



## Getting Started with *featherlite* Web Projects

Author: Robert von Burg  
Type: *featherlite* Documentation  
Date: 2011-02-18  
Version: 0.2.0

## **Abstract**

This paper explains the steps necessary to setup the development environment to run and create *featherlite* Plug-Ins as a Web project in a Java Servlet Container version 2.5 with the help of the Eclipse 3.5 integrated development environment (IDE).

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Installation</b>	<b>1</b>
2.1	Required Software Packages . . . . .	1
2.1.1	Installing the Eclipse IDE . . . . .	2
2.1.2	Installing the Eclipse Web Tools . . . . .	3
2.1.3	Installing the XML Plug-in . . . . .	3
2.1.4	Installing the Subclipse SVN Plug-in . . . . .	4
2.2	Installing the <i>featherlite</i> libraries . . . . .	4
<b>3</b>	<b><i>featherlite</i> Web Core Plug-ins</b>	<b>7</b>
3.1	WebCore20 . . . . .	7
3.2	WebTemplate . . . . .	7
<b>4</b>	<b>Configure Eclipse</b>	<b>8</b>
<b>5</b>	<b><i>featherlite</i> in Web Projects</b>	<b>13</b>
5.1	Custom Web Project from WebTemplate . . . . .	13
5.2	Adding <i>featherlite</i> functionality to an existing web project . . . . .	17
<b>6</b>	<b>Running a web project in Eclipse</b>	<b>19</b>

## 1 Introduction

This document shows which *featherlite* Plug-Ins are needed for web development and how these plug-ins are to be installed and used to create *featherlite* web applications.

It is highly recommended that the GettingStarted document which describes the *featherlite* Plug-Ins for Server and Rich Client development is read beforehand, as many parts also apply to web development. To avoid redundant explanations, the *featherlite* Core Plug-Ins are not re-explained in this document.

*featherlite* web projects are created as Eclipse dynamic web projects. The view technology is Java ServerFaces 2 with Facelets as the default viewing technology. The reference implementation of the JSF2 API used is Mojarra<sup>1</sup>.

As an add-on for more advanced and feature rich components, PrimeFaces<sup>2</sup> in the version 2.0.x is used.

## 2 Installation

Installation of the a development environment for *featherlite* web applications can be done in two ways. Either the *featherlite* web SDK is downloaded from the *featherlite* download page or the development environment is put together manually, tailoring it to ones personal needs.

If the SDK is not used, then one must carry on with section 2.1 as that section describes the procedure to install Eclipse for *featherlite* web development.

If the *featherlite* SDK Kit is used, then the installation is straight forward:

- download the *featherlite* web SDK Kit from the *featherlite* download page<sup>3</sup>
- unpack the download zip file
- run the start\_eclipse.xxx file, depending on the platform

This concludes the Eclipse installation and one may continue with the section 2.2.

### 2.1 Required Software Packages

To install the Eclipse IDE for Web Development the installation of the following software packages are recommended:

- Either Eclipse for Java EE developers or Eclipse Classic IDE - The EE edition of Eclipse already contains all the necessary eclipse plug-ins to develop and start

---

<sup>1</sup> <https://javaserverfaces.dev.java.net/>

<sup>2</sup> <http://www.primefaces.org>

<sup>3</sup> [http://featherlite-framework.com/download#Development\\_Kits](http://featherlite-framework.com/download#Development_Kits)

web applications, yet if *featherlite* rich clients are going to be developed, then the Eclipse RCP tools will have to be installed as well<sup>4</sup>. The package can be downloaded free of charge from the eclipse download site. Version of writing is 3.5.x

- Eclipse Web Tools - These plug-ins are preinstalled in the Eclipse for Java EE developers, and thus are only needed if the Eclipse Classic IDE is used. The web tools contain the plug-ins needed to validate JSF pages and start web servers. The packages can be installed free of charge by using the Eclipse Software Update functionality from within the running eclipse application
- Eclipse XML Editors and Tools - These plug-ins are preinstalled in the Eclipse for Java EE developers, and thus are only needed if the Eclipse Classic IDE is used. These tools are needed to write the JSF web pages as they are used to validate the wellformedness of the XML files. The package can be installed free of charge by using the Eclipse Software Update functionality from within the running eclipse application
- Subclipse Plug-in - the revision control system software the *featherlite* developers use in-house. The package can be installed free of charge by using the Eclipse Software Update functionality from within the running eclipse application. Version of writing is 1.6.x
- Apache Tomcat 6.0.x - Although web applications run in different servlet containers, the *featherlite* team has tested and deployed *featherlite* web applications using the Apache Tomcat Servlet Container and thus recommends using this servlet container
- *featherlite* reference applications - useful for development. These applications are pre-compiled for different operating systems and can be directly started
- *featherlite* WebTemplate - recommended for web development. This is a pre-configured web project which can be used for a new dynamic web project in Eclipse. It also includes all the needed libraries for a *featherlite* instance

### 2.1.1 Installing the Eclipse IDE

To install the Eclipse for Java EE developers or the Classic IDE proceed as follows:

- download the Eclipse IDE from the eclipse download page<sup>5</sup>
- unzip the Eclipse package to your preferred location on the hard disk
- start the Eclipse application from the installation folder
- when the Workspace Launcher dialog asks to select a workspace, press the 'Browse...' button and create a new folder for your workspace

---

<sup>4</sup> See the Getting Started document for more information on this

<sup>5</sup> <http://www.eclipse.org/downloads>

### 2.1.2 Installing the Eclipse Web Tools

These plug-ins are preinstalled in the Eclipse for Java EE developers; thus, they only have to be installed when the Eclipse Classic IDE is used.

