**Integrating Apache CXF, JAX-RS and Spring**

**Author**: Horatiu Stanciu

**Email:** [hstanciu@gmail.com](mailto:hstanciu@gmail.com)

**Date:** 2013-03-22

In this article we will build a simple REST application using Apache CXF, JAX-RS and integrated with Spring. As build tool we will be using Maven.

I used for this article the following tools:

Eclipse Luna 4.4.2

Tomcat 7.0.59

JDK 1.7

Maven 3.0

To understand the article, you’ll need Java, JEE, REST web service and Spring knowledge.

This REST web service will model a house hold payments as CRUD operations.

Once the tools are properly installed, we’ll do these steps:

1. Create a web application
2. Import the web application in Eclipse and configure
3. Configure the Maven POM file
4. Create Java artifacts needed by the web service
5. Create and configure the Spring application context
6. Configure web.xml
7. Deploy and use the service
8. **Create a web application**
9. On hard disk, move to a folder where you will create the project. For me, it was **C:\java\projects\webServiceWorkspace**
10. Create the project skeleton using the following maven command:

*mvn archetype:generate -DgroupId=biz.korwin -DartifactId=home-web-service -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false*

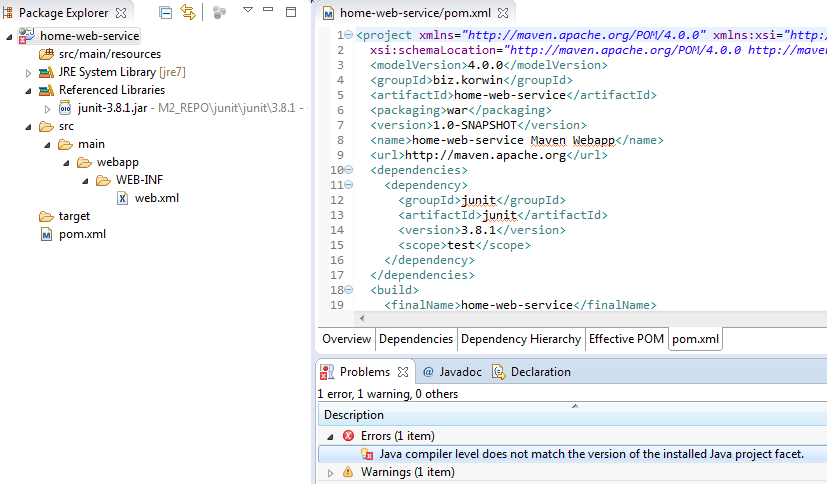
This would create a basic web application structure. The project will be contained in **home-web-service** folder.

1. **Import the web application in Eclipse**
2. Now in order to quickly import this application in Eclipse run this in **home-web-service** folder:

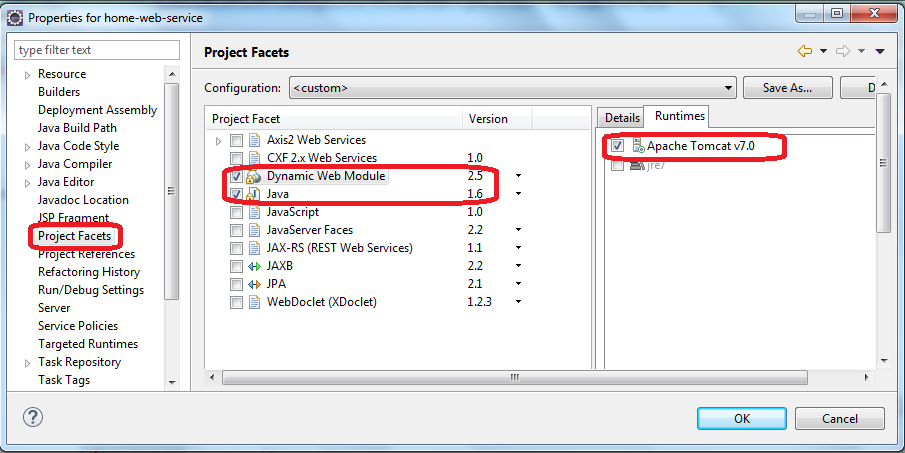
*mvn eclipse:eclipse -Dwtpversion=2.0*

This will generate the **.project** and **.classpath** Eclipse files.

