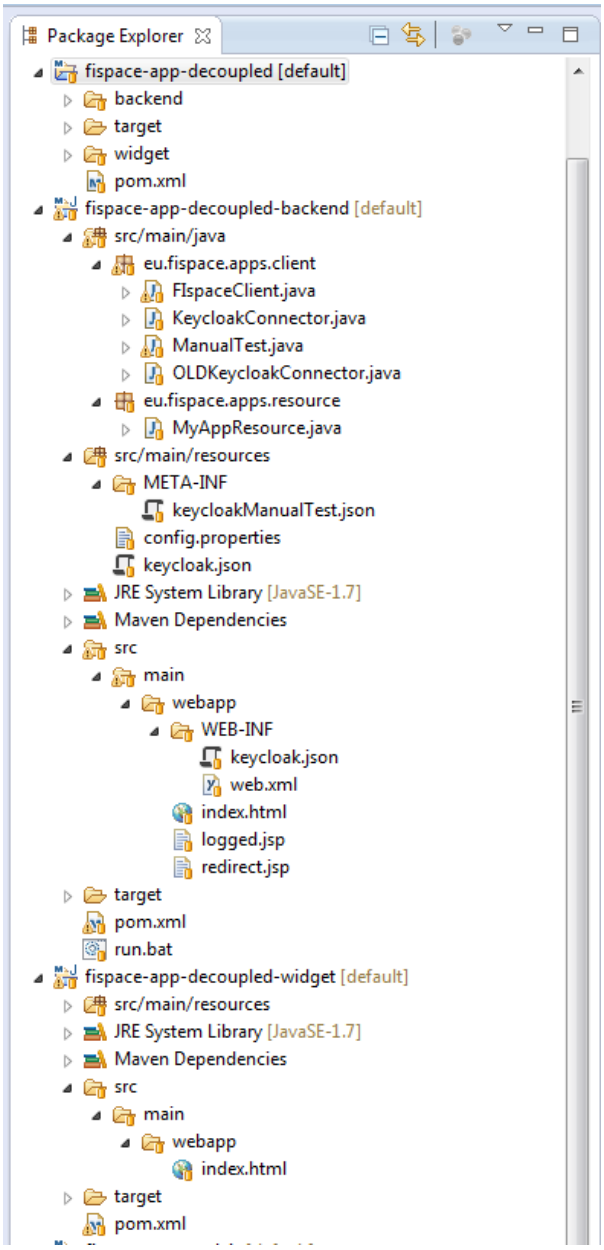


Figure 37: Decoupled archetype structure

2.3.16.5 Node.js archetype

This archetype is used to create a node.js project.

The Node.js archetype represents a basic Node.js project, which helps users to start coding. Node.js is a runtime environment and a library for running applications written in JavaScript outside the browser (for example, on the server). Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

An already created project has been used for that, the *Product Information App*, used in the Tailored Information for Consumers pilot.

The structure of the project is compatible to Node.js projects, and also compatible to express, a web application framework for node, widely used.

This archetype has the structure represented in next figure:

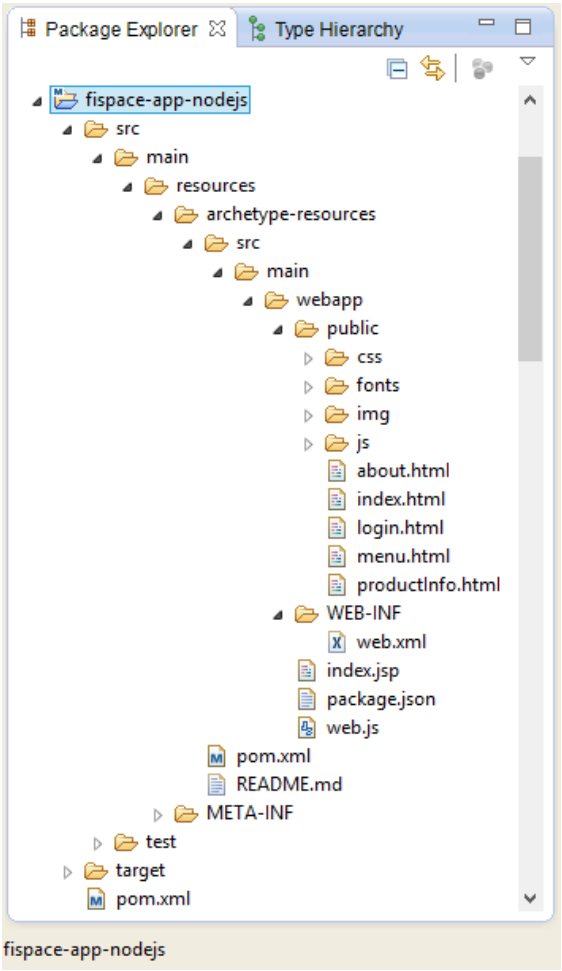


Figure 38: Node.js archetype structure

2.3.16.6 JavaFX archetype

This archetype is used to create a JavaFX project.

JavaFX is a rich client platform for enterprise business applications. It reuses Java libraries and allows developers to access native system capabilities. The JavaFX archetype provides a basic JavaFX application to interconnect to FIspace capabilities through the SDI. To do that it uses the *NotificationMessage* class to send a basic message when a button is pressed in the Java GUI, as can be seen in Figure 40.

This archetype has the structure represented in next figure:

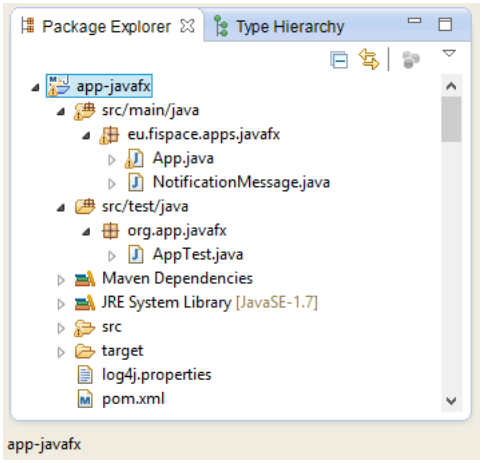


Figure 39: JavaFX archetype structure

Once the project is launched as an “Eclipse Application”, the application “JAVAFX Archetype” is displayed:

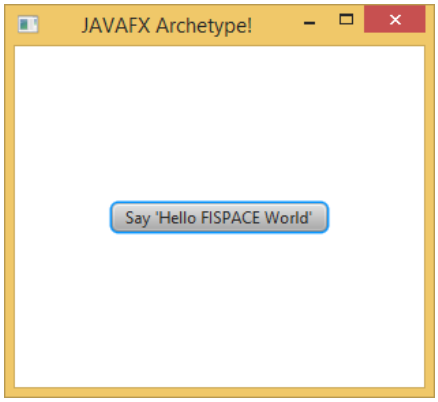


Figure 40: JavaFX archetype GUI

2.3.16.7 RCP app archetype

This archetype is used to create a RCP Project [33].

RCP (*Rich Client Platform*) constitutes the minimal set of plug-ins needed to build a rich client application (platform application with a UI). The RCP project enables Eclipse to be used in a wide range of end-user applications that are not Integrated Development Environments (IDEs).

This archetype has the structure represented in next figure:

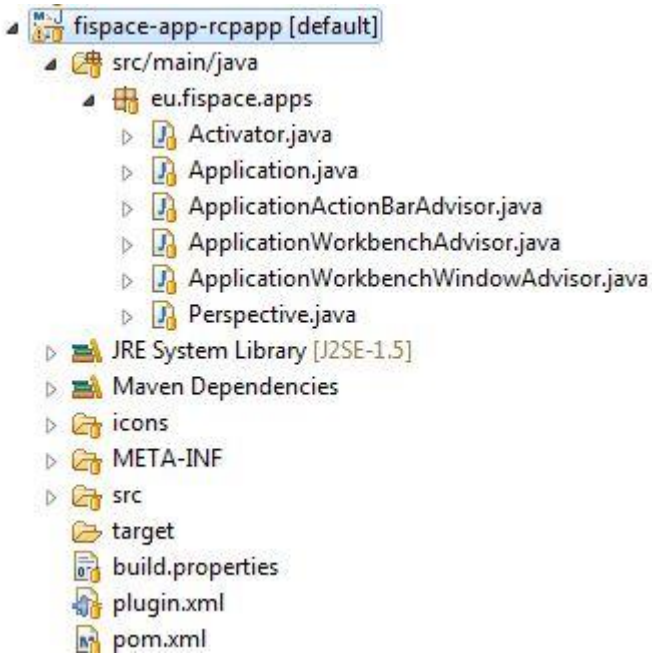


Figure 41: RCP app archetype structure

Once the project is launched as an “Eclipse Application”, the application “Hello RCP” is displayed:

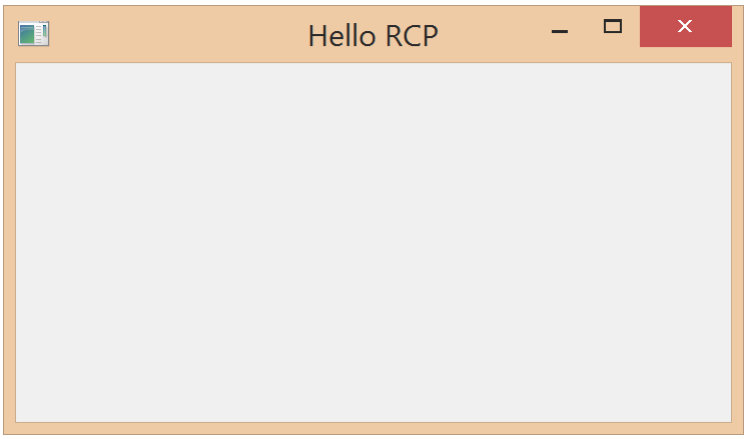


Figure 42: RCP archetype GUI

2.3.17 FIspace-xmleditor plugin

FIspace XML editor provides structured form-based UI for viewing and editing the Wire-cloud xml file (config.xml):

- XML editor
- Form-based UI
- Dependency Hierarchy viewer

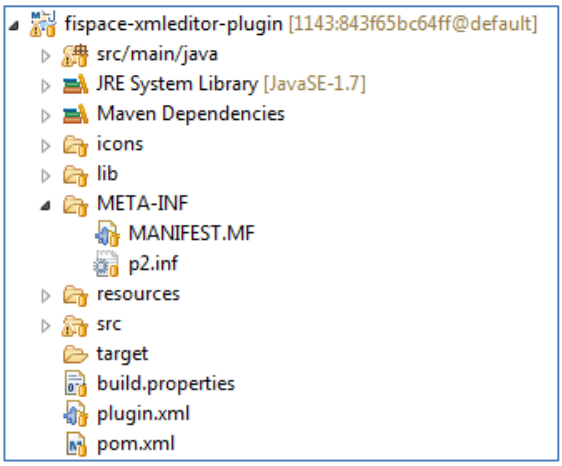


Figure 43: FIspace XML editor plugin project

Pom.xml

fispace-xmleditor/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"
    xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <modelVersion>4.0.0</modelVersion>

