Nuxeo IDE Documentation

Table of Contents

Nuxeo IDE Documentation Center

Welcome to the documentation center for Nuxeo IDE!

Download this documentation in PDF

Getting started guide

This guide provides the major steps to start working with the Eclipse Nuxeo IDE plugin:

1. Install the Nuxeo IDE plugin
2. Step up a Nuxeo SDK
3. Create a Nuxeo project
4. Build your project

Running your Projects on the Server

Installing Nuxeo IDE adds a new perspective to Eclipse, called Nuxeo, which enables you to deploy your changes and control the runtime.

1. Configuring the Server Launcher
2. Deployment profiles
3. Lauching the Server
4. Remote debugging

Working with Studio Projects

- Configuring a Nuxeo Connect Account
- Binding Studio Projects to an Nuxeo IDE Project
- Deploying a Project with Studio Dependencies
- Settings Eclipse Development Templates

Wizard Index

Studio

Nuxeo

Getting started with Nuxeo IDE

You should now follow the Quick Start page for getting started with Nuxeo Studio and Nuxeo IDE.

Installing Nuxeo IDE

Nuxeo IDE is available from the Eclipse Marketplace and from the following Eclipse update site: http://community.nuxeo.com/static/nuxeo-ide/stable/site/

Prerequisites

- Eclipse 4.3 (Kepler) ,
- Java 7,
- A Tomcat based distribution of Nuxeo Platform 5.8 or later.

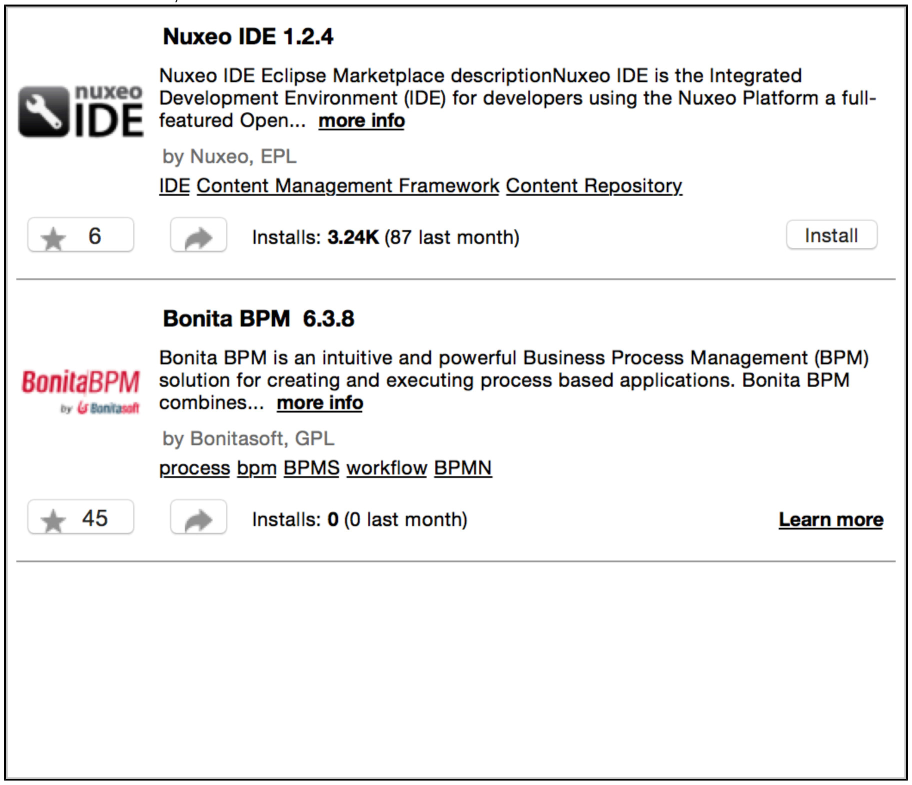
This installation guide assumes that you have already installed some flavor of a supported version of Eclipse. If not, then you can download Eclipse from http://www.eclipse.org/downloads.

Installation

Installing Nuxeo IDE from the Eclipse Marketplace.

This procedure can only be used for **Nuxeo 6.0 and later** versions. **For Nuxeo 5.8**, please refer to the **manual installation procedure** below in this page.

1. In Eclipse, go into the **Help**, **Eclipse Marketplace** menu.
2. The Eclipse Marketplace window opens.
3. Search for **Nuxeo**, select **Nuxeo IDE** and click on the **Install** button.



Nuxeo IDE 1.2.4

Nuxeo IDE Eclipse Marketplace descriptionNuxeo IDE is the Integrated Development Environment (IDE) for developers using the Nuxeo Platform a full-featured Open... more info

by Nuxeo, EPL

IDE Content Management Framework Content Repository

6 Installs: **3.24K** (87 last month) Install

Bonita BPM 6.3.8

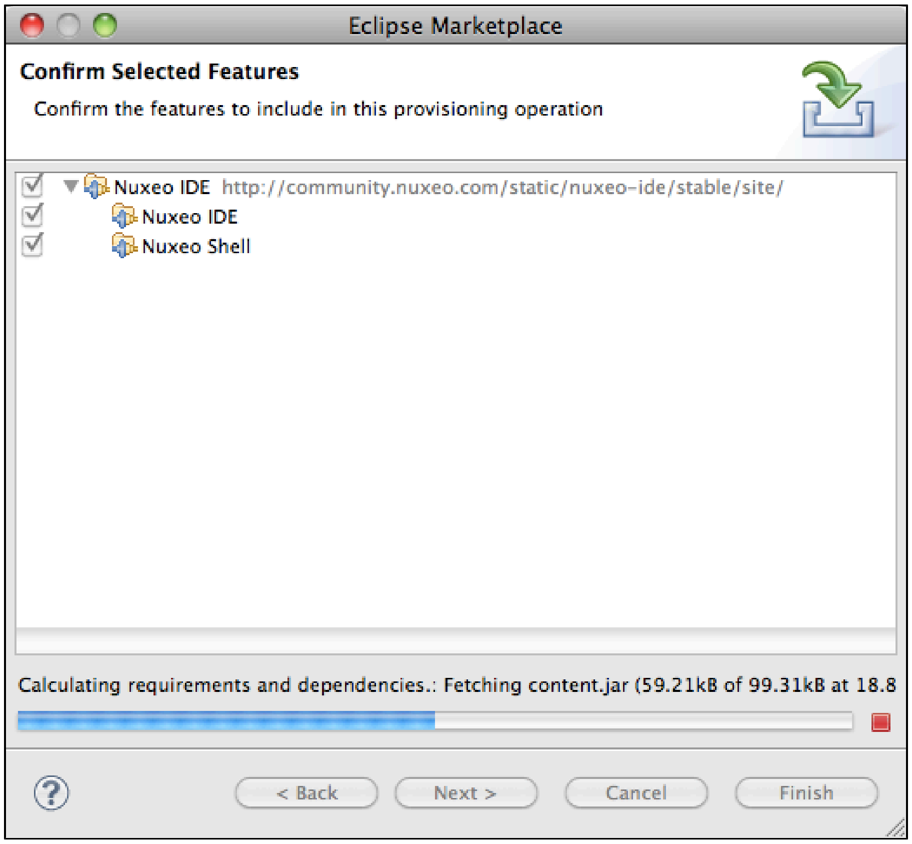
Bonita BPM is an intuitive and powerful Business Process Management (BPM) solution for creating and executing process based applications. Bonita BPM combines... more info

by Bonitasoft, GPL

process bpm BPMS workflow BPMN

45 Installs: **0** (0 last month) Learn more

4. Nuxeo IDE and Nuxeo Shell are automatically selected and downloaded.



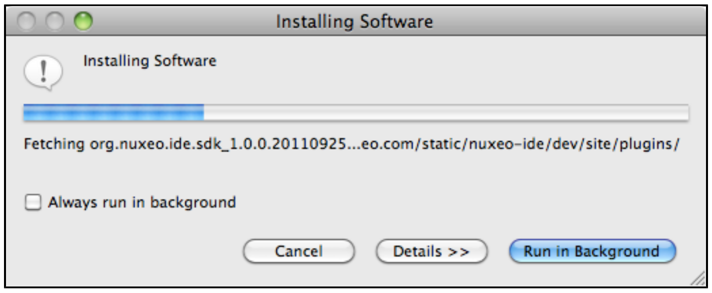
Confirm Selected Features

Confirm the features to include in this provisioning operation

- [✓] Nuxeo IDE http://community.nuxeo.com/static/nuxeo-ide/stable/site/
  - [✓] Nuxeo IDE
  - [✓] Nuxeo Shell

Calculating requirements and dependencies.: Fetching content.jar (59.21kB of 99.31kB at 18.8

5. When Nuxeo IDE and Nuxeo Shell are downloaded, click on the **Next** button.
6. Accept license when prompted.
   Installation begins. After a few seconds, a security warning is prompted.
7. On the security warning window, click on **OK**.
   Installation continues.



Installing Software

Fetching org.nuxeo.ide.sdk_1.0.0.20110925...eo.com/static/nuxeo-ide/dev/site/plugins/

[ ] Always run in background

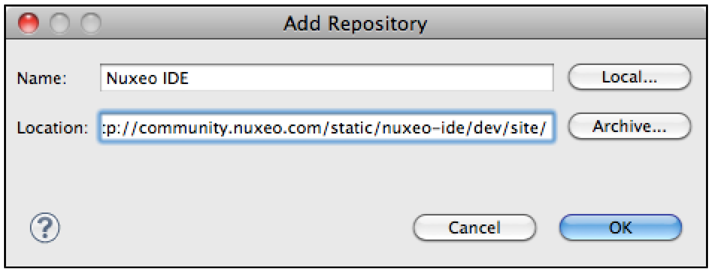
8. Restart Eclipse when prompted.

   Nuxeo IDE is installed. A new button is available in the Eclipse toolbar:

Now you can start to explore and use the Nuxeo IDE. For instance, let's take a look at The Nuxeo IDE perspective.

Installing Nuxeo IDE manually from Eclipse menu

1. Open the install dialog from the Eclipse menu: **Help** > **Install New Software...**
2. Click on **Add...** button and enter the Update Site URL (stable for the latest released version, or development for a development snapshot) and name and click the **OK** button.
   - Update site for **latest Nuxeo 5.8 compatible version**: http://community.nuxeo.com/static/nuxeo-ide/releases/1.2.4.R12x_v20141120_1114/site/
   - Update site for latest Nuxeo IDE snapshot version: http://community.nuxeo.com/static/nuxeo-ide/dev/site/
   - Update site for latest Nuxeo IDE stable version: http://community.nuxeo.com/static/nuxeo-ide/stable/site/
   - To get latest Nuxeo IDE from QA: http://qa.nuxeo.org/jenkins/job/nuxeo-ide-master/ws/nuxeo-ide/sites/org.nuxeo.ide.site/target/site/

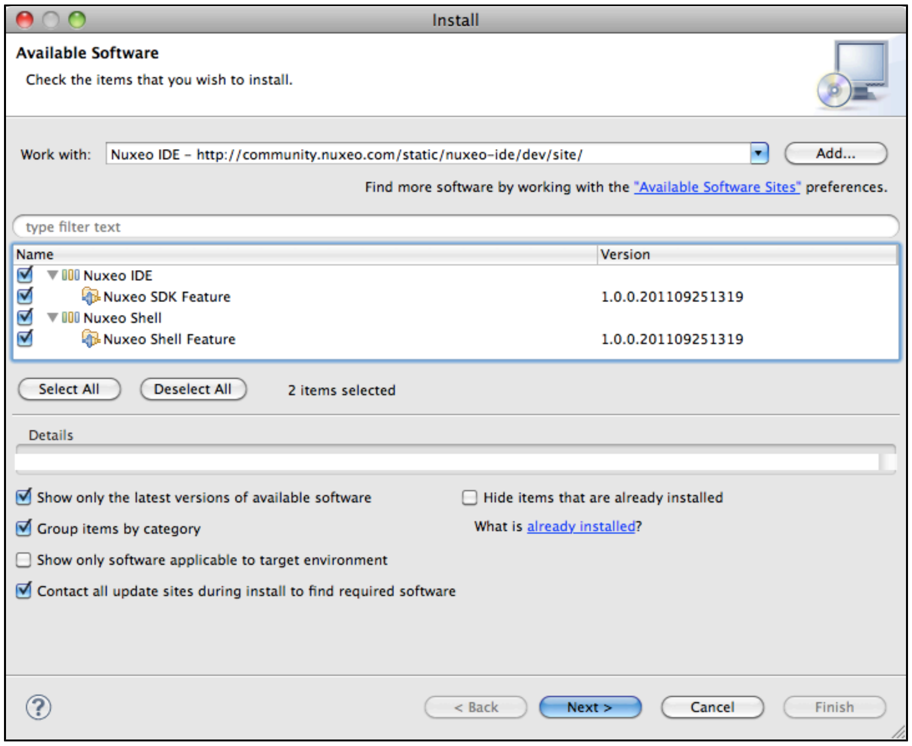


Add Repository

Name: Nuxeo IDE

Location: :p://community.nuxeo.com/static/nuxeo-ide/dev/site/

3. If nothing appears, uncheck the **Group items by category** box.
4. Check both **Nuxeo IDE** and **Nuxeo Shell** and click **Next**.



Install

Available Software

Check the items that you wish to install.

Work with: Nuxeo IDE - http://community.nuxeo.com/static/nuxeo-ide/dev/site/

Find more software by working with the "Available Software Sites" preferences.

Name	Version
☑ Nuxeo IDE	
☑ Nuxeo SDK Feature	1.0.0.201109251319
☑ Nuxeo Shell	
☑ Nuxeo Shell Feature	1.0.0.201109251319

2 items selected

Details

☑ Show only the latest versions of available software

☐ Hide items that are already installed

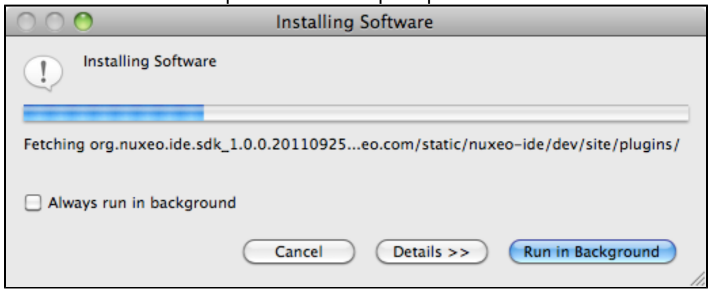
☑ Group items by category

What is already installed?

☐ Show only software applicable to target environment

☑ Contact all update sites during install to find required software

5. Follow the wizard: accept license when prompted and click **Next** until the software is downloaded.



Installing Software

Installing Software

Fetching org.nuxeo.ide.sdk_1.0.0.20110925...eo.com/static/nuxeo-ide/dev/site/plugins/

☐ Always run in background

6. Restart Eclipse when prompted.

   Nuxeo IDE is installed. A new button is available in the Eclipse toolbar:

After the IDE is installed you can Setup a Nuxeo SDK.