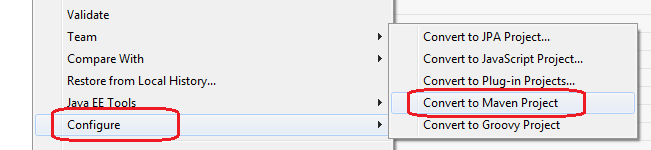
1. Import the application in Eclipse. Remove the **index.jsp** file (located in **home-web-service\src\main\webapp**) since we will not need this file. The project structure at this step would look like this:



1. We still have to make couple of settings. Choose a proper JDK, Dynamic Web Module 2.5 and select an Apache Runtime.



1. Convert the Eclipse project in a Maven project:



1. **Configure the Maven POM file**

Replace the existing **pom.xml** with this one. This file contains the CXF, Spring, Servlet and Jackson (for JSON transformation) dependencies:

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>biz.korwin</groupId>

<artifactId>home-web-service</artifactId>

<packaging>war</packaging>

<version>1.0-SNAPSHOT</version>

<name>home-web-service Maven Webapp</name>

<url>http://maven.apache.org</url>

<properties>

<cxf.version>2.7.15</cxf.version>

<spring.version>4.1.3.RELEASE</spring.version>

<jackson.version>1.9.13</jackson.version>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.codehaus.jackson</groupId>

<artifactId>jackson-jaxrs</artifactId>

<version>${jackson.version}</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<scope>provided</scope>

<version>2.5</version>

</dependency>

<dependency>

<groupId>org.apache.cxf</groupId>

<artifactId>cxf-bundle-jaxrs</artifactId>

<version>${cxf.version}</version>

</dependency>

</dependencies>

<build>

<finalName>home-web-service</finalName>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.2</version>

<configuration>

<source>1.6</source>

<target>1.6</target>

</configuration>

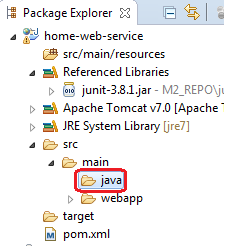
</plugin>

</plugins>

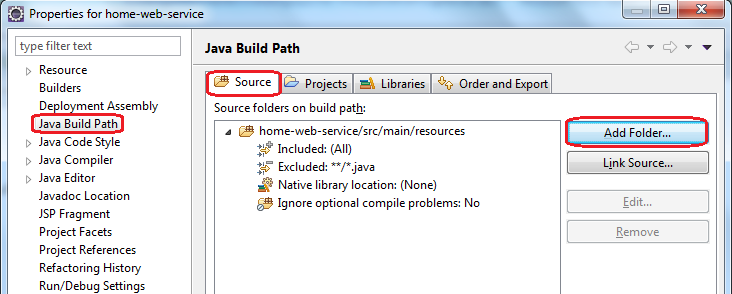
</build>

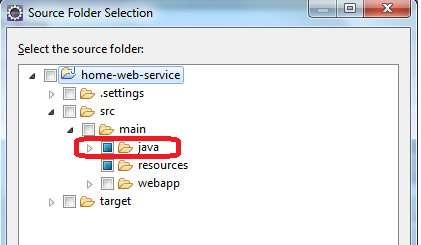
</project>

1. **Create Java artifacts needed by the web service**
2. Create the java folder (is missing because of a bug in Maven archetype)



Add this folder to the project sources:





1. Create the package (for example) **biz.korwin.web.service.home** as base package and

**biz.korwin.web.service.home.model** as the model package

1. Create in **biz.korwin.web.service.home.model** the following model classes:

**PaymentVO.java**

**package** biz.korwin.web.service.home.model;

**import** java.util.Date;

**import** javax.ws.rs.FormParam;

**import** javax.xml.bind.annotation.XmlAccessorType;

**import** javax.xml.bind.annotation.XmlAccessType;

**import** javax.xml.bind.annotation.XmlRootElement;

/\*\*

\* Model class. Represents a payment.

\*

\* \*/

@XmlRootElement(name="payment")

@XmlAccessorType(XmlAccessType.***FIELD***)

**public** **class** PaymentVO {

/\* Payment ID\*/

**private** **long** id;

/\* Who will be paid; ex: Canada Bank\*/

**private** String payee;

/\* The payment product; ex: Master Card Petro-Points \*/

**private** String payeeProduct;

/\* The number of days a new statement will be issued, or -1 if is not cyclical\*/

