## SHORTCUT

<http://bit.ly/be-clipboard>

# IntelliJ Setup

<http://bit.ly/intellij-setup>

## SETUP LOGGING

1. Open the file:
   1. configuration/src/main/resources/logback.xml
2. Include a <logger> in it:
   1. <logger name="com.backbase.training" level="DEBUG" />

## PROTECT SERVICES

1. Edit the portalserver/src/main/resources/META-INF/spring/backbase-portal-presentation-security.xml file.
2. And add the following interceptor in the default <http> Spring Security element:

<intercept-url pattern="/services/\*\*" access="hasAnyRole('role\_admin', 'role\_manager', 'role\_user')"/>

## EXERCISE PROPERTIES

configuration/src/main/resources/backbase.properties

#Training Server Host  
training.server.host=${training.server.host}  
training.server.http.port=${training.server.http.port}  
training.server.mq.port=${training.server.mq.port}

configuration/src/main/filters/local.properties

#Training Server Host  
training.server.host=localhost  
training.server.http.port=9999  
training.server.mq.port=61616

## REBUILD CXP

Go to the root of your CXP project and execute:

mvn clean install

## 

## 

## INSTALLING TRAINING SERVER

<https://github.com/Backbase/training-server>

## INSTALL JOLOKIA

1. Jolokia is a JMX-HTTP bridge
2. Download the WAR-Agent from:
   1. <https://jolokia.org/download.html>
3. Copy the war file to the webapps folder
4. Edit the webapps/pom.xml file, including the following content (in red). Make sure you are using the correct filename (in this example, jolokia-war-1.3.5.war):

|  |
| --- |
| Sample webapp/pom.xml  ...  </webApp>  <!-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* -->  <!-- NOTE: this must be outside the <webApp> tag -->  <!-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* -->  <contextHandlers>       <contextHandler implementation="org.eclipse.jetty.maven.plugin.JettyWebAppContext">           <war>${project.basedir}/../jolokia-war-1.3.5.war</war>           <contextPath>/jolokia</contextPath>       </contextHandler>  </contextHandlers>  </configuration>  <dependencies>  ... |

1. Start the portalserver, contentservices and orchestrator web servers
2. Use Hawtio
   1. To use the hawtio just access the URL:
      1. <http://localhost:9999/hawtio/>
   2. Click on Connect tab
      1. Connecting to the Portal Server
         1. Name: Portal Server
         2. Scheme: http
         3. Host: localhost
         4. Port: 7777
         5. Path: jolokia
         6. User name: <leave it blank>
         7. Password: <leave it blank>
         8. Use proxy: <does not matter - using localhost>
         9. Click on Save
            1. It will become available in the Saved Connections for future use
         10. Click on Connect to remote server
      2. Connecting to the Content Services or Orchestrator
         1. Name: <an appropriate name here>
         2. Port: 8082 (Content Services) or 8083 (Orchestrator)
         3. The remaining fields are exactly the same
   3. Now you can just start using the hawtio connected to the selected server

## INSTALLING THE TRAINING WIDGETS

<https://github.com/Backbase/training-be-module-01/tree/master/training-collection>

## STOCK PRICE QUERY

### CREATE A SERVICE USING ARCHETYPE

1. We will create a simple service to return the stock price
   1. Service to consume. Some samples:
      1. Yahoo: <https://www.google.com/finance/info?q=YHOO>
      2. Apple: <https://www.google.com/finance/info?q=AAPL>
      3. Google: <https://www.google.com/finance/info?q=GOOGL>
      4. Amazon: <https://www.google.com/finance/info?q=AMZN>
      5. Microsoft: <https://www.google.com/finance/info?q=MSFT>
2. Reference:
   1. Visit:
      1. <http://my.backbase.com>
   2. Select:
      1. Docs -> Product Documentation -> CXP Documentation
   3. And then:
      1. Development Guide -> Integration Services -> How To Create an Integration Service With Camel Spring XML DSL -> Generate Project
3. Steps:
   1. Go to the services directory:
      1. cd services
   2. Generate the service from the archetype:
      1. Execute the command:
         1. mvn archetype:generate -DarchetypeArtifactId=integration-service-spring-archetype -DarchetypeGroupId=com.backbase.portal.archetypes -DarchetypeVersion=5.6.2.0
      2. And answer the following:
         1. groupId = com.backbase.training
         2. artifactId = stock-price-service
         3. version = (leave the default value)
         4. package = (leave the default value)
      3. Open the services/pom.xml file
         1. Check if stock-price-service is included as a <module>
   3. Open the generated Camel XML file:
      1. Location:
         1. services/stock-price-service/src/main/resources/META-INF/spring/backbase-integration-service.xml
      2. Add the following route:

<camel:routeContext id="com.backbase.portal.integration.service.StockPriceService"

xmlns="http://camel.apache.org/schema/spring">

<route id="com.backbase.training.StockPriceRoute">

<from uri="restlet:/stock/{code}/price?restletMethods=GET" />

<log message="Code received: ${in.header.code}" loggingLevel="DEBUG" />

<removeHeaders pattern="CamelHttp\*" excludePattern="CamelHttpMethod"/>

<setHeader headerName="CamelHttpQuery">

<simple>q=${in.header.code}</simple>

</setHeader>

<to uri="https://www.google.com/finance/info?throwExceptionOnFailure=false"/>

<choice>

<when>

<simple>${header.CamelHttpResponseCode} == 200</simple>

<convertBodyTo type="String"/>

<transform>

<simple>${body.replaceAll("//", "")}</simple>

</transform>

<unmarshal>

<json library="Jackson" unmarshalTypeName="java.util.List"/>

</unmarshal>

<log message="body: ${body}" loggingLevel="DEBUG"/>

<setHeader headerName="price">

<simple>${body.get(0)[l]}</simple>

</setHeader>

</when>

<when>

<simple>${header.CamelHttpResponseCode} == 400</simple>

<setHeader headerName="CamelHttpResponseCode">

<constant>200</constant>

</setHeader>

<setHeader headerName="price">

<constant>null</constant>

</setHeader>

</when>

</choice>

<setBody>

<simple>{"code" : "${header.code}", "price" : ${header.price}}</simple>

</setBody>

<log id="com.backbase.training.content.log.body" message="body: ${body}" loggingLevel="DEBUG"/>

</route>

</camel:routeContext>

1. Add camel-jackson dependency
   1. Open services/stock-price-service/pom.xml
   2. Add the following <dependency>

<dependency>  
 <groupId>org.apache.camel</groupId>  
 <artifactId>camel-jackson</artifactId>  
 <version>2.12.4</version>  
</dependency>

1. Include dependency in webapps/portalserver/pom.xml

<dependency>

<groupId>com.backbase.training</groupId>

<artifactId>stock-price-service</artifactId>

<version>1.0-SNAPSHOT</version>

</dependency>

1. Restart Portal
   1. Build configuration, services, & portal
   2. Run portal
   3. Test the service:
      1. <http://localhost:7777/portalserver/services/rest/stock/YHOO/price>

### INSTALL THE STOCK PRICE QUERY WIDGET

1. Download the Stock Price Query Widget:
   1. <https://drive.google.com/file/d/0B7dzciN5c-MwbVJnc1hVVWpIR3c/view?usp=sharing>
2. Go to the Enterprise Catalog
3. Import the stock-price-widget-v2.zip file
4. Import it to the Portal Catalog
5. Use it in some page based on Launchpad
6. Test the widget

## REGISTER PLAYER

1. Go to the services directory
2. Generate the service from the archetype:
   1. Execute the command:
      1. mvn archetype:generate -DarchetypeArtifactId=integration-service-spring-archetype -DarchetypeGroupId=com.backbase.portal.archetypes -DarchetypeVersion=5.6.2.0
   2. And answer the following:
      1. groupId = com.backbase.training
      2. artifactId = player-service
      3. version = (leave the default value)
      4. package = (leave the default value)
   3. Open the services/pom.xml file
      1. Check if player-service is included as a <module>
3. Open the generated Camel XML file:
   1. Location:
      1. services/player-service/src/main/resources/META-INF/spring/backbase-integration-service.xml
   2. Add the following route:

<**route id="com.backbase.training.PlayerService"**>  
 <**from uri="restlet:/player/register?restletMethods=POST"** />  
  
 <**convertBodyTo type="String"**/>  
 <**log message="headers: ${in.headers}" loggingLevel="DEBUG"** />  
  
 <**removeHeaders pattern="CamelHttp\*" excludePattern="CamelHttpMethod"**/>  
  
  
 <**to uri="http://localhost:9999/training-server/rest/player"**/>  
  
 <**convertBodyTo type="String"**/>  
 <**log message="body: ${body}" loggingLevel="DEBUG"**/>  
</**route**>