In this section

Setting up a Nuxeo SDK

Before starting to create your first project you must configure a Nuxeo SDK.
A Nuxeo SDK is a regular Tomcat distribution of the Nuxeo Platform that is used as the development server, where your projects will be deployed. The Nuxeo SDK is providing the classpath for your Nuxeo projects. All the JARs on the server will be visible in your project and you will be able to reference them from your code. Only Nuxeo Platform 5.5 versions and later are supported.

Do not use a production server as a Nuxeo SDK.

Installing a Nuxeo SDK

To install a Nuxeo distribution you can:

- download a Nuxeo SDK and install it:
  - Nuxeo Platform 6.0 SDK
  - Nuxeo Platform 5.8 SDK.
- checkout the Nuxeo Platform source code and build a Tomcat distribution from the Nuxeo Platform trunk (you must activate the `sdk` Maven profile: `mvn package -Ptomcat,sdk`).

It is far simpler to download the SDK, and there is generally no need to build your own distribution. The Nuxeo Platform is designed to be customized *without* changing the SDK.

Installing a downloaded distribution

1. UNzip the downloaded Nuxeo Platform distribution.
2. In the **Nuxeo SDK** preference page (Eclipse Menu -> Window -> Preferences -> Nuxeo -> Nuxeo SDK), click on **Add...**.
3. Select the directory where you unzipped your Nuxeo distribution.
   Be sure to select the right directory: the one containing the `bin` and `nxserver` directories.
4. **Check** the added Nuxeo SDK and click on the **OK** button to finish.
   Checking the box at the left of the SDK is required to mark it as the current Nuxeo SDK used in the Eclipse Workspace.

Using multiple Nuxeo SDKs

You can install multiple Nuxeo SDKs (with different flavors and versions). But you cannot use them in parallel in the same workspace.
Only one Nuxeo SDK can be used in an Eclipse workspace at a time: the one that you check in the SDK preferences view.

You can switch to another Nuxeo SDK at anytime, by checking the SDK of your choice in the preferences. This will recompile all of your Nuxeo projects against the new Nuxeo SDK.

You can for example use this method to check if your projects are compiling on different Nuxeo Platform versions.

If you need to work on projects that target different Nuxeo Platform versions, then you must use a different workspace for each Nuxeo Platform version you need. Checking a Nuxeo SDK - i.e. making an SDK the default one - is relative to an Eclipse workspace.

In this section

Creating your first Nuxeo project

Now, that you have setup a Nuxeo SDK, you can create your first Nuxeo project.

Nuxeo IDE is providing several wizards: some wizards are used to create new Nuxeo projects, others to create new components inside existing Nuxeo projects.

Two types of project creation wizards are available:

- a wizard to create a regular Nuxeo project,
- a wizard to create a Nuxeo WebEngine project.

In this section we will create a new Nuxeo WebEngine project.

To open the Nuxeo wizards you can:

- click on the icon NX in the Eclipse toolbar;
- or from the menu, click on **Nuxeo** > **Nuxeo Artifact**;
- or you can use the Eclipse way: click on **File** > **New** > **Other...** and select the **Nuxeo** folder. There you will find all the Nuxeo IDE wizards.

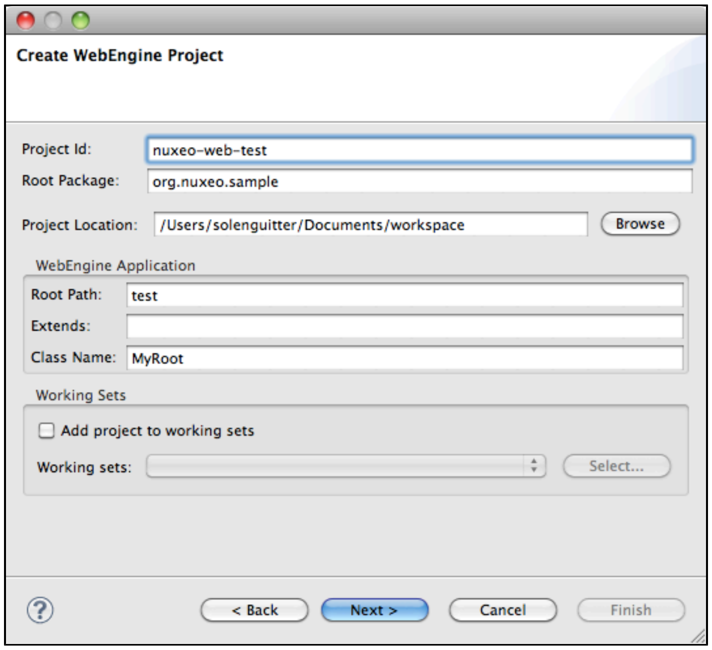
Creating a Nuxeo WebEngine project

So, let's create a new Nuxeo WebEngine project.

1. Click on the button NX in the Eclipse toolbar.
2. Click on **Nuxeo WebEngine Project** and then **Next**.
3. Enter a **Project ID**. This will be the name of the project root directory (and also of the plugin JAR).

   > Use project names like `company-feature-classifier` or `org.company.feature.classifier`.
   > Example: `nuxeo-automation-core`

   Let's choose `nuxeo-web-test` as the project ID.
4. Optionally change the project location - i.e. the parent directory (the default one points to the Eclipse workspace).
5. Enter a **Root Path** for your WebEngine Application (the path will be relative to `/nuxeo/site`). Let's use **test** as the path.



Create WebEngine Project

Project Id: nuxeo-web-test

Root Package: org.nuxeo.sample

Project Location: /Users/solenguitter/Documents/workspace

WebEngine Application

Root Path: test

Extends:

Class Name: MyRoot

Working Sets

Add project to working sets

Working sets:

6. Optionally change the class name that will be generated (and used as the root entry point to your web application).
7. Click on **Next**.
8. On this page you can modify the information that will be used to generate the project `pom.xml` file. Some of them are already filled with default values.
   You can skip this page for now and click on **Finish**.

Now you have a new project in the Project Explorer view that contains the generated `MyRoot` class and other stuff used to configure the WebEngine project.

If you open the `MyRoot` class, you will notice that there is a regular JAX-RS root resource with some WebEngine annotations.
This object will respond to **GET** HTTP requests by returning a view named "index". The view source is located in `src/main/resources/skin/views/MyRoot/index.ftl`.

Deploying the project on the server

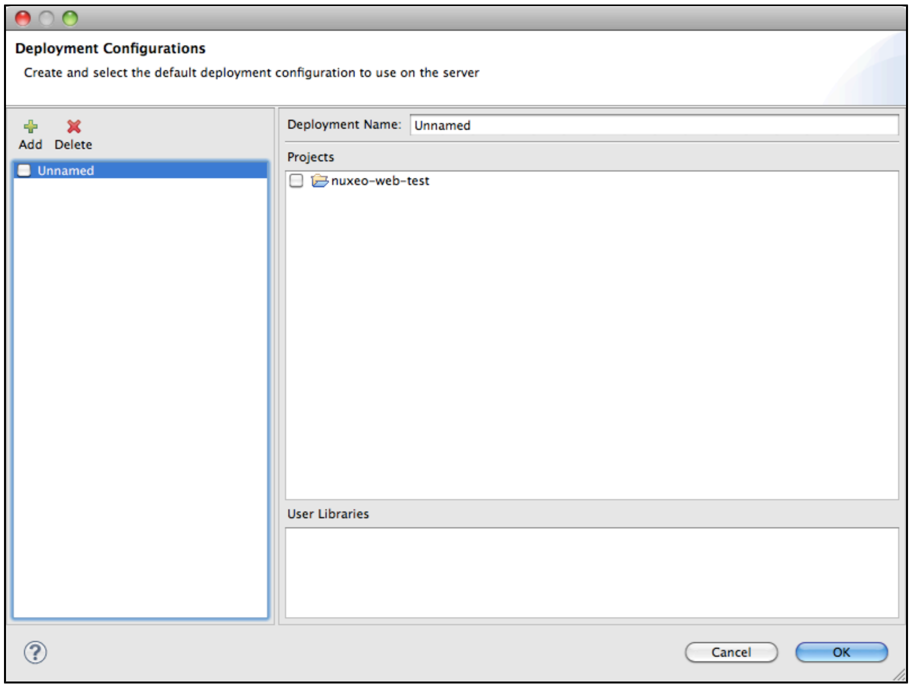
The configured Nuxeo SDK is providing a Nuxeo server that you can use to test your projects. Now, we want to deploy our new WebEngine project on the server and see what happens. This takes two steps:

1. Tell to the IDE to deploy your project. For this you must create a new **deployment profile** .
2. Start your server.

Creating a deployment profile

Here we want to create a new default profile, called "My Projects" that includes the `nuxeo-web-test` project.

1. Go into the **Nuxeo Server** view (on the bottom right).
2. Click on the button in the view toolbar. The button tooltip is "Select projects to deploy on server".
3. Click on **Add**.


Deployment Configurations

Create and select the default deployment configuration to use on the server

Deployment Name: Unnamed

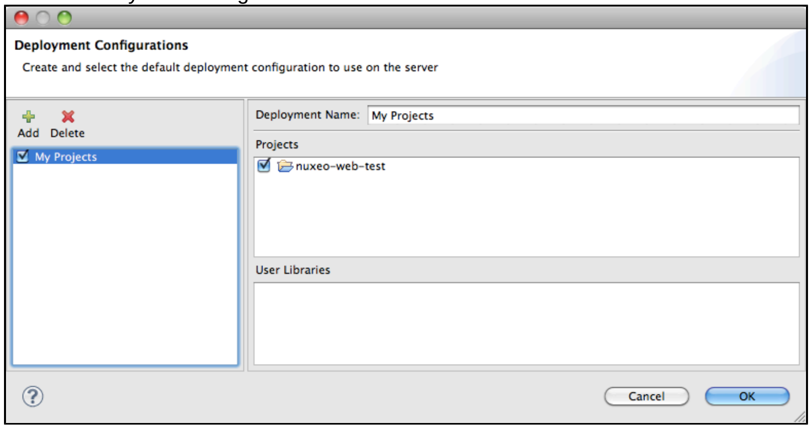
Projects

☐ nuxeo-web-test

User Libraries

4. Enter a name (in the right panel) for your deployment profile. Example: "My Projects".
5. In the left panel click on the newly created profile.
6. In the right panel, check the projects that will be deployed with this profile.
7. If you want the new profile to be the default deployment profile, check it.
8. Click on the **OK** button.

Here is what you should get:


Deployment Configurations

Create and select the default deployment configuration to use on the server

Deployment Name: My Projects

Projects

✓ nuxeo-web-test

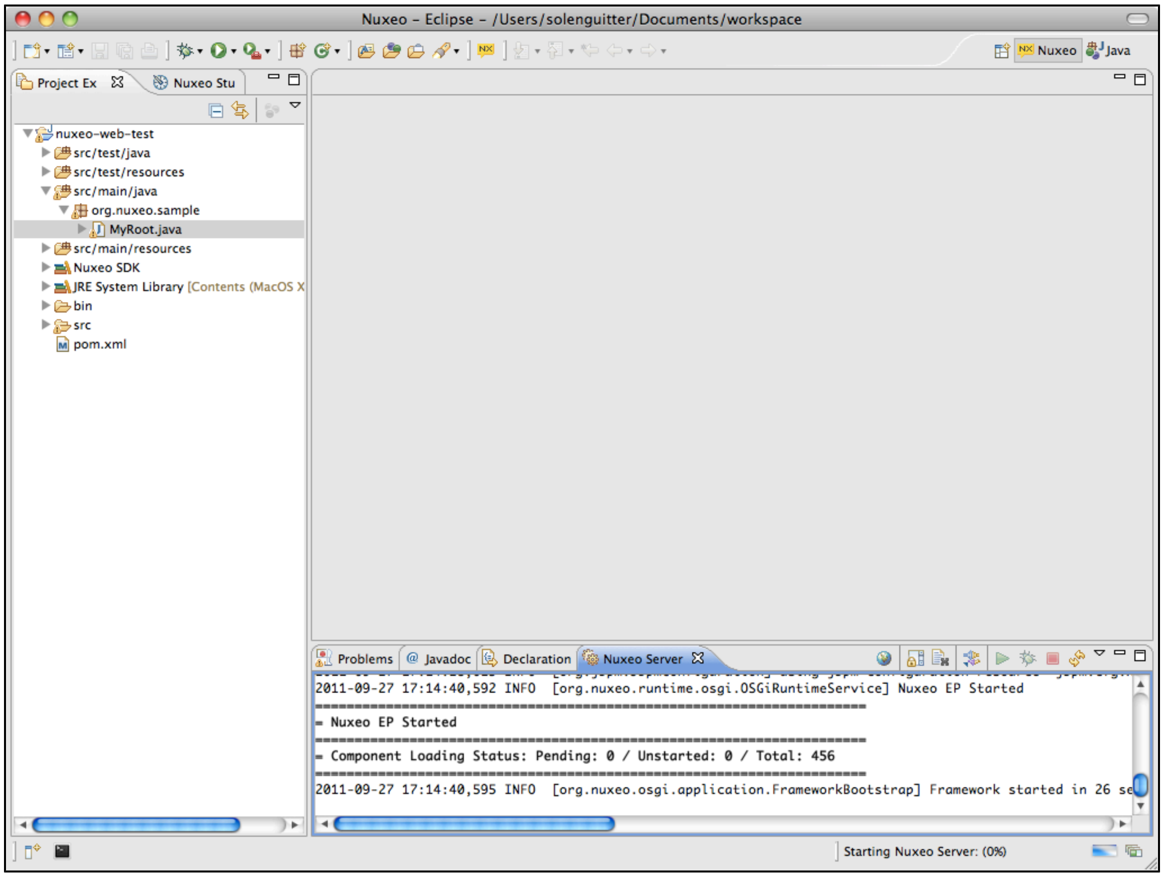
User Libraries

Now every time you start the Nuxeo server or you click on the **Refresh** button in the Nuxeo Server view toolbar, the projects selected in the current profile will be (re)deployed on the server.

