**FIRST CONNECT - PRESENTATION LAYER**

**DEVELOPER GUIDE**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version No | Date | Prepared by / Modified by | Significant Changes |
| 1.0 | 29-Apr-14 | Kanimuthu | Initial Version |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

Contents

[1. Getting Access 5](#_Toc389233217)

[1.1 Share Folder 5](#_Toc389233218)

[1.2 SVN 5](#_Toc389233219)

[2. Environment Setup 5](#_Toc389233220)

[2.1 JDK 5](#_Toc389233222)

[2.1.1 Prerequisites 5](#_Toc389233223)

[2.1.2 Configuration 5](#_Toc389233224)

[2.2 IDE 6](#_Toc389233225)

[2.2.1 Prerequisites 6](#_Toc389233226)

[2.2.2 Installation 6](#_Toc389233227)

[2.2.3 Configuration 6](#_Toc389233228)

[2.2.4 Verification 10](#_Toc389233229)

[2.3 Application Server 11](#_Toc389233230)

[2.3.1 Prerequisites 11](#_Toc389233231)

[2.3.2 Installation 11](#_Toc389233232)

[2.3.3 SSL Key Store Generation 11](#_Toc389233233)

[2.3.4 Configuration 12](#_Toc389233234)

[2.3.5 Verification 13](#_Toc389233235)

[2.4 Build Management 13](#_Toc389233236)

[2.4.1 Prerequisites 13](#_Toc389233237)

[2.4.2 Installation 14](#_Toc389233238)

[2.4.3 Configuration 15](#_Toc389233239)

[2.5 Version Control System 16](#_Toc389233240)

[2.5.1 Prerequisites 16](#_Toc389233241)

[2.5.2 Installation 16](#_Toc389233242)

[2.5.3 Configuration 16](#_Toc389233243)

[2.6 Web Service Framework 18](#_Toc389233244)

[2.6.1 Prerequisites 18](#_Toc389233245)

[3. Presentation Layer Implementation 18](#_Toc389233246)

[3.1 Prerequisites 18](#_Toc389233249)

[3.1.1 Identify Services 18](#_Toc389233250)

[3.1.2 Identify Request-Response 18](#_Toc389233251)

[3.1.3 Java Project Creation 19](#_Toc389233252)

[3.1.4 POM Creation 20](#_Toc389233253)

[3.1.5 Building Project 21](#_Toc389233254)

[3.2 Configuring the Web Application 22](#_Toc389233255)

[3.2.1 Files Required 22](#_Toc389233256)

[The following files are required. 22](#_Toc389233257)

[3.2.1.1 Servlet Configurations in web.xml 22](#_Toc389233258)

[3.2.1.2 index.html 23](#_Toc389233259)

[3.2.1.3 web-application-config.xml 23](#_Toc389233260)

[Create an xml, web-application-config.xml in src/main/webapp/WEB-INF/config. 23](#_Toc389233261)

[3.2.1.4 Bean configuration – sample-beans-config.xml 23](#_Toc389233262)

[3) SampleController.java 25](#_Toc389233263)

[8) Trouble Shooter 27](#_Toc389233264)

[a. Build Management 27](#_Toc389233265)

[b. Application Servers 27](#_Toc389233266)

[c. Deployment 28](#_Toc389233267)

# Getting Access

Access to the share folders and SVN repositories is required. Follow the instructions in the below sections to get the required access.

## Share Folder

Access to the following share folders is required. Get it from Archana ([Archana-r@hcl.com](mailto:Archana-r@hcl.com)).

1. 10.105.188.30
2. 10.105.130.10

## SVN

Access to the following repositories is required. Get it from Anitha Ramaiah ([Anitha.Ramaiah@hcl.com](mailto:Anitha.Ramaiah@hcl.com)).

1. http://10.105.188.28/fgbrepo
2. http://10.105.188.28/fgbdocs
3. <http://172.20.211.15/fbpsource> (or) http://10.105.67.138/fbpsource

# Environment Setup

Follow the instructions described in the below sections to install and configure the development Environment of First Connect. After executing each section return back to the appropriate section and continue the flow.



## JDK

### Prerequisites

1. Get the tarmac approval
2. Raise a remedy to install

### Configuration

Set java environment variables.

|  |
| --- |
| JAVA\_HOME=D:\FGBDEV\tools\ jdk1.6.0\_25  PATH=%PATH%;%JAVA\_HOME%\bin |

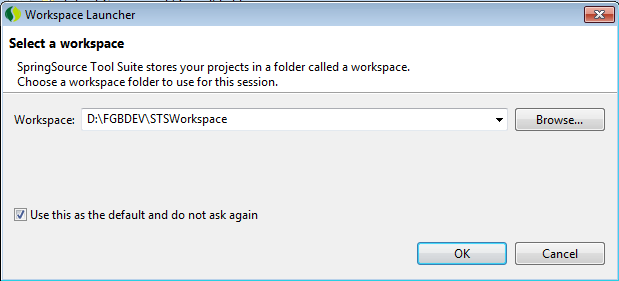
## IDE

### Prerequisites

1. Make sure you have access to server 10.105.188.30. If not, refer the section [Getting Access](#_Getting_Access_1).
2. Make sure Eclipse/STS/JBOSS developer studio exists in share folder 10.105.188.30/Softwares.
3. Get Tarmac approval for all the software to be used.
4. Ensure JDK is installed.

### Installation

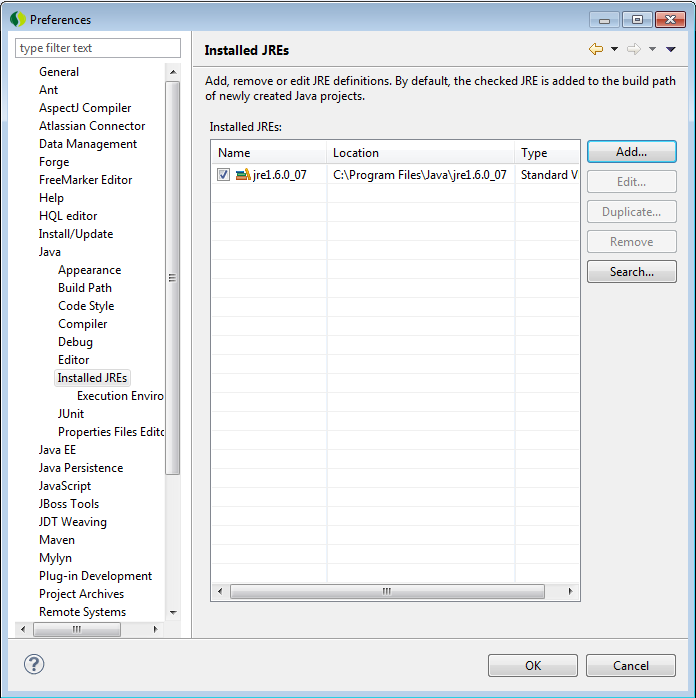
1. Copy the Eclipse/STS zip or JBOSS developer studio jar file from the share folder depending on your system configuration.
2. For Eclipse/STS, extract it to a folder in local drive (For JBOSS developer studio, install it by executing the installer).
3. Launch the application by opening executable file.
4. On prompt, specify the workspace.