1. Include dependency in webapps/portalserver/pom.xml

<dependency>

<groupId>com.backbase.training</groupId>

<artifactId>player-service</artifactId>

<version>1.0-SNAPSHOT</version>

</dependency>

1. Restart Portal
   1. Build configuration, services, & portal
   2. Run portal
2. Test the service using Postman
   1. Method:
      1. POST
   2. URL:
      1. <http://localhost:7777/portalserver/services/rest/player/register>
   3. In Authorization tab:
      1. Type:
         1. Basic Auth
      2. Username:
         1. admin
      3. Password:
         1. admin
      4. Click on Update Request button
   4. In Headers tab:
      1. Set a JSON content type header
         1. Key = content-Type
         2. Value = application/json
   5. In Body tab:
      1. Select raw
      2. Include a valid player JSON. See sample below:

{  
 "username": "bruce",  
 "fullname": "Bruce Lee",  
 "password": "bruce",  
 "birthDay": "1940-11-27"  
}

* 1. Click on Send button
  2. Check if the Status is 200 OK
  3. Add more players as you wish

1. Test the widget
   1. Add the 2048 widgets to the Portal Catalog
   2. Create a new page
   3. Include the 2048 Register Widget in the page
   4. Add players using the page
2. Check if the player was really registered in the training server:

The players you are registering must be listed in this URL:

<http://localhost:9999/training-server/rest/player/players>

## LIST PLAYERS

1. Add this bean in the player-service project

<bean id="sortProcessor" class="com.backbase.training.processor.PlayerSortProcessor"/>

1. Add this route in the player-service project

<route id="com.backbase.training.PlayersListRoute">  
 <from uri="restlet:/player/list?restletMethods=GET&amp;restletBinding=#queryStringToHeadersRestletBinding" />  
  
 <convertBodyTo type="String"/>  
  
 <removeHeaders pattern="CamelHttp\*" excludePattern="CamelHttpMethod"/>  
  
 <to uri="http://localhost:9999/training-server/rest/player/players" />  
  
 <convertBodyTo type="String"/>  
  
 <log message="body before unmarshall: ${body}" loggingLevel="DEBUG"/>  
  
 <unmarshal>  
 <json library="Jackson" unmarshalTypeName="com.backbase.training.to.Players"/>  
 </unmarshal>  
  
 <log message="body after unmarshall: ${body}" loggingLevel="DEBUG"/>  
  
 <process ref="sortProcessor"/>  
  
 <log message="body after sortProcessor: ${body}" loggingLevel="DEBUG"/>  
  
 <marshal>  
 <json library="Jackson"/>  
 </marshal>  
  
 <log message="body: ${body}"/>  
  
 </route>

1. Create the Player class

package com.backbase.training.to;

public class Player {

private Integer id;

private Integer uid;

private String username;

private String fullname;

private String password;

private String emailAddress;

private String profileUrl;

private String photoUrl;

private String thumbnalUrl;

private String gender;

private Long birthDay;

private String firstName;

private String lastName;

private String middleName;

private Integer highScore;

// implement getter and setters here

}

1. Create the Players class

package com.backbase.training.to;

import java.util.ArrayList;

import java.util.List;

public class Players {

private List<Player> players;

public Players() {

setPlayers(new ArrayList<Player>());

}

public List<Player> getPlayers() {

return players;

}

public void setPlayers(List<Player> players) {

this.players = players;

}

}

1. Create the PlayerSortProcessor (Java 8 - using lambda)

package com.backbase.training.processor;  
  
import com.backbase.training.to.Player;  
import com.backbase.training.to.Players;  
import org.apache.camel.Exchange;  
import org.apache.camel.Processor;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
import java.lang.reflect.Field;  
import java.util.Collections;  
import java.util.List;  
  
public class PlayerSortProcessor implements Processor {  
  
 private final static Logger logger = LoggerFactory.getLogger(PlayerSortProcessor.class);  
  
 @Override  
 public void process(Exchange exchange) throws Exception {  
  
 Players body = exchange.getIn().getBody(Players.class);  
 List<Player> players = body.getPlayers();  
 final String field = (String) exchange.getIn().getHeader("sort");  
  
 Collections.sort(players, (Object p1, Object p2) -> {  
 return comparePlayers(p1, p2, field);  
 });  
  
 logger.debug("Players: " + players);  
 }  
  
 private int comparePlayers(Object player1, Object player2, String field) {  
 int compare = -1;  
  