Starting the server

To start the server:

1. Click on the **Run** button in the toolbar (▶).
2. Wait a minute (or more) until the server starts.

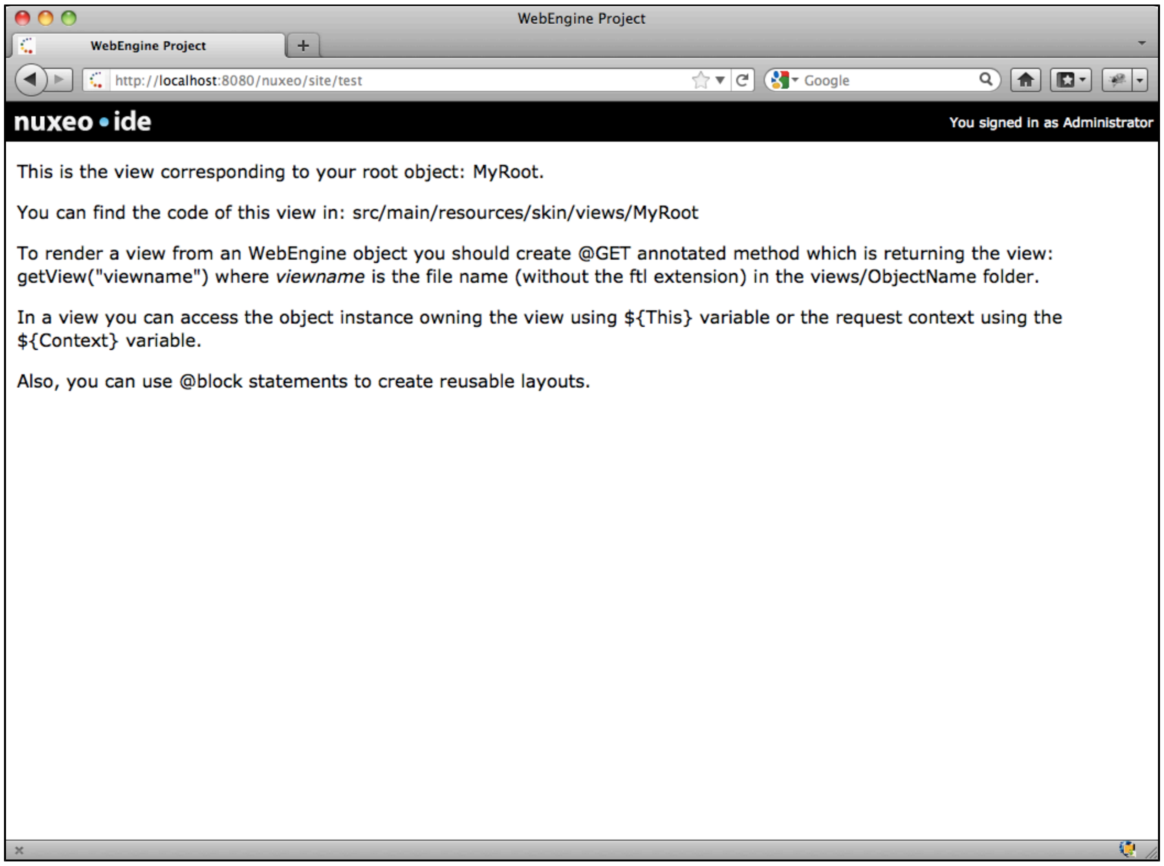


```
2011-09-27 17:14:40,592 INFO  [org.nuxeo.runtime.osgi.OSGiRuntimeService] Nuxeo EP Started
======================================================================
= Nuxeo EP Started
======================================================================
= Component Loading Status: Pending: 0 / Unstarted: 0 / Total: 456
======================================================================
2011-09-27 17:14:40,595 INFO  [org.nuxeo.osgi.application.FrameworkBootstrap] Framework started in 26 se
```

3. When done (i.e when the **Stop** button is enabled), open a browser and login using the 'Administrator/Administrator' account.

> You can open the browser by clicking the **Browse** button ( ).

4. After logging in, go to http://localhost:8080/nuxeo/site.
   There is a link **test** to your web module.
5. Click on it.
   You will be redirected to http://localhost:8080/nuxeo/site/test.
   This is the default view for the `MyRoot` root class generated by Nuxeo IDE.


WebEngine Project
http://localhost:8080/nuxeo/site/test
Google
nuxeo • ide
You signed in as Administrator

This is the view corresponding to your root object: MyRoot.

You can find the code of this view in: src/main/resources/skin/views/MyRoot

To render a view from an WebEngine object you should create @GET annotated method which is returning the view: getView("viewname") where *viewname* is the file name (without the ftl extension) in the views/ObjectName folder.

In a view you can access the object instance owning the view using ${This} variable or the request context using the ${Context} variable.

Also, you can use @block statements to create reusable layouts.

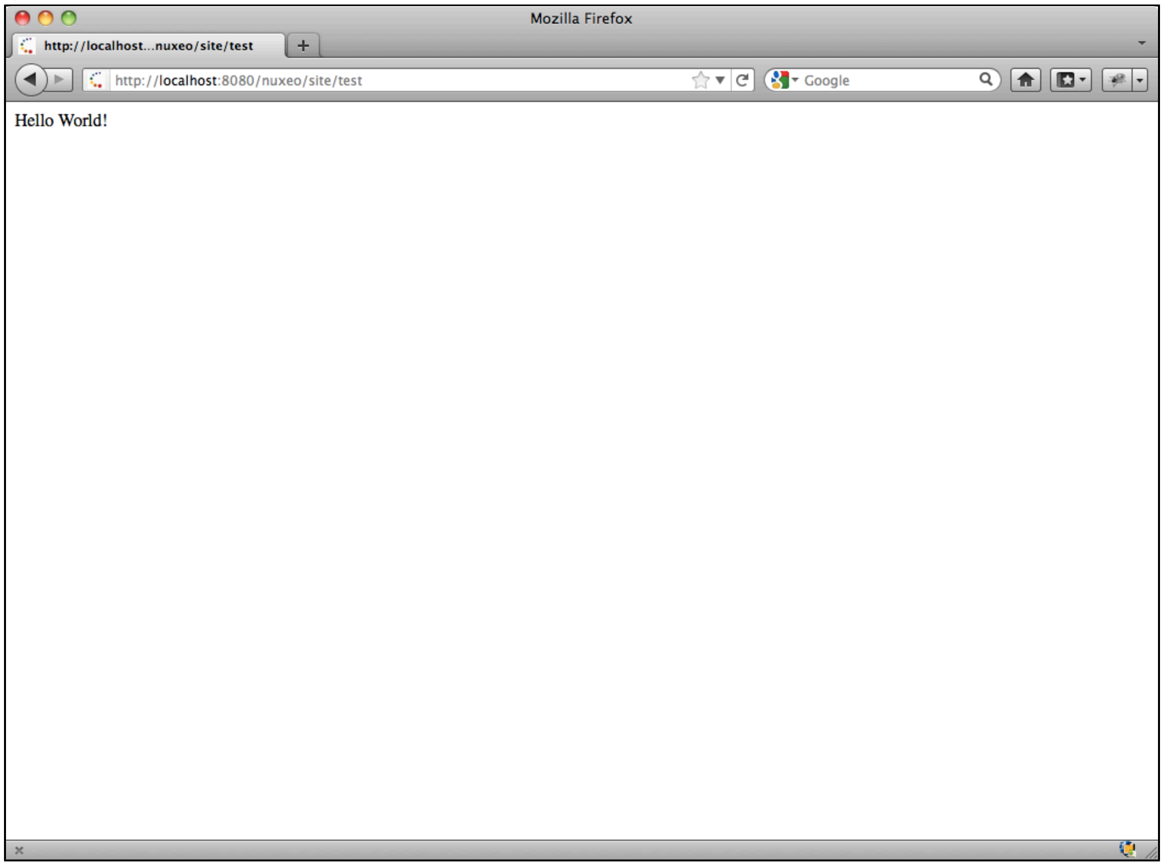
Reloading projects on server (Hot Deploy)

Now we want to change this default page. We want for example to display a string: "Hello World!" instead of that page.

1. Open the `MyRoot` class from your project and replace the content of the `doGet` method by this one:

```
@GET
    public Object doGet() {
        return "Hello World!";
    }
```

2. Click on the **Refresh** button in the **Nuxeo Server** view (🔄).
3. Go back in the browser, and refresh the http://localhost:8080/nuxeo/site/test page.
   You now see "Hello World!" on the page.


Mozilla Firefox
http://localhost...nuxeo/site/test
http://localhost:8080/nuxeo/site/test
Google

Hello World!

If nothing happens click once again: the server has a delay of two seconds before it reloads the projects.

In this section

Preparing your project for building

Now that we've done our first project, we want to be able to build it from the command line using Maven.
For this, we need to update the `pom.xml` file of the project with all the dependencies we used in our project.

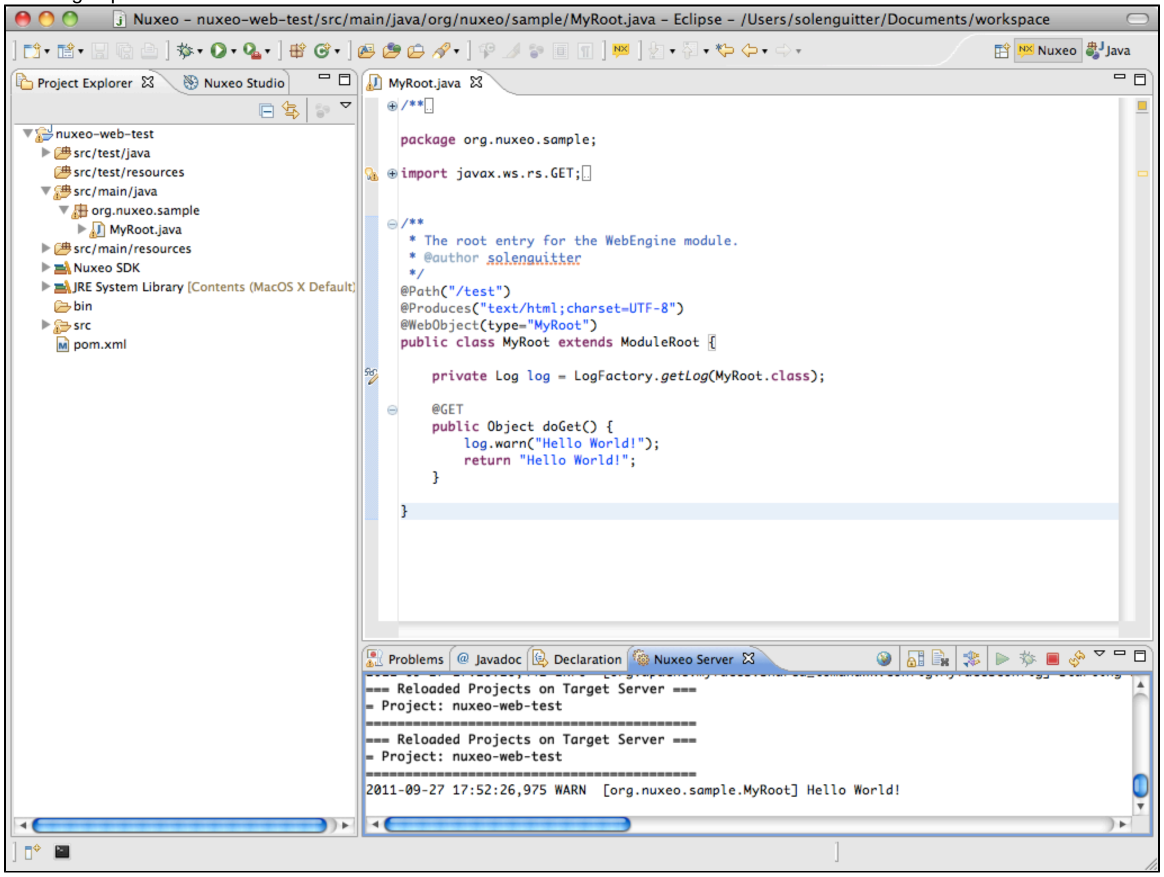
But first, let's go back into the `MyRoot` class to add a log in the `doGet` method.

1. Edit the file as following.

```
@Path("/test")
@Produces("text/html;charset=UTF-8")
@WebObject(type="MyRoot")
public class MyRoot extends ModuleRoot {

 private Log log = LogFactory.getLog(MyRoot.class);

    @GET
    public Object doGet() {
     log.warn("Hello World!");
        return "Hello World!";
    }

}
```

> You can write this code by using Eclipse auto-completion. All the Nuxeo dependencies are already in your classpath so you can use any class available in the Nuxeo SDK.
> **Be sure** to use the `org.apache.commons.logging.Log` class for logging as it is the one used in Nuxeo.

2. Now, click again on the **Refresh** button to reload your project on the server. Then refresh your web page: http://localhost:8080/nuxeo/site/test.
   Your log is printed in the **Nuxeo Server** console.

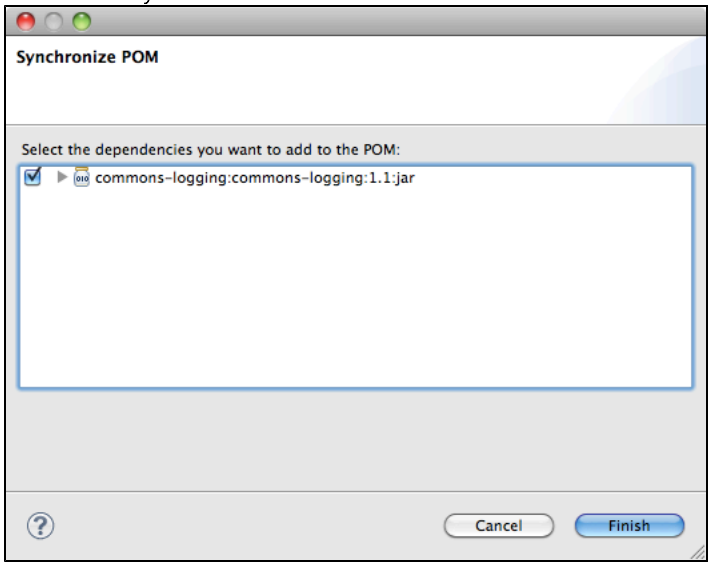


```
/**
package org.nuxeo.sample;

import javax.ws.rs.GET;

/**
 * The root entry for the WebEngine module.
 * @author solenguitter
 */
@Path("/test")
@Produces("text/html;charset=UTF-8")
@WebObject(type="MyRoot")
public class MyRoot extends ModuleRoot {

    private Log log = LogFactory.getLog(MyRoot.class);

    @GET
    public Object doGet() {
        log.warn("Hello World!");
        return "Hello World!";
    }

}
```

```
=== Reloaded Projects on Target Server ===
= Project: nuxeo-web-test
==========================================
=== Reloaded Projects on Target Server ===
= Project: nuxeo-web-test
==========================================
2011-09-27 17:52:26,975 WARN  [org.nuxeo.sample.MyRoot] Hello World!
```

Ok, but as we've seen, we added a new dependency to our project: the `Log` class from the Apache commons logging JAR.
What will happen if we try to build our project from the command line using Maven? It will fail of course, since the `pom.xml` file of our project doesn't mention any dependency to the Apache commons logging JAR. You can open the `pom.xml` file (located in the project root) and see with your eyes: there is no reference to the Apache logging artifact. So, before being able to build our project from the command line using Maven, we need to update the `pom.xml` file.

Thankfully, Nuxeo IDE provides you with a tool to do this:

1. Right-click on the `pom.xml` file file then click on **Nuxeo** > **Synchronize POM**
2. You will be prompted to select the artifacts you want to add to the `pom.xml` file.
   The missing artifacts are discovered by the Nuxeo IDE by scanning the sources of your project, and finding any class that you reference and that is not yet in the POM.
   
   Synchronize POM
   Select the dependencies you want to add to the POM:
   ☑ commons-logging:commons-logging:1.1:jar

3. Click **Finish**.
   Then look again at the `pom.xml` file. Now you have your dependency there.

Now, you can go in the command line and build your project using Maven. Have fun!