**private** **int** cycle;

/\* Amount to be paid (in cents) \*/

**private** **int** amountInCents;

/\* Due date \*/

**private** String dueDate;

/\* In order for payment to occur at due date, the payment should occur

\* in advance with a couple of days. This date is the limit date when

\* this occurs in order for due date not to be passed.\*/

**private** String paymentLimitDate;

/\* Date when payment happened. If it didn't happen yet,

\* then this attribute will be null.\*/

**private** String paidDate;

**public** **long** getId() {

**return** id;

}

//@FormParam("id")

**public** **void** setId(**long** id) {

**this**.id = id;

}

**public** String getPayee() {

**return** payee;

}

@FormParam("payee")

**public** **void** setPayee(String payee) {

**this**.payee = payee;

}

**public** String getPayeeProduct() {

**return** payeeProduct;

}

@FormParam("payeeProduct")

**public** **void** setPayeeProduct(String payeeProduct) {

**this**.payeeProduct = payeeProduct;

}

**public** **int** getCycle() {

**return** cycle;

}

@FormParam("cycle")

**public** **void** setCycle(**int** cycle) {

**this**.cycle = cycle;

}

**public** **int** getAmountInCents() {

**return** amountInCents;

}

@FormParam("amountInCents")

**public** **void** setAmountInCents(**int** amountInCents) {

**this**.amountInCents = amountInCents;

}

**public** String getDueDate() {

**return** dueDate;

}

@FormParam("dueDate")

**public** **void** setDueDate(String dueDate) {

**this**.dueDate = dueDate;

}

**public** String getPaymentLimitDate() {

**return** paymentLimitDate;

}

@FormParam("paymentLimitDate")

**public** **void** setPaymentLimitDate(String paymentLimitDate) {

**this**.paymentLimitDate = paymentLimitDate;

}

**public** String getPaidDate() {

**return** paidDate;

}

@FormParam("paidDate")

**public** **void** setPaidDate(String paidDate) {

**this**.paidDate = paidDate;

}

}

This class contains couple of attributes linked to a payment.

**PaymentsVO.java**

**package** biz.korwin.web.service.home.model;

**import** java.util.List;

**import** javax.xml.bind.annotation.XmlAccessorType;

**import** javax.xml.bind.annotation.XmlAccessType;

**import** javax.xml.bind.annotation.XmlRootElement;

/\*\*

\* This is a way to deal with returning an array in the response. The array

\* of PaymentVOs is wrapped inside a PaymentsVO.

\* \*/

@XmlRootElement(name="paymentsList")

@XmlAccessorType(XmlAccessType.***FIELD***)

**public** **class** PaymentsVO {

**public** List<PaymentVO> payments;

**public** List<PaymentVO> getPayments() {

**return** **this**.payments;

}

**public** **void** setPayments(List<PaymentVO> payments) {

**this**.payments = payments;

}

}

This class is a container for a list of payments.

**IPaymentWS.java**

**package** biz.korwin.web.service.home;

**import** javax.ws.rs.BeanParam;

**import** javax.ws.rs.Consumes;

**import** javax.ws.rs.DELETE;

**import** javax.ws.rs.GET;

**import** javax.ws.rs.POST;

**import** javax.ws.rs.PUT;

**import** javax.ws.rs.Path;

**import** javax.ws.rs.PathParam;

**import** javax.ws.rs.Produces;

**import** javax.ws.rs.core.MediaType;

**import** javax.ws.rs.core.Response;

**import** biz.korwin.web.service.home.model.PaymentVO;

**import** biz.korwin.web.service.home.model.PaymentsVO;

@Path("/payments")

**public** **interface** IPaymentWS {

/\* Returns the list of payments that have the same payment limit date. \*/

@GET

@Consumes({MediaType.***APPLICATION\_JSON***, MediaType.***APPLICATION\_XML***})

@Produces({MediaType.***APPLICATION\_JSON***, MediaType.***APPLICATION\_XML***})

@Path("/date/{date}")

**public** PaymentsVO getPaymentsByDate(@PathParam("date") String date);

/\* Returns the list of payments that have the same payment limit date. \*/

@GET

@Consumes({MediaType.***APPLICATION\_JSON***, MediaType.***APPLICATION\_XML***})

@Produces({MediaType.***APPLICATION\_JSON***, MediaType.***APPLICATION\_XML***})

@Path("/{id}")

**public** PaymentVO getPaymentById(@PathParam("id") **long** id);

/\* Creates a payment. \*/

@POST

@Consumes({MediaType.***APPLICATION\_FORM\_URLENCODED***})

@Produces({MediaType.***APPLICATION\_JSON***, MediaType.***APPLICATION\_XML***})

**public** PaymentVO create(@BeanParam PaymentVO newPayment);

/\* Deletes a payment. \*/

@DELETE

@Path("/{id}")

@Produces({MediaType.***APPLICATION\_JSON***, MediaType.***APPLICATION\_XML***})

**public** Response delete(@PathParam("id") **long** id);

/\* Updates a payment. \*/

@PUT

@Path("/{id}")

@Produces({MediaType.***APPLICATION\_JSON***, MediaType.***APPLICATION\_XML***})

**public** PaymentVO update(@PathParam("id") **long** id);

}

Interface for our web services. Contains CRUD operations for payments.