    <groupId>eu.fispace.plugin.fispace-xmleditor</groupId>
    <artifactId>fispace-xmleditor</artifactId>
    <version>0.8.1</version>
    <name>FIspace-XmlEditor-Plugin-Parent</name>
    <packaging>pom</packaging>
    <modules>
        <module>fispace-xmleditor-plugin</module>
        <module>fispace-xmleditor-feature</module>
        <module>fispace-xmleditor-updatesite</module>
    </modules>
    <build>
        <plugins>
            <plugin>
                <groupId>com.googlecode</groupId>
                <artifactId>maven-idea-plugin</artifactId>
                <version>1.6.1</version>
            </plugin>
        </plugins>
    </build>
    <parent>
        <groupId>eu.fispace.plugin</groupId>
        <relativePath>../pom.xml</relativePath>
        <version>0.8.1</version>
        <artifactId>plugin-parent</artifactId>
    </parent>
</project>
```

Figure 44: FIspace-xmleditor pom file

Key Package Diagram:

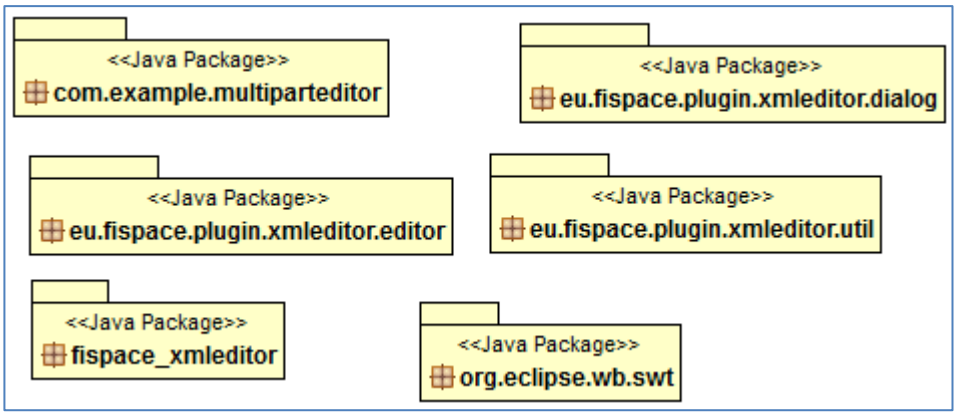


Figure 45: XML editor package diagram

2.3.18 FIspace-logging-feature plugin

The FIspace logging plugin enables users to store and classify the logs generated by FIspace apps in order to make easy the app development process.

The FIspace.logging plugin has the structure defined in the following figure:

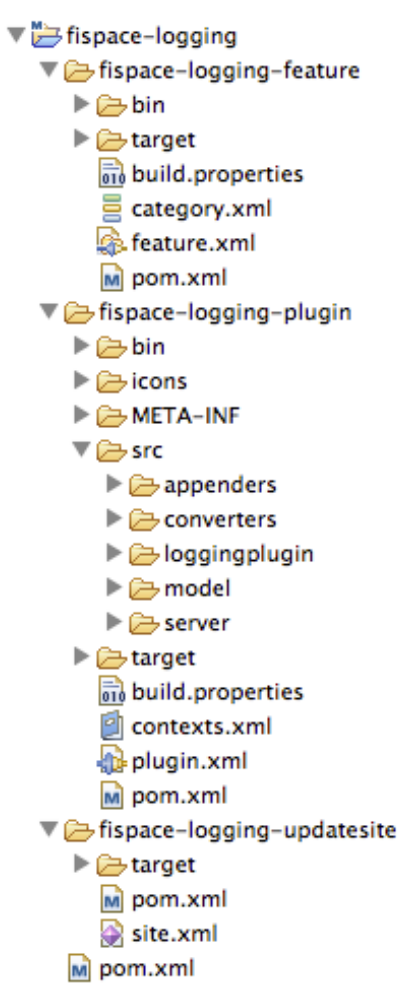


Figure 46: FIspace-logging plugin structure

Project modules are defined in the POM file:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>eu.fispace.plugin.fispace-logging</groupId>
  <artifactId>fispace-logging</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>pom</packaging>
  <name>FIspace-Logging_Plugin-Parent</name>
  <modules>
        <module>fispace-logging-feature</module>
        <module>fispace-logging-plugin</module>
        <module>fispace-logging-updatesite</module>
  </modules>
    <repositories>
                <repository>
                        <id>eclipse keppler</id>
```

```
                        <layout>p2</layout>
                        <url>http://download.eclipse.org/releases/kepler</url>
                </repository>
        </repositories>
        <pluginRepositories>
                <pluginRepository>
                        <id>maven-idea-plugin-repo</id>
                        <url>http://maven-idea-plugin.googlecode.com/svn/maven-repo</url>
                </pluginRepository>
        </pluginRepositories>
</project>
```

Plugin.xml of the Logging plugin is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.4"?>
<plugin>
   <extension
         point="org.eclipse.ui.views">
      <category
            name="FIspace"
            id="fispace-tool">
      </category>
      <view
            name="FIspace Logging"
            icon="icons/fispace_icon.png"
            category="fispace-tool"
            class="loggingplugin.views.LoggingView"
            id="loggingplugin.views.LoggingView">
      </view>
   </extension>
   <extension
         point="org.eclipse.ui.perspectiveExtensions">
      <perspectiveExtension
            targetID="org.eclipse.jdt.ui.JavaPerspective">
         <view
               ratio="0.5"
               relative="org.eclipse.ui.views.ProblemView"
               relationship="right"
               id="loggingplugin.views.LoggingView">
         </view>
      </perspectiveExtension>
   </extension>
   <extension
         point="org.eclipse.help.contexts">
      <contexts
            file="contexts.xml">
      </contexts>
   </extension>
</plugin>
```

Main dependencies declared:

- org.eclipse.core.commands: Application programming interfaces for commands and handlers.
- org.eclipse.core.resources: Provides basic support for managing a workspace and its resources.

Design decisions:

1. Preferences functionality

We have three main files:

- loggingplugin.preferences.**LoggingPreferencePage**
- loggingplugin.preferences.**PreferenceInitializer**
- loggingplugin.preferences.**PreferenceConstants**

The LoggingPreferencePage configures Preferences Views, labels, text fields, etc. The following figure provides a summary of this class.

```
public class LoggingPreferencePage
        extends FieldEditorPreferencePage
        implements IWorkbenchPreferencePage {

        public LoggingPreferencePage() {
                super(GRID);
                setPreferenceStore(Activator.getDefault().getPreferenceStore());
                setDescription("Edit preferences for FIspace logging");
        }
        /**
         * Creates the field editors. Field editors are abstractions of
         * the common GUI blocks needed to manipulate various types
         * of preferences. Each field editor knows how to save and
         * restore itself.
         */
        public void createFieldEditors() {
        addField(new DirectoryFieldEditor(PreferenceConstants.P_LOGS_FILE_PATH,
                                "&Directory preference:", getFieldEditorParent()));

        addField(new StringFieldEditor(PreferenceConstants.P_USER_NAME, "FIspace
Username:", getFieldEditorParent()));

        addField(new StringFieldEditor(PreferenceConstants.P_USER_PASSWORD, "FIspace
Password:", getFieldEditorParent()));

        addField(new StringFieldEditor(PreferenceConstants.P_SERVER_URL, "FIspace Server
URL:", getFieldEditorParent()));

        }

        public void init(IWorkbench workbench) {
        }

}
```

The preference initializer provides a way to set default preference values. The following Figure provides a summary of this class.

```
public class PreferenceInitializer extends AbstractPreferenceInitializer {

        public void initializeDefaultPreferences() {
                IPreferenceStore store = Activator.getDefault().getPreferenceStore();
                store.setDefault(PreferenceConstants.P_USER_NAME,
                                "Default Test");
                store.setDefault(PreferenceConstants.P_USER_PASSWORD,
                                "");
                store.setDefault(PreferenceConstants. P_LOGS_FILE_PATH,
                                System.getProperty("user.home")+"/fispace-logs");
        }
}
```

The class PreferenceConstants provides access to preferences values

```
public class PreferenceConstants {
	public static final String P_LOGS_FILE_PATH = "logFilePreference";
	public static final String P_USER_NAME = "userNamePreference";
	public static final String P_USER_PASSWORD = "userPasswordPreference";
	public static final String P_SERVER_URL = "urlPreference";
}
```

If we want to access to plugin preferences from Java Classes we should do like this:

```
IPreferenceStore preferenceStore = Activator.getDefault().getPreferenceStore();

final String logsPath = preferenceStore.getString("logFilePreference");
```

2. View functionality

FIspace Logging Plugin let users, among others, monitor logs from different fispace projects in a single view (Java Perspective).

The involved class is loggingplugin.views.***LoggingView***:

```
public class LoggingView extends ViewPart {

public LoggingView() {
	}

	/**
	 * This is a callback that will allow us
	 * to create the viewer and initialize it.
	 */
public void createPartControl(Composite parent) {

	viewer = new TableViewer(parent, SWT.MULTI | SWT.H_SCROLL | SWT.V_SCROLL);

	createColumns(parent, viewer);
	final Table table = viewer.getTable();
	table.setHeaderVisible(true);
	table.setLinesVisible(true);

	LogFileReader logReader=LogFileReader.getInstance();
	logReader.startMonitor();

	viewer.setContentProvider(ArrayContentProvider.getInstance());
	 viewer.setInput(ModelProvider.INSTANCE.getLogs());

		// Make the selection available to other Views
	getSite().setSelectionProvider(viewer);

	}

	// Used to update the viewer from outsite
	public void refresh() {
		viewer.refresh();
	}

	public static TableViewer getViewer() {
		return viewer;
	}


	// colums for the table
	private void createColumns(final Composite parent, final TableViewer viewer2) {


		String[] titles = {"Date", "Source", "Mesage Level", "Message" };
		int[] bounds = { 150, 275, 100, 300 };