 Object value1 = getPrivateField(player1, field);  
 Object value2 = getPrivateField(player2, field);  
  
 if (value1 == null) {  
 // null at the beginning  
 compare = -1;  
 } else if (value2 == null) {  
 // null at the beginning  
 compare = 1;  
 } else {  
 compare = ((Comparable) value1).compareTo((Comparable) value2);  
 }  
  
 return compare;  
 }  
  
 private Object getPrivateField(Object object, String field) {  
 Object value = null;  
 try {  
 Field f = object.getClass().getDeclaredField(field);  
 f.setAccessible(true);  
 value = f.get(object);  
 } catch (NoSuchFieldException e) {  
 logger.error("The field [" + field + "] does not exist", e);  
 } catch (IllegalAccessException e) {  
 logger.error("Cannot access the field [" + field + "]", e);  
 } catch (Exception e) {  
 logger.error("Unexpected error trying to access field [" + field + "]", e);  
 }  
 return value;  
 }  
  
}

1. Create the PlayerSortProcessor (Java 7 - no lambda)

package com.backbase.training.processor;  
  
import com.backbase.training.to.Player;  
import com.backbase.training.to.Players;  
import org.apache.camel.Exchange;  
import org.apache.camel.Processor;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
import java.lang.reflect.Field;  
import java.util.Collections;  
import java.util.Comparator;  
import java.util.List;  
  
public class PlayerSortProcessor implements Processor {  
  
 private final static Logger logger = LoggerFactory.getLogger(PlayerSortProcessor.class);  
  
 @Override  
 public void process(Exchange exchange) throws Exception {  
  
 Players body = exchange.getIn().getBody(Players.class);  
 List<Player> players = body.getPlayers();  
 final String field = (String) exchange.getIn().getHeader("sort");  
  
 PlayerComparator comparator = new PlayerComparator(field);  
  
 Collections.sort(players, comparator);  
  
 logger.debug("Players: " + players);  
 }  
  
 private static class PlayerComparator implements Comparator<Player> {  
  
 String field;  
  
 public PlayerComparator(String field) {  
 this.field = field;  
 }  
  
 @Override  
 public int compare(Player p1, Player p2) {  
 return comparePlayers(p1, p2, this.field);  
 }  
  
 private int comparePlayers(Object player1, Object player2, String field) {  
 int compare = -1;  
  
 Object value1 = getPrivateField(player1, field);  
 Object value2 = getPrivateField(player2, field);  
  
 if (value1 == null) {  
 // null at the beginning  
 compare = -1;  
 } else if (value2 == null) {  
 // null at the beginning  
 compare = 1;  
 } else {  
 compare = ((Comparable) value1).compareTo((Comparable) value2);  
 }  
  
 return compare;  
 }  
  
 private Object getPrivateField(Object object, String field) {  
 Object value = null;  
 try {  
 Field f = object.getClass().getDeclaredField(field);  
 f.setAccessible(true);  
 value = f.get(object);  
 } catch (NoSuchFieldException e) {  
 logger.error("The field [" + field + "] does not exist", e);  
 } catch (IllegalAccessException e) {  
 logger.error("Cannot access the field [" + field + "]", e);  
 } catch (Exception e) {  
 logger.error("Unexpected error trying to access field [" + field + "]", e);  
 }  
 return value;  
 }  
  
 }  
  
}

1. Restart Portal
   1. Build configuration, services, & portal
   2. Run portal
2. Test the service using a browser
   1. <http://localhost:7777/portalserver/services/rest/player/list?sort=username>
   2. <http://localhost:7777/portalserver/services/rest/player/list?sort=fullname>
   3. <http://localhost:7777/portalserver/services/rest/player/list?sort=birthDay>
   4. <http://localhost:7777/portalserver/services/rest/player/list?sort=id>
3. Test the widget
   1. Create a new page
   2. Include the 2048 Players Widget in the page
   3. See if the page show the list of players and scores

## GAME SESSION CREATE

1. Go to the services directory
2. Generate the service from the archetype:
   1. Execute the command:
      1. mvn archetype:generate -DarchetypeArtifactId=integration-service-spring-archetype -DarchetypeGroupId=com.backbase.portal.archetypes -DarchetypeVersion=5.6.2.0
   2. And answer the following:
      1. groupId = com.backbase.training
      2. artifactId = game-service
      3. version = (leave the default value)
      4. package = (leave the default value)
   3. Open the services/pom.xml file
      1. Check if game-service is included as a <module>
3. Add camel-jackson dependency
   1. Open services/game-service/pom.xml
   2. Add the following <dependency>