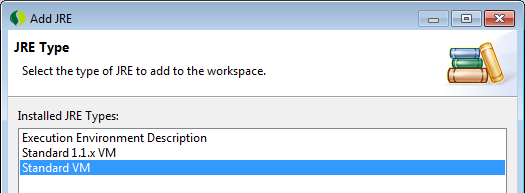
### Configuration

#### JRE Configuration

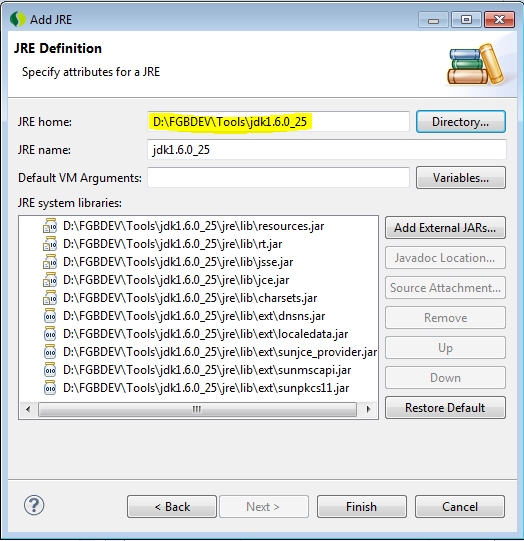
1. Open preferences window from menu windows-preferences.
2. Expand the option “java-installed JRE” from left navigation bar.



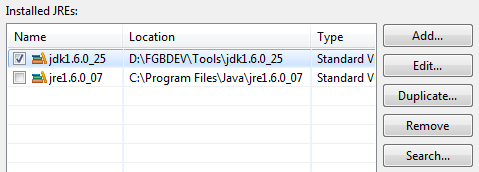
1. Click the add button to add the Standard VM.
2. Select the Standard VM option from the window as shown below.



1. Click next to proceed.
2. Specify the JRE Home as “D:\jdk1.6.0\_25”

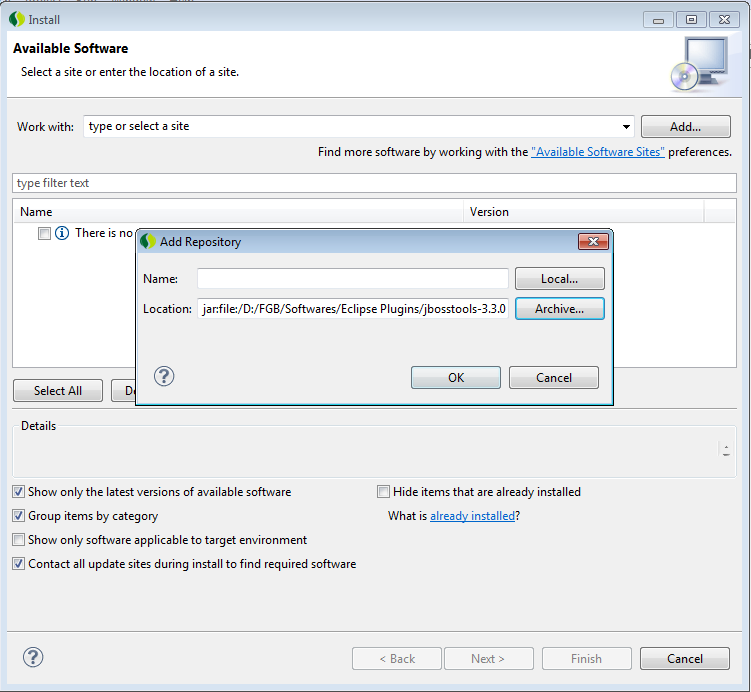


1. Click Finish.
2. Select the newly added JRE as shown below to make it the default JRE for the IDE.



#### JBoss Tools Plugins configuration

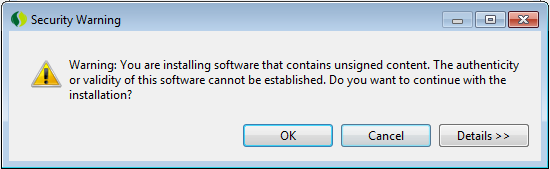
1. Select Help->Install New Software from menu option.
2. From the Install window, click the button Add.
3. From the Add Repository window click the button Archive and select the file jbosstools-3.3.0.M5.aggregate-Update-2011-12-14\_19-42-13-H61.zip from the share folder.



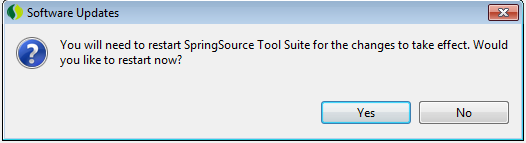
1. Click Ok to show the List of Available Softwares in a new Window.
2. Un Check the following Softwares from the list,

* JBoss GWT Integration
* JBoss BIRT Integration
* JBoss Cloud Development Tools

1. Accept the license terms and click ok to continue.
2. Click ok when the Security Warning pops up.

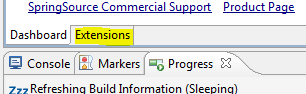


1. On Completion, it will prompt to reboot the workbench. Select the Yes button to restart STS.



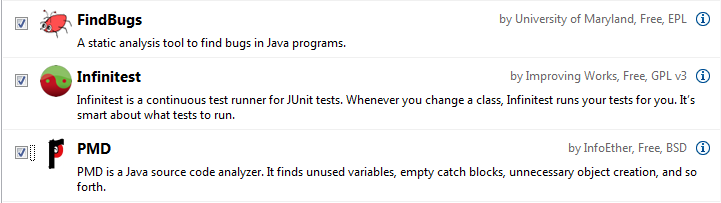
#### Additional Plugins Configuration

1. Select Help->Dashboard from the menu.
2. At the left bottom of the Dashboard window, select the Extensions tab as shown below,



1. Select the Following optional Plugins from the list and click Install button.

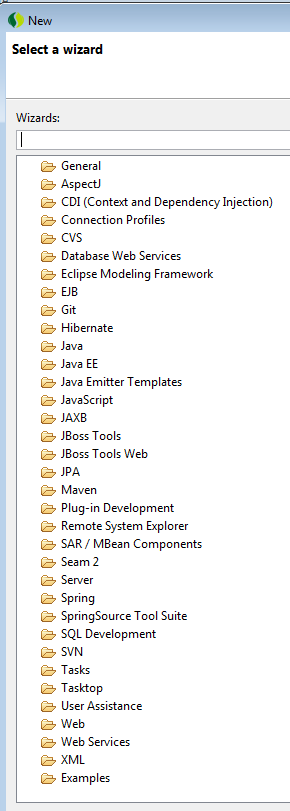
* FindBugs
* Infinitest
* PMD



1. Select all the softwares from the list that it shows and click next to proceed.
2. Accept the license and click next to proceed.
3. Click Ok button when the Security Warning window pops up and Click Yes when it prompts for reboot of STS.

### Verification

1. Start the IDE.
2. Ensure it doesn’t report any errors.
3. Verify whether the new Project window opens up a similar window with JBoss Tools and other plugin options as shown below (Rotate and see).