	TableViewerColumn col = createTableViewerColumn(titles[0], bounds[0], 0);
```

```
            col.setLabelProvider(new ColumnLabelProvider() {
              @Override
              public String getText(Object element) {
                Log log = (Log) element;
                return log.getLogDate().toString();
              }
            });
            // ... Same code for the other three colums ...
        }
        }
```

The Logging View class is responsible to create the viewer and initialize it. To do that it follows these steps:

1. Set the view as a TableViewer
2. Call **createColumns** to populate table with content
   a. Format Logs from content provider to columns text
3. Initialize LogFileReader to start monitor logs
4. Set content provider
   a. When plugin is loaded and each time we call viewer.refresh, tableViewer will remove existing content and will load new content from ModelProvider (Array of Logs obtained from Logs file)

Next figure describes the workflow of how the plugin works.

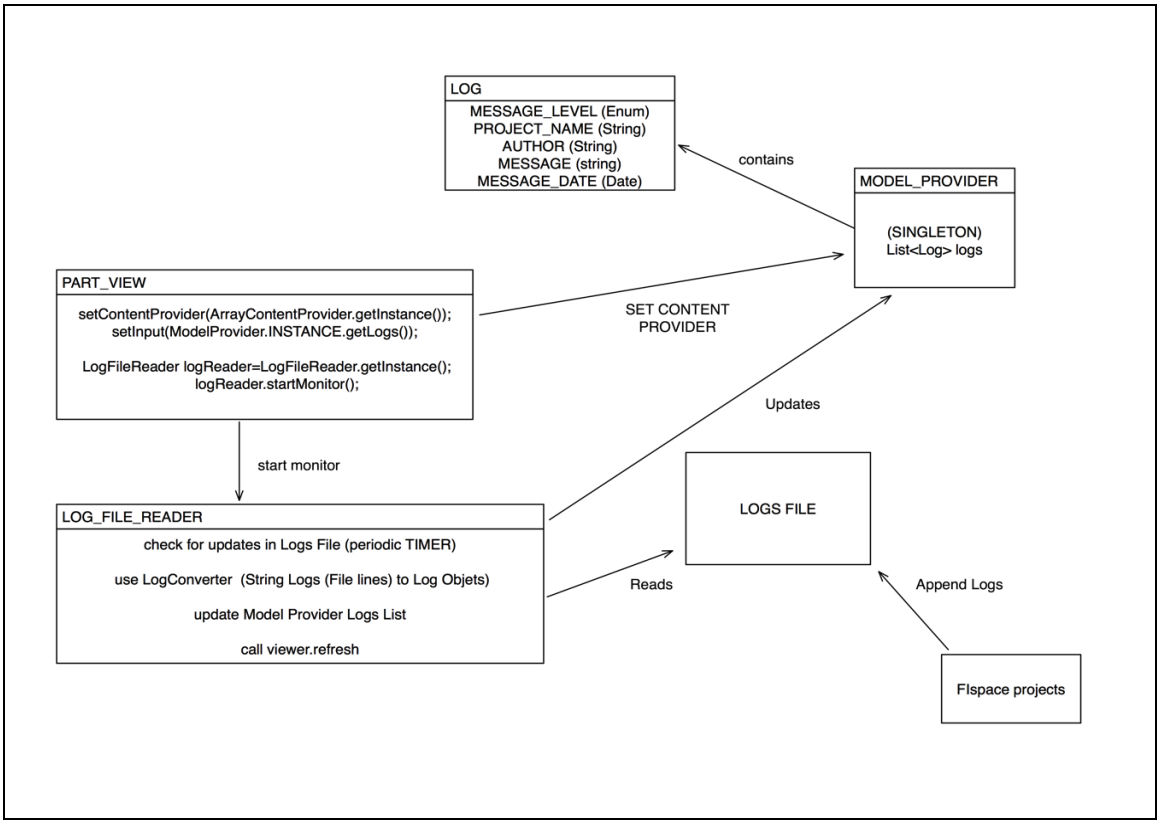


Figure 47: Logging plugin design diagram

5. Key Package Diagram:

In the next figure, key packages are shown:

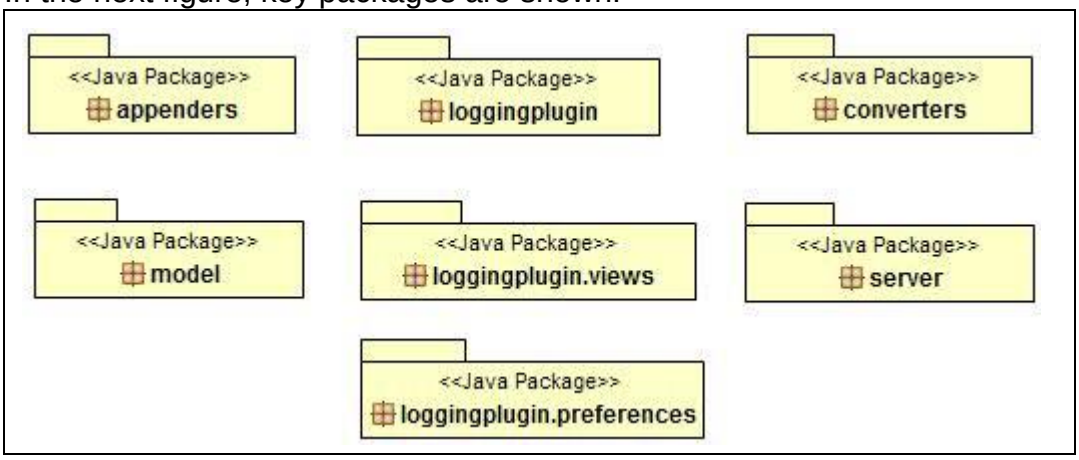


Figure 48: Logging plugin package diagram

The “loggingplugin” class along “views” and “preferences”, are the main plugin packages, where the Activator.java and viewers are declared.
The “server” package includes the logic of the server connection and log sender.
The remaining packages involve helper functions and model interface to log generation.

Logging server

In order to keep the generated logs and be able to review them, a log server will be deployed. The logging plugin will send generated logs in real time to the server for visualization.

Server interface

The communication between these two components will be performed with a REST interface. A POST request will be initiated in the plugin and handled server side in order to store the received log in the DB.
The content is sent in the body of the request, including all the parameters generated:

- Username:
  - Parameter: user
  - Type: String
- Date generated:
  - Parameter: date
  - Type: Long (millis format)
- Project name:
  - Parameter: project
  - Type: String
- Warning level:
  - Parameter: level
  - Type: String
- Log message:
  - Parameter: message
  - Type: String

An example body content of a POST request would be like follows:

```
Us-
er="+logginguser+"&date="+messageDateString+"&project="+messageSource+"&level="+messageLevel+"&message="+
message
```

Server design

Server is developed in nodeJs, using a MVC domain and Express framework with the tree files that are shown in the next figure.

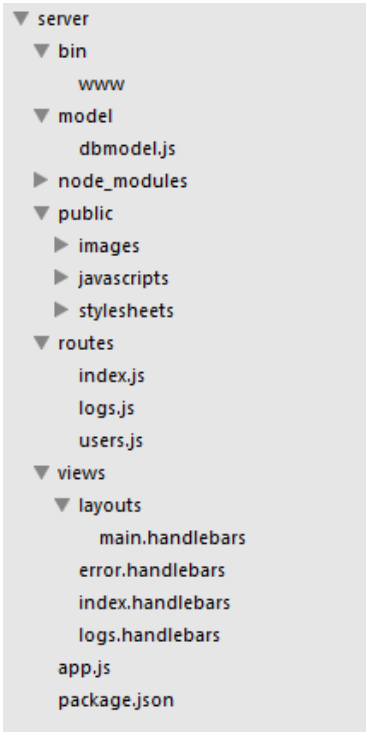


Figure 49: Logging plugin design diagram

MongoDB is used as the main DB, storing logs and users. Two collections will be created with the next schema:

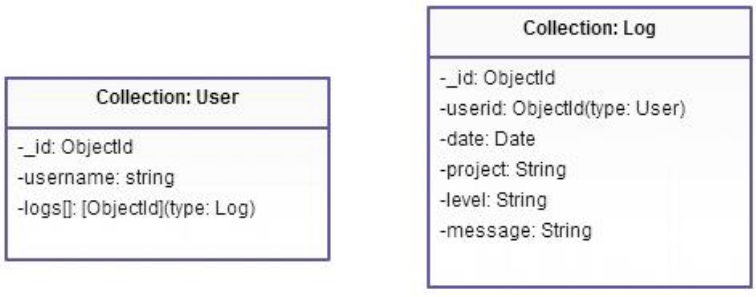


Figure 50: Logging plugin design diagram

The application will handle web request for displaying purposes and also the POST request to insert a new log in the DB.

To prevent duplicated users in the BD, a unique is assign to each user, creating a new entry if the username is not found in the DB. MongoDB [34] has to be configured in or-

der to prevent duplicates in the User collection by adding a unique index to the username field like follows:

```
db.User.ensureIndex( { “username”: 1 }, { unique: true } )
```

Express framework allows adding more functionalities as required.

An actual deployment of the server can be found at http://gisai.dit.upm.es:3004/, where the REST service can be called at http://gisai.dit.upm.es:3004/api/log.

2.3.19 FIspace pluginX feature

An Eclipse feature project contains *features*. A feature describes a list of plug-ins and other features which can be understood as a logical unit. It also has a name, version number and license information assigned to it.

A feature is described via a *feature.xml* file. For example the file might look like the following in a text editor.

The grouping of plug-ins into logical units makes it easier to handle a set of plug-ins. Instead of adding a large number of individual plug-ins to your product configuration file you can group them using features. That increases the visibility of your application structure.

Features can be used by the Eclipse update manager, the build process and optionally for the definition of Eclipse products. Features can also be used as the basis for a launch configuration.

Features are needed to deploy the app via FIspace update site (see 2.3.20).

```
<?xml version="1.0" encoding="UTF-8"?>
<feature
      id="fispace_logging_feature"
      label="Fispace_logging_feature"
      version="1.0.0.qualifier"
      provider-name="FIspace">

   <description url="http://www.example.com/description">
      [Enter Feature Description here.]
   </description>
   <copyright url="http://www.example.com/copyright">
      [Enter Copyright Description here.]
   </copyright>
   <license url="http://www.example.com/license">
      [Enter License Description here.]
   </license>
         <requires>
      <import plugin="org.eclipse.ui"/>
      <import plugin="org.eclipse.core.runtime"/>

   </requires>
   <plugin
         id="fispace-logging-plugin" #plugin
         download-size="0"
         install-size="0"
         version="1.0.0.qualifier"
         unpack="false"/>

</feature>
```

fispace-logging-feature master feature.xml file

2.3.20 FIspace pluginX update-site

FIspace SDK plugins have their personal update-site as a reference point for installation and update of the corresponding plugin.

The update-site is responsible to package and make available on the web the feature of a plugin.

The update-site project of a FIspace plugin looks like this:

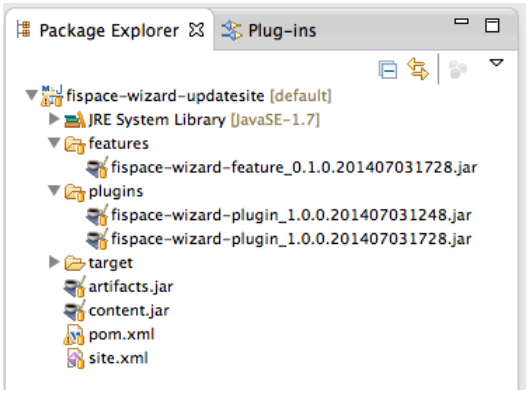


Figure 51: "FIspace project Wizard" update-site project structure

Update-site of a plugin can be easily updated into the latest version of the plugin through the GUI part of its site.xml configuration file.

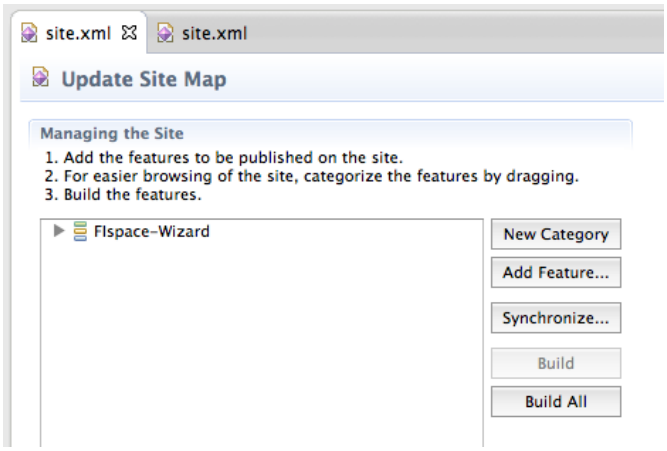


Figure 52: Graphic view of site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<site>
   <description name="FIspace wizard update site">
      A FIspace IDE plugin
   </description>
   <feature url="features/fispace-wizard-feature_0.1.0.201407031728.jar" id="fispace-wizard-feature" ver-
sion="0.1.0.201407031728">
      <category name="FIspace-Wizard"/>
   </feature>
   <category-def name="FIspace-Wizard" label="FIspace-Wizard"/>
</site>
```

FIspace project wizard site.xml

2.3.21 FIspace general update-site Tool

General update-site is used to package all of the existed plugins under one reference point. FIspace developers are able to install the whole FIspace SDK in their eclipse IDE through this update-site.

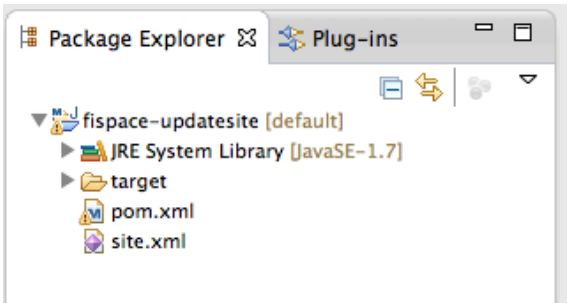


Figure 53: General update site project structure

2.3.22 FIspace Studio Tool

FIspace Studio is a binary distribution providing all of FIspace SDK plugins. Specifically it is a customised eclipse IDE designed and preconfigured for the specific needs of FIspace SDK.

Its creation is based on eclipse Plug-in development and eclipse RCP 3. Through the eclipse RCP 4 [33] compatibility layer the platform code can be easily migrated into eclipse RCP 4.