<dependency>  
 <groupId>org.apache.camel</groupId>  
 <artifactId>camel-jackson</artifactId>  
 <version>2.12.4</version>  
</dependency>

1. Open the generated Camel XML file:
   1. Location:
      1. services/game-service/src/main/resources/META-INF/spring/backbase-integration-service.xml
   2. Add the following route:

<route id="com.backbase.training.GameSessionCreate">  
 <from uri="restlet:/game/session?restletMethods=post&amp;restletBinding=#queryStringToHeadersRestletBinding"/>  
  
 <removeHeaders pattern="CamelHttp\*" excludePattern="CamelHttpMethod"/>  
  
 <convertBodyTo type="String"/>  
 <log message="Request: ${body}." loggingLevel="DEBUG"/>  
 <to uri="http://{{training.server.host}}:{{training.server.http.port}}/training-server/rest/game/session?bridgeEndpoint=true&amp;throwExceptionOnFailure=false"/>  
  
 <convertBodyTo type="String"/>  
 <log message="Response: ${body}." loggingLevel="DEBUG"/>  
 </route>

1. Include dependency in webapps/portalserver/pom.xml

<dependency>

<groupId>com.backbase.training</groupId>

<artifactId>game-service</artifactId>

<version>1.0-SNAPSHOT</version>

</dependency>

1. Restart Portal
   1. Build configuration, services, & portal
   2. Run portal
2. Test the widget
   1. Create a new page
   2. Include the 2048 Game Widget in the page
   3. Preview the page in a browser
   4. Start debugging in the page
   5. Refresh the page
   6. Check the POST to the REST service:
      1. <http://localhost:7777/portalserver/services/rest/game/session>

## GAME SESSION UPDATE

1. Create ActiveMQ beans in the game-service project

<bean id="jmsConnectionFactory"  
 class="org.apache.activemq.ActiveMQConnectionFactory">  
 <property name="brokerURL" value="tcp://${training.server.host}:${training.server.mq.port}"/>  
 </bean>  
  
 <bean id="pooledConnectionFactory"  
 class="org.apache.activemq.pool.PooledConnectionFactory" init-method="start" destroy-method="stop">  
 <property name="maxConnections" value="8"/>  
 <property name="connectionFactory" ref="jmsConnectionFactory"/>  
 </bean>  
  
 <bean id="jmsConfig"  
 class="org.apache.camel.component.jms.JmsConfiguration">  
 <property name="connectionFactory" ref="pooledConnectionFactory"/>  
 <property name="concurrentConsumers" value="1"/>  
 </bean>  
  
 <bean id="activemq"  
 class="org.apache.activemq.camel.component.ActiveMQComponent">  
 <property name="configuration" ref="jmsConfig"/>  
 </bean>

1. Add this route to game-service project

<route id="com.backbase.training.GameSessionUpdate">  
 <from uri="restlet:/game/session/{sessionId}?restletMethods=get,put&amp;restletBinding=#queryStringToHeadersRestletBinding"/>  
  
 <convertBodyTo type="String"/>  
  
 <log message="Request: ${body}"/>  
  
 <to uri="jms:queue:UpdateGameSession"/>  
 </route>

1. Test the service
   1. Open the page with the 2048 Game Widget
   2. Start debugging in the page
   3. Start playing
   4. Check the PUT calls to the REST service:
      1. <http://localhost:7777/portalserver/services/rest/game/session/73>
2. Open the hawtio
   1. Go to: Queue -> UpdateGameSession
   2. Check if these counters increase as you play the game:
      1. Dequeue count
      2. Dispatch count
      3. Enqueue count

## UPDATE HIGHSCORES

1. Add this route the the game-service

<route id="com.backbase.training.PlayerHighscoreUpdate">

<from uri="restlet:/player/{username}/highscore?restletMethods=get,put&amp;restletBinding=#queryStringToHeadersRestletBinding"/>

<!-- Convert Body to String because input streams can only be read once -->

<convertBodyTo type="String"/>

<log message="Request: ${body}"/>

<to uri="jms:queue:UpdatePlayerHighScore"/>

<log message="Response: ${body}"/>

</route>

1. Test with Postman
   1. Method:
      1. PUT
   2. URL:
      1. http://localhost:7777/portalserver/services/rest/player/{player}/highscore
   3. Body (any number):
      1. 4000
2. Check if counters from UpdatePlayerHighScore queue using the hawtio (<http://localhost:9999/hawtio>)