## Application Server

### Prerequisites

1. Make sure you have access to server 10.105.188.30. If not, refer the section [Getting Access](#_Getting_Access_1).
2. Make sure jboss-6.1.0.Final.zip exists in share folder 10.105.188.30/Softwares

### Installation

1. Copy the jboss-6.1.0.Final.zip file from the share folder.
2. Extract it to a folder in local drive.

### SSL Key Store Generation

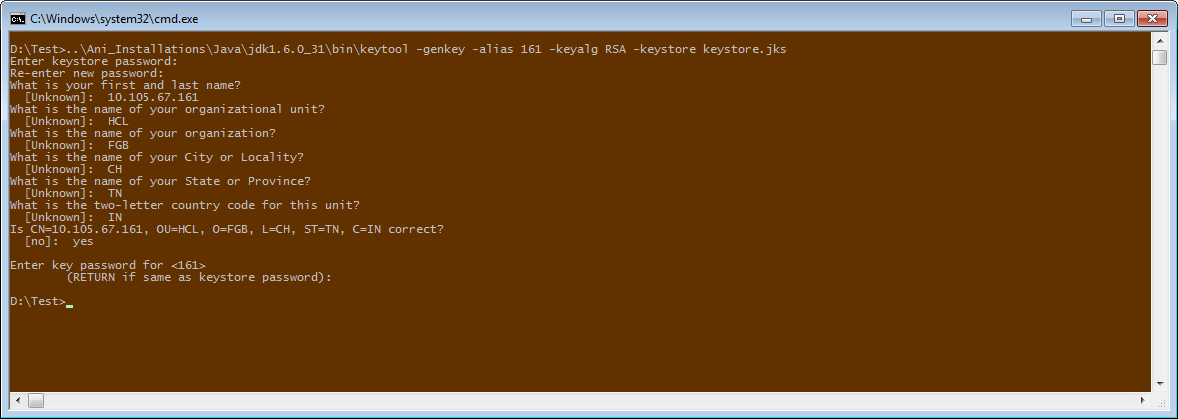
The below two sections can be omitted as of now. These are required when the services are exposed with SSL.

1. Create a new folder to save the generated certificates.
2. Create a new keystore using the Java supplied Keytool

**keytool -genkey -alias jboss -keyalg RSA -keystore keystore.jks**

**Password :** **fgb@123**

1. Enter the following details CN, FN, City Code, Country code etc



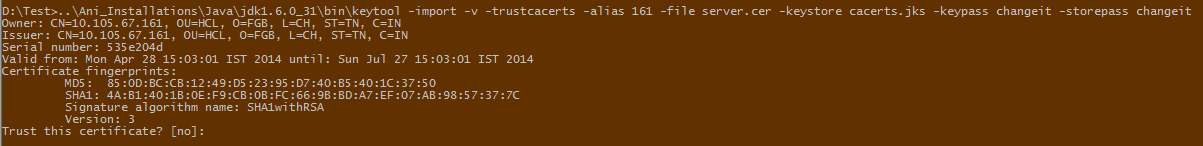
1. Export the generated server certificate from the keystore to server.cer.

**keytool –export –alias 161 -storepass fgb@123 -file server.cer -keystore keystore.jks**



1. Create a trust-store file and add the server cert to it.

**keytool –import –v –trustcacerts -alias 161 -file server.cer -keystore cacerts.jks –keypass fgb@123 -storepass changeit**



1. Bind the certificate to your java

D:/ProgramFiles/Java/jdk1.6.0\_21/bin/keytool –import –file D:/Certs/server.cer -alias localjboss -keystore D:/ProgramFiles/Java/jdk1.6.0\_21/jre/lib/security/cacerts

1. Finally copy those certificates from the folder which you created in the Step1. Place these certificate in the path jboss server.

Ie. Server->all ->conf -> cert-> jboss certs

|  |
| --- |
| Note : |
| 1. If you are having SL server in your system then copy those certificates and place in both and PL and SL server like this   Ie. Server->all ->conf -> cert-> jboss certs |
| 1. If not , then get the certificates generated with their IP and place it in your PL server. And also bind their certificate to java |

### Configuration

1. Edit the file josso-agent-config.xml in conf folder as described below

Change the IP to your System IP

<gatewayLoginUrl>https://10.105.67.38:8443/josso/signon/login.do</gatewayLoginUrl><gatewayLogoutUrl>https://10.105.67.38:8443/josso/signon/logout.do</gatewayLogoutUrl>

<!-- Gateway service locator -->

<gatewayServiceLocator>

<!-- Other properties for ws-service-locator :

username, password, servicesWebContext, transportSecurity

-->

<!-- <transportSecurity>Confidential</transportSecurity> -->

<protocol:ws-service-locator endpoint="10.105.67.38:8443" transportSecurity ="confidential"/>

</gatewayServiceLocator>

1. Edit the josso-gateway-web.xml in conf as described below.

change josso IP and add the trusted hosts as business server’s IP.

<web-configuration id="josso-web-configuraiton"

rememberMeEnabled="false"

sessionTokenOnClient="true"

sessionTokenSecure="false"

sendP3PHeader="false"

customLoginURL="https://10.105.67.38:8443/SSORedirect/login-redirect.jsp">

<trustedHosts>

<s:value>10.105.67.161</s:value>

<s:value>10.105.188.29</s:value>

<s:value>10.105.67.38</s:value>

<s:value>10.105.188.28</s:value>

<s:value>10.105.15.6</s:value>

<s:value>10.105.67.200</s:value>

<s:value>10.105.67.167</s:value>

<s:value>10.105.67.164</s:value>

</trustedHosts>

</web-configuration>

1. Edit theserver.xml in deploy->jbossweb.sar as described below.

Change the keystore and truststorePass password as you mention when you are generating the certificates.

<Connector SSLEnabled="true" address="${jboss.bind.address}" clientAuth="false" keystoreFile="${jboss.server.home.dir}/conf/cert/jbosscerts/keystore.jks" keystorePass="fgb@123" port="${jboss.web.https.port}" protocol="HTTP/1.1" scheme="https" secure="true" sslProtocol="TLS" truststoreFile="${jboss.server.home.dir}/conf/cert/jbosscerts/cacerts.jks" truststorePass="fgb@123"/>

### Verification

1. Go to Command Prompt.
2. Start the server.
3. Hit the URL <https://localhost:8443> in browser. It should open the server page.

## Build Management

### Prerequisites

1. Make sure you have access to server 10.105.188.30. If not, refer the section [Getting Access](#_Getting_Access_1).
2. Make sure apache-maven-3.0.4.zip and repository.Final.zip exists in share folder 10.105.188.30/Softwares

### Installation

1. Copy the apache-maven-3.0.4.zip file from the share folder.
2. Extract it to a folder in local drive.
3. Copy the repository.Final.zip from the share folder.
4. Extract it to MAVEN\_HOME/respository folder.
5. Open Settings.xml from MAVEN\_HOME/conf.
6. Search for the word <localRepository>.
7. Comment it as below.

<!-- <localRepository>/path/to/local/repo</localRepository> -->

1. Add the below line to reflect your repository.

<localRepository>D:\Tools\apache-maven-3.0.4\repository </localRepository>

Before Change:

<localRepository>/path/to/local/repo</localRepository>

After Change:

<!-- <localRepository>/path/to/local/repo</localRepository> -->

<localRepository>D:\Tools\apache-maven-3.0.4\repository </localRepository>

1. Search for the word ‘proxies’.
2. Comment the section ‘proxy’ as below

<!--

<proxy>

<id>optional</id>

<active>true</active>

<protocol>http</protocol>

<username>proxyuser</username>

<password>proxypass</password>

<host>proxy.host.net</host>

<port>80</port>

<nonProxyHosts>local.net|some.host.com</nonProxyHosts>

</proxy>

-->

1. Add the below section inside the tag <proxies></proxies>.

<proxy>

<id>optional</id>

<active>true</active>

<protocol>http</protocol>

<username>anitha.ramaiah</username>

<password>MarPas@1</password>

<host>arihant-proxy</host>

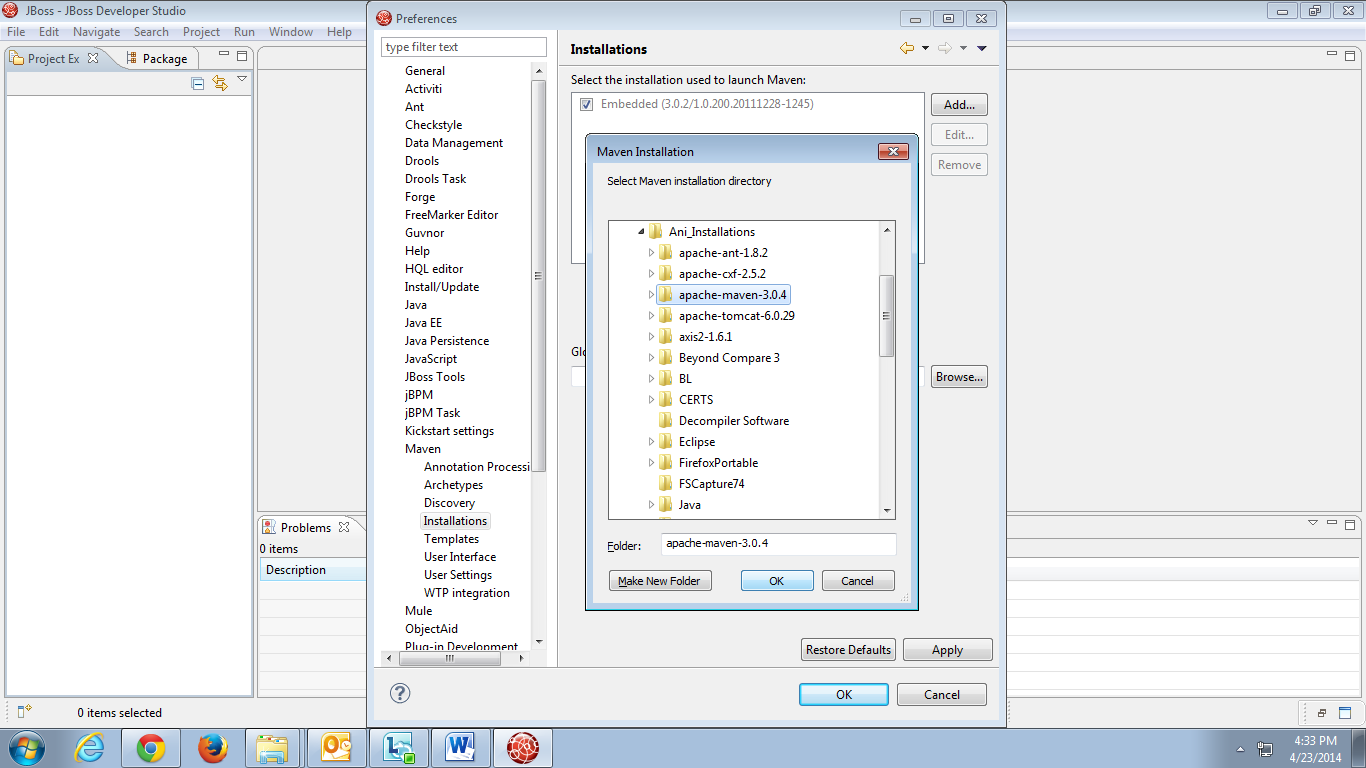
<port>80</port>

<nonProxyHosts>local.net|some.host.com</nonProxyHosts>

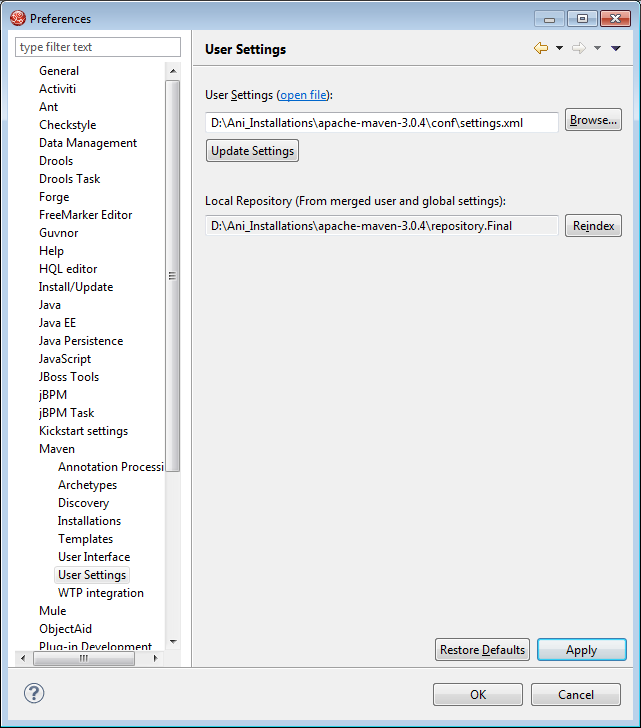
</proxy>

### Configuration

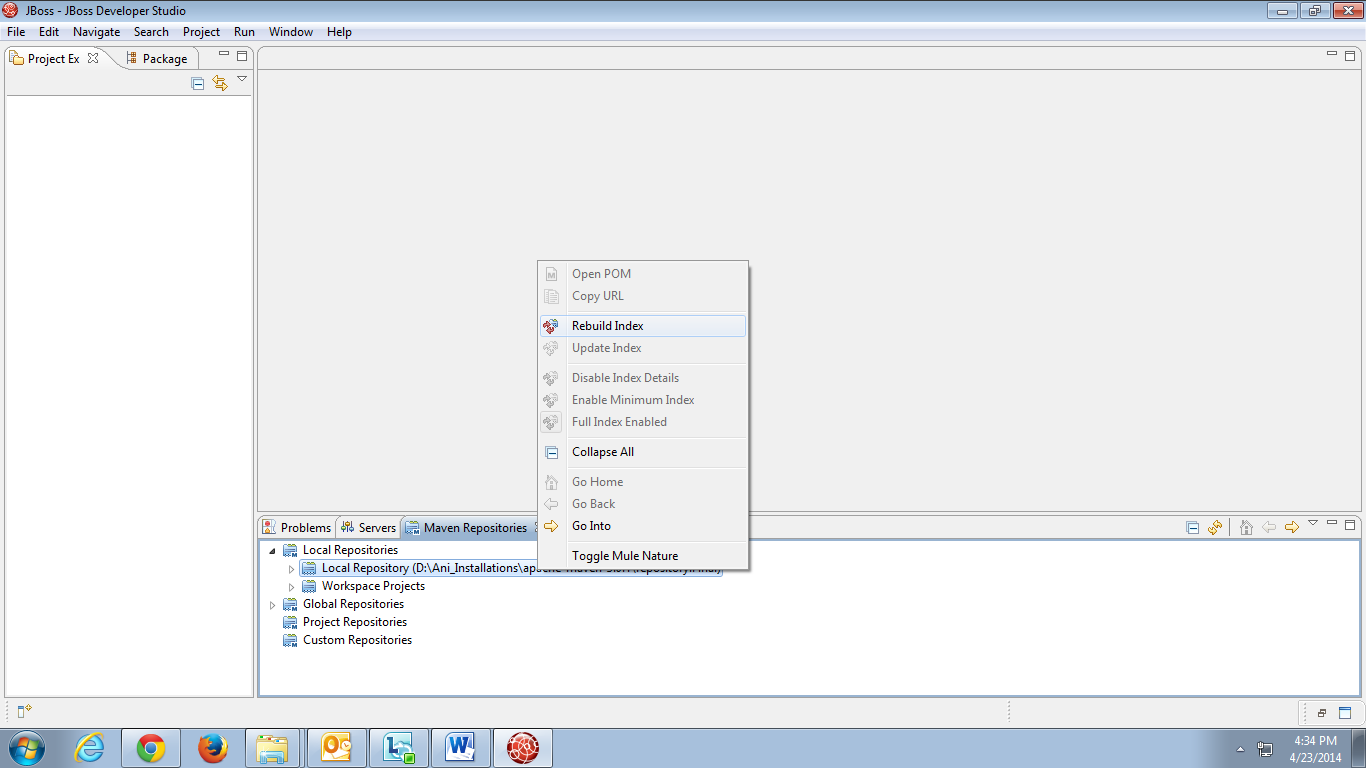
1. Open Window-Preferences-Maven in IDE.
2. Expand Maven.
3. Click on Installations.



1. Add the maven installation from your local system.
2. Click on User Settings.



1. Browse and choose the settings.xml from conf folder inside MAVEN\_HOME.
2. Right Click and Rebuild Index to get the local repository reflected in IDE.



**Note: The highlighted ones need to be updated with respect to your system.**

## Version Control System

### Prerequisites

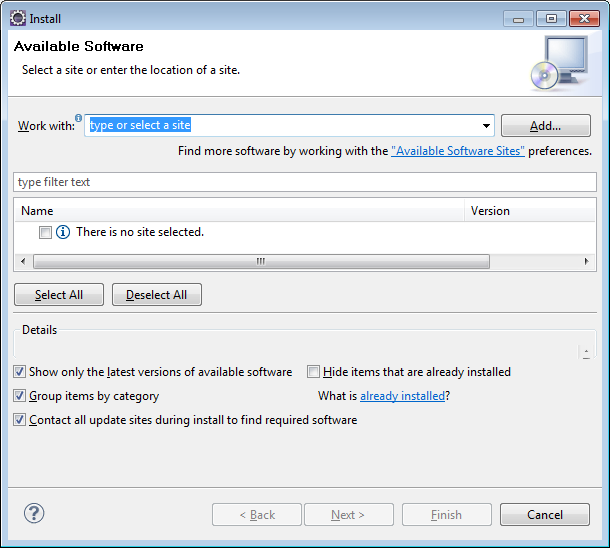
1. Make sure you have access to server 10.105.188.30 and SVN repositories. If not, refer the section [Getting Access](#_Getting_Access_1).
2. Make sure eclipse-SVN-site-1.8.5.zip exists in share folder 10.105.188.30/Softwares

### Installation

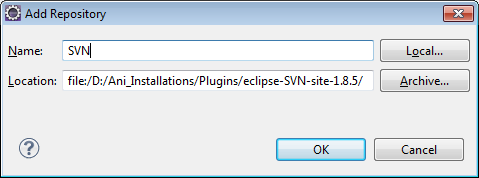
1. Copy the eclipse-SVN-site-1.8.5.zip file from the share folder.
2. Extract it to a folder in local drive.

### Configuration

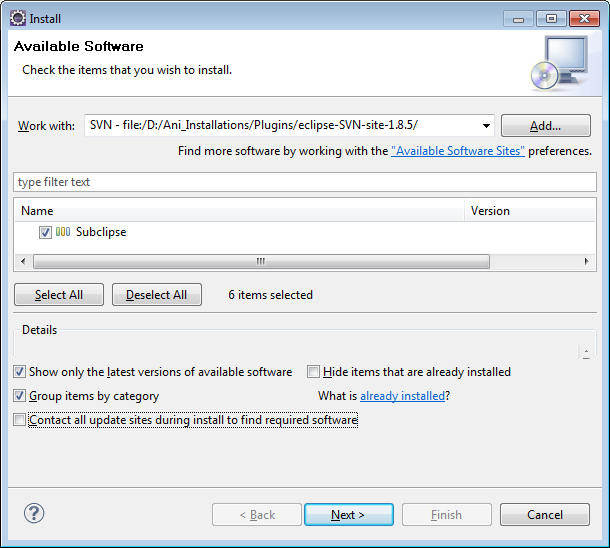
1. Open ‘Install New Software’ from Help-Install New Software in IDE.
2. Click Add.



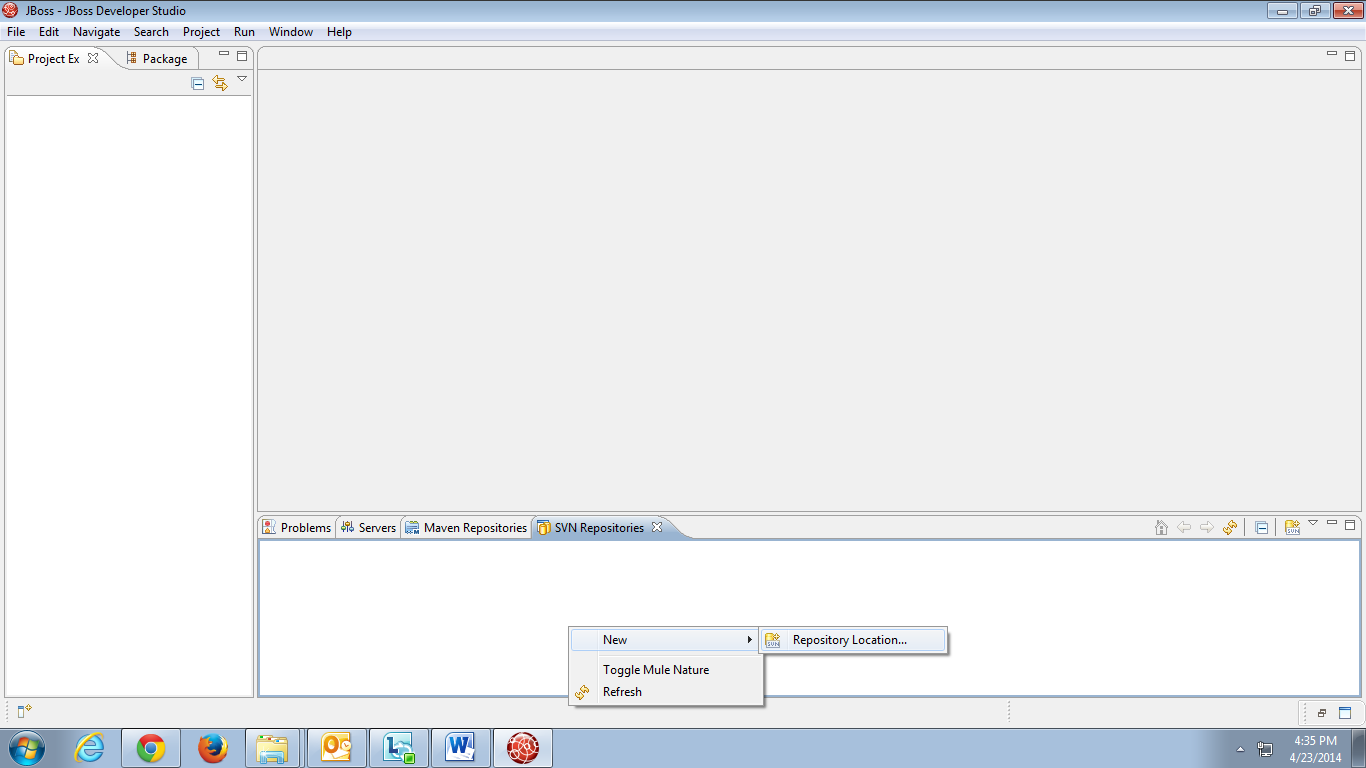
1. Choose the extracted folder and click OK.