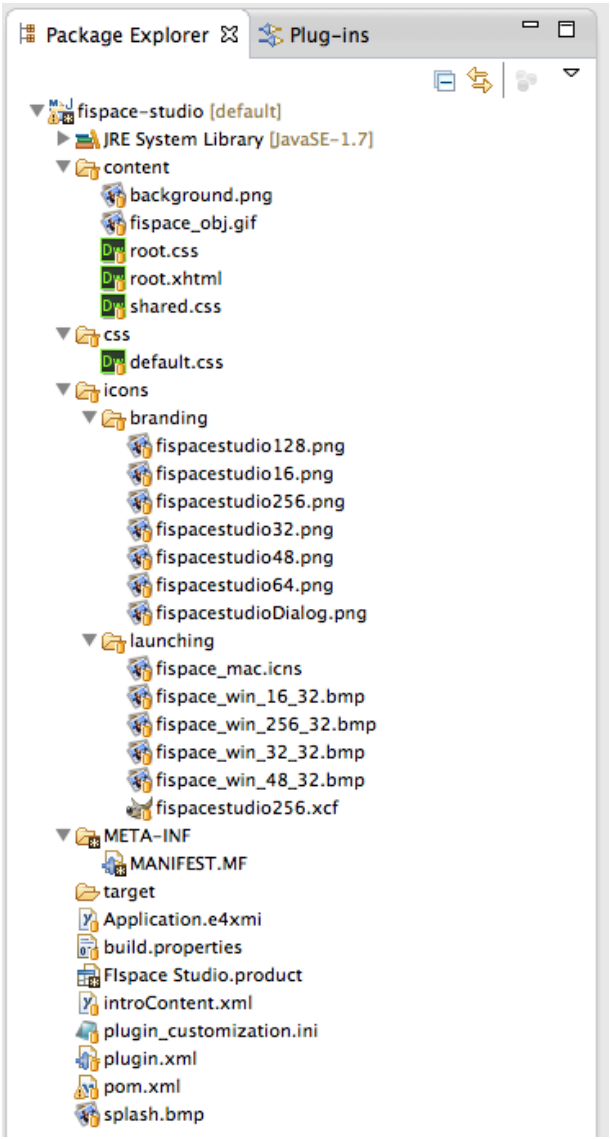


Figure 54: FIspace Studio project structure

The visual customisation of Studio includes cross platform launching icons, branding images, splash screen, welcome page and about dialog.

All of the customised visual parts can be easily configured and updated through the FIspaceStudio.product configuration file.

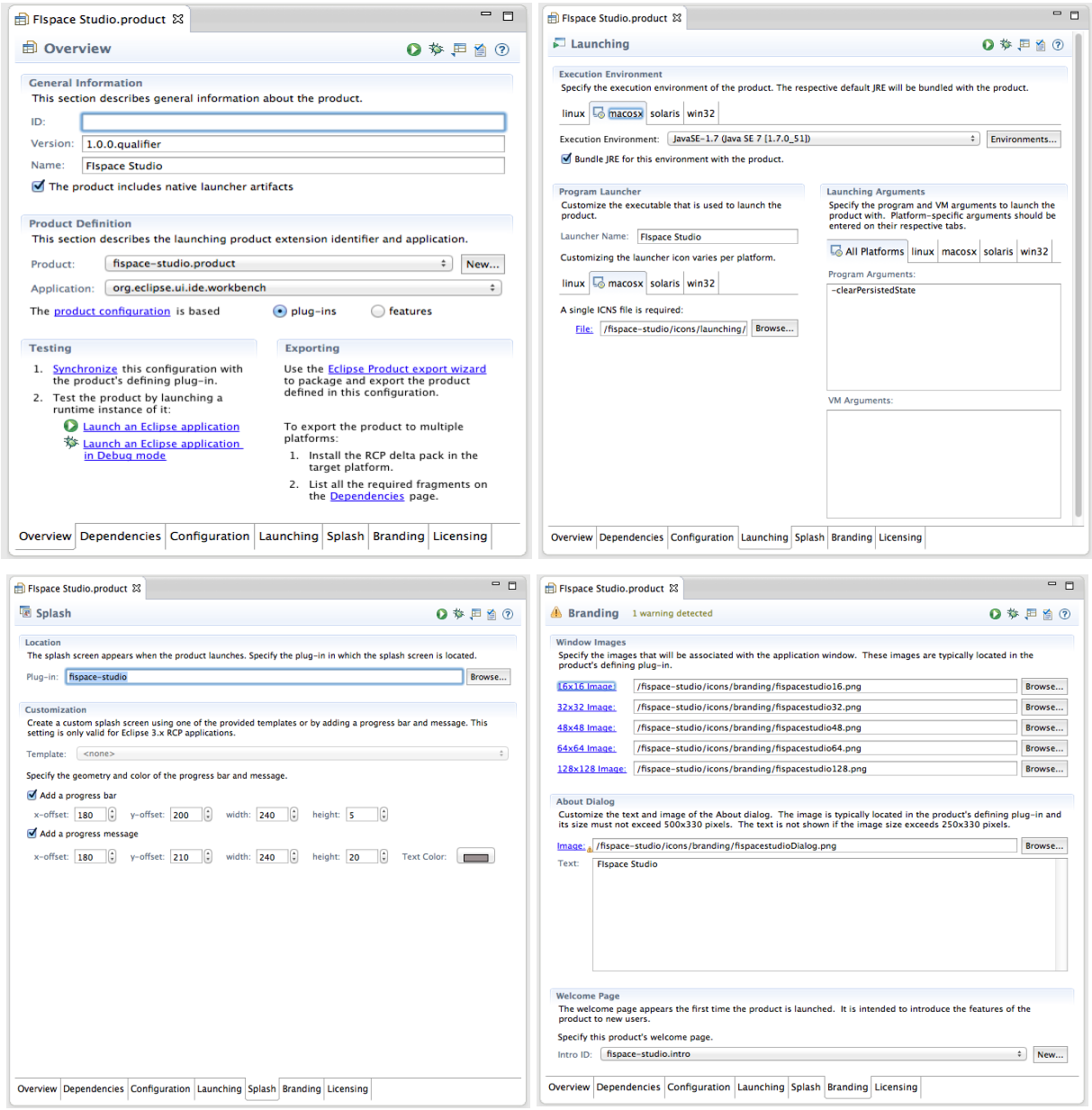


Figure 55: FIspace Studio product configuration

The plugin.xml file of FIspace Studio is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>


   <extension
         id="product"
         point="org.eclipse.core.runtime.products">
      <product
            application="org.eclipse.ui.ide.workbench"
            name="FIspace Studio">

         <property
               name="preferenceCustomization"
               value="plugin_customization.ini">
         </property>
         <property
               name="appName"
               value="FIspace Studio">
         </property>
         <property
               name="aboutImage"
               value="icons/branding/fispacestudioDialog.png">
         </property>
         <property
               name="aboutText"
               value="FIspace Studio">
         </property>
         <property
               name="windowImages"
               val-
ue="icons/branding/fispacestudio16.png,icons/branding/fispacestudio32.png,icons/branding/fispacestudio48.p
ng,icons/branding/fispacestudio64.png,icons/branding/fispacestudio128.png">
         </property>
         <property
               name="startupForegroundColor"
               value="9A9091">
         </property>
         <property
               name="startupMessageRect"
               value="180,210,240,20">
         </property>
         <property
               name="startupProgressRect"
               value="180,200,240,5">
         </property>
      </product>
   </extension>



      <extension
         point="org.eclipse.ui.views">
      <view
            name="View"
            class="org.eclipse.ui.navigator.CommonNavigator"
            id="example.view">
      </view>
   </extension>
      <extension
            point="org.eclipse.ui.intro">
         <intro
               class="org.eclipse.ui.intro.config.CustomizableIntroPart"
               id="fispace-studio.intro">
         </intro>
         <introProductBinding
               introId="fispace-studio.intro"
               productId="fispace-studio.product">
         </introProductBinding>
      </extension>
      <extension
            point="org.eclipse.ui.intro.config">
         <config
               content="introContent.xml"
```

```
               id="fispace-studio.introConfigId"
               introId="fispace-studio.intro">
            <presentation
                  home-page-id="root">
               <implementation
                     kind="html"
                     os="win32,linux,macosx"
                     style="content/shared.css">
               </implementation>
            </presentation>
         </config>
      </extension>
      <extension
            id="id1"
            point="org.eclipse.core.runtime.applications">
         <application
               cardinality="singleton-global"
               thread="main"
               visible="true">
         </application>
      </extension>
      <extension
            id="product1"
            point="org.eclipse.core.runtime.products">
         <product
               application="org.eclipse.ui.ide.workbench"
               name="FIspace Studio">
            <property
                  name="windowImages"
                  val-
ue="icons/branding/fispacestudio16.png,icons/branding/fispacestudio32.png,icons/branding/fispacestudio48.p
ng,icons/branding/fispacestudio64.png,icons/branding/fispacestudio128.png">
            </property>
            <property
                  name="aboutText"
                  value="FIspace Studio">
            </property>
            <property
                  name="aboutImage"
                  value="icons/branding/fispacestudioDialog.png">
            </property>
            <property
                  name="appName"
                  value="FIspace Studio">
            </property>
         </product>
      </extension>