To install the Eclipse Web tools proceed as follows:

- From the eclipse SDK 'Help' menu select 'Install New Software'
- Select '–All Available Sites–' from the list of repositories
- Select the following packages:
  - JavaScript Developer Tools
  - Eclipse Java EE Developer Tools
  - Eclipse Web Developer Tools
  - Eclipse XSL Developer Tools
  - JST Server Adapters
  - JST Server UI
  - JST Web UI
  - Web Page Editor (Optional)
  - WST Server Adapters
- In the 'Install' wizard review the license agreement, mark 'I accept the license agreement' and press 'Finish' to install

### 2.1.3 Installing the XML Plug-in

These plug-ins are preinstalled in the Eclipse for Java EE developers, and thus are only needed if the Eclipse Classic IDE is used.

To install the 'Eclipse XML Editors and Tools' plug-in proceed as follows:

- from the eclipse SDK 'Help' menu select 'Install New Software'
- in the 'Install' dialog select 'All Available Sites' from the selection list in the upper part of the dialog
- wait for the list of available plug-ins to be updated by eclipse
- scroll to 'Web, XML and Java EE Development' and open the tree node to see all plug-ins available for that topic
- select 'Eclipse XML Editors and Tools' and proceed with 'Next'
- from the 'Install Details' select 'Eclipse XML Editors and Tools' and click 'Next' to proceed
- in the 'Install' wizard review the license agreement, mark 'I accept the license agreement' and press 'Finish' to install

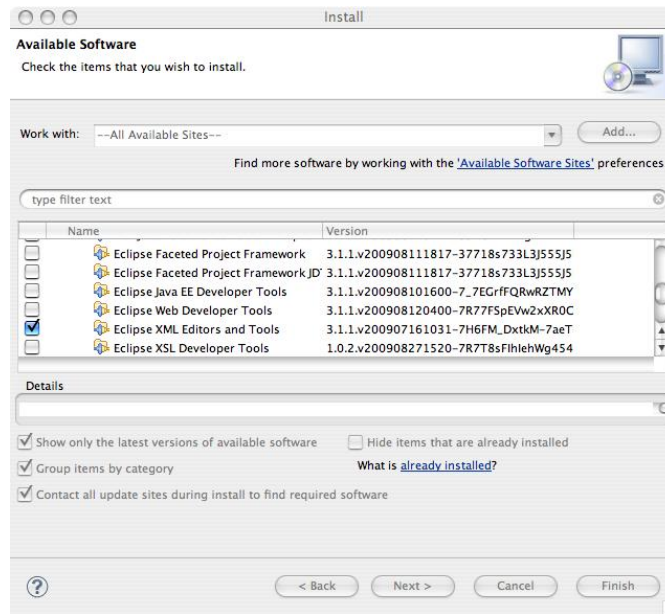


Figure 1: Install dialog settings to install the 'Eclipse XML Editors and Tools' Plug-in.

### 2.1.4 Installing the Subclipse SVN Plug-in

To install the Subclipse Plug-in proceed as follows:

- from the eclipse SDK 'Help' menu select 'Install New Software'
- press 'Add' to open a dialog to add the subclipse site
- in the 'Add Site' dialog type 'subclipse' into the name field and use 'subclipse.tigris.org/update\_1.6.x' as the address
- from the available packages select the 'Core SVNKit Library', 'SVNKit Client Adapter' and 'Subclipse' and press 'Next' to continue
- from the 'Install Details' select 'Subclipse' and click 'Next' to proceed
- in the 'Install' wizard review the license agreement, mark 'I accept the license agreement' and press 'Finish' to install

## 2.2 Installing the *featherlite* libraries

In contrast to the *featherlite* server plug-in model which is explained in the GettingStarted guide, web projects are not OSGI plug-ins. Thus the *featherlite* plug-ins are deployed as simple JARs which are deployed in the servlets lib folder:

```
<eclipse_web_project>/WebContent/WEB-INF/lib/
```