The Nuxeo IDE perspective

The Nuxeo perspective is the proposed layout to work on Nuxeo projects. It extends the default Java perspective by adding some specific Nuxeo IDE views like:

Nuxeo Server

Let's start by looking at the Nuxeo Server tab . It gives you the ability to start , stop , start in debug mode . As you can see when starting Nuxeo, the tab starts displaying the server log. There are two associated buttons to Lock the scrolling and to Clear the console. Once you see the "Server started" line in the logs, you can click on the Open Nuxeo In Web Browser button. We'll talk about the other two (hot reload and deployment profile) later.

Nuxeo components

The Components tab gives you an overview of the different Components available in the SDK. What's a component you might ask? It's an XML file declaring a Service, an Extension Point (XP) or a contribution to an XP. This is what makes Nuxeo easily extensible. Basically a Service will provide some business logic that can be modified or extend using XPs. The service knows how to handle and register contribution to XP. Here's an example. In Nuxeo there is a service that handles Document Type. It knows how to handle several XPs. One of them is used to register new Document Type for Nuxeo. You can find out more on our component model in our documentation.

Nuxeo Studio

The Studio tab lists the different Studio project you have access to. You can browse their content to see what configuration has been added to the project. Click on a feature and you'll be sent directly to the corresponding Studio tab. Notice the two icons on the upper right corner. One is used to refresh the list and the other is used to export the operations you develop in the IDE into your Studio project.

Nuxeo Shell

Nuxeo Shell is, as its name suggest, a shell. You can use it to log in to Nuxeo and realize different actions. You can use it to connect and do maintenance work on any remote Nuxeo server.

To switch to the Nuxeo perspective, click on **Window** > **Open Perspective** and choose **Nuxeo**.
Of course, you can always use the default Java perspective and add the Nuxeo IDE views you want.

Related pages

- Running your Projects on the Server
- Browsing a Studio Project Content

Managing Project Dependencies

All the dependencies of your projects are managed using Nuxeo IDE.

Nuxeo SDK Classpath

A Nuxeo SDK is necessary to create a Nuxeo project. A Nuxeo SDK is a Nuxeo-based Tomcat distribution (5.5 or more recent) of Nuxeo on which you will deploy your customizations. It provides the classpath for your projects.

You can install several Nuxeo SDKs for your different projects. For instance, you can use a Nuxeo CAP SDK for one project, a Nuxeo CMF SDK for another, etc. However you can use only one SDK at a time in an Eclipse workspace.

Nuxeo SDK are installed from the Eclipse preferences. This is also where you select the Nuxeo SDK to use in the workspace.

To install a Nuxeo SDK:

1. UNzip the downloaded Nuxeo Platform distribution.
2. In the **Nuxeo SDK** preference page (Eclipse Menu -> Window -> Preferences -> Nuxeo -> Nuxeo SDK), click on **Add...**.
3. Select the directory where you unzipped your Nuxeo distribution.
   Be sure to select the right directory: the one containing the `bin` and `nxserver` directories.
4. **Check** the added Nuxeo SDK and click on the **OK** button to finish.
   Checking the box at the left of the SDK is required to mark it as the current Nuxeo SDK used in the Eclipse Workspace.

User Libraries

As you develop your new features, you may want to use external libraries that are not provided by the Nuxeo SDK. In that case you can add new JARs to the project's classpath by declaring custom **User Libraries**. You can easily do so by using the Eclipse preferences:

1. Open Eclipse preferences and go to **Nuxeo**.
2. Click on **User Libraries**.
3. Click on the **Add** button and select the library's JAR on your computer.
   The JAR is added in the libraries list. The library's metadata should be automatically filled in. If not, fill them in, as they they are required to deploy your project correctly.
4. Click on the **Apply** button.
   The library's classes are now available in Eclipse and you can use them.

As we will see in the next section, Nuxeo IDE provides a mean to automatically update the POM with the project dependencies. This also includes any user library you declared.

⚠ You **must** fill the user library metadata (the Maven GAV information) otherwise synchronizing the POM will ignore your additional library.

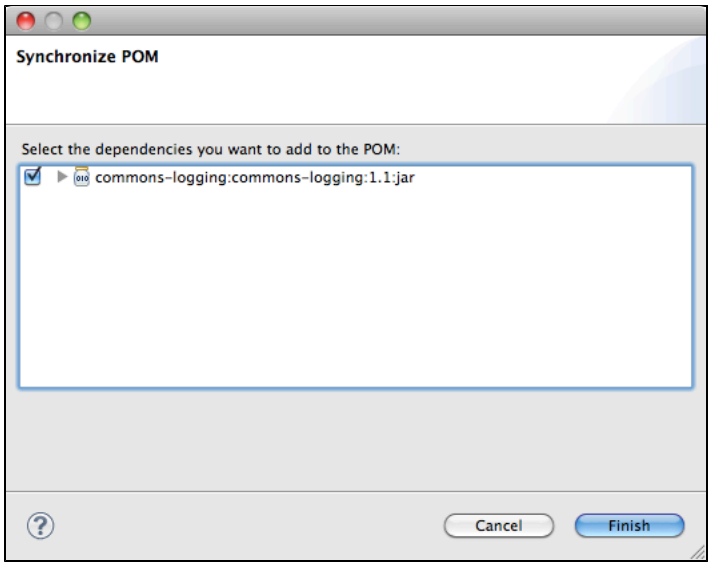
Synchronizing POMs

As you develop your customizations, you add new dependencies. All the dependencies are listed in the `pom.xml` file. Usually, you need to update your `pom.xml` manually and might forget a dependency, which will make your project build fail.

Nuxeo IDE provides a way to update your `pom.xml` file with the new dependencies. Before deploying your project, you can synchronize your project's `pom.xml` file. Then, Nuxeo IDE scans your project with all undeclared dependencies and adds them to the POM.

To synchronize your project's POM:

1. Right-click on the `pom.xml` file file then click on **Nuxeo** > **Synchronize POM**
2. You will be prompted to select the artifacts you want to add to the `pom.xml` file.
   The missing artifacts are discovered by the Nuxeo IDE by scanning the sources of your project, and finding any class that you reference and that is not yet in the POM.



Select the dependencies you want to add to the POM:

☑ commons-logging:commons-logging:1.1:jar

3. Click **Finish**.
   Then look again at the `pom.xml` file. Now you have your dependency there.

Attaching Sources for Project Dependencies

You may surely want at some point to be able to browse the sources of the dependencies used by your project. This is useful when you need to understand the code or if you want to put breakpoints in the code outside your project. As we've seen the project dependencies are either the ones provided by the current Nuxeo SDK, or the ones you explicitly added as user libraries.

In order to do this, you must download the sources (if they are not yet already present in the Nuxeo SDK) and attach it to the corresponding JAR from the project classpath.
You can do that by using the **Nuxeo SDK** classpath properties dialog as follows:

1. Right-click on the **Nuxeo SDK** entry in your project tree and then click on **Properties** to open the Nuxeo SDK classpath properties dialog.
2. Select the JAR for which you want to attach sources. You can use the filter text box to quickly find the JAR by name.
3. Click on **Download**.
   Sources are downloaded from Nuxeo Maven repositories. This means, the dependencies must have a correct Maven GAV information (in case the JAR is a user library).

   For now, only Nuxeo Maven repositories are scanned for sources.

   In the case the dependency is not located in a Maven repository known by Nuxeo IDE, you can add sources by simply copying the corresponding source JAR from Maven to `NuxeoSDK/sdk/sources` directory (where "NuxeoSDK" is the install directory of the Nuxeo SDK). Then you need to reload the SDK by removing and adding it again in Nuxeo SDK preferences page.

In this section

Running your Projects on the Server

Installing Nuxeo IDE adds a new perspective to Eclipse, called Nuxeo. This perspective is composed of three views:

- the Studio view,
- the Shell view,
- the server view.

The server view enables you to deploy your changes and control the runtime.

Configuring the Server Launcher

The server launcher can be configured from the Eclipse preferences.

1. In the Eclipse preferences, click on **Nuxeo** > **Run/Debug**.
2. Change the parameters you need (see below).
3. Click on the **Apply** or **OK** button.

Run/Debug parameters

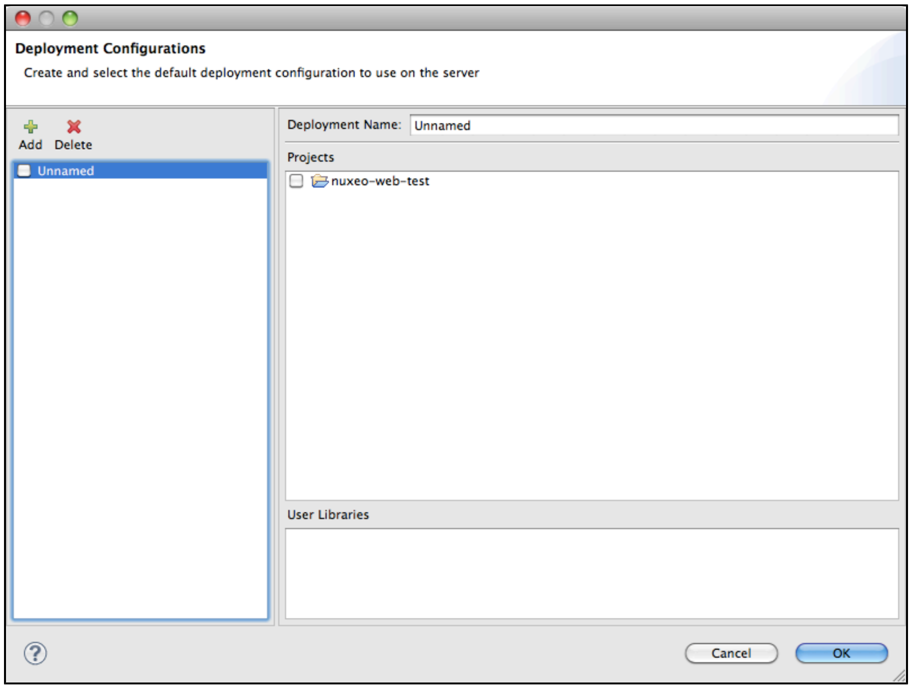
Field	Description
VM Arguments	System properties of the JVM
Debug port	Port used to debug
Suspend server	Indicate if you want the server to wait for the debugger to be attached to it before actually starting.

Deployment Profiles

As you develop new features, you want to be able to easily test them. To do so, you need to deploy them on the Nuxeo SDK. However, you often don't want to deploy all the features or projects you are working on. You may want to deploy only some of them. To make this easy, Nuxeo IDE includes a deployment profile configuration. A deployment profile is a set of Nuxeo IDE projects that will be deployed at the same time.

To create a new deployment profile:

1. Go into the **Nuxeo Server** view (on the bottom right).
2. Click on the button in the view toolbar. The button tooltip is "Select projects to deploy on server".
3. Click on **Add**.


Deployment Configurations

Create and select the default deployment configuration to use on the server

Deployment Name: Unnamed

4. Enter a name (in the right panel) for your deployment profile. Example: "My Projects".
5. In the left panel click on the newly created profile.
6. In the right panel, check the projects that will be deployed with this profile.
7. If you want the new profile to be the default deployment profile, check it.
8. Click on the **OK** button.

Launching the Server

The server can be started either in **Run** or **Debug** mode. In both cases, when launched the server will automatically deploy (at the end of the start process) all the projects and user libraries you configured in the current deployment profile.

You can see in the server console messages that are detailing which projects and user libraries were deployed.

Remote Debugging

If you need to check your developments or watch Nuxeo source code behavior this is possible throw the remote debugging.

With Eclipse and Nuxeo IDE

First configure the port the JVM will listen for remote debug:

1. Go to Preferences > Nuxeo > Run/Debug
2. Set a value for the debug port (choose one free 🙂
   Then start your Nuxeo Server from the Nuxeo perspective as debug and connect Eclipse to the JVM:
3. Go to Run > Debug configurations... > Remote Java Application
4. Create a new debug configurations
5. Type the host name as localhost and the port as the one set below

You can set breakpoints and have fun.

With Other IDE or with a Server Not Local

First configure the port the JVM will listen for remote debug:

1. Open the nuxeo.conf file, by default is here `$TOMCAT_HOME/bin/nuxeo.conf or /etc/nuxeo.conf`.
2. Uncomment the following line:

```
#JAVA_OPTS=$JAVA_OPTS -Xdebug
-Xrunjdwp:transport=dt_socket,address=8787,server=y,suspend=n
```

3. You can change the port number here.
4. Start your server.
5. Add source code you want to watch into your IDE.
6. Create a remote debug connection with the port set.

You can set breakpoints and have fun.

Hot Reload

If you are modifying the sources of a project while the server is running, you may reload your changes on the server by simply clicking on the **refresh** button (in the Server View).
A message will appear in the server console detailing the projects that were reloaded.

Be sure that the project you modify is selected in the current deployment profile, otherwise it will not be reloaded on the server.

You can modify the current deployment profile at any time by removing or adding projects. The next time you will click on the **refresh** button, the old profile will be uninstalled form the server and the new one will be installed. Projects that are common to both old and new deployment profiles will be reloaded.

For more information about Nuxeo hot reload support, see Supporting Hot Reload.

A bug affects hot reload when using Nuxeo 6.0 SDK (fixed in Nuxeo 6.0 HF01). Refer to the following JIRA ticket for a workaround if you have not installed any hotfix yet: fix dev bundles not taken in account

In this section

Working with Studio Projects

Nuxeo IDE enables developers to extend the Nuxeo application with new features, but they also use Nuxeo Studio in parallel.
Nuxeo IDE provides a way to use each customization tool's content into the other one. For instance, use operations created in Eclipse using Nuxeo IDE from Nuxeo Studio, or exploit the document types defined in Nuxeo Studio in your code in Eclipse.