</plugin>
```

2.3.23 Libraries to connect FIspace platform

There was a necessity for making easy the connection among FIspace platform and external applications. We are trying to provide a basic development kit for the most relevant programming languages.

A *Java* client is currently implemented and used, for example, to connect the decoupled-backend project to FIspace.

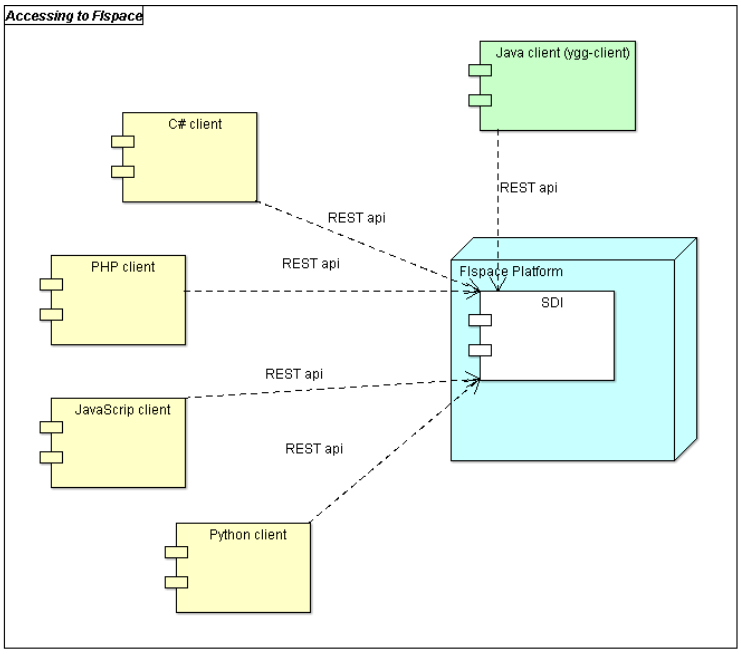


Figure 56: Access to FIspace Platform

Each client (or connector) consists of a set of functions that invoke the **REST services** provided by FIspace API (https://bitbucket.org/fispace/core/wiki/capabilities/api), and an **OAuth** client for implementing the authentication process necessary to access FIspace platform. A test program will be also provided:

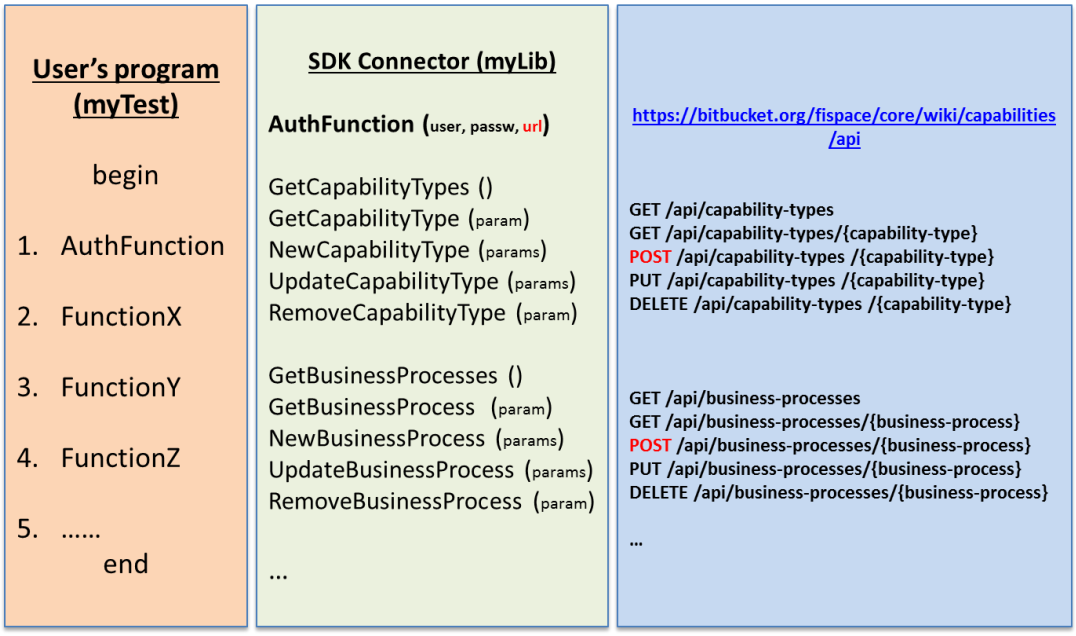


Figure 57: Library to access FIspace Platform

There are several considerations to be taken in account to use those libraries.

First, since OAuth protocol is used, your application (in this case the test just provided with each library) has to be registered like an **OAuth client** in the FIspace SPT component (FIspace realm).

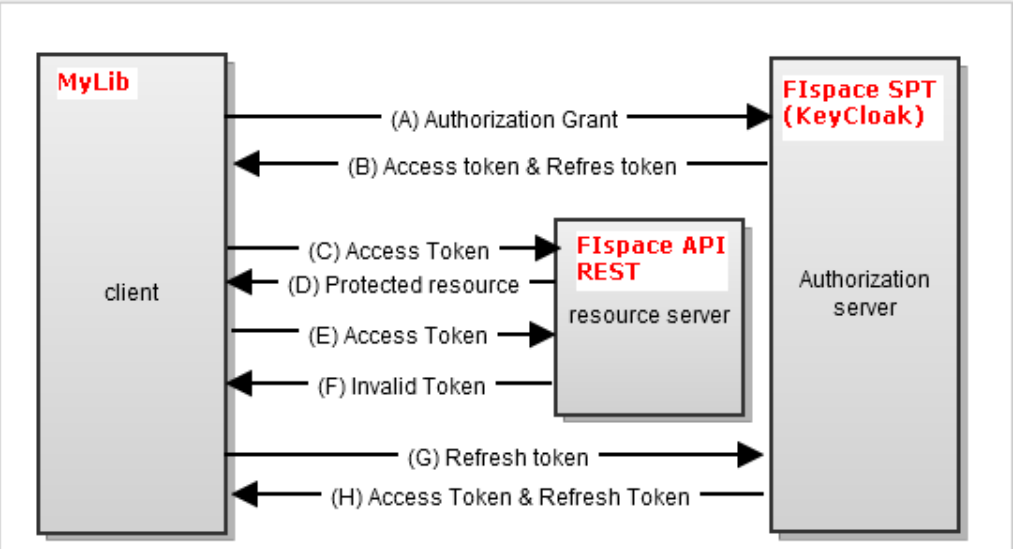


Figure 58: OAuth protocol in the library to access FIspace Platform

Second, you have to arrange the necessary OAuth client **scope** for using the services provided by FIspace API, application developer role and business architect role.

The following libraries are being implemented:

2.3.23.1 C# FIspace connector

C# FIspace Connector provides some features to access to FIspace platform apart from implementing the OAuth protocol developed in C#.

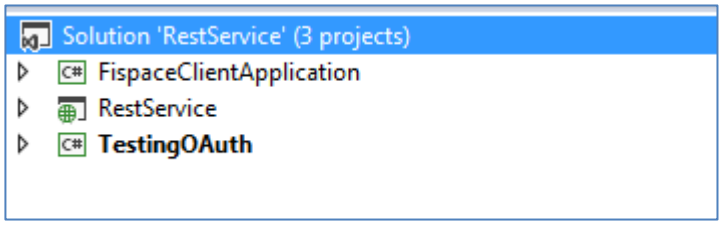


Figure 59: FIspace C# Connector

FIspace C# Solution is composed by three types of projects:

Project FIspaceClientApplication:

It provides constants and static methods for consume and connect with FIspace Platform and other common functions. The classes and interfaces are shown in the next figure:

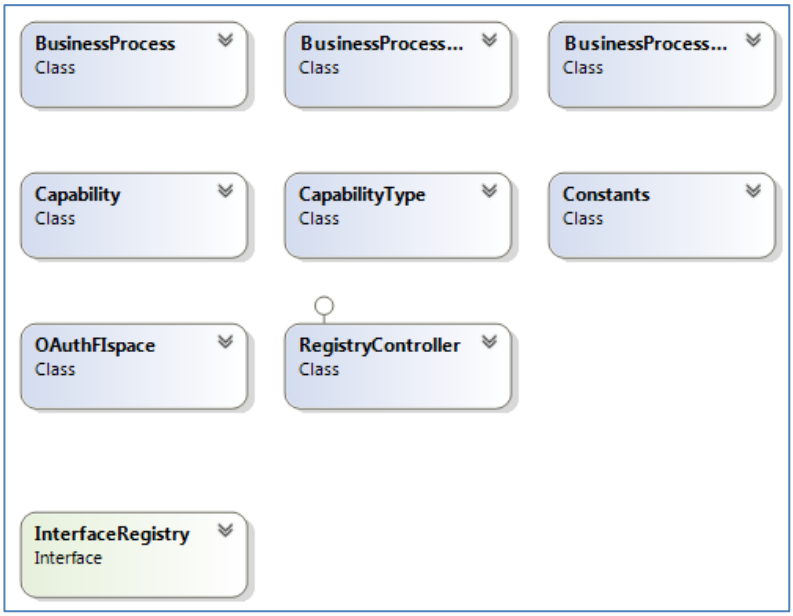


Figure 60: Diagram Class Project FIspaceClientApplication

A brief description follows:

InterfaceRegistry.cs: It's an interface, with the FIspace related methods.

RegistryController.cs: It's the main class which implements all methods of the **InterfaceRegistry.cs** interface. See next figure:

Name	Type	Modifier
Methods		
addBusinessProcess	void	public
addBusinessProcessTemplate	void	public
addCapability	void	public
addCapabilityType	void	public
getBusinessProcess	string	public
getBusinessProcesses	string	public
getBusinessProcessTemplate	string	public
getBusinessProcessTemplates	string	public
getCapabilities	string	public
getCapability	string	public
getCapabilityType	string	public
getCapabilityTypes	string	public
RegistryController		private
removeBusinessProcess	void	public
removeBusinessProcessTemplate	void	public
removeCapability	void	public
removeCapabilityType	void	public
setBusinessProcess	void	public
setBusinessProcessTemplate	void	public
setCapability	void	public
setCapabilityType	void	public

Figure 61: RegistryController.cs methods

BusinessProcess.cs: represents a FIspace BusinessProcess object.

BusinessProcessTemplate.cs: represents a FIspace BusinessProcessTemplate object.

BusinessProcessTemplateList.cs: represents a List of BusinessProcessTemplate objects.

Capability.cs: represents a FIspace Capability.

CapabilityType.cs: represents FIspace Capability Type.

OAuthFIspace.cs: implements the OAuth capability.

Constants.cs: manages the constants.

Project RestService:

It provides an ASP NET web application for testing easily the web services.

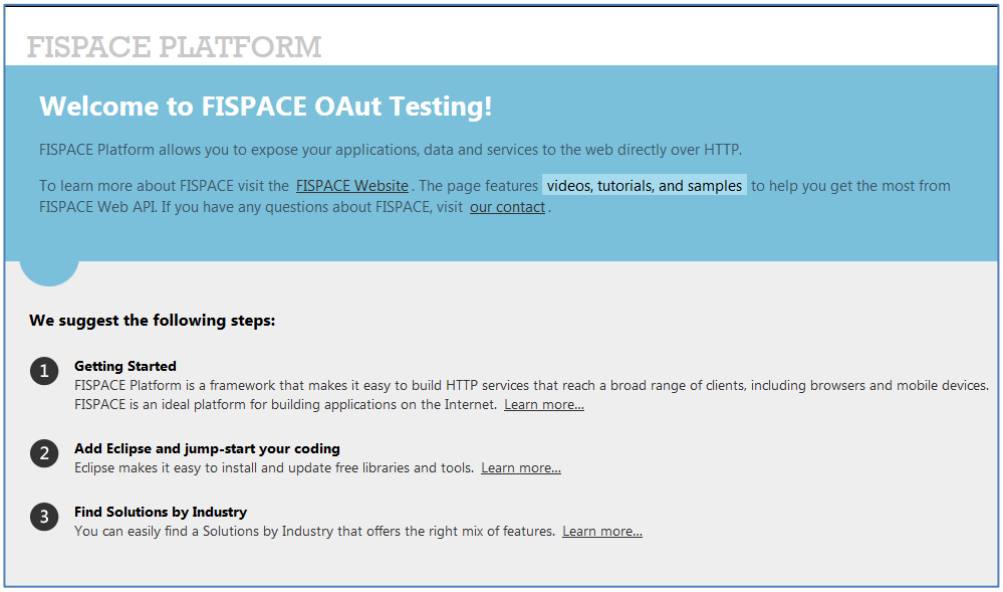