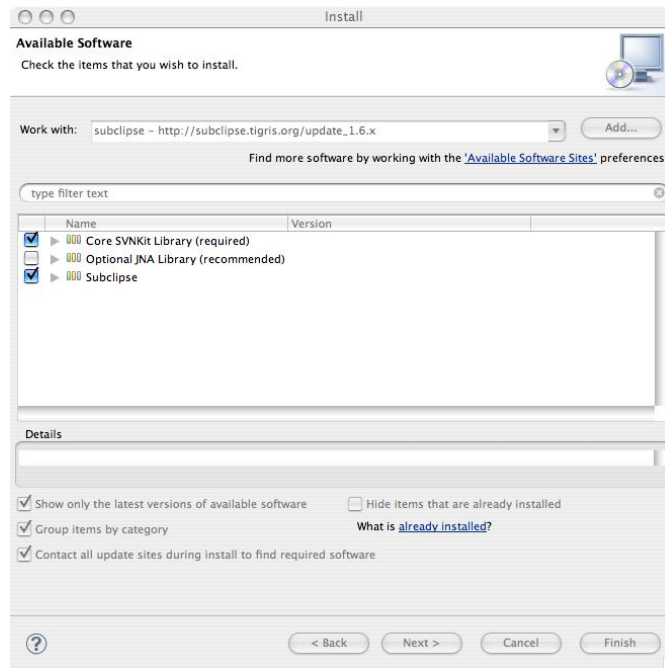


Figure 2: The dialog settings to install the 'Subclipse' plug-in.

On the *featherlite* download page for the web components<sup>6</sup> the components are distributed as zip files. Aside of the actual *featherlite* web component, the archive might contain additional JAR dependencies in a "lib" directory. These must also be copied to the web application.

Additionally, the archive might contain example configuration files and models which can help getting started with the web component.

The two reference clients *PlanningClient* and *ExecutionClient* can aid in debugging, as running *featherlite* web instance still allow RMI calls as long as the proper *featherlite* components are registered.

To simplify the dependency resolution, there is a template web project which contains all the needed JARs including a minimal working setup, thus the template contains the following JARs in the `../lib/` folder:

***featherlite* JARs:**

- bedplate.jar
- RSP.jar
- JEPlugin.jar

***featherlite* dependencies:**

- asm-2.2.x.jar

<sup>6</sup> [http://featherlite-framework.com/download#Web\\_Components](http://featherlite-framework.com/download#Web_Components)

- asm-commons-2.2.x.jar
- asm-util-2.2.x.jar
- ocutil-2.4.x-with-src.jar
- dom4j-1.6.x.jar
- je-4.0.x.jar
- log4j-1.2.x.jar
- meparser-0.0.x.jar
- org.eclipse.core.runtime\_3.5.x.jar
- org.eclipse.equinox.common\_3.5.x.jar
- org.eclipse.osgi\_3.5.x.jar
- poi-3.x-with-src.jar

***featherlite* web core:**

- WebCore20.jar

***featherlite* web core dependencies:**

- facestrace-1.1.x.jar
- jsf-api-2.0.x.jar
- jsf-impl-2.0.x.jar
- primefaces-2.0.x.jar

The use of the web template is not mandatory. If an existing web project should be converted to a *featherlite* instance, then one must only copy the appropriate libraries to the existing web project. How this is done is explained in section 5.2

### 3 featherlite Web Core Plug-ins

In this document only the WebCore Plug-Ins are explained. For information to the other *featherlite* Core Plug-Ins, see the GettingStarted document.

#### 3.1 WebCore20

The WebCore20 JAR defines commonly used features in a *featherlite* instance, e.g.:

- JSF managed bean “ResourcesBean” for the resourcesView.xhtml page
- JSF managed bean “OrdersBean” for the ordersView.xhtml page
- JSF managed bean “SettingsBean” for some default settings
- JSF managed bean “UserBean” for the login mechanism
- ExceptionHandler to handle ViewExpired Exceptions navigating the request to a session expired page
- StartStopListener to start and stop the *featherlite* instance

It is important to know that the WebCore20 is a completely optional JAR and is only needed if the WebTemplate is used. *featherlite* functionality can be added to an existing web project. On how this is done, refer to section 5.2.

#### 3.2 WebTemplate

The WebTemplate Plug-In is a fully functioning Web project that can be deployed and tested. It contains default views for viewing *featherlite* Resource and Order objects.

These views are all placed under the WebTemplate/WebContent/pages/views directory.

The web template includes authentication which evaluates session timeouts and illegal actions. The authentication is configured using the *XMLUserMgmtHandler featherlite* component. This component uses a configuration file called *users.xml*. The **default user** has the username/password combination **admin / admin**.

The directory WebTemplate/WebContent/WEB-INF/config and ../models have the same functionality as in a *featherlite* server. This has already been described in the GettingStarted document.

Section 5.1 shows the steps needed to import the WebTemplate into Eclipse and how to start it.