1. The features will get listed in the screen.
2. Unselect ‘Contact all update sites during install to find required software’ check box.
3. Click Next.



1. Accept the agreement.
2. Once installed, restart the IDE if prompted.
3. Click on New- Repository Location.



Specify the URL and click Finish.

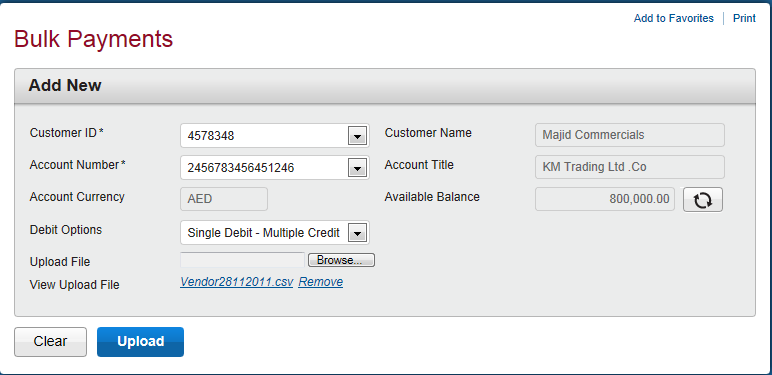
## Web Service Framework

### Prerequisites

1. Make sure you have access to server 10.105.188.30. If not, refer the section [Getting Access](#_Getting_Access_1).

# Presentation Layer Implementation

Below is the sample screen for which the services are identified, developed and explained in this guide.



Follow the instructions described in the below sections to get familiarize with the business layer implementation of First Connect.



## Prerequisites

### Identify Services

The first step is to identify the services that need to be consumed. For the above screen, the below services are required.

1. Service to fetch customer and account details.
2. Service to fetch account details.
3. Service to fetch debit options.
4. Service to accept the uploaded file along with other details.

### Identify Request-Response

The input and output parameters needs to be identified.

1. Customer/Account details Service – Need to fetch customer ids and account numbers for which the logged in user is having access to.

**Request** – User Id

**Response** – List of customer Object with customer Id, name as fields and List of account numbers.

1. Account details Service – Need to fetch the account title, currency and balance from T24.

**Request** – Account Number

**Response** – List of Account object with account details.

1. Debit options Service – Need to fetch the available debit option from database.

**Request** – NA

**Response** – List of debit options

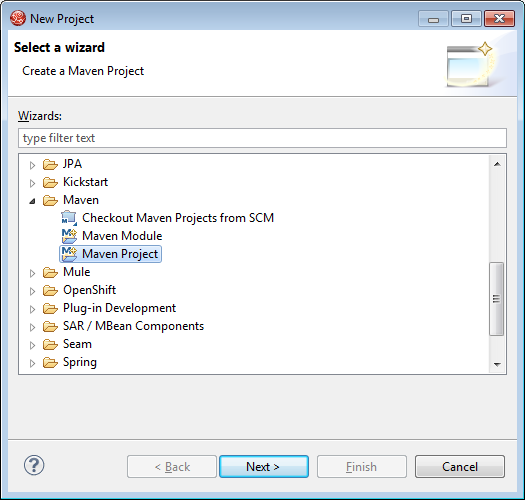
1. Upload file Service – Need to accept the user selected details along with the file uploaded and provide a reference number as response.

**Request** - File, customer id, account number, debit option.

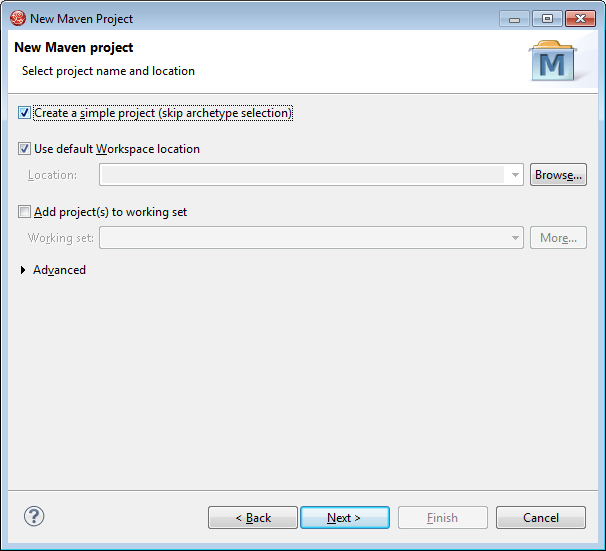
**Response**- Reference number

### [Java Project Creation](#_Web_Project_Creation)

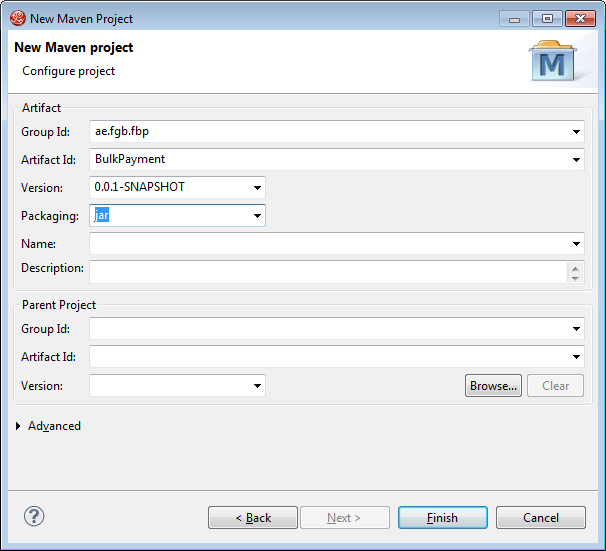
1. Click New – Maven Project.



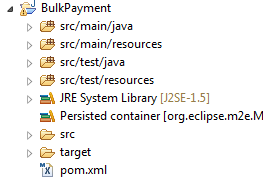
1. Select ‘Create a simple project’.



1. Enter the Group Id, Artifact Id and select Packaging as war.



1. Click Finish and a Java Project is created with below structure.



Similarly create java projects for BulkModel and BulkDao.

### POM Creation

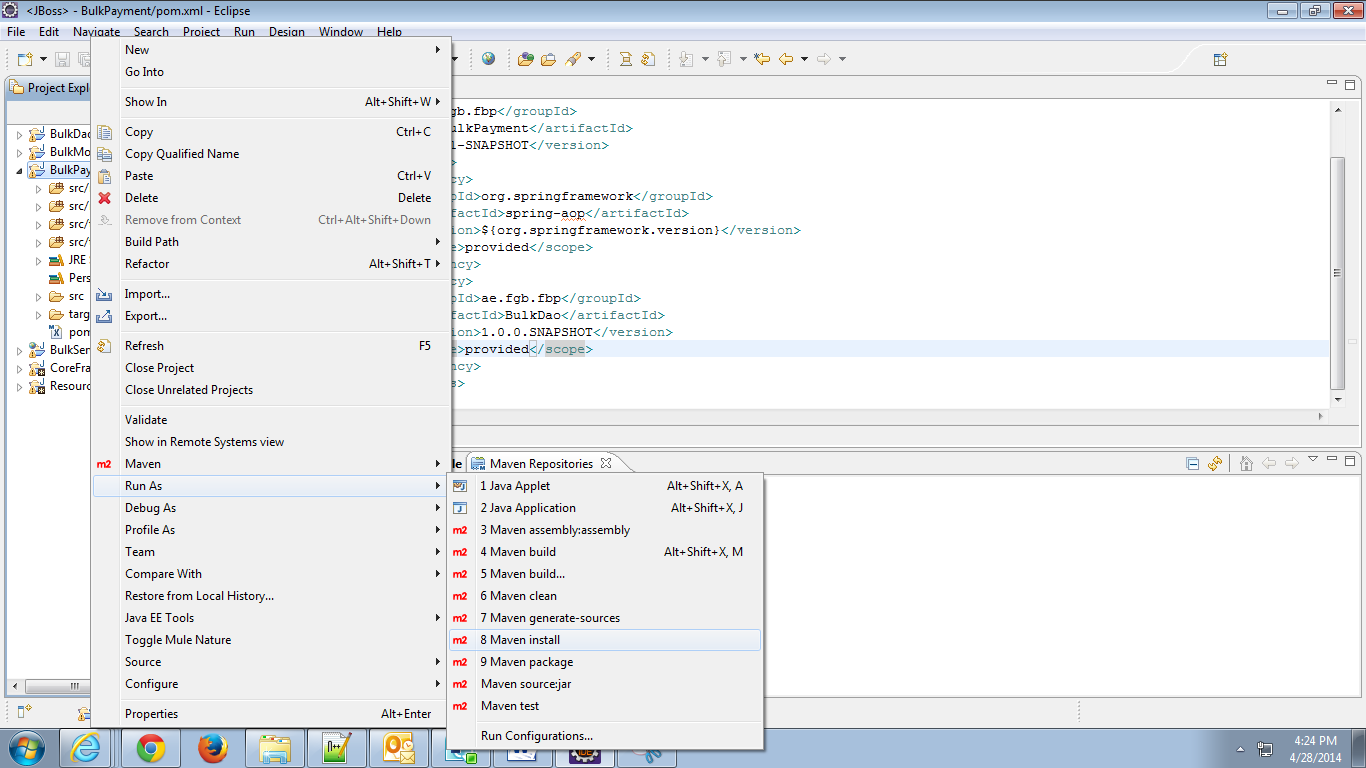
On creating a project, an xml with name pom.xml is created. This is used for managing dependencies automatically using Maven. All the other project dependencies can also be added to this file and hence managed. This include CommonsPL , CoreFramework , model , ResourceBundle , RichFaces , Servlet , Spring , ViewRequestHandler, Security ect..



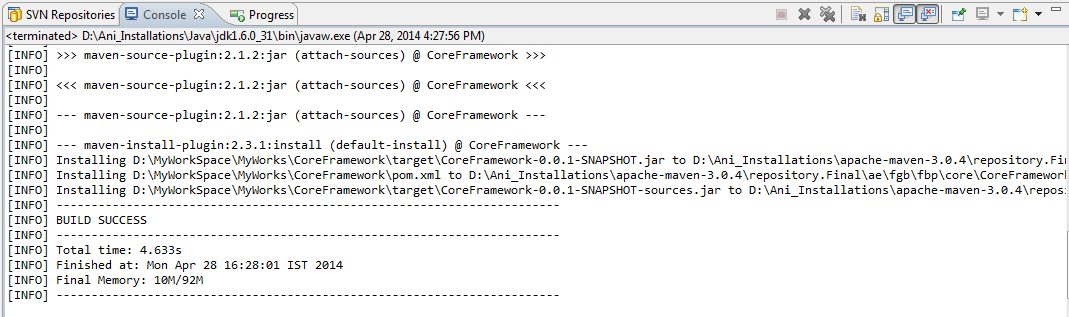
### Building Project

The Maven Project can be built to obtain either the jar or war as per the configuration. This will ensure that all the dependencies are also added.

For building the project, right click-run as – Maven install.



On successful build, it will show as ‘Build Success’.



If any errors occurred, refer Trouble Shooter.

## Configuring the Web Application

### Files Required

### The following files are required.

1. Web.xml – xml having the servlet configurations.
2. web-application-config.xml – For importing the other xml files.
3. Controller – For dispatching the request and response.
4. ViewHelper – creation of request with the values and implementation for UI.
5. Delegate – to call the business methods through endpoint interface by passing the request.
6. End Point – Interface containing the service method signatures
7. ModelVO’s – for Setting and getting the UI values.
8. Templates – Page creations (.xhtml files)
9. Flow.xml – for page navigation

### Servlet Configurations in web.xml

1. Add the below in web.xml which is located in WEB-INF under src/main/webapp.

|  |
| --- |
| <servlet>  <servlet-name>SpringMVCDispatcherServlet</servlet-name>  <servlet-class>  org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value></param-value>  </init-param>  <load-on-startup>2</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>SpringMVCDispatcherServlet</servlet-name>  <url-pattern>/fbp/\*</url-pattern>  </servlet-mapping> |

1. Pass the application Context path as a context param and configure a contextLoaderListener.

|  |
| --- |
| <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>  /WEB-INF/config/web-application-config.xml  </param-value>  </context-param>  <listener>  <listener-class> org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <servlet> |

3.Need to give the welcome file list

|  |
| --- |
| <welcome-file-list>  <welcome-file>index.html</welcome-file>  </welcome-file-list> |

### index.html

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv=*"Refresh"* content=*"0; URL=fbp/loginFlow"*>  </html> |

### web-application-config.xml

### Create an xml, web-application-config.xml in src/main/webapp/WEB-INF/config.

1. The beans instatiation for all classes and web flow registries . Here just import that xml.

|  |
| --- |
| <import resource="webflow-config.xml" />  <import resource="sample-beans-config.xml" /> |

### Bean configuration – sample-beans-config.xml

c

|  |
| --- |
| <bean id=*"sampleController"* class=*"ae.fgb.fbp.pl.controller.payment.SampleController"*>  <property name=*"sampleViewHelper"* ref=*"sampleViewHelper"*></property>  </bean>  <bean id=*"sampleViewHelper"* class=*"ae.fgb.fbp.pl.vhelper.payment.SampleViewHelper"*>  </bean> |