**PaymentWSImpl.java**

The implementation for this class is dummy. You can add a service or DAO layer that can do really useful stuff.

**package** biz.korwin.web.service.home;

**import** java.text.SimpleDateFormat;

**import** java.util.ArrayList;

**import** java.util.List;

**import** javax.ws.rs.core.Response;

**import** javax.ws.rs.core.Response.Status;

**import** org.springframework.stereotype.Component;

**import** biz.korwin.web.service.home.model.PaymentVO;

**import** biz.korwin.web.service.home.model.PaymentsVO;

/\*\*

\* Web service implementation.

\* \*/

@Component("paymentsService")

**public** **class** PaymentWSImpl **implements** IPaymentWS {

SimpleDateFormat formatter = **new** SimpleDateFormat("yyyyMMdd");

@Override

**public** PaymentsVO getPaymentsByDate(String date) {

PaymentsVO payments = **new** PaymentsVO();

List<PaymentVO> paymentsList = **new** ArrayList<PaymentVO>();

PaymentVO payment1 = **new** PaymentVO();

payment1.setAmountInCents(12344);

payment1.setCycle(22);

payment1.setDueDate("20150330");

payment1.setPaidDate(**null**);

payment1.setPayee("CIBC");

payment1.setPayeeProduct("Master Card Petro-Points");

payment1.setPaymentLimitDate("20150328");

PaymentVO payment2 = **new** PaymentVO();

payment2.setAmountInCents(1234);

payment2.setCycle(25);

payment2.setDueDate("20150330");

payment2.setPaidDate(**null**);

payment2.setPayee("RBC");

payment2.setPayeeProduct("VISA Gold");

payment2.setPaymentLimitDate("20150328");

paymentsList.add(payment1);

paymentsList.add(payment2);

payments.setPayments(paymentsList);

**return** payments;

}

@Override

**public** PaymentVO create(PaymentVO newPayment) {

**return** newPayment;

}

@Override

**public** Response delete(**long** id) {

**return** Response.*status*(Status.***OK***).build();

}

@Override

**public** PaymentVO update(**long** id) {

PaymentVO payment = **new** PaymentVO();

payment.setId(id);

payment.setAmountInCents(12344);

payment.setCycle(22);

payment.setDueDate("20150330");

payment.setPaidDate(**null**);

payment.setPayee("CIBC");

payment.setPayeeProduct("Master Card Petro-Points");

payment.setPaymentLimitDate("20150328");

**return** payment;

}

@Override

**public** PaymentVO getPaymentById(**long** id) {

PaymentVO payment = **new** PaymentVO();

payment.setId(id);

payment.setAmountInCents(12344);

payment.setCycle(22);

payment.setDueDate("20150330");

payment.setPaidDate(**null**);

payment.setPayee("CIBC");

payment.setPayeeProduct("Master Card Petro-Points");

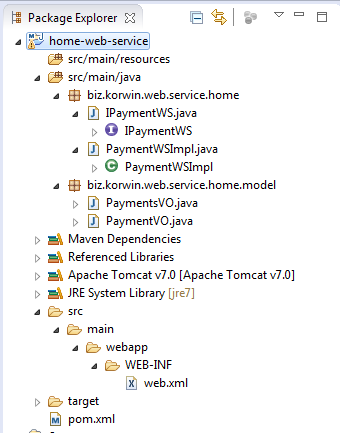
payment.setPaymentLimitDate("20150328");

**return** payment;

}

}

The Java sources would look like this:



1. **Create and configure the Spring application context**

Create the **applicationContext.xml** in WEB-INF

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

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:jaxrs=*"http://cxf.apache.org/jaxrs"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://cxf.apache.org/jaxrs http://cxf.apache.org/schemas/jaxrs.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.1.xsd"*>