## 4 Configure Eclipse

Before a web project can be started, Eclipse must be configured. This is done by configuring a Server Runtime, adding a web application server and subsequently setting a web project to run on the defined server:

- Under 'Window/Preferences' switch to the 'Server' configuration point and choose 'Runtime Environments' (Figure 3)
- By clicking on 'Add...' create a new Server Runtime configuration for 'Apache Tomcat v6.0' (Figure 4)
- In the next wizard page change the Tomcat installation path to reflect the actual directory where tomcat has been extracted to (Figure 5). Now the wizard can be finished
- Now a concrete server must be configured. This is done by opening the 'Servers' view. It might have to be opened through the menu 'Window/ Show View/ Other...' and then selecting 'Servers'
- By using the context menu and choosing 'New/Server' the wizard to create a new Server is opened (Figure 6)
- Choose the server type 'Tomcat v6.0 Server' and then finish the wizard (Figure 7)
- Now the 'Servers' view has a new entry. Open the server to edit certain options (Figure 8)
- This view allows changing of the behaviour of the server and the runtime configuration (Figure 9)
- To not have Eclipse restart the server everytime code changes are saved, choose 'Never publish automatically' under 'Publishing'
- The maximum amount of memory for the virtual machine can be modified by clicking on 'Open launch configuration' and appending the appropriate '-XmxXXXm' (e.g. -Xmx512m for 512MB of memory) argument to the 'Arguments view' in the new dialog
- This concludes the configuration of a new server (Figure 10)

Section 6 will go into detail on how to run a specific web project.

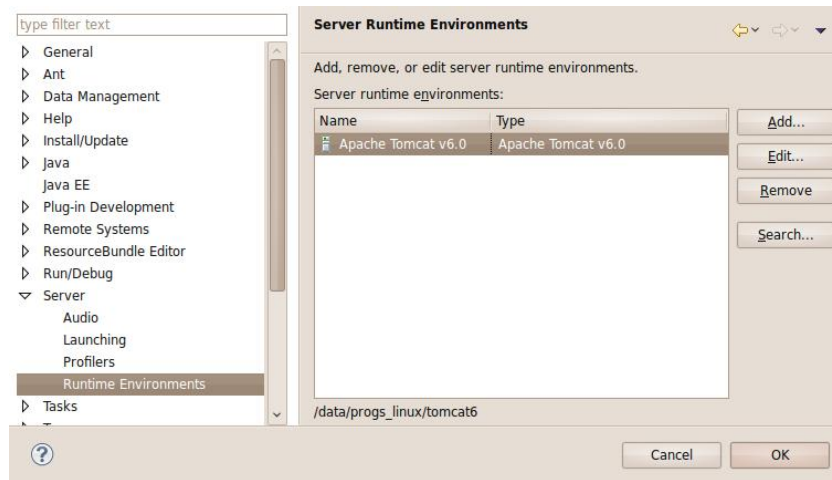


Figure 3: Create new Server Runtime

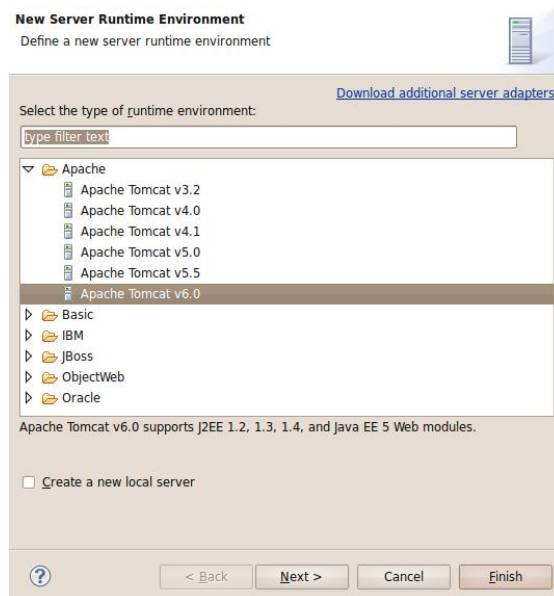


Figure 4: Choose Server Runtime Type

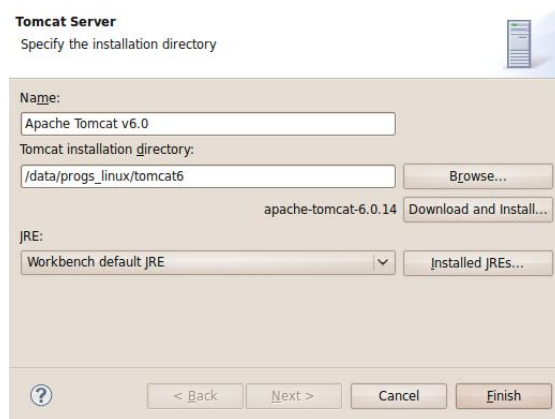


Figure 5: Define Server Runtime installation path



Figure 6: Create new Server

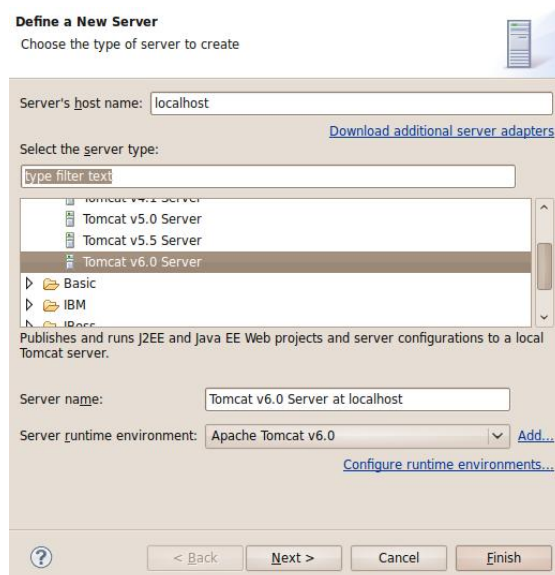


Figure 7: Choose Server Type

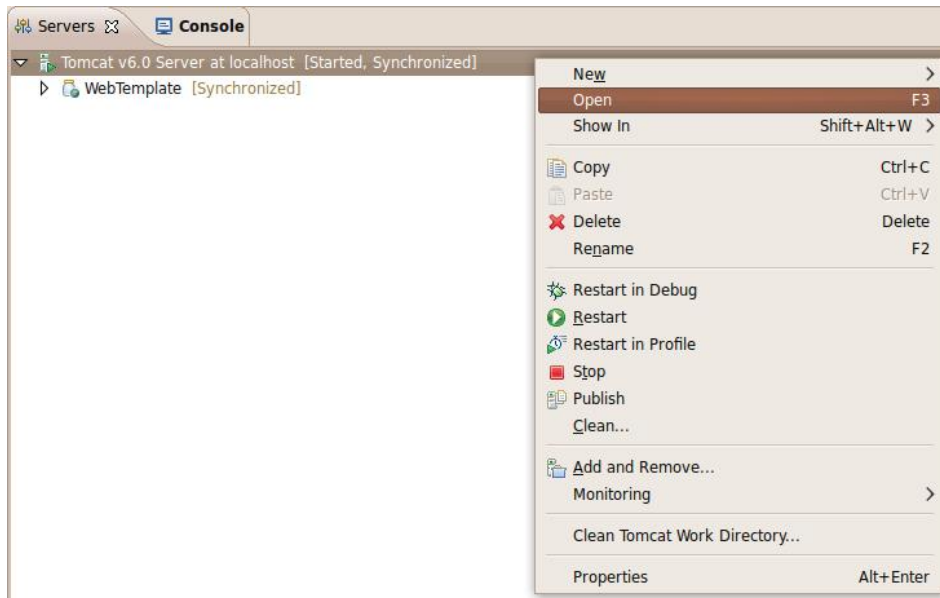


Figure 8: Open configuration of created server

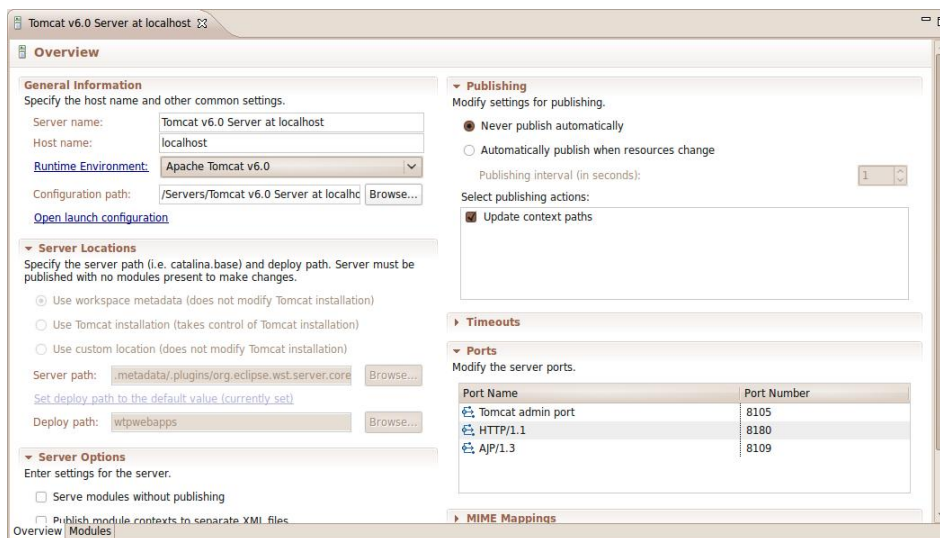


Figure 9: Configuration of the server

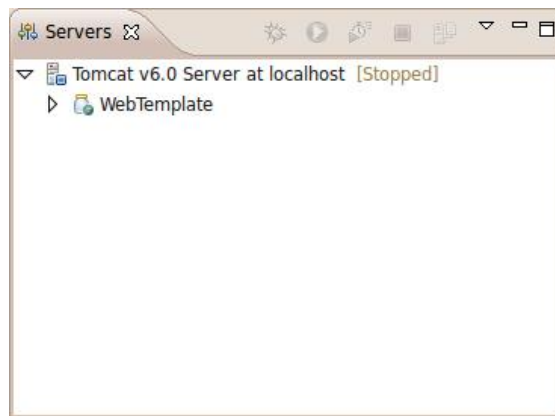


Figure 10: Configured Server

## 5 featherlite in Web Projects

Adding *featherlite* functionality to a web project can be done in different ways.