Available how-tos:

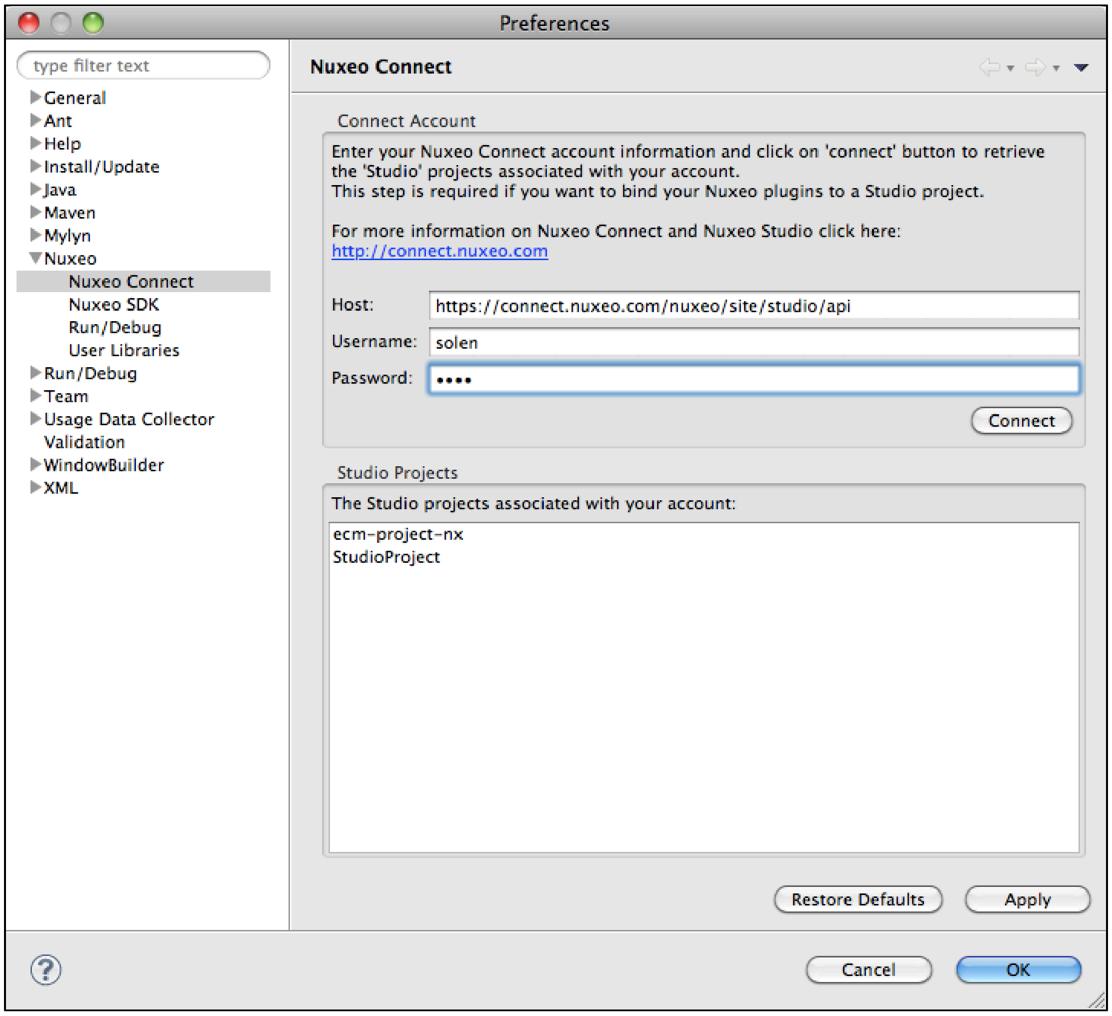
- Configuring a Nuxeo Connect Account — To enable Nuxeo IDE to leverage your Nuxeo Studio customization, you need to configure your Nuxeo Connect account so Nuxeo IDE can connect to Studio.
- Browsing a Studio Project Content — The Studio view enables you to browse the list of your Studio customizations. You can see all the document types, schemas, content views, automation chains, etc, that you have configured in Nuxeo Studio.
- Binding Studio Projects to an Nuxeo IDE Project — The whole purpose of enabling to work with Nuxeo Studio is to be able to use elements configured in Studio from Nuxeo IDE, and to benefit from Eclipse key features such as code completion. As you may have several projects (both in Nuxeo Studio and in Nuxeo IDE), you need to indicate which Studio project the Nuxeo IDE project should be bound to.
- Uploading Custom Operations in Nuxeo Studio — Nuxeo IDE enables you to code new features, that can be complementary to your Studio customizations. Typically, you can use Nuxeo IDE to develop new Automation operations, that want to leverage in the Nuxeo Studio automation chain builder. Just like you can browse your Nuxeo Studio project from Nuxeo IDE, you can also use your operation in Nuxeo Studio.
- Deploying a Project with Studio Dependencies — For now, when deploying your projects on the server (using the a deployment profile) the Studio projects are not automatically deployed. In order to deploy both your Nuxeo IDE projects and their Studio project dependencies, your need to:
- Creating a Custom Marketplace Package — Nuxeo IDE enables you to create Marketplace package for your Nuxeo project. A Marketplace Package is the easiest way to distribute a plugin, as it contains all the bundles, libraries and runtime properties that would be required to make your new plugin work, all in one single ZIP file.
- Settings Eclipse Development Templates

Configuring a Nuxeo Connect Account

To enable Nuxeo IDE to leverage your Nuxeo Studio customization, you need to configure your Nuxeo Connect account so Nuxeo IDE can connect to Studio.

To configure a Nuxeo Connect account:

1. Open Eclipse preferences and go to **Nuxeo** > **Nuxeo Connect**.
   The Host name is already filled in with Nuxeo Connect address.
2. Type your Connect login and password and click on the **Connect** button.
   The lists of Studio projects associated to your account is displayed.



Enter your Nuxeo Connect account information and click on 'connect' button to retrieve the 'Studio' projects associated with your account.
This step is required if you want to bind your Nuxeo plugins to a Studio project.

For more information on Nuxeo Connect and Nuxeo Studio click here:
http://connect.nuxeo.com

Host: https://connect.nuxeo.com/nuxeo/site/studio/api
Username: solen
Password: ••••

The Studio projects associated with your account:
ecm-project-nx
StudioProject

3. Click on **OK**.
   The Preferences window closes.
   You can now go to the Studio view to browse your Studio customizations.

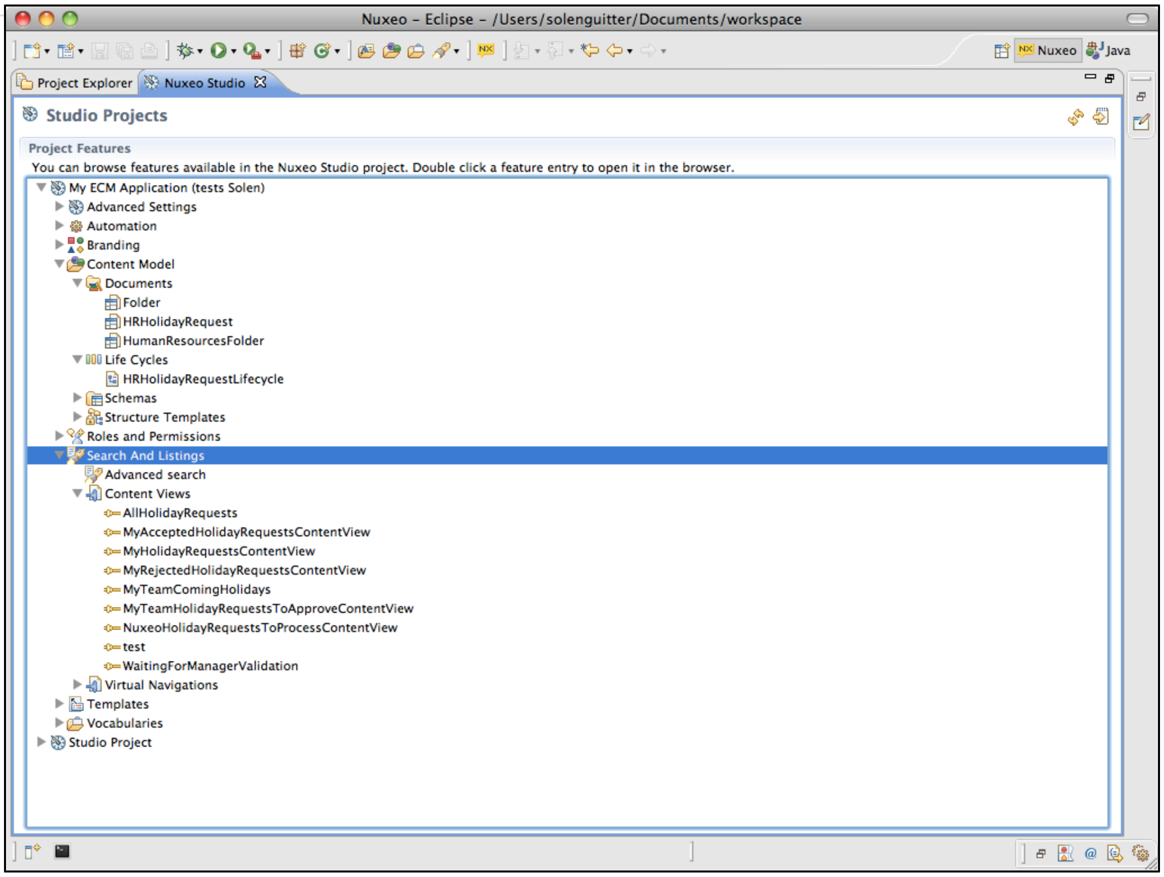
Related pages

- Deploying a Project with Studio Dependencies
- Uploading Custom Operations in Nuxeo Studio
- Binding Studio Projects to an Nuxeo IDE Project
- Configuring a Nuxeo Connect Account
- Browsing a Studio Project Content

Browsing a Studio Project Content

The Studio view enables you to browse the list of your Studio customizations. You can see all the document types, schemas, content views, automation chains, etc, that you have configured in Nuxeo Studio.

Double-clicking on any of these customizations opens Nuxeo Studio in a browser. It is not possible to edit Studio customizations from Nuxeo IDE.


Nuxeo – Eclipse – /Users/solenguitter/Documents/workspace

Studio Projects

Project Features
You can browse features available in the Nuxeo Studio project. Double click a feature entry to open it in the browser.

```
My ECM Application (tests Solen)
  Advanced Settings
  Automation
  Branding
  Content Model
    Documents
      Folder
      HRHolidayRequest
      HumanResourcesFolder
    Life Cycles
      HRHolidayRequestLifecycle
    Schemas
    Structure Templates
  Roles and Permissions
  Search And Listings
    Advanced search
    Content Views
      AllHolidayRequests
      MyAcceptedHolidayRequestsContentView
      MyHolidayRequestsContentView
      MyRejectedHolidayRequestsContentView
      MyTeamComingHolidays
      MyTeamHolidayRequestsToApproveContentView
      NuxeoHolidayRequestsToProcessContentView
      test
      WaitingForManagerValidation
    Virtual Navigations
  Templates
  Vocabularies
Studio Project
```

Related pages

- Deploying a Project with Studio Dependencies
- Uploading Custom Operations in Nuxeo Studio
- Binding Studio Projects to an Nuxeo IDE Project
- Configuring a Nuxeo Connect Account
- Browsing a Studio Project Content

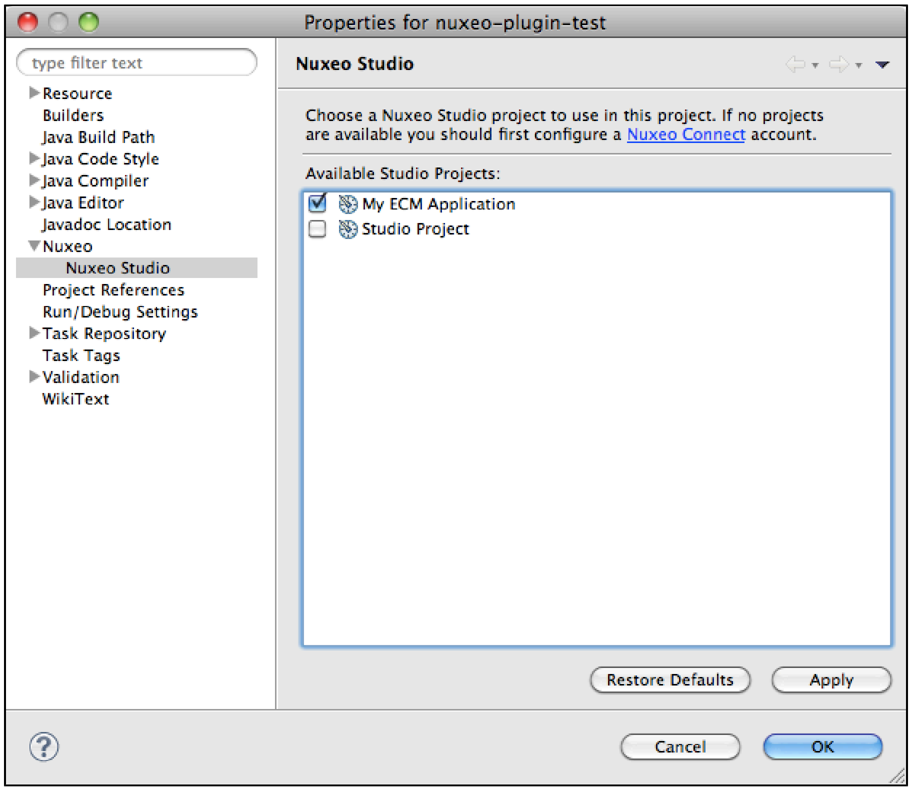
Binding Studio Projects to an Nuxeo IDE Project

The whole purpose of enabling to work with Nuxeo Studio is to be able to use elements configured in Studio from Nuxeo IDE, and to benefit from Eclipse key features such as code completion. As you may have several projects (both in Nuxeo Studio and in Nuxeo IDE), you need to indicate which Studio project the Nuxeo IDE project should be bound to.

Binding Nuxeo IDE and Studio Projects

To bind a Studio project and a Nuxeo IDE project:

1. In the Project explorer view, right-click on the project name and click on **Properties**.
2. Click on **Nuxeo** > **Nuxeo Studio**.
3. Check the project you want to use.



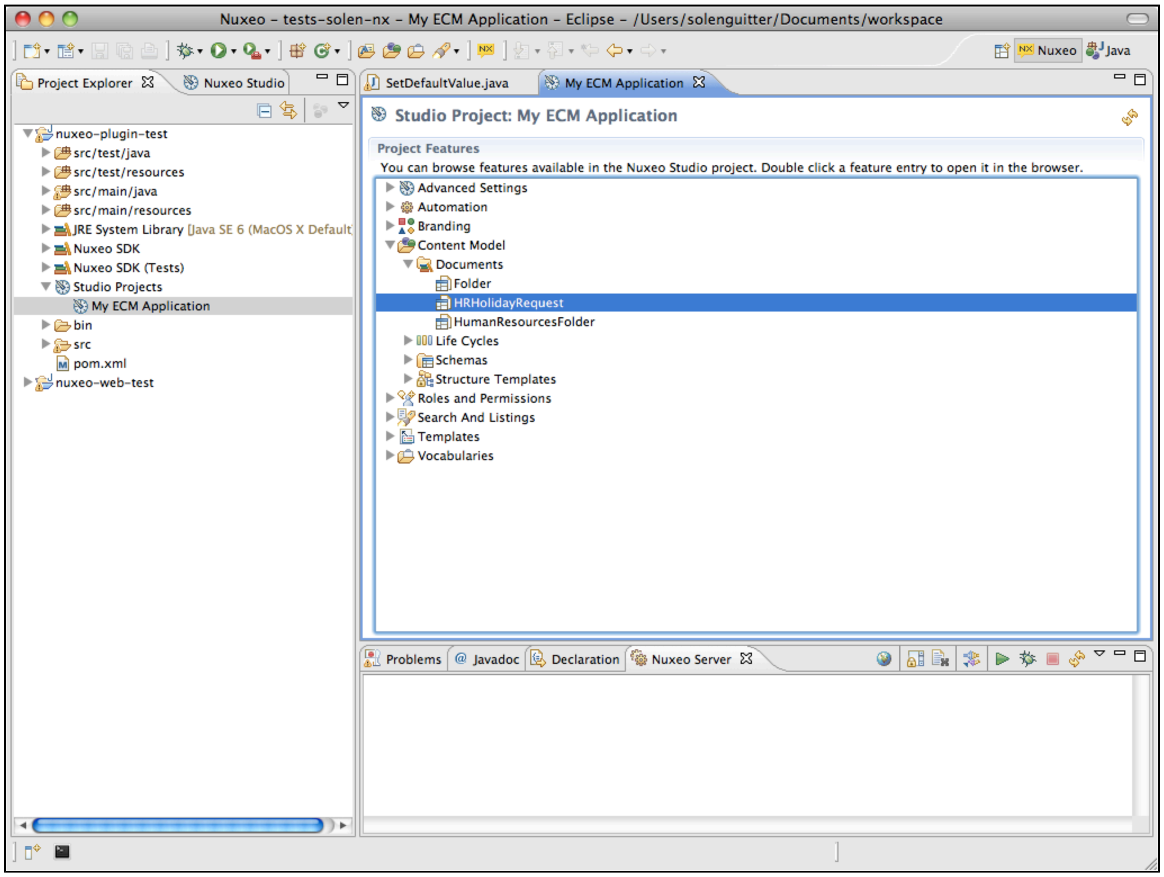
Choose a Nuxeo Studio project to use in this project. If no projects are available you should first configure a Nuxeo Connect account.