<import resource=*"classpath:META-INF/cxf/cxf.xml"*/>

<import resource=*"classpath:META-INF/cxf/cxf-servlet.xml"*/>

<context:component-scan base-package=*"biz.korwin"*/>

<jaxrs:server id=*"restfulId"* address=*"/rest"*>

<jaxrs:serviceBeans>

<ref bean=*"paymentsService"*/>

</jaxrs:serviceBeans>

<jaxrs:providers>

<bean id=*"jacksonProvider"* class=*"org.codehaus.jackson.jaxrs.JacksonJsonProvider"*/>

</jaxrs:providers>

</jaxrs:server>

</beans>

1. **Configure web.xml**

Modify the **web.xml** file from WEB-INF folder:

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

<web-app xmlns=*"http://java.sun.com/xml/ns/javaee"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*

version=*"2.5"*>

<display-name>Payments Web Service</display-name>

<servlet>

<servlet-name>CXF Servlet</servlet-name>

<servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>CXF Servlet</servlet-name>

<url-pattern>/rest/\*</url-pattern>

</servlet-mapping>

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/applicationContext.xml</param-value>

</context-param>

</web-app>

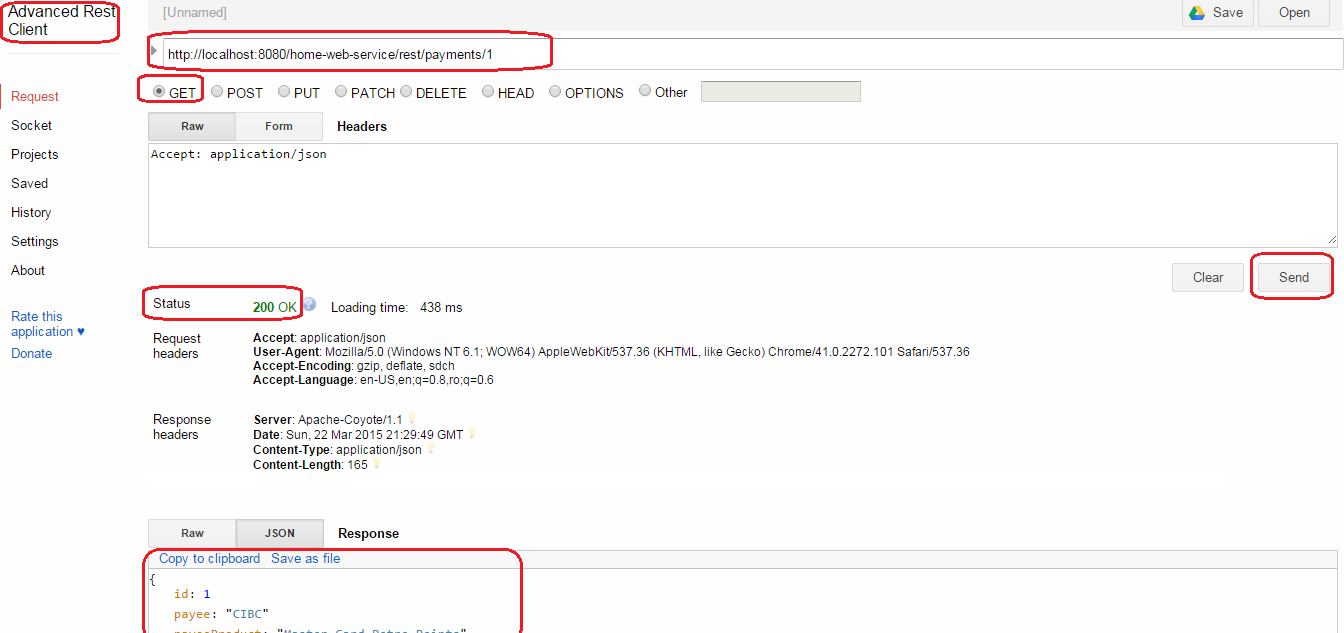
1. **Deploy and use the service**

In order to build and package the web service go to **C:\java\projects\webServiceWorkspace** and type

*mvn clean package*

This will create in **target** folder home-web-service.war file. Take this file and drop it into Tomcat’s **webapps** folder. Restart Tomcat.

For testing I used Google’s Chrome and I installed a Google application called Advanced Rest Client. Any REST client that allows you to send different types of HTTP requests, would do it.



This is a GET request that will trigger **IPaymentsWS**’s

public PaymentVO getPaymentById(@PathParam("id") long id);

Similar with this you can make PUT and DELETE calls. For POST, you can call the web service this way:

