

FUSE and Camel Workshop

RHTE 2014

Christina Lin
JBoss Technology Evangelist