* + - 1. **loginFlow.xml**

1. Create new view view id for login page. On entry in before loading the page we need to call a method in controller called *createScope()- for creating a new flow scope to store userid and password.*

|  |
| --- |
| <view-state id=*"loginView"* view=*"sample/loginFlow.xhtml"* redirect=*"true"*>  <on-entry>  <evaluate expression=*"sampleController.createScope()"*></evaluate>  </on-entry>  <transition on=*"goToNextPage"* to=*"sampleSubFlow"* >  <evaluate expression=*"sampleController.createCustomerReq()"*></evaluate>  </transition>  </view-state> |

1. Create loginFlow.xhtml in flow/sample folder.

|  |
| --- |
| <ui:composition xmlns=*"http://www.w3.org/1999/xhtml"*  xmlns:h=*"http://java.sun.com/jsf/html"*  xmlns:f=*"http://java.sun.com/jsf/core"*  xmlns:ui=*"http://java.sun.com/jsf/facelets"*  xmlns:a4j=*"http://richfaces.org/a4j"*  xmlns:rich=*"http://richfaces.org/rich"*  template=*"../../template/sample/LoginTemplate.xhtml"*>  </ui:composition> |

1. create LoginTemplate.xhtml in template/sample

This h:inputText is a conversion of <input type= “text”> in HTML. This LoginScope is the one which is a flow scope created in java file. This is called the backing bean in JSF. This userId is the attribute in VO. Which may be a getter or setter.

|  |
| --- |
| h:inputText styleClass="tboxMed" value="#{LoginScope.userId}" /> |

|  |
| --- |
| <a4j:commandButton value=*"Login* action=*"#{sampleViewHelper.validateCredientials()}"* oncomplete=*"#{LoginScope.valid?'nextPage()':'error()'}"*>  </a4j:commandButton>  <a4j:jsFunction name=*"nextPage"* action=*"goToNextPage"* />  <script>  **function** error() {  alert('Please enter user id and password');  }  </script> |

1. Java method in controller for creating this scope called LoginScope.

|  |
| --- |
| **public** **void** createSampleScope() {  sampleViewHelper.createSampleScope();  } |

1. Creating viewHelper method createSampleScope().

|  |
| --- |
| **public** **void** createSampleScope() {  **SampleVO** sampleVO = **new** **SampleVO**();  **RequestContextHolder**.getRequestContext().getFlowScope()  .put("SampleScope", sampleVO);  } |

3) If an action button is pressed , this transition on will happen and transition to is another view id which is a subflow here.

|  |
| --- |
| <subflow-state id="sampleSubFlow" subflow="sampleFlow"></subflow-state> |

* + - 1. **sampleFlow.xml**

The same way of loginFlow we need to create this xml

1. In this we have MainTemplate which is the IFrame Template to load all the templates.

|  |
| --- |
| <view-state id=*"MainTemplateView"* view=*"sample/Main.xhtml"* redirect=*"true"*>  </view-state> |

Bellow attacments for IFrame Template insample/ flow and sample/Template.

** **

1. This ia the view id for the next page after login page which is the bulk payments page**.**

|  |
| --- |
| <view-state id=*"sample"* view=*"sample/sampleFlow.xhtml"* redirect=*"true"*>  <on-entry>  <evaluate expression=*"sampleController.createCustomerReq()"*></evaluate>  </on-entry>  </view-state> |

### SampleController.java

* Just for an example to get customer response. Create a method named createCustomerReq

|  |
| --- |
| **public** **void** createCustomerReq() **throws** BaseException, Throwable {  **GenericRequest** genericRequest = sampleViewHelper.createGenericRequest();  **GenericResponse** genericResponse = sampleDelegate  .getCustomerResponse(genericRequest);  sampleViewHelper.setCutomerResp(genericResponse);  } |

* This file should contain only the request and response creation method and no implementation. All implementations should be done in viewHelper

1. SampleViewHelper.java

* Preparing the request based on the signature.

|  |
| --- |
| **public** **GenericRequest** createCustReq(**String** custId) **throws** BaseException,  Throwable {  **GenericRequest** genericRequest = **new** **GenericRequest**();  **ServiceContext** serviceContext = setServiceContext();  **String** userID = serviceContext.getUserContext().getUserId();  genericRequest.setUserId(userID);  **return** genericRequest;  } |

Setting the response values based on the application.

|  |
| --- |
| **public** **void** setCutomerResp(**GenericResponse** genericResponse) {  List<**String**> custIds = **new** **ArrayList**<**String**>();  **CustomerInfoVO** customerInfo = **new** **CustomerInfoVO**();  **RequestContextHolder**.getRequestContext().getFlowScope()  .put("CustomerResponse", genericResponse);  **if** (genericResponse != **null** && genericResponse.getCustomers() != **null**  && !genericResponse.getCustomers().isEmpty()) {  **for** (**FbpCust** fbpcust : genericResponse.getCustomers()) {  **if** (!**StringUtils**.isEmpty(fbpcust.getCustId()))  custIds.add(fbpcust.getCustId());  }  custIds.add(0, "Please Select");  customerInfo.setCustIds(custIds);  customerInfo.setSelCustID(custIds.get(0));  }  **RequestContextHolder**.getRequestContext().getFlowScope()  .put("CustomerDetails", customerInfo);  } |

1. SampleDelegate.java

|  |
| --- |
| **public** **GenericResponse** getCustomerDetails(**GenericRequest** genericRequest) **throws** Throwable {  **GenericResponse** genericResponse = bulkEndPoint.getCustomers(genericRequest);  **return** genericResponse;  } |

1. SampleVo

|  |
| --- |
| private String userId ;  **public** **String** getUserId() {  **return** userId;  }  **public** **void** setUserId(**String** userId) {  **this**.userId = userId;  } |

1. BulkEndPoint.java

|  |
| --- |
| @WebMethod  @WebResult(name = "CustResponse")  **GenericResponse** getCustomers(  @WebParam(name = "CustRequest") **GenericRequest** request)  **throws** Throwable; |

# Trouble Shooter

## Build Management

1. **On building a project, if an exception of ‘dependency cannot be resolved’ comes**

* Check the local repository for the jar file.
* If found in system, rebuild the repository index in eclipse.
* If not found, check the proxy and build the project again. The jar should get downloaded.
* If not downloaded, check the internet settings in eclipse that it points to proxy.
* If not, download manually and place it in repository.

## Application Servers

1. **Address already bind exception**

* Check whether the server is already running. If yes, stop it.
* Check if any other services are running in that port. If yes, change the server’s port and then restart it.
* If no services are running but still the exception comes, execute the below command in command prompt.

netstat -n -a –o

* It will list out all the services running. Find out the PID of the service running in the port specified. Kill it as below.

taskkill /pid <PID>

Note: Kill the process only if you are sure that it is not required.

1. **Perm gen space**

* Increase the xmx and xms values in server configuration.

1. **Error opening in zip file**

* Stop the server.
* Remove the deployed war/ear.
* Deploy it again.
* Restart the server.

1. **Certificate Exception**

* Check whether the required certificate is present.
* Check whether the certificate is not expired.
* Check whether the certificate location is pointed properly.

## Deployment

1. **Marshalling exception**

* Check whether the same endpoints, requests, response are used in both server and client.

1. **Could not send message**

* Check whether the server is up and running.
* Check whether url pointing to is correct.
* Check whether the server is reachable from your side.
* Check whether any firewall is blocking.

1. **Could not resolve a binding when exposing web service**

* Check whether the cxf jars are available in jboss lib.