For existing web projects, one must only add all the libraries, and *featherlite* configuration files and then install a `ServletContextListener` which starts the *featherlite* instance.

For new web projects, one can either use the supplied `WebTemplate`, or one can create a new web project and then add the functionality as with existing web projects.

### 5.1 Custom Web Project from WebTemplate

Creating a web project from the `WebTemplate` is the simplest and fastest way of getting started with *featherlite* and web projects. Using the `WebTemplate` does bind the *featherlite* web application to use the web technology as has been described in section 1.

If this is not desirable, then see section 5.2 to add *featherlite* functionality to an existing project.

The following describes the procedure in creating a new web project from the `WebTemplate`:

- Download the `WebTemplate` from the *featherlite* download page<sup>7</sup>
- Extract the contents of the archive to your Eclipse workspace directory
- Rename the extracted directory so that the name fits the new web project e.g. "MyWeb"
- Now in Eclipse from the "File" menu, select "New" and "Other",
- from the "New" dialog select "Dynamic Web Project" which is in the "Web" folder (Figure 11)
- In the 'New Dynamic Web Project' dialog set the name of the project to the name of the renamed `WebTemplate` e.g. "MyWeb" (Figure 12)
- In the "Target Runtime" select the "Apache Tomcat v6.0" runtime, or create a new runtime using the "New..." button. Creating a new runtime is described in section 4 where one would proceed from figure 7
- Select Version 2.5 for the "Dynamic web module version" and then continue to the next page
- in the next page select "Add Folder..." and add the source folder 'i18n' to the project (Figure 13). This folder contains the bundles for internationalizing the web application.
- Now finish the wizard

---

<sup>7</sup> <http://featherlite-framework.com/download>

Once the project has been created and Eclipse has finished building the project, the project can be started in the way described in Section 6. Figure 14 shows what the structure of the newly created web project should look like in Eclipse.

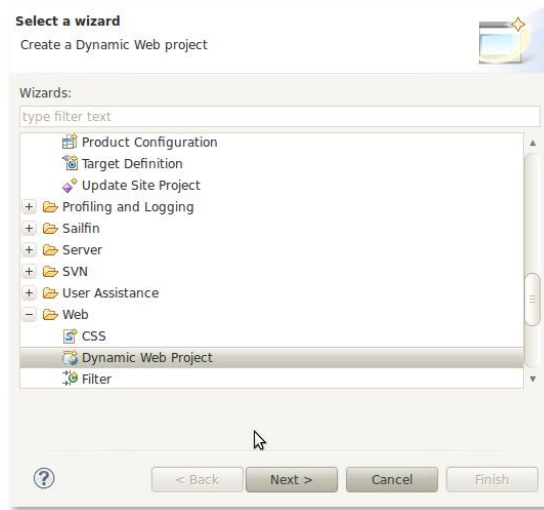


Figure 11: Select 'Dynamic Web Project' from the "New" dialog

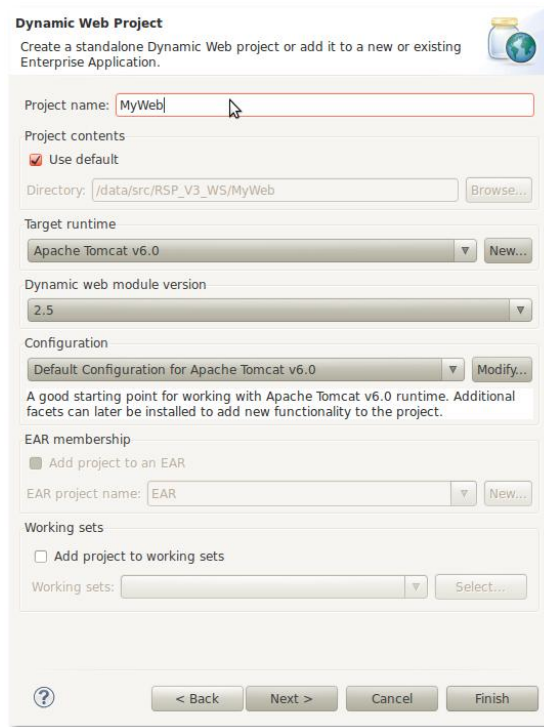


Figure 12: The new 'Dynamic Web Project' wizard where the project name must be defined