Available Studio Projects:

[✓] My ECM Application
[ ] Studio Project

4. Click on the **Apply** or **OK** button.

A new **Studio Projects** item is available in the project tree. This item contains an item for each of the Studio project you bound to your project. This way, you can browse the content of each individual Studio project dependency by opening it in the editor area (i.e. double-click on the Studio project item). Double-clicking on the Studio items will redirect you to Nuxeo Studio in your browser.



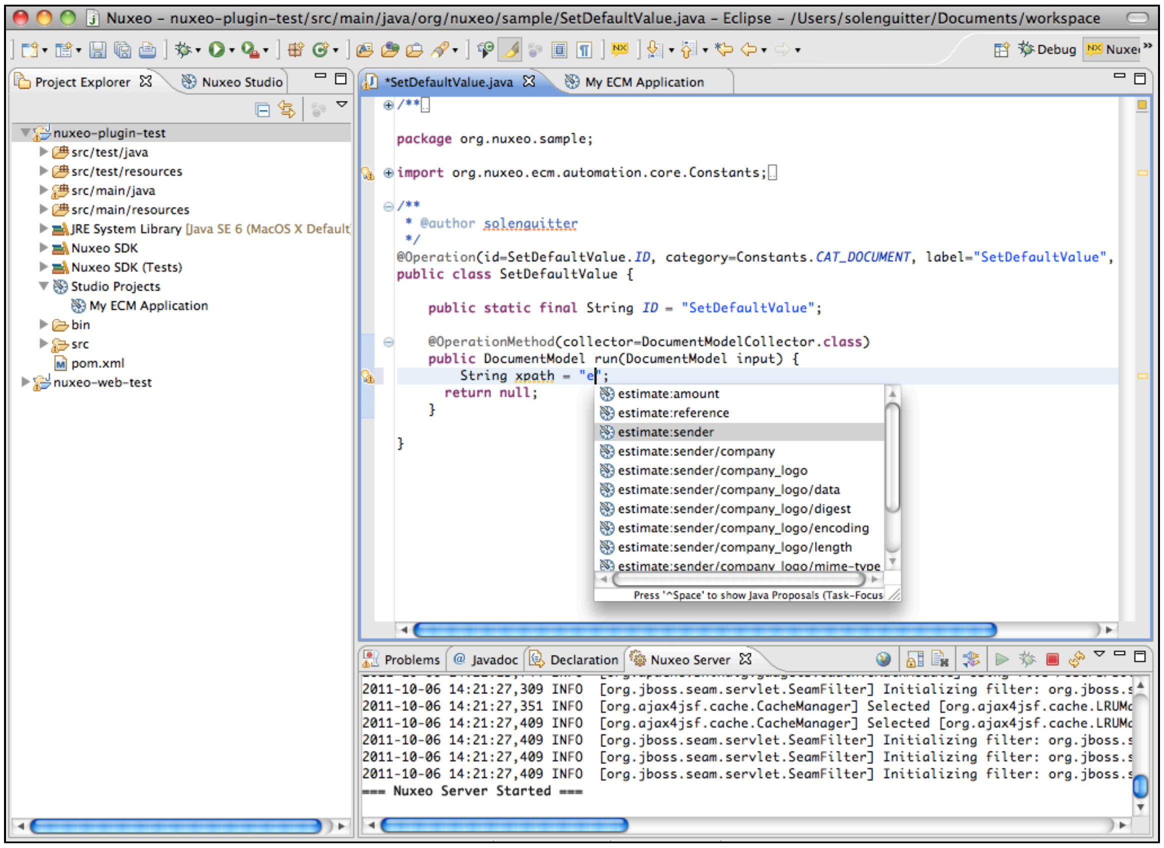
You can browse features available in the Nuxeo Studio project. Double click a feature entry to open it in the browser.

Also, you can export to one of the studio projects you bound the operations that are present in your current project: right-click on the Studio

project and then click on **Export Operations**.
To export operations from multiple projects use the global export operation feature as explained in Uploading Custom Operations in Nuxeo Studio.

XPath Autocompletion

The code autocompletion now takes your content model configuration into account.
So, anywhere you want to use a document property XPath in your code you can use the Eclipse autocompletion feature to cycle through possible completion - one of them being provided by the Studio projects you bound to the project.



Nuxeo - nuxeo-plugin-test/src/main/java/org/nuxeo/sample/SetDefaultValue.java - Eclipse - /Users/solenguitter/Documents/workspace

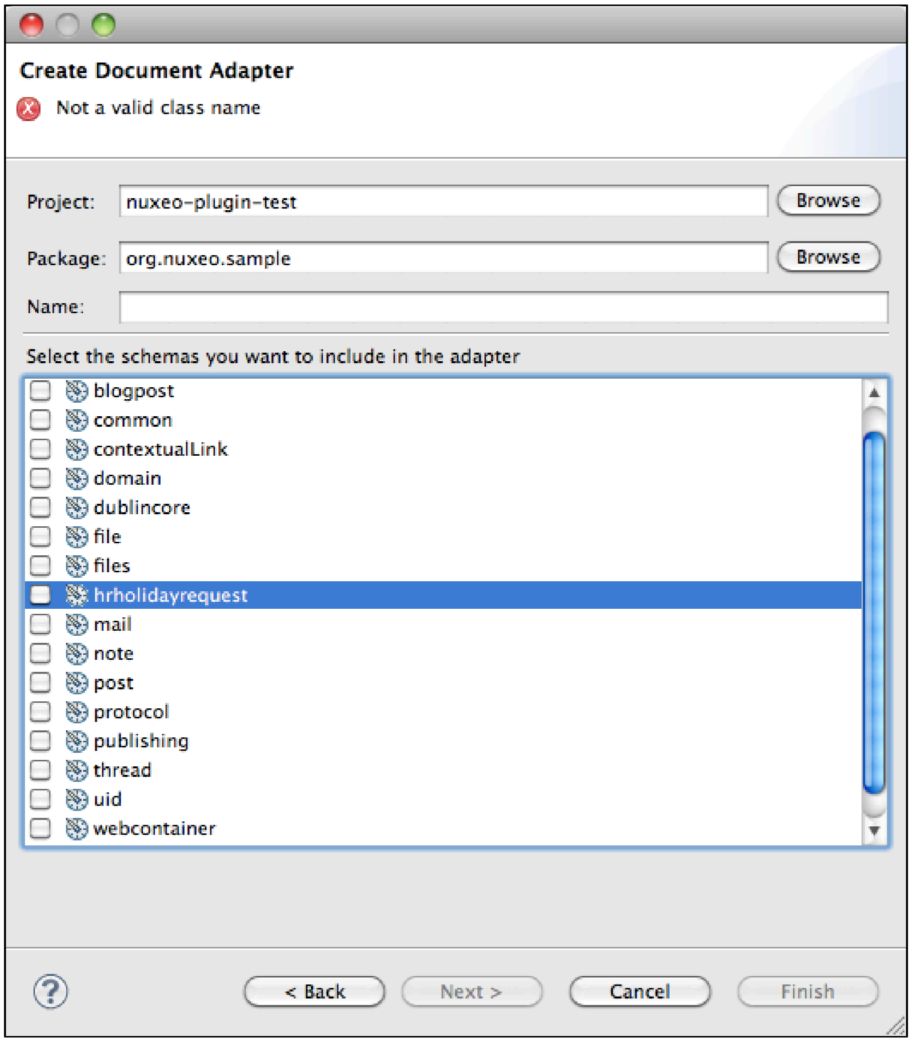
```
/**

package org.nuxeo.sample;

import org.nuxeo.ecm.automation.core.Constants;

/**
 * @author solenguitter
 */
@Operation(id=SetDefaultValue.ID, category=Constants.CAT_DOCUMENT, label="SetDefaultValue",
public class SetDefaultValue {

    public static final String ID = "SetDefaultValue";

    @OperationMethod(collector=DocumentModelCollector.class)
    public DocumentModel run(DocumentModel input) {
        String xpath = "e";
      return null;
    }

}
```

estimate:amount
estimate:reference
estimate:sender
estimate:sender/company
estimate:sender/company_logo
estimate:sender/company_logo/data
estimate:sender/company_logo/digest
estimate:sender/company_logo/encoding
estimate:sender/company_logo/length
estimate:sender/company_logo/mime-type
Press '^Space' to show Java Proposals (Task-Focus
Problems
Javadoc
Declaration
Nuxeo Server

```
2011-10-06 14:21:27,309 INFO  [org.jboss.seam.servlet.SeamFilter] Initializing filter: org.jboss.s
2011-10-06 14:21:27,351 INFO  [org.ajax4jsf.cache.CacheManager] Selected [org.ajax4jsf.cache.LRUMc
2011-10-06 14:21:27,409 INFO  [org.ajax4jsf.cache.CacheManager] Selected [org.ajax4jsf.cache.LRUMc
2011-10-06 14:21:27,409 INFO  [org.jboss.seam.servlet.SeamFilter] Initializing filter: org.jboss.s
2011-10-06 14:21:27,409 INFO  [org.jboss.seam.servlet.SeamFilter] Initializing filter: org.jboss.s
2011-10-06 14:21:27,409 INFO  [org.jboss.seam.servlet.SeamFilter] Initializing filter: org.jboss.s
=== Nuxeo Server Started ===
```

Document Adapters

You can also create Document adapters for your Studio document types.



Create Document Adapter

Not a valid class name

Project: nuxeo-plugin-test

Package: org.nuxeo.sample

Name:

Select the schemas you want to include in the adapter

- blogpost
- common
- contextualLink
- domain
- dublincore
- file
- files
- hrholidayrequest
- mail
- note
- post
- protocol
- publishing
- thread
- uid
- webcontainer

Related pages

- Deploying a Project with Studio Dependencies
- Uploading Custom Operations in Nuxeo Studio
- Binding Studio Projects to an Nuxeo IDE Project
- Configuring a Nuxeo Connect Account
- Browsing a Studio Project Content

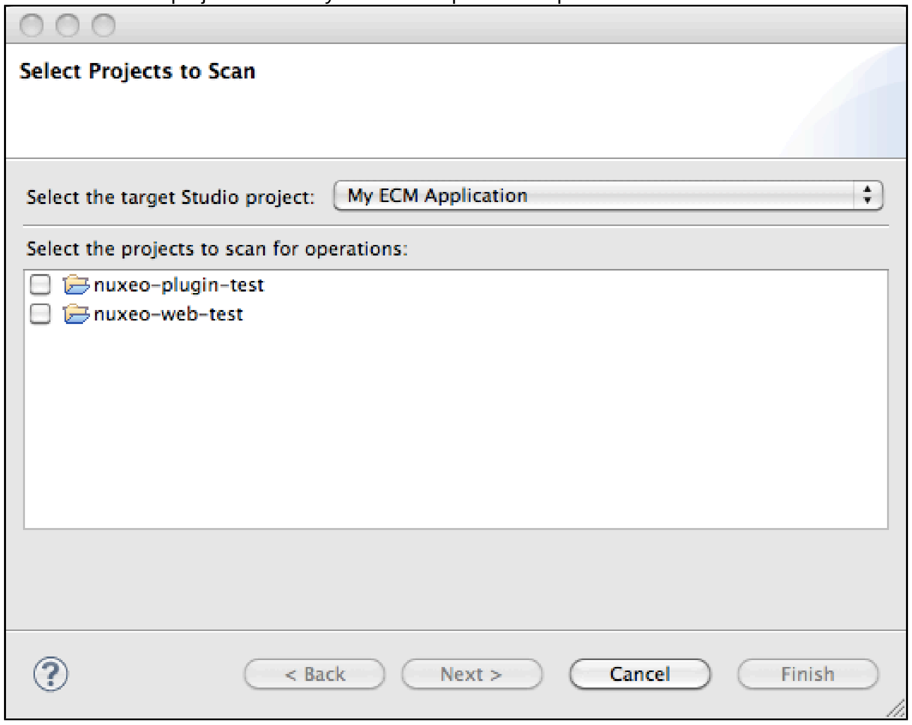
Uploading Custom Operations in Nuxeo Studio

Nuxeo IDE enables you to code new features, that can be complementary to your Studio customizations. Typically, you can use Nuxeo IDE to develop new Automation operations, that want to leverage in the Nuxeo Studio automation chain builder. Just like you can browse your Nuxeo Studio project from Nuxeo IDE, you can also use your operation in Nuxeo Studio.

To upload your operations in Nuxeo Studio:

1. If you haven't yet, configure a Nuxeo Connect account.

2. From the Studio view, click on the icon (Export operations).
   A window called **Select projects to scan** opens.
3. Select the Studio project to which you want to upload the operation.



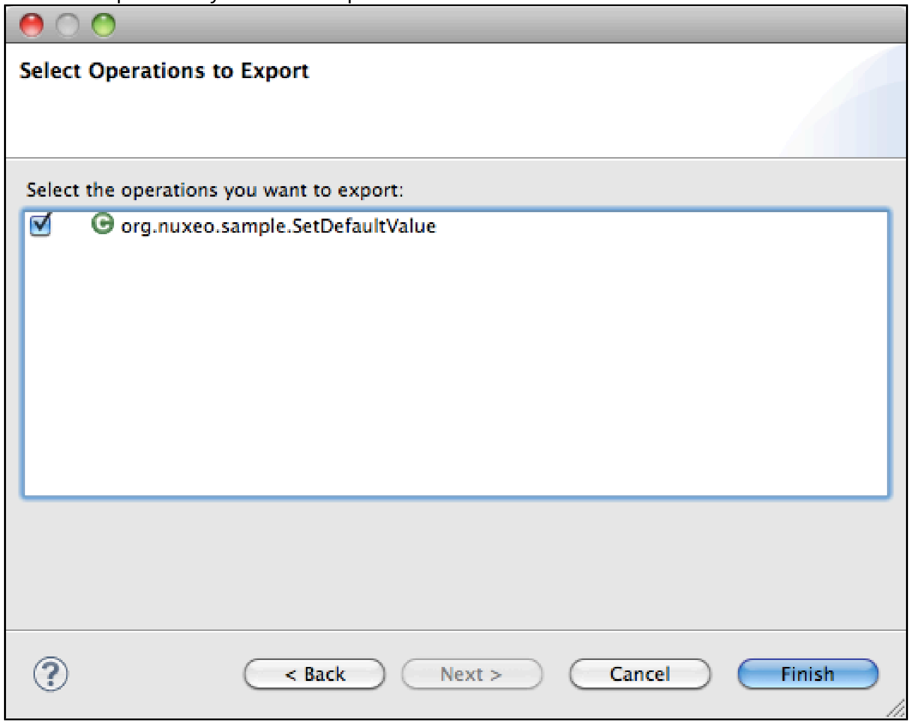
Select Projects to Scan

Select the target Studio project: My ECM Application

Select the projects to scan for operations:

- [ ] nuxeo-plugin-test
- [ ] nuxeo-web-test

4. Select the Nuxeo IDE project that holds the operations to export.
5. Click on the **Next** button.
6. Select the operations you want to export to Nuxeo Studio and click on the **Finish** button.