## LISTENING FOR HIGHSCORES UPDATES

1. Add this route the the game-service

<route id="com.backbase.training.PlayerHighscoreUpdated">  
 <from uri="activemq:topic:Player.HighScoreUpdated"/>  
 <!-- Convert Body to String because input streams can only be read once -->  
 <convertBodyTo type="String"/>  
 <log message="Event received : ${body}." loggingLevel="DEBUG"/>  
 <to uri="file:inbox/highscoreupdate"/>  
 </route>

1. Repeat the same test of UPDATE HIGHSCORES
2. Check the HighSCoreUpdate topic counters
3. Check if the message was created in the inbox/highscoreupdate folder

## PLAYER CREATE EVENT

1. Check the Player.Created topic
   1. Open hawtio <http://localhost:9999/hawtio>
   2. Check the Player.Create topic
      1. Consumer count = 0 (no consumer in this topic)
2. Create the following route in the player-service project

<route id="com.backbase.training.PlayerCreatedTopic">  
 <from uri="activemq:topic:Player.Created"/>  
 <convertBodyTo type="String"/>  
 <log message="Player created event received : ${body}." loggingLevel="DEBUG"/>  
 <to uri="file:inbox/players/created"/>  
 </route>

1. Rebuild services & portal server
2. Restart portal server
3. Test the route
   1. Check the Player.Create topic if Consumer count = 1
   2. Create a new player
   3. Check the portal server log if this route was executed
   4. Check if there is some message file created in the portalserver/inbox/players/created folder

## PLAYER LOGIN

1. Go to the services directory
2. Generate the service from the archetype:
   1. Execute the command:
      1. mvn archetype:generate -DarchetypeArtifactId=integration-service-spring-archetype -DarchetypeGroupId=com.backbase.portal.archetypes -DarchetypeVersion=5.6.2.0
   2. And answer the following:
      1. groupId = com.backbase.training
      2. artifactId = authentication-service
      3. version = (leave the default value)
      4. package = (leave the default value)
3. Service pom.xml

<dependency>  
 <groupId>org.apache.camel</groupId>  
 <artifactId>camel-jackson</artifactId>  
 <version>${camel.version}</version>  
 </dependency>

<build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.cxf</groupId>  
 <artifactId>cxf-codegen-plugin</artifactId>  
 <version>2.7.8</version>  
 <executions>  
 <execution>  
 <id>generate-sources</id>  
 <phase>generate-sources</phase>  
 <configuration>  
 <sourceRoot>${project.build.directory}/generated-sources/cxf</sourceRoot>  
 <wsdlOptions>  
 <wsdlOption>  
 <wsdl>http://localhost:9999/training-server/PlayerManager?wsdl</wsdl>  
 <bindingFiles>  
 <bindingFile>${basedir}/src/main/resources/binding.xml</bindingFile>  
 </bindingFiles>  
 </wsdlOption>  
 </wsdlOptions>  
 </configuration>  
 <goals>  
 <goal>wsdl2java</goal>  
 </goals>  
 </execution>  
 </executions>  
 </plugin>  
 </plugins>  
 </build>

1. Create src/main/resources/binding.xml

<jaxb:bindings jaxb:extensionBindingPrefixes="xjc" version="2.1"  
 xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"  
 xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc">  
 <jaxb:globalBindings>  
 <xjc:simple/>  
 </jaxb:globalBindings>  
</jaxb:bindings>

1. Open the generated Camel XML file:
   1. Location:
      1. services/authentication-service/src/main/resources/META-INF/spring/backbase-integration-service.xml
   2. Add the following route:

<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:camel="http://camel.apache.org/schema/spring" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd  
 http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd">  
  
 <bean id="playerManagementService"  
 class="org.springframework.remoting.jaxws.JaxWsPortProxyFactoryBean">  
 <property name="serviceInterface"  
 value="com.backbase.expert.training.services.webservice.PlayerManagementService"/>  
 <property name="wsdlDocumentUrl" value="http://${training.server.host}:${training.server.http.port}/training-server/PlayerManager?wsdl"/>  
 <property name="namespaceUri" value="http://webservice.services.training.expert.backbase.com/"/>  
 <property name="serviceName" value="PlayerManagementWebServiceImplService"/>  
 <property name="portName" value="PlayerManagementServicePort"/>  
 <property name="endpointAddress" value="http://${training.server.host}:${training.server.http.port}/training-server/PlayerManager?wsdl"/>  
 </bean>  
  
 <camel:routeContext id="com.backbase.portal.integration.service.PlayerLoginService" xmlns="http://camel.apache.org/schema/spring">  
  
 <route id="com.backbase.training.PlayerLoginRoute">  
 <from uri="restlet:/player/login?restletMethods=POST&amp;restletBinding=#queryStringToHeadersRestletBinding"/>  
 <log message="Incoming body message: ${body}." loggingLevel="DEBUG"/>  
 <log message="Incoming header message: ${headers}." loggingLevel="DEBUG"/>  
  
 <bean ref="playerManagementService"  
 method="login(${header.sessionId}, ${header.username}, ${header.password})"/>  
  
 <log message="Received message: ${body}." loggingLevel="DEBUG"/>  
 <marshal>  
 <json library="Jackson"/>  
 </marshal>  
 </route>  
  
 </camel:routeContext>  
</beans>

1. Include dependency in webapps/portalserver/pom.xml

<dependency>

<groupId>com.backbase.training</groupId>

<artifactId>authentication-service</artifactId>

<version>1.0-SNAPSHOT</version>

</dependency>

1. Restart Portal
   1. Build configuration, services, & portal
   2. Run portal
2. Test the service using Postman
   1. Method:
      1. POST
   2. URL:
      1. http://localhost:7777/portalserver/services/rest/player/login?username=john&password=backbase
   3. Click on Send button
   4. Check if the response is something like this:

{

"id": 33,

"uid": null,

"username": "bruce",

"fullname": "Bruce Lee",

"password": "bruce",

"emailAddress": null,

"profileUrl": null,

"photoUrl": null,

"thumbnalUrl": null,

"gender": null,

"birthDay": -923097600000,

"firstName": null,

"lastName": null,

"middleName": null,

"highScore": 0

}

## CAMEL CACHE

1. Include this dependency in pom.xml

<dependency>  
 <groupId>org.apache.camel</groupId>  
 <artifactId>camel-cache</artifactId>  
 <version>2.12.4</version>  
 </dependency>

1. Sample route

<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:camel="http://camel.apache.org/schema/spring" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd  
 http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd">  
  
 <camel:routeContext id="com.backbase.portal.integration.service.exercise06"  
 xmlns="http://camel.apache.org/schema/spring">  
 <route id="com.backbase.training.route.player.list">  
 <from uri="restlet:/player/listWithCache?restletMethod=GET&amp;restletBinding=#queryStringToHeadersRestletBinding"/>  
 <removeHeaders pattern="CamelHttp\*" excludePattern="CamelHttpMethod"/>  
  
 <!-- Cache Here -->  
 <setHeader headerName="CamelCacheOperation"><camel:simple>CamelCacheGet</camel:simple></setHeader>  
 <setHeader headerName="CamelCacheKey"><camel:simple>PLAYERS\_KEY</camel:simple></setHeader>  
 <to uri="cache://playerscache?timeToLiveSeconds=60"/>  
 <log message="Headers: ${headers}" loggingLevel="DEBUG"/>  
 <choice>  
 <when>  
 <simple>${headers.CamelCacheElementWasFound} == null</simple>  
 <to uri="http://localhost:9999/training-server/rest/player/players"/>  
 <convertBodyTo type="String"/>  
 <log message="After calling the training-server: body = ${body}" loggingLevel="DEBUG"/>  
  
 <unmarshal>  
 <json library="Jackson" />  
 </unmarshal>  
 <process ref="playerSort"/>  
 <marshal>  
 <json library="Jackson"/>  
 </marshal>  
  
 <!-- add it into cache -->  
 <log message="Content was added into the cache"/>  
 <setHeader headerName="CamelCacheOperation"><camel:simple>CamelCacheUpdate</camel:simple></setHeader>  
 <setHeader headerName="CamelCacheKey"><camel:simple>PLAYERS\_KEY</camel:simple></setHeader>  
 <to uri="cache://playerscache"></to>  
 </when>  
 <otherwise>  
 <log message="hey! there was something in the cache" loggingLevel="DEBUG"/>  
 </otherwise>  
 </choice>  
 </route>  
 </camel:routeContext>  
</beans>

## COMPLETE SOLUTION

1. The complete solution is at GitHub
   1. URL: <https://github.com/Backbase/training-be-module-01>
   2. Folder: enterprise-integration-module