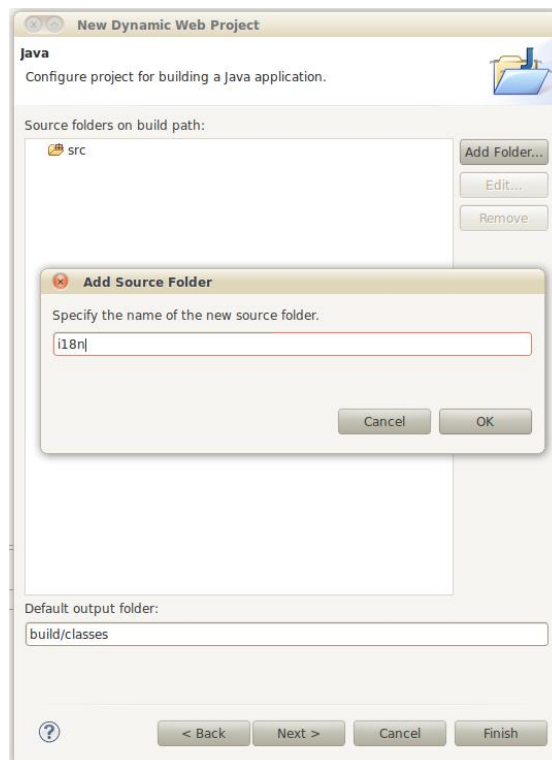


Figure 13: Add the 'i18n' source folder for the internationalization property bundles

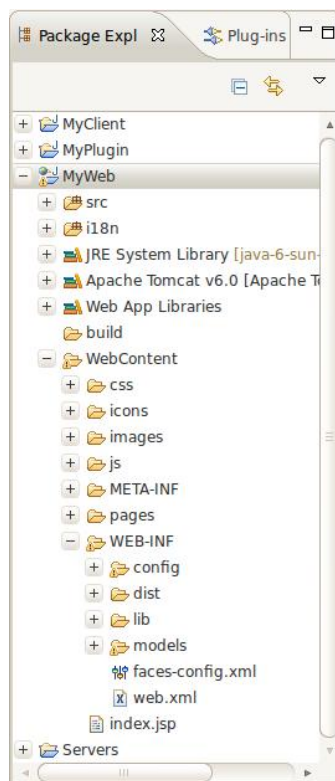


Figure 14: The structure of the newly created web project

## 5.2 Adding *featherlite* functionality to an existing web project

Adding *featherlite* functionality to an existing web project is done by adding all the *featherlite* libraries and their dependencies, adding the *featherlite* configuration files and adding a ServletContextListener which starts the *featherlite* instance:

- Download the WebTemplate from the *featherlite* download page<sup>8</sup>
- Extract the contents of the archive to a temporary place
- From the 'WebTemplate/WebContent/WEB-INF/lib' copy the following JARs to the existing web project's lib/ folder as they are defined in section 2.2:
  - The *featherlite* JARs
  - The *featherlite* dependencies
- From the WEB-INF/ folder of the WebTemplate copy the config/, models/ and dist/ folders to the WEB-INF/ folder of the existing web project
- Create a new Java class which implements the ServletContextListener interface in the existing web project. Implement the class as is shown in listing 1.
- In the existing web project's web.xml file add a listener declaration which calls the new StartStopListener:

```
<listener>
  <listener-class>
    com.featherlite.web.base.StartStopListener
  </listener-class>
</listener>
```

Now when the existing web application is started, then the *featherlite* instance should be initialized and in the Eclipse console, a log similar to figure 18 should be seen.

---

<sup>8</sup> <http://featherlite-framework.com/download>

```

1 package com.featherlite.web.base;
2
3 import java.io.File;
4
5 import javax.servlet.ServletContextEvent;
6 import javax.servlet.ServletContextListener;
7
8 import org.apache.log4j.Logger;
9
10 import com.rsp.core.base.RSPComponentContainer;
11 import com.rsp.core.base.RSPComponentContainerMBean;
12 import com.rsp.core.base.exception.RSPException;
13 import com.rsp.core.base.init.RSPLog4jConfigurator;
14 import com.rsp.core.base.init.RSPStarterInit;
15
16 /**
17  * @author robertb
18  */
19 public class StartStopListener implements ServletContextListener {
20
21     private static final Logger logger = Logger.getLogger(StartStopListener.class);
22
23     /**
24      * @see javax.servlet.ServletContextListener#contextInitialized(javax.servlet.ServletContextEvent)
25      */
26     public void contextInitialized(ServletContextEvent event) {
27
28         try {
29             RSPLog4jConfigurator.configure();
30             String pcoRootWeb = event.getServletContext().getRealPath("/WEB-INF");
31             File rootPath = new File(pcoRootWeb);
32             RSPStarterInit.startRSP(rootPath);
33
34         } catch (Exception e) {
35             logger.error(e, e);
36             RSPException pcoe = new RSPException("Startup Exception while starting the web project", "rsp.initFailed",
37                 e);
38             pcoe.addProperty("className", "Web Project");
39             throw pcoe;
40         }
41     }
42
43     /**
44      * @see javax.servlet.ServletContextListener#contextDestroyed(javax.servlet.ServletContextEvent)
45      */
46     public void contextDestroyed(ServletContextEvent event) {
47
48         logger.info("try to shutdown featherlite container");
49
50         RSPComponentContainerMBean container = RSPComponentContainer.getInstance();
51
52         container.stop();
53         container.shutdown();
54     }
55 }