Figure 62: Web Application

Project TestingOAuth:

It provides a set of methods for testing an OAuth provider.

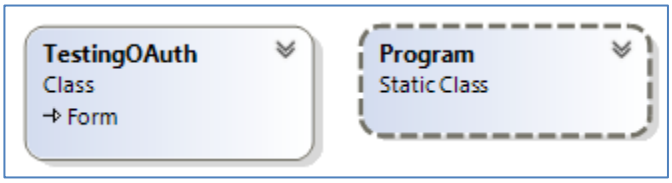


Figure 63: Diagram Class Project TestingOAuth

Note: Others secured FIspace connector are currently in development and will be provided in the next releases cycles. It will allow creating FIspace connectors for JavaScript and Python languages.

2.4 Description and use of features

The following updating procedures are used in the FIspace SDK development.

2.4.1 Update procedure for plugins

XXX Update-site:

The update-site of a specific plugin can be updated through its corresponding update-site project.

Specifically the site.xml file is used by following the next steps:

- Under the “Managing the Site” window select the “Fispace-XXX” category and then “Add feature”.
- Click on Build All.
- The update-site can be found under target folder of the project ready to be uploaded with all of its contents on a public server.

General updatesite:

The general update-site can be updated by following the next steps:

- Clone the plugin repository (hg clone ssh://hg@bitbucket.org/fispace/plugin)
- Build the plugin project (mvm clean install)
- Import the plugin projects as existing maven projects inside your eclipse workspace. (File -> Import -> Maven -> Existing Maven Projects -> Navigate to the plugin folder -> Next -> Finish)
- Open the site.xml file under the fispace-updatesite project (fispace-updatesite/site.xml)
- Under the “Managing the Site” window select the “Fispace” category and then “Add feature”.
- Add all of the FIspace features.
- Click on Build All.
- The general update-site can be found under target folder of the project ready to be uploaded with all of its contents on a public server.

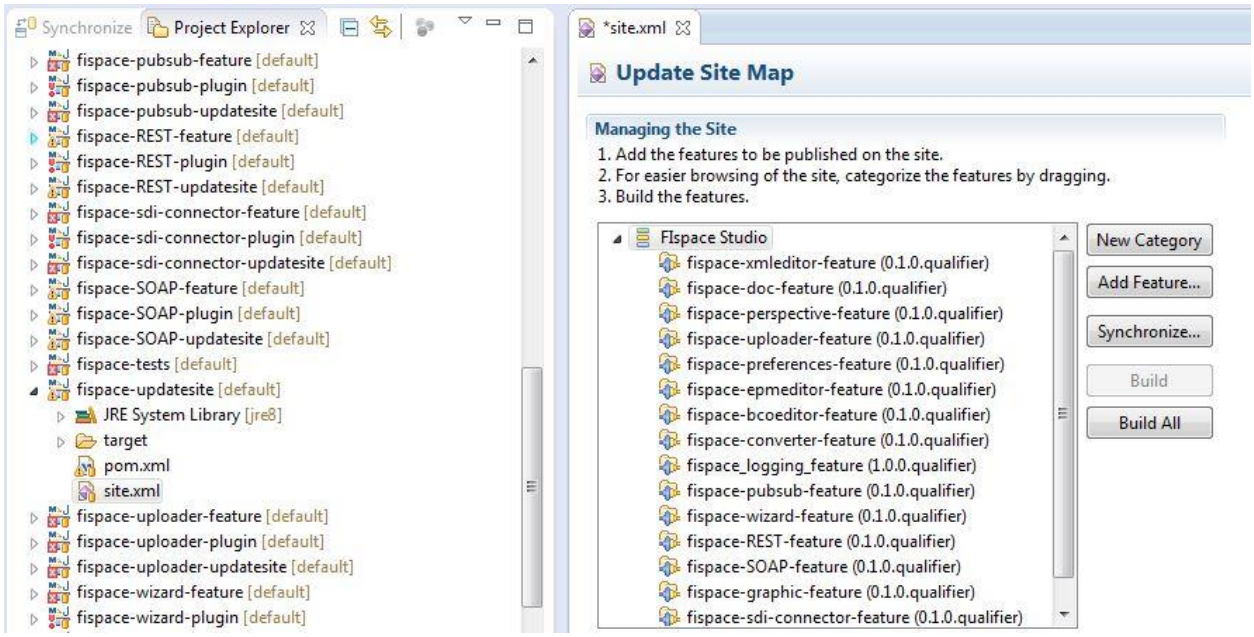


Figure 64: Updatesite procedure

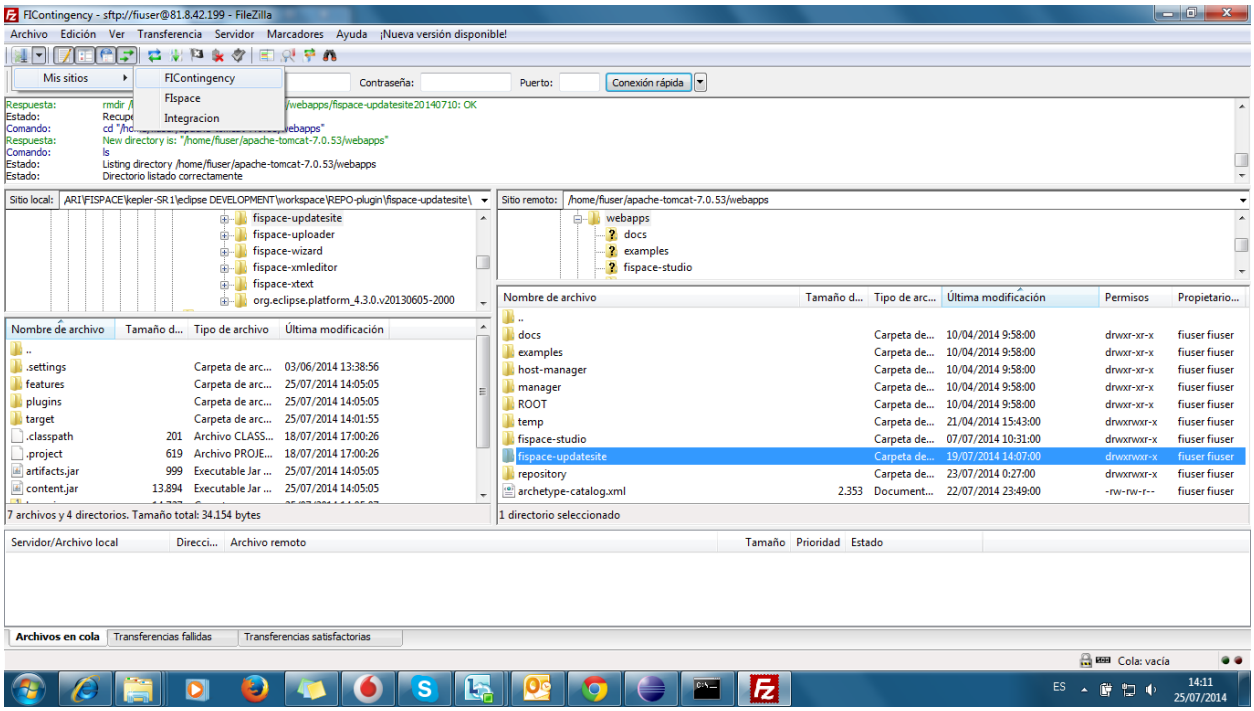


Figure 65: Uploading to a public server

2.4.2 Update procedure for FIspaceStudio Tool

Preparation

- Download (http://download.eclipse.org/e4/downloads/) and install the required plug-ins in your eclipse platform.
- Download the delta-pack for your eclipse IDE version and add it in your "Active platform" (Preferences -> Plug-in Development -> Target Platform)

Steps

- Clone the "plugin" repository in your local environment outside eclipse.

- Import as an "Existing Maven project" the plugin project in your eclipse workspace.
- Navigate to the “fispace-studio” project.
- Open the FIspace Studio.product file.
- In the "Dependencies" tab click the "Add required Plug-ins" button and save the file.
- In the "General" tab, under the “Exporting” window, click the “Eclipse Product export wizard”.
- Export FIspace Studio for the platform of your preference.

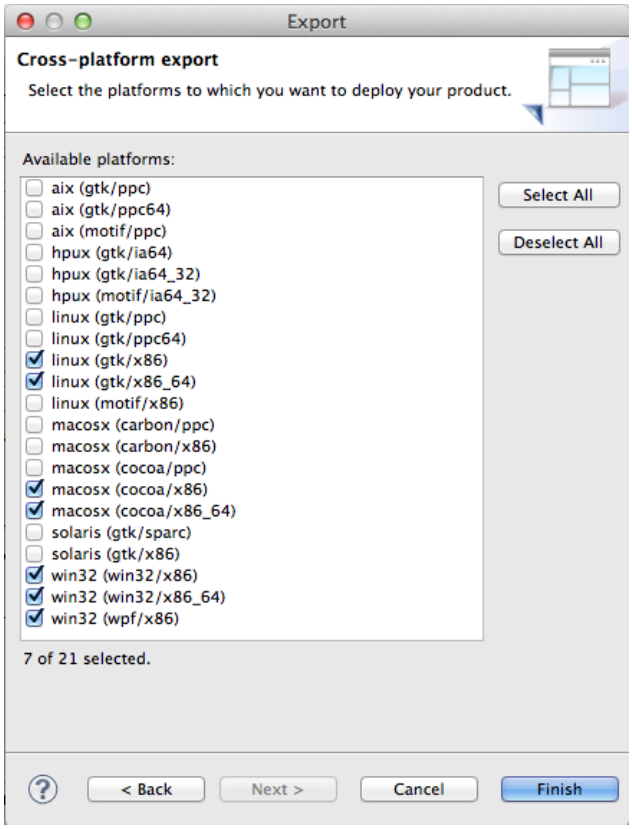


Figure 66: FIspace Studio exporting wizard

2.4.3 Update procedure for archetypes

With each code iteration, we change the archetype version according to the version of all the FIspace software. This action generates an update procedure that is explained below.

If we have modified the archetype project, we need to regenerate the archetype:

In order to generate an archetype from a project we need to follow the following steps:

1. Export and install archetypes from: https://bitbucket.org/fispace/archetypes. Obtaining a structure as described in the section 2.3.16.1FIspace-archetypes.

2. Once exported the archetype project, you could modify and create more folders and files inside.
3. Open a command window and, inside the project (for example the archetype folder “app-compact”), execute this command (to clean the target folder).