## CMIS

1. Download Apache Chemistry CMIS Workbench
   1. <http://chemistry.apache.org/>
2. Open CMIS Workbench
   1. URL: <http://localhost:8081/contentservices/atom>
   2. Binding: AtomPub
   3. Username: admin
   4. Password: admin
3. Check the repositories
   1. Click on Load Repositories button
   2. Click on the List of repositories
   3. Show:
      1. There is a repository for each created portal
         1. Stores Portal Content
      2. Configuration Repository
         1. Stores CXP internal configuration
      3. Content Repository
         1. Stores Shared Content
      4. Resources Repository
         1. Stores Enterprise Catalog (when configured to be stored in content services)
4. Select Content Repository
5. Click Login
6. Make sure you have some Shared Content
7. Click on some content
   1. Click on Content URL
   2. See the image rendered in the browser (provide username/password if required)
8. Make some queries
   1. Click on Query button
   2. SELECT \* FROM cmis:document WHERE cmis:path='/banner\_home.jpg'
   3. SELECT \* FROM cmis:document WHERE cmis:objectTypeId='bb:image'

## CMIS JAVA API

1. Visit:
   1. <https://github.com/Backbase/training-be-module-02>
2. Clone this project
3. Click on the camel-cmis-upload folder
4. Follow the instructions

## VALIDATORS

1. Visit:
   1. <https://github.com/Backbase/training-be-module-02>
2. Clone this project (if not cloned yet)
3. Click on the contentservices-validator folder
4. Follow the instructions

## 

## 

## LDAP AUTHENTICATION PROVIDER

1. Install Apache Directory Studio
   1. URL: <http://directory.apache.org/studio/>
2. Open Directory Studio
3. Create a new connection to the LDAP from Training Server
   1. Connection name: Training Server
   2. Hostname: localhost
   3. Port: 33389
   4. Authentication Method: no authentication
4. Browse the users and groups

|  |
| --- |
| \* Note:  To create your own Embedded LDAP Server (just a reference, not to do in classroom).   * Go to: <http://my.backbase.com> * Follow: Docs -> How To Guides * Select the Category: Security * Click on: Integration and Configuring Spring Security Providers – LDAP Integration |

## CUSTOM LDAP AUTHENTICATION PROVIDER

1. Visit:
   1. <https://github.com/Backbase/training-be-module-03>
2. Clone this project
3. Click on the security-ldap folder
4. Follow the instructions
5. Note for CXP 5.6.2:
   1. I had to include this dependency in portalserver/pom.xml in order to avoid EhcacheCacheManager class not found exception when starting the portal server:

<dependency>  
 <groupId>com.backbase.portal.foundation</groupId>  
 <artifactId>business</artifactId>  
 <version>5.6.2.0</version>  
 </dependency>

1. CXP Manager
   1. Open Groups app
   2. Check if employees group exist, if not, create one group
   3. Give Can Manager Portals privilege to employees
   4. Go to your portal server's permissions and grant Can Administer privilege for employees group
   5. Logout from CXP Manager
   6. Login as:
      1. Username: john
      2. Password: backbase
   7. Check if you have access to your portal
   8. Check the portal server log to see UserDetailsContextMapperImpl in action

## TRACKING

1. Enable tracking:
   1. Add to backbase.properties

optional.tracking.profile=trackingStoreCookie  
foundation.tracking.tags.maxAge=3

1. Build configuration & portal server
2. Restart portal server
3. Open Explorer App
   1. Go to your portal
   2. Add the property:
      1. Name: trackingTagTypes
      2. Value: 2048,game
4. Go to the settings of your game page
   1. Set Tracking = 2048
   2. Save
5. Create a page with a target container using this tracking counter
6. Use incognito window in google chrome

## CUSTOM COLLECTOR

1. Visit:
   1. <https://github.com/Backbase/training-be-module-04>
2. Clone this project
3. Click on the targeting-service-module folder
4. Follow the instructions
5. Test:
   1. Create a targeting container with the Age collector
   2. Consult the list of players
   3. Create some players in CXP Manager with the same username as manager
      1. This is necessary because we did not have time to integrate the game authentication with CXP
   4. Log with these users and see the page

## PUBLISHING CHAINS

1. Configuring Publishing Chains:
   1. Visit: <http://my.backbase.com>
   2. Go to: Docs -> How To Guides
   3. And then: Categories -> Publishing
   4. Publishing across multiple environments:
      1. Click on:
         1. Configuring Publishing Across Multiple Environments
   5. Publishing in a single environment:
      1. Click on:
         1. Install and Configure Orchestrator for Publishing
2. More references:
   1. Configuration guide:
      1. Visit: <http://my.backbase.com>
      2. Go to: Docs -> Product Documentation -> CXP Documentation
      3. And then: Configuration Guide -> Publishing Services -> Publishing Chains