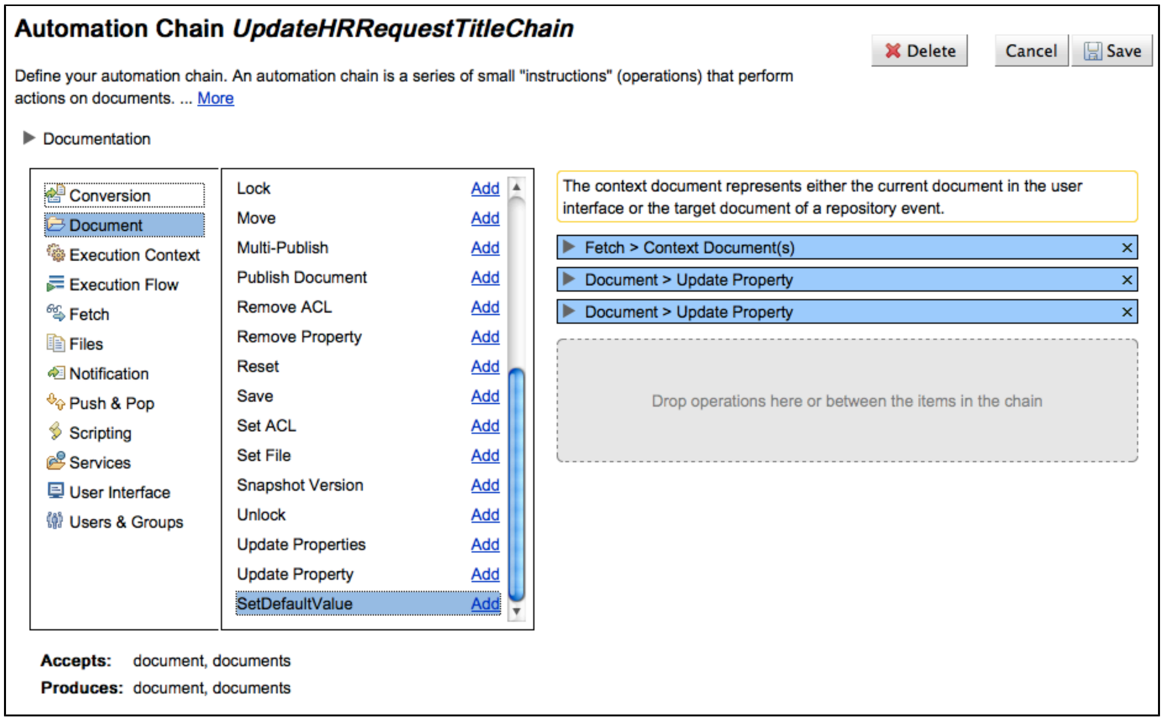


Select Operations to Export

Select the operations you want to export:

- [x] org.nuxeo.sample.SetDefaultValue

You can now log in to Nuxeo Studio and use your operations in automation chains.


Automation Chain *UpdateHRRequestTitleChain*

Define your automation chain. An automation chain is a series of small "instructions" (operations) that perform actions on documents. ... More

The context document represents either the current document in the user interface or the target document of a repository event.

Fetch > Context Document(s)
Document > Update Property
Document > Update Property

Drop operations here or between the items in the chain

Accepts: document, documents
Produces: document, documents

Related pages

- Deploying a Project with Studio Dependencies
- Uploading Custom Operations in Nuxeo Studio
- Binding Studio Projects to an Nuxeo IDE Project
- Configuring a Nuxeo Connect Account
- Browsing a Studio Project Content

Deploying a Project with Studio Dependencies

For now, when deploying your projects on the server (using the a deployment profile) the Studio projects are not automatically deployed. In order to deploy both your Nuxeo IDE projects and their Studio project dependencies, your need to:

1. Start your SDK server.
2. Deploy the Studio projects using the regular **Admin Center** web interface.
3. Reload the Nuxeo IDE projects on the server by clicking on the **refresh** button in the server view.

Of course, if you haven't modified your Nuxeo IDE projects, you don't need to reload the Nuxeo IDE project.

Related pages

- Deploying a Project with Studio Dependencies
- Uploading Custom Operations in Nuxeo Studio
- Binding Studio Projects to an Nuxeo IDE Project
- Configuring a Nuxeo Connect Account
- Browsing a Studio Project Content

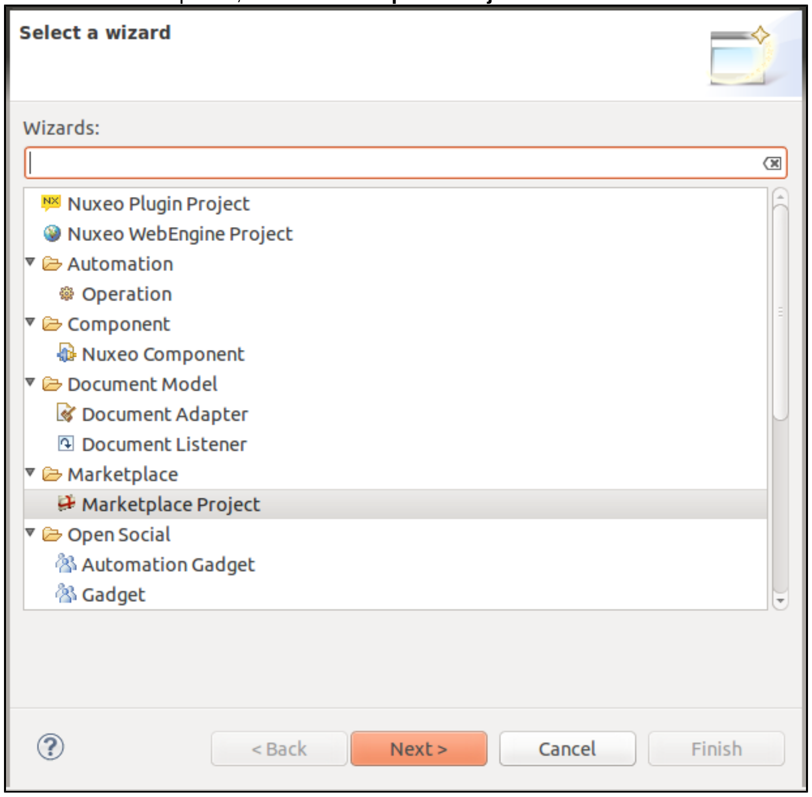
Creating a Custom Marketplace Package

Nuxeo IDE enables you to create Marketplace package for your Nuxeo project. A Marketplace Package is the easiest way to distribute a plugin, as it contains all the bundles, libraries and runtime properties that would be required to make your new plugin work, all in one single ZIP file.

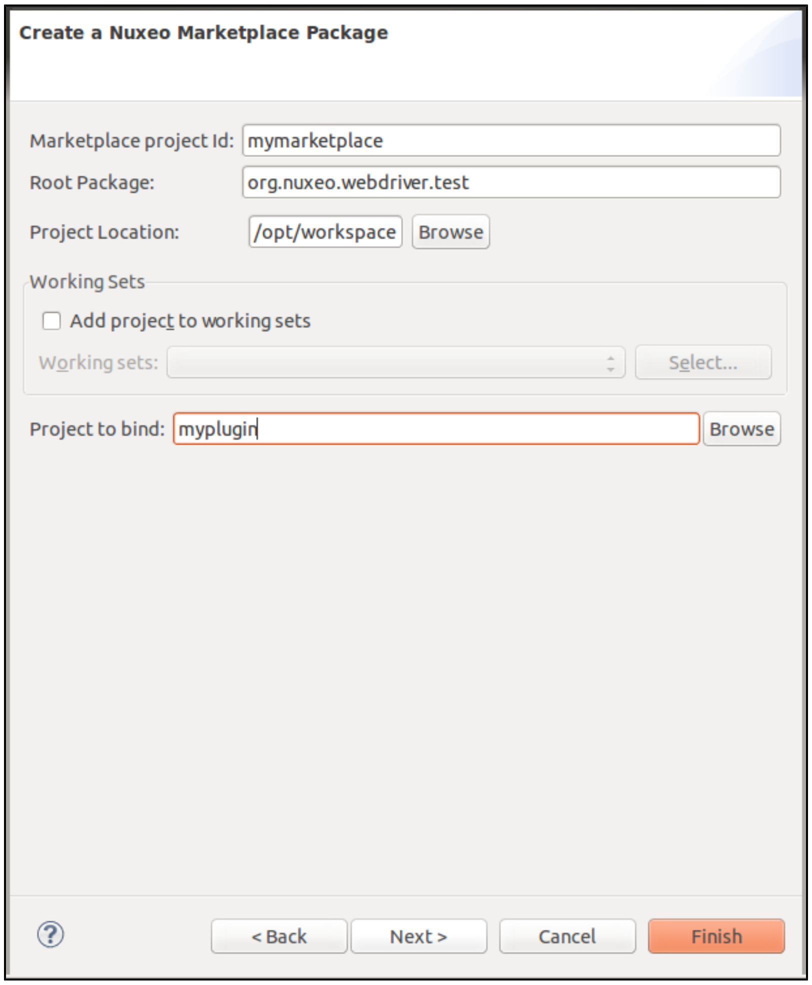
The whole Marketplace package structure and metadata are detailed on the page Creating Marketplace Packages.

Please note that this functionality may not be fully compliant on Nuxeo IDE versions 1.2.1 to 1.2.3 included when using a Nuxeo Platform 5.8 SDK. For users using this configuration, an upgrade to Nuxeo IDE 1.2.4 is recommended. A workaround can also be found in the following JIRA ticket : Allow marketplace package generation with Nuxeo Platform 5.8 / Maven 3

To create your marketplace package in Nuxeo IDE:

1. If you haven't yet, create a new Nuxeo plugin in your Eclipse.
2. From the Wizard options, choose **Marketplace Project**.


3. On the Marketplace creation form, define:
   - the id for your Marketplace package,
   - a namespace package for your Java test classes,
   - project binding to attach your Nuxeo plugin to the Marketplace package.



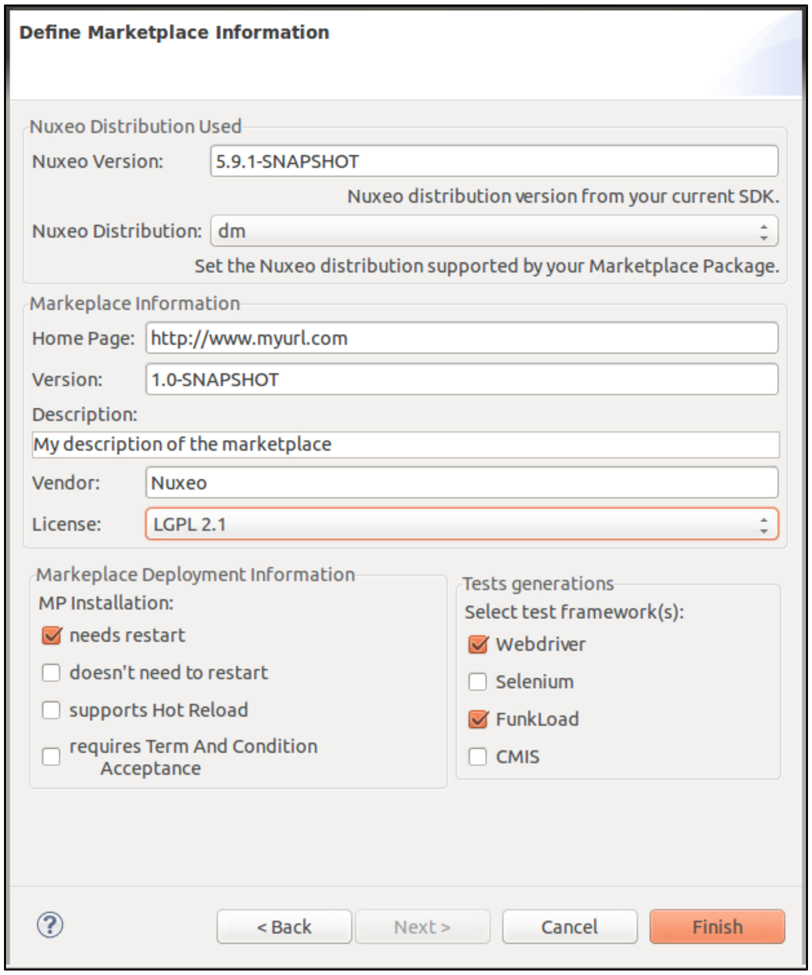
Create a Nuxeo Marketplace Package

Marketplace project Id: mymarketplace
Root Package: org.nuxeo.webdriver.test
Project Location: /opt/workspace

Working Sets
☐ Add project to working sets

Project to bind: myplugin

4. On the Marketplace information form, specify the Marketplace package metadata such as:
   - the distribution type (`cap`, `dm`, `dam`...),
   - the Marketplace package version, licence, description,
   - the Marketplace package deployment options (need to restart the instance...)
   - the functionnal tests deployment (WebDriver, Selenium and FunkLoad).


Define Marketplace Information

Nuxeo Distribution Used

Nuxeo Version: 5.9.1-SNAPSHOT

Nuxeo distribution version from your current SDK.

Nuxeo Distribution: dm

Set the Nuxeo distribution supported by your Marketplace Package.