```
C:\Users\archetypes\app-compact> mvn clean
```

4. Execute this command to create a project with the structure defined in “app-compact”.

```
C:\Users\archetypes\app-compact> mvn archetype:create-from-project
```

5. Inside the target folder that was created, execute this command:

```
C:\Users\app-compact\target\generated-sources\archetype>mvn install
```

6. Once installed in your .m2 directory of your local machine you could generate the archetype, using the following command:

```
mvn archetype:generate -DarchetypeCatalog=local -DarchetypeGroupId=eu.fispace.apps -
DarchetypeArtifactId=fispace-app-compact-archetype -DarchetypeVersion=0.9.0-SNAPSHOT -
DgroupId=eu.fispace.apps -DartifactId=app-bus-compact -Dconfirm=N -DinteractiveMode=false
```

7. Verify the creation of the archetype. A folder structure like the one described in Figure 67 will be obtained.

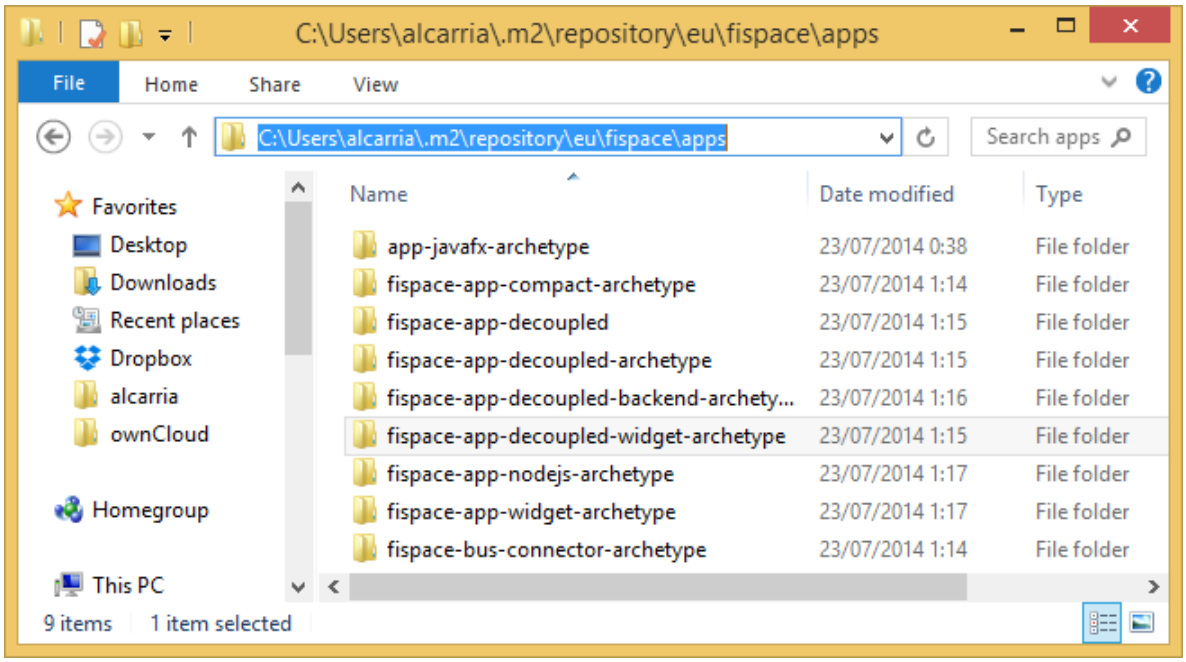


Figure 67: Archetype folders in local repository

Once the archetypes have been generated the binaries are stored in the local maven repository folder (“./.m2”). It is time to upload these archetypes to FIspace server. The IP server that we are currently using to store archetypes is: 81.8.42.199. We will use a WinSCP program to upload the archetypes installed on our local machine (Figure 68).

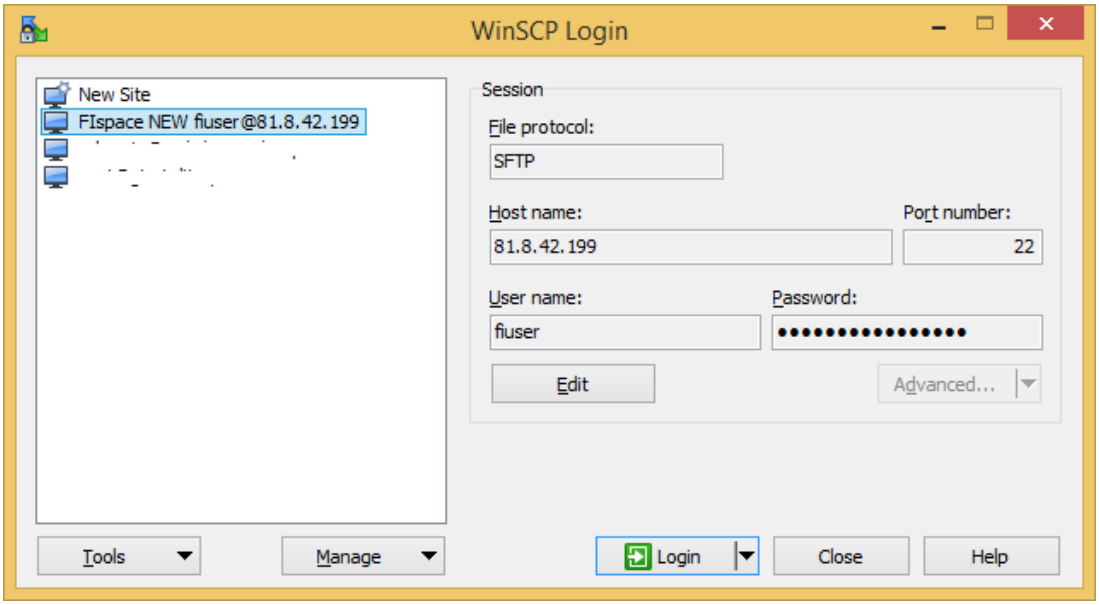


Figure 68: WinSCP connection to archetype repository

To connect to the provided repository we need some access credentials (Figure 69).

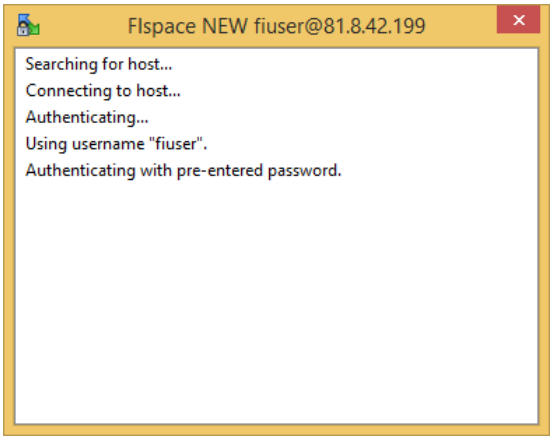


Figure 69: Repository authentication with FIspace credentials

The archetype repository is a virtual machine that has an Apache Tomcat (version 7.0.53). Archetype files must be uploaded to the webapps/repository. Figure 70 describes the folder structure when archetypes have been uploaded:

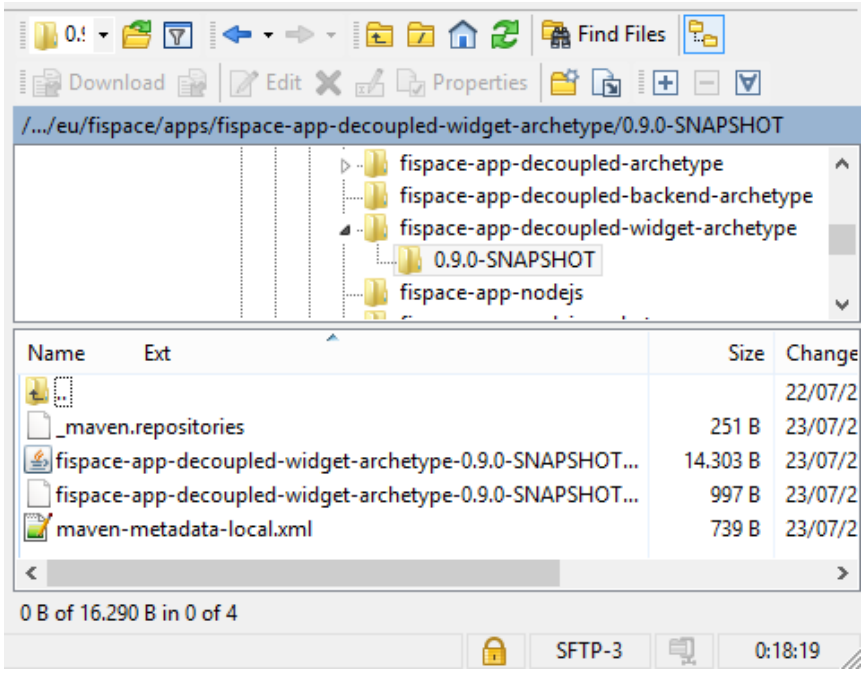


Figure 70: Archetype folder structure in remote repository

Once the upload is completed the following maven commands will work and they will create a project with the same structure than the archetype:

#To create a compact project

mvn archetype:generate -DarchetypeCatalog=http://37.131.251.110:8080/repository/ -DarchetypeRepository=http://37.131.251.110:8080/repository/ -DarchetypeGroupId=eu.fispace.apps -DarchetypeArtifactId=fispace-app-compact-archetype -DarchetypeVersion=0.14.0-SNAPSHOT -DgroupId=eu.fispace.apps -DartifactId= myAppName -Dconfirm=N -DinteractiveMode=false

#To create a decoupled project

mvn archetype:generate -DarchetypeCatalog=http:// 37.131.251.110:8080/repository/ -DarchetypeRepository=http:// 37.131.251.110:8080/repository/ -DarchetypeGroupId=eu.fispace.apps -DarchetypeArtifactId=fispace-app-decoupled-archetype -DarchetypeVersion=0.14.0-SNAPSHOT -DgroupId=eu.fispace.apps -DartifactId= myAppName -Dconfirm=N -DinteractiveMode=false

#To create a javafx project

mvn archetype:generate -DarchetypeCatalog=http:// 37.131.251.110:8080 /repository/ -DarchetypeRepository=http:// 37.131.251.110:8080 /repository/- DarchetypeGroupId=eu.fispace.apps -DarchetypeArtifactId=app-javafx-archetype -DarchetypeVersion=0.14.0-SNAPSHOT -DgroupId=eu.fispace.apps -DartifactId= myAppName -Dconfirm=N -DinteractiveMode=false

#To create a node.js project

mvn archetype:generate -DarchetypeCatalog=http:// 37.131.251.110:8080 /repository/ -DarchetypeRepository=http:// 37.131.251.110:8080 /repository/ -DarchetypeGroupId=eu.fispace.apps -DarchetypeArtifactId=fispace-app-nodejs-archetype -DarchetypeVersion=0.14.0-SNAPSHOT -DgroupId=eu.fispace.apps -DartifactId= myAppName -Dconfirm=N -DinteractiveMode=false

#To create a widget project

mvn archetype:generate -DarchetypeCatalog=http:// 37.131.251.110:8080 /repository/ -DarchetypeRepository=http:// 37.131.251.110:8080 /repository/ -DarchetypeGroupId=eu.fispace.apps -DarchetypeArtifactId=fispace-app-widget-archetype -DarchetypeVersion=0.14.0-SNAPSHOT -DgroupId=eu.fispace.apps -DartifactId= myAppName -Dconfirm=N -DinteractiveMode=false

Archetype maven commands for remote repository

Figure 71: Archetype maven commands

2.5 User SDK Guidance and resources

There is a tutorial available at https://bitbucket.org/fispace/apps/wiki/tutorial/getting-started/sdk-tutorial for the plugins installation/use, and another one for the FIspaceStudio tool at https://bitbucket.org/fispace/apps/wiki/tutorial/getting-started/FIspaceStudio-tutorial.

In addition the document D200.7 Annex *"FIspace SDK User and Developer Guide"* [11] describes FIspace Software Development Kit (SDK) presented as a user and developer guideline related to the environment of development and corresponding Integrated Development Environment (IDE) based on Eclipse [26] SDK tools for Java. This annex document describes the functionalities, wizards, FIspace Eclipse plugins developed and provided by FIspace SDK to facilitate the tasks of the developers through a set of graphical user interface (GUI) which allows the developer to interact with the tools, following the necessary steps to achieve the desired work or carry out some tasks and testing purposes

The online documentation for SDK is also available in the following web site: http://dev.fispace.eu/doc/wiki/sdk

3 Glossary

The glossary provides the coherent terminological framework used in this document.

3.1 Terms and definitions

This section provides definitions of any terms that may be needed in order for the reader to understand the terminology used in the document. The author should define any definition/acronym or technical term used in the document that may be unfamiliar to the reader, and it is best to err on the side of too many rather than too few definitions. This also allows the author to frame a word within a specific context, which provides the reader with a common understanding of the author's definition.

Access control

Authorisation (or denegation) for performing a certain action (based on privileges management). The access control is carried out once the Identification and Authentication procedures have been performed.

Accounting

Process of gathering information about the usage of resources by subjects.

Acceptance and trust

Acceptability indicates the degree of approval of a technology by the users. It depends on whether the technology can satisfy the needs and expectations of its users and potential stakeholders. Within the framework of introducing new technologies, acceptability relates to social and individual aspects as well.

Application

Use of capabilities, including hardware, software and data, provided by an information system specific to the satisfaction of a set of user requirements in a given application domain.

Application Domain

Integrated set of problems, terms, information and tasks of a specific thematic domain that an application (e.g. an information system or a set of information systems) has to cope with.

Application Schema [ISO/FDIS 19109:2003]

Conceptual schema for data required by one or more applications.

Architecture (of a system) [ISO/IEC 10746-2:1996]

Set of rules to define the structure of a system and the interrelationships between its parts.

Architecture (of a system) [ISO/IEC 10746-2:1996]

Set of rules to define the structure of a system and the interrelationships between its parts.

Authentication

Process of verifying the identity of a certain subject. In other words authentication indicates whether a subject is who/what it seems to be.

Generally speaking, this proof can depend on a secret that can be, e.g. what somebody has (key, smart card, …), what somebody knows (password, …), what somebody is (biometrical data, …)

Authorisation

Process of determining whether a subject is allowed to have the specified types of access to a particular resource. This is done by evaluating applicable access control information contained in a so called authorisation context. Usually, authorisation is carried out after the identification and authentication. Once a subject is identified and authenticated, it may be authorized (or not) to perform different types of access.

Availability

Availability refers to the degree to which a system, subsystem, or equipment is in a specified operable and committable state at the start of a mission, when the mission is called for at an unknown, i.e., a random time. So, availability is the proportion of time that a system is in operating condition.

Capability

Capabilities are a set of functionalities, through a combination of software and hardware, used to provide services and data. They can reside in a system or for example in a terminal itself as embedded capabilities or they can be available through the network services and infrastructure and others communication technologies as external capabilities.

Catalogue [derived from http://www.opengeospatial.org/resources/?page=glossary]

Collection of entries, each of which describes and points to a feature collection. Catalogues include indexed listings of feature collections, their contents, their coverages, and of meta-information. A catalogue registers the existence, location, and description of feature collections held by an Information Community. Catalogues provide the capability to add and delete entries. A minimum Catalogue will include the name for the feature collection and the locational handle that specifies where these data may be found. Each catalogue is unique to its Information Community.

Certificate Authority

A Trusted Third Party, responsible for ensuring the binding between the public keys and the personal data of their respective owners.

Component

Hardware component (device) or Software Component.

Conceptual model [ISO/FDIS 19109:2003(E); ISO 19101]

Model that defines concepts of a universe of discourse.

Conceptual schema [ISO/FDIS 19109:2003(E); ISO 19101]

Formal description of a conceptual model.

Coverage [ISO 19123]

Function from a spatial, temporal or spatiotemporal domain to an attribute range. A coverage associates a position within its domain to a record of values of defined data types. Thus, a coverage is a feature with multiple values for each attribute type, where each direct position within the geometric representation of the feature has a single value for each attribute type.

Data acquisition

Methods of data acquisition include methods to collect background data, digitally acquire data from sensors, and subjective data (such as data acquired from questionnaires). In addition, data in the form of manually or automatically transcribed data and reductions of collected data is also considered sensor acquired data (but with a manual sensor – the analyst).

Description Logics

Family of logic based knowledge representation languages that are a decidable subset of first order logic with well-defined semantics and inferencing (problem decision procedures). In Description Logics, a distinction is made between the terminological knowledge and the assertional knowledge. This distinction is useful for knowledge base modelling and engineering: for modelling it is just natural to distinguish between concepts and individuals; for engineering it helps by separating key inference problems.

Digital Certificate

A kind of digital document that contains structured information about the identity of its owner along with her/his public key, signed all together with a Certificate Authority's private key.

Digital Signature

The encrypted form of a message with the private key of the owner, indicating in a secure way the creator of the message, as well as the identity of a signed data.

Encryption

The act of modifying the contents of a message in an algorithmic and secure way, so that it can not be observed or altered in while in transit.

End-User

All users that are involved in an application domain and that use the applications, the services built by the system users according to the system and service Architecture.

Feature [derived from ISO 19101]

Abstraction of a real world phenomenon [ISO 19101] perceived in the context of an Application. In this general sense, a feature corresponds to an "object" in analysis and design models.

Framework [http://www.opengeospatial.org/resources/?page=glossary]

An information architecture that comprises, in terms of software design, a reusable software template, or skeleton, from which key enabling and supporting services can be selected, configured and integrated with application code.

Generic

A service is generic, if it is independent of the application domain. A service infrastructure is generic, if it is independent of the application domain and if it can adapt to different organisational structures at different sites, without programming (ideally).

Identification

The identification process allows relating a person/device with the service environment. The “electronic identity” is something like a credential or a “business card”, suitable to be verified throughout the authentication process.

Implementation [http://www.opengeospatial.org/resources/?page=glossary]

Software package that conforms to a standard or specification. A specific instance of a more generally defined system.

Info-structure Service

Service that is required to operate a system oriented service in the sense that it plays an indispensable role in the operation of an architecture or system oriented service.

Interface [ISO 19119:2005; http://www.opengis.org/docs/02-112.pdf]

Named set of operations that characterize the behaviour of an entity.

The aggregation of operations in an interface, and the definition of interface, shall be for the purpose of software reusability. The specification of an interface shall include a static portion that includes definition of the operations. The specification of an interface shall include a dynamic portion that includes any restrictions on the order of invoking the operations.

Interoperability [ISO 19119:2005 or OGC; http://www.opengeospatial.org/resources/?page=glossary]

Capability to communicate, execute programs, or transfer data among various functional units in a manner that require the user to have little or no knowledge of the unique characteristics of those units [ISO 2382-1]. (http://www.opengeospatial.org/ogc/glossary/i)

Loose coupling [W3C; http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#loosecoupling]

Coupling is the dependency between interacting systems. This dependency can be decomposed into real dependency and artificial dependency: Real dependency is the set of features or services that a system consumes from other systems. The real dependency always exists and cannot be reduced. Artificial dependency is the set of factors that a system has to comply with in order to consume the features or services provided by other systems. Typical artificial dependency factors are language dependency, platform dependency, API dependency, etc. Artificial dependency always exists, but it or its cost can be reduced. Loose coupling describes the configuration in which artificial dependency has been reduced to the minimum.

Middleware [http://www.opengeospatial.org/resources/?page=glossary]

Software in a distributed computing environment that mediates between clients and servers.

Open Architecture [based on (Powell 1991)] [35]

Architecture whose specifications are published and made freely available to interested vendors and users with a view of widespread adoption of the architecture. An open ar-

chitecture makes use of existing standards where appropriate and possible and otherwise contributes to the evolution of relevant new standards.

Operation [ISO 19119:2005; http://www.opengis.org/docs/02-112.pdf]

Specification of a transformation or query that an object may be called to execute. An operation has a name and a list of parameters.

Performance indicators definition (PI)

PIs are quantitative or qualitative measurements, agreed on beforehand, expressed as a percentage, index, rate or other value, which is monitored at regular or irregular intervals and can be compared with one or more criteria.

Platform (Service)

Set of infrastructural means and rules that describe how to specify service interfaces and related information and how to invoke services in a distributed system.

Reference Model [ISO Archiving Standards; http://ssdoo.gsfc.nasa.gov/nost/isoas/us04/defn.html]

A reference model is a framework for understanding significant relationships among the entities of some environment, and for the development of consistent standards or specifications supporting that environment. A reference model is based on a small number of unifying concepts and may be used as a basis for education and explaining standards to a non-specialist.

Reliability

Reliability is the ability of a system or component to perform its required functions in routine circumstances, as well as hostile or unexpected circumstances, under stated conditions for a specified period of time.

Resource

Functions (possibly provided through services) or data objects.

Service [ISO 19119:2005; ISO/IEC TR 14252; http://www.opengis.org/docs/02-112.pdf]

Distinct part of the functionality that is provided by an entity through interfaces.

REST

Representational state transfer (REST) is an abstraction of the architecture of the World Wide Web; more precisely, REST is an architectural style consisting of a coordinated set of architectural constraints applied to components, connectors, and data elements, within a distributed hypermedia system. REST ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements.

Service [ISO 19119:2005; ISO/IEC TR 14252; http://www.opengis.org/docs/02-112.pdf]

Distinct part of the functionality that is provided by an entity through interfaces.

Session

Temporary association between a subject and a principal as a result of an authentication process initiated by the subject. Information about a session is stored in authentication session information.

SOAP

Simple Object Access protocol is a protocol specification for exchanging structured information in the implementation of web services in computer networks. It uses XML Information Set for its message format, and relies on other application layer protocols, most notably Hypertext Transfer Protocol (HTTP) or Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission.

Software Component [derived from component definition of http://www.opengeospatial.org/resources/?page=glossary]

Software program unit that performs one or more functions and that communicates and interoperates with other components through common interfaces.

Source System

Container of unstructured, semi-structured or structured data and/or a provider of functions in terms of services. The source systems are of very heterogeneous nature and contain information in a variety of types and formats.

Support Service

Service that facilitates the operation of an architecture or system oriented service, e.g. providing an added value by combining the usage of Info-Structure Services.

System [ISO/IEC 10746-2:1996]

Something of interest as a whole or as comprised of parts. Therefore a system may be referred to as an entity. A component of a system may itself be a system, in which case it may be called a sub-system.

Note: For modelling purposes, the concept of system is understood in its general, system theoretic sense. The term "system" can refer to an information processing system but can also be applied more generally.

System User

Provider of services that are used for an application domain as well as IT architects, system developers, integrators and administrators that conceive, develop, deploy and run applications for an application domain.

Terminal

Terminals are a mobile device that is capable of running mobile services and/or mobile applications.

Use case

A common definition of use cases is the one described by Jacobson (Jacobson et al (1995) [36]): "*When a user uses the system, she or he will perform a behaviourally related sequence of transactions in a dialogue with the system. We call such a special sequence a use case*". In Other words, a use case is a textual presentation or a story about the usage of the system told from an end user's perspective.

The use cases provide some tools for people, with different skills (e.g. software developers and non-technology oriented people), to communicate with each other. The use

cases are general descriptions of needs or situations that often are related to basic scenarios and that are independent of the technologies and implementations of the underlying system.

User

Human acting in the role of a system user or end user of the service and system.

WADL

The Web Application Description Language is a machine-readable XML description of HTTP-based web applications (typically REST web services) WADL models the resources provided by a service and the relationships between them. WADL is intended to simplify the reuse of web services that are based on the existing HTTP architecture of the Web. It is platform and language independent and aims to promote reuse of applications beyond the basic use in a web browser.

Web Service

Self-contained, self-describing, modular service that can be published, located, and invoked across the Web. A Web service performs functions, which can be anything from simple requests to complicated business processes. Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service.

W3C Web Service [W3C, http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice]

Software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

4 References

The following references are used as background documents for the preparation of this document. References are categorized standards (i.e. standards and specifications from the consortium working groups or alliances and specifications or drafts standardization bodies) and other documents, publications and technical or scientific books.

[1]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. Deliverable D200.1 *"FIspace Design and Release Plan"*, 2014.
[2]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. Deliverable D200.2 *"FIspace Technical Architecture and Specification"*, 2014
[3]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. Deliverable D200.3 *"FIspace Integrated Release V1"*, 2014
[4]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. Deliverable D200.4 *"FIspace Development Progress Report and V1 Updates"*, *2014*
[5]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. Deliverable D200.5 *"FIspace Integrated Release V2"*, 2014.
[6]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. Deliverable D200.6 *"FIspace Development Progress Report and V2 Updates"*, *2014*
[7]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. D200.5 Annex *"FIspace Front-End User Guide"*
[8]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. D200.5 Annex *"FIspace SDK User and Developer Guide"*
[9]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. Deliverable D200.7 *"FIspace Integrated Release V3"*, 2014.
[10]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. D200.7 Annex *"FIspace Front-End User Guide"*
[11]	FIspace project. FIspace: Future Internet Business Collaboration Networks in Agri-Food, Transport and Logistics. D200.7 Annex *"FIspace SDK User and Developer Guide"*

[12]	FIspace Business collaboration web site. http://www.fispace.eu/
[13]	FIspace Developer Documentation web site. http://dev.fispace.eu/doc/wiki/Home
[14]	FIspace Deliverables web site. http://www.fispace.eu/deliverable.html
[15]	FIspace Tutorial web site. http://www.fispace.eu/tutorials.html
[16]	FIspace Front-End Users Information web site. http://dev.fispace.eu/doc/wiki/gui
[17]	FIspace Front-End User Guide web site. http://dev.fispace.eu/doc/wiki/gui/gui-guide
[18]	FIspace App Developer Intro web site. http://dev.fispace.eu/doc/wiki/App%20Developer%20Intro
[19]	FIspace SDK Guide web site. http://dev.fispace.eu/doc/wiki/sdk
[20]	FIspace apps for newbies web site. https://bitbucket.org/fispace/apps/wiki/FIspace%20apps%20for%20newbies
[21]	FIWARE web site. http://www.fi-ppp.eu/projects/fi-ware/
[22]	FIWARE Catalogue of the Generic Enablers (GEs). http://catalogue.fi-ware.org/
[23]	FIWARE community web site. http://www.fi-ware.org/community/
[24]	FIWARE - Catalogue - Application Mashup – Wirecloud web site. http://catalogue.fi-ware.org/enablers/application-mashup-wirecloud
[25]	FIWARE - Catalogue - Store – Wstore web site. http://catalogue.fi-ware.org/enablers/store-wstore
[26]	Eclipse web site. https://www.eclipse.org/ , https://www.eclipse.org/downloads/
[27]	Maven web site. http://maven.apache.org/ , http://maven.apache.org/download.cgi
[28]	RabbitMQ web site. http://www.rabbitmq.com/
[29]	Express web site. http://expressjs.com/
[30]	ACSI EU-project web site. http://www.acsi-project.eu/
[31]	Proton - CEP recorded webinar and tutorial. http://edu.fi-ware.eu/course/view.php?id=58 Proton - CEP user and programmers guide. http://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/CEP_-

	_User_and_Programmer_Guide
[32]	OAuth 2.0. Open standard Authentication protocol specification. http://tools.ietf.org/html/rfc6749
[33]	Eclipse 4 RCP. http://www.vogella.com/tutorials/EclipseRCP/article.html
[34]	MongoDB. Open-source document database, NoSQL database. http://www.mongodb.org/
[35]	Powell, D. (Ed.) (1991). Delta-4: A Generic Architecture for Dependable Distributed Computing. Re-search Reports ESPRIT. Project 818/2252 Delta-4 Vol.1. ISBN 3-540-54985-4 Springer-Verlag 1991.
[36]	Jacobson, I., Bylund, S., Jonsson, P., and Ehneboom, S. (1995), “Mod-eling with Use Cases: Using contracts and use cases to build plugable architectures”. Journal of Object Oriented Programming, Vol. 8, No. 2, pp. 18-24.


FIspace
Business Collaboration