```

Listing 1: A `ServletContextListener` implementation which starts and stops the *featherlite* instance

## 6 Running a web project in Eclipse

The following is the procedure which needs to be followed to start an existing *featherlite* web application which is imported into Eclipse as an Eclipse plug-in:

- To start a project, it must be added to a server, thus set a project on the server by using the context menu on the server and selecting 'Add and Remove...' (Figure 15)
- Now add the project to the 'Configured' side (Figure 16)
- Once Eclipse is configured with a server and a project is associated to a server, the server can be started
- Figure 17 shows the buttons used to control a server:
  1. Start a server which is not yet running
  2. Stop a running server
  3. Publish changes to a server

**Note:** If the server is running, and Java classes have been changed, then publishing will also restart the server. Yet if the changes are only to XHTML pages, then the changes are published and are immediately active
- Once the server is finished starting, the console should have more or less the output as is shown in figure 18

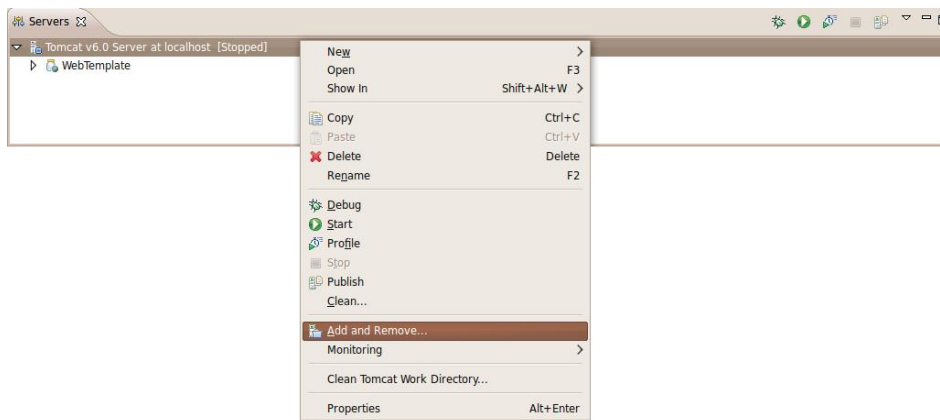


Figure 15: Add a project to server 1/2

Unless the server was configured differently, you can access your web project through a browser using the url `http://localhost:8080/Project_Name` where you must substitute 'Project\_Name' with the name of the project you have configured for the web project (e.g., MyWeb for the example of the Section 5.1).

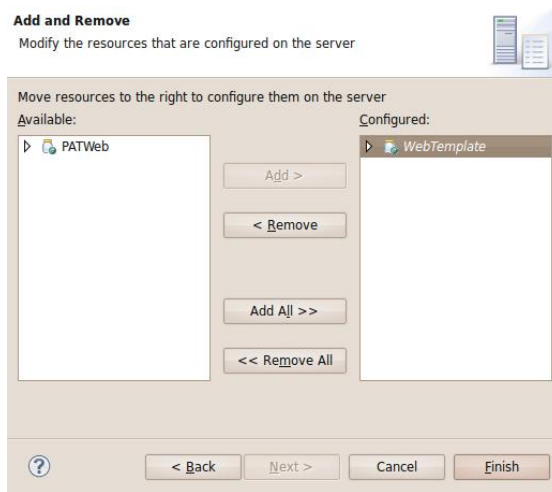


Figure 16: Add a project to server 2/2



Figure 17: Running a Server

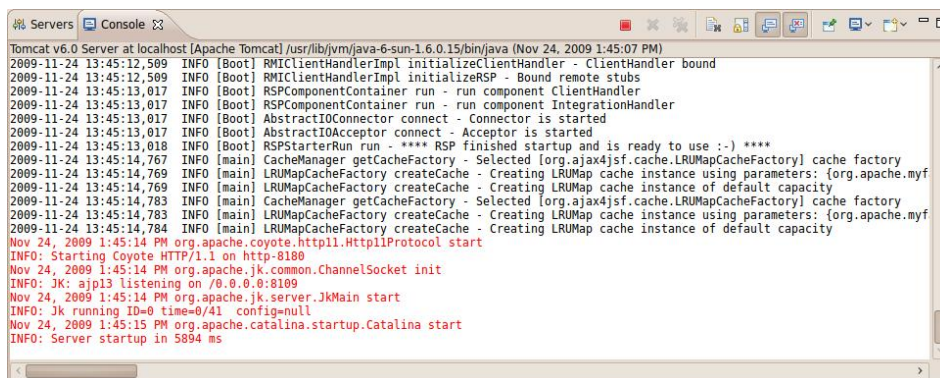


Figure 18: Successfully started *featherlite* server in Tomcat