Markeplace Information

Home Page: http://www.myurl.com

Version: 1.0-SNAPSHOT

Description:

My description of the marketplace

Vendor: Nuxeo

License: LGPL 2.1

Markeplace Deployment Information

MP Installation:

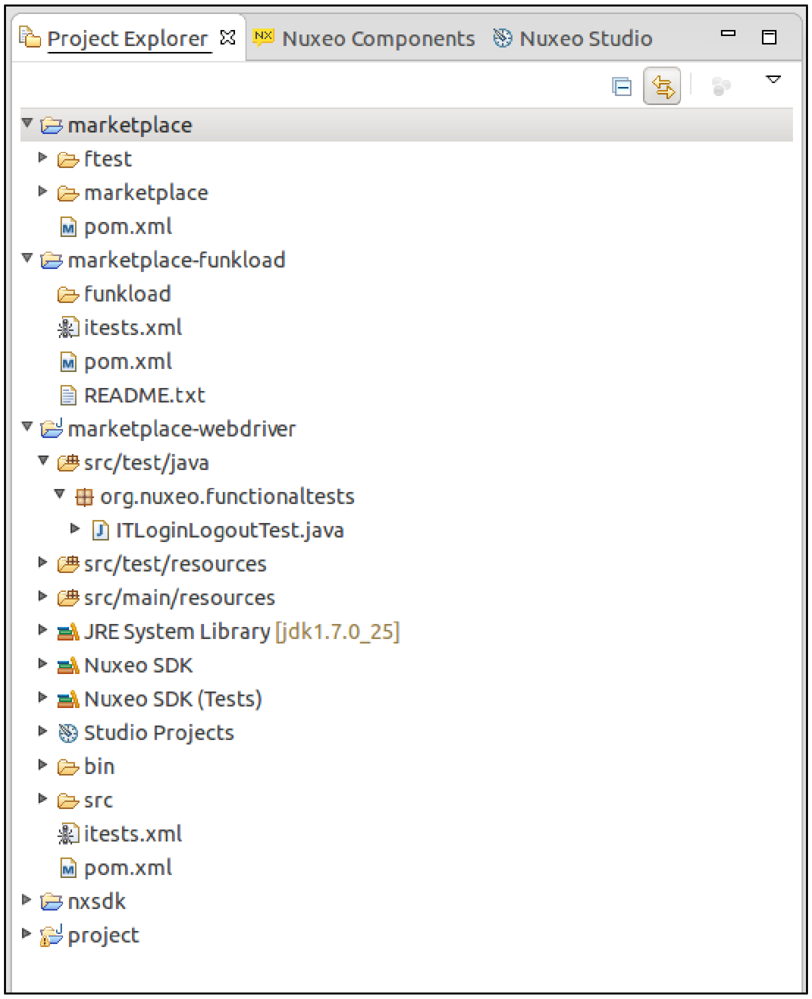
- [✓] needs restart
- [ ] doesn't need to restart
- [ ] supports Hot Reload
- [ ] requires Term And Condition Acceptance

Tests generations

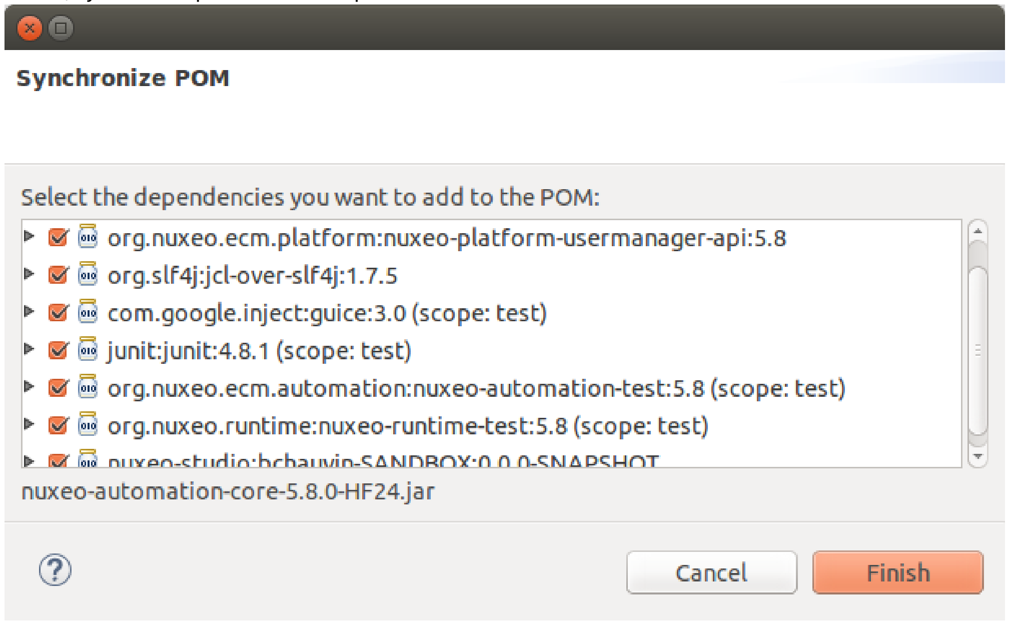
Select test framework(s):

- [✓] Webdriver
- [ ] Selenium
- [✓] FunkLoad
- [ ] CMIS

5. Click on the **Finish** button to create your Marketplace package project.
   Here is what you get:
   - One project is created for whole Marketplace package (link) container;
   - One project for each test framework selected (included WebDriver tests in Java).



6. To generate the actual marketplace package:
   a. Make sure the bound project's pom.xml file contains all necessary dependencies. You can right click on your project and choose Nuxeo, synchronise pom to ease the process.



Synchronize POM

Select the dependencies you want to add to the POM:

- [✓] org.nuxeo.ecm.platform:nuxeo-platform-usermanager-api:5.8
- [✓] org.slf4j:jcl-over-slf4j:1.7.5
- [✓] com.google.inject:guice:3.0 (scope: test)
- [✓] junit:junit:4.8.1 (scope: test)
- [✓] org.nuxeo.ecm.automation:nuxeo-automation-test:5.8 (scope: test)
- [✓] org.nuxeo.runtime:nuxeo-runtime-test:5.8 (scope: test)
- [✓] nuxeo-studio:bchauvin-SANDBOX:0.0.0-SNAPSHOT

nuxeo-automation-core-5.8.0-HF24.jar

   b. From the bound project, use maven to compile and install the project's jar file into your local repository (*mvn clean install* command)
   c. From the eclipse marketplace project, use maven to generate the marketplace package (*mvn clean package* command)

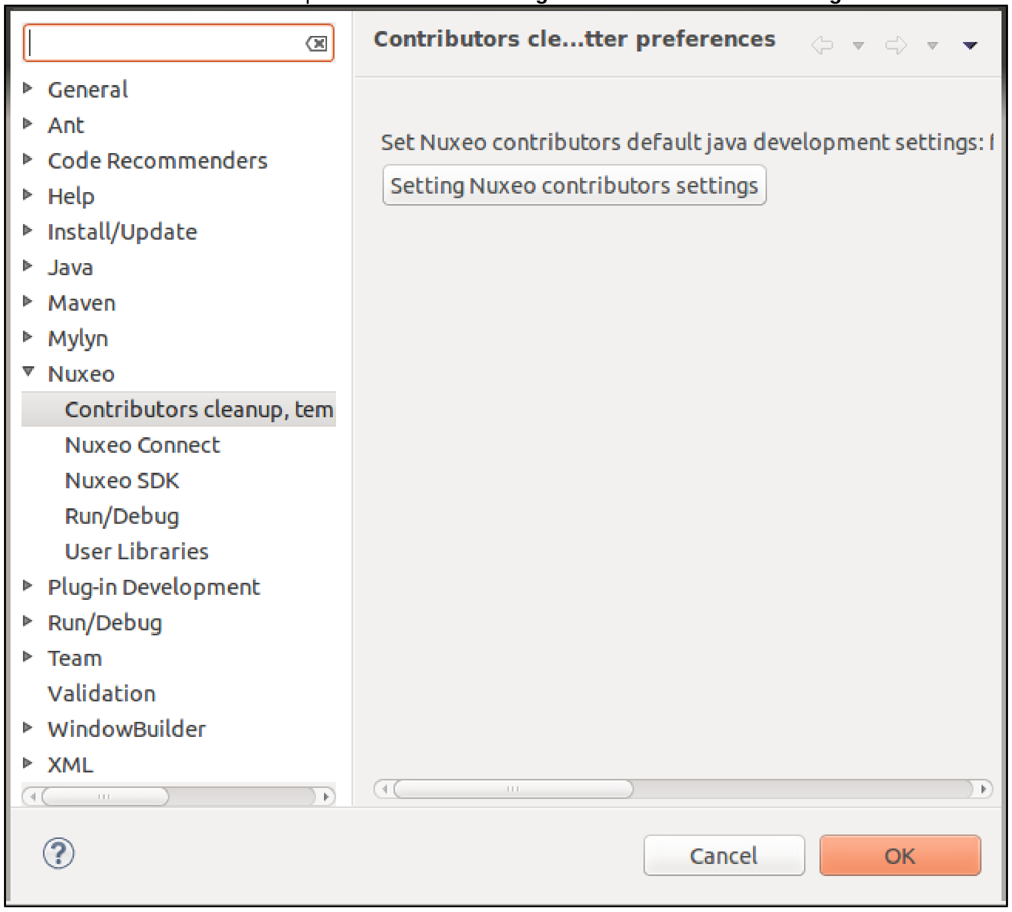
Since Nuxeo IDE 1.2.1, maven 3 is being used to compile projects and generate marketplace packages. Refer to the Use

Maven 3 JIRA ticket for more information about switching from maven 2 to maven 3 impacts.

Settings Eclipse Development Templates

In general of if you want to contribute to the Nuxeo Platform as a contributor, Nuxeo IDE provides the ability to set up all code templates inside your Eclipse.

1. Go to the Eclipse Preferences.
2. Select **Nuxeo** > **Contributors** option and click on **Settings Nuxeo Contributors settings**.



Set Nuxeo contributors default java development settings: I

The Following files are installed to override your current Eclipse configuration:

- `nuxeo_cleanups.xml`
- `nuxeo_codetemplates.xml`
- `nuxeo_formatter.xml`

These files are also provided in the Nuxeo GitHub repository.

Wizard Index

Here is the list of all wizards available on the IDE:

Projects

Name	Description	Generated files
Nuxeo Plugin	Creates an empty Nuxeo plugin project.	• `src/test/java/packagePath` • `src/main/resources/META-INF/MANIFEST.MF` • `src/main/java/packagePath` • `src/main/resources/META-INF/MANIFEST.MF` • `pom.xml`

Nuxeo WebEngine	Creates an empty WebEngine project.	• `src/test/java/packagePath` • `src/main/resources/META-INF/MANIFEST.MF` • `src/main/java/packagePath/MyModule.java` • `src/main/resources/META-INF/MANIFEST.MF` • `src/main/resources/skin/base.ftl` • `src/main/resources/skin/resources/README.txt` • `src/main/resources/skin/resources/css/site.css` • `src/main/resources/skin/resources/img/logo.png` • `src/main/resources/skin/views/MyRoot/index.ftl` • `pom.xml`

Automation

Name	Description	Generated files
operation	Creates the operation and its XML contribution. You can choose the input/output of the operation, its ID, category etc...	• `src/main/java/packagePath/MyOperation.java` • `src/main/resources/OSGI-INF/extensions/extensions.xml`

Component

Name	Description	Generated files
Nuxeo Component	Creates a Java class extending `DefaultComponent` that can be exposed as a service.	• `src/main/java/packagePath/MyComponent.java` • `src/main/resources/OSGI-INF/extensions/extensions.xml`

DocumentModel

Name	Description	Generated files
Adapter	Generates an adapter and the appropriate XML contribution. The adapter will have a getter and a setter for the different properties of the selected schema. To select a schema, you need to bind the project to a Studio account.	• `src/main/java/packagePath/MyDocumentAdapter.java` • `src/main/java/packagePath/MyDocumentAdapterFactory.java` • `src/main/resources/OSGI-INF/extensions/extensions.xml`
Listener	Generates a listener and the appropriate XML contribution.	• `src/main/java/packagePath/MyListener.java` • `src/main/resources/OSGI-INF/extensions/extensions.xml`

OpenSocial

Name	Description	Generated files

Automation Gadget	Creates a gadget based on an automation chain.	• `resources/gadget/gadgetName/` • `resources/gadget/gadgetName/dynamic_messages.properties` • `resources/gadget/gadgetName/gadgetName.css` • `resources/gadget/gadgetName/gadgetName.js` • `resources/gadget/gadgetName/gadgetName.png` • `resources/gadget/gadgetName/gadgetName.xml` • `resources/gadget/gadgetName/gadgets-contrib.xml`
Gadget	Creates a gadget structure. Only difference with the Automation Gadget is in the JS file.	• `resources/gadget/gadgetName/dynamic_messages.properties` • `resources/gadget/gadgetName/gadgetName.css` • `resources/gadget/gadgetName/gadgetName.js` • `resources/gadget/gadgetName/gadgetName.png` • `resources/gadget/gadgetName/gadgetName.xml` • `resources/gadget/gadgetName/gadgets-contrib.xml`

Seam

Name	Description	Generated files
Action Bean	Code skeleton for a Seam bean that will manage a simple action. This can be used to: • do a navigation, • do some modification on the `currentDocument` (or other docs), • create new documents, • send/retrieve info from an external service.	• `src/main/resources/extensions/MyAction-extensions.xml` • `src/main/resources/OSGI-INF/l10n/MyAction-messages.properties` • `src/main/resources/OSGI-INF/deployment-fragment.xml` • `src/main/resources/web/nuxeo.war/icons/MyAction-action.png` • `src/main/resources/seam.properties` • `src/main/seam/packagePath/MyActionBean.java`
Controller Bean	Code skeleton for a Bean that is controlling a new Tab like action (a XHTML fragment). This can be used: • to display additional information about a document, • to provide a screen that allow to update the document. The bean is by default Conversation scoped so that: • if you have expensive computation it will be run only when needed, • you can manage view/update/view cycle. All states should be managed as member variables of the beans. The default invalidation policy is to reset the state when the `currentDocument` changes.	• `/src/main/resources/OSGI-INF/l10n/MyController-messages.properties` • `src/main/resources/OSGI-INF/deployment-fragment.xml` • `src/main/resources/web/nuxeo.war/icons/MyController-tab.png` • `src/main/resources/web/nuxeo.war/incl/tabs/MyController-tab.xhtml` • `src/main/resources/seam.properties` • `src/main/seam/packagePath/MyControllerBean.java`
Service Bean	This is a simple service wrapper that allow to inject a Nuxeo Runtime service as a Seam component.	• `src/main/seam/packagePath/MyServiceWrapperBean.java`

Security

Name	Description	Generated files
Permission	Creates an XML contribution containing the new permission.	• `src/main/resources/OSGI-INF/extensions/permissions.xml`
Permission Visibility	Creates an XML file that associates different permission to a document type.	• `src/main/resources/OSGI-INF/extensions/permissions-visibility.xml`
Security Policy	Creates a Java file extending `AbstractSecurityPolicy` and the appropriate XML contribution.	• `src/main/java/packagePath/MySecurityPolicy.java` • `src/main/resources/OSGI-INF/extensions/extensions.xml`

Frequently Asked Questions

Nuxeo IDE cannot deploy. It just waits and waits!

Nuxeo IDE is generating a dedicated template for running the SDK. Did you started the SDK by hand ? If yes, you should activate that template by setting in your environment the following variable

```
NUXEO_CONF=bin/nuxeo-sdk.conf
```

Can you check also that the profile is correctly activated ? In your SDK, the file conf/Catalina/localhost/nuxeo.xml should contains the line

```
<Loader className="org.nuxeo.runtime.tomcat.NuxeoWebappLoader"
loaderClass="org.nuxeo.runtime.tomcat.dev.NuxeoDevWebappClassLoader" />
```

```
In all cases, if you run Nuxeo by hand -> "./nuxeoctl(.bat) console" you should see much more information
to debug it.
```

My server is started with Nuxeo IDE and my Seam components / XHTML views are not deployed.

Just refresh once with the "Reload projects on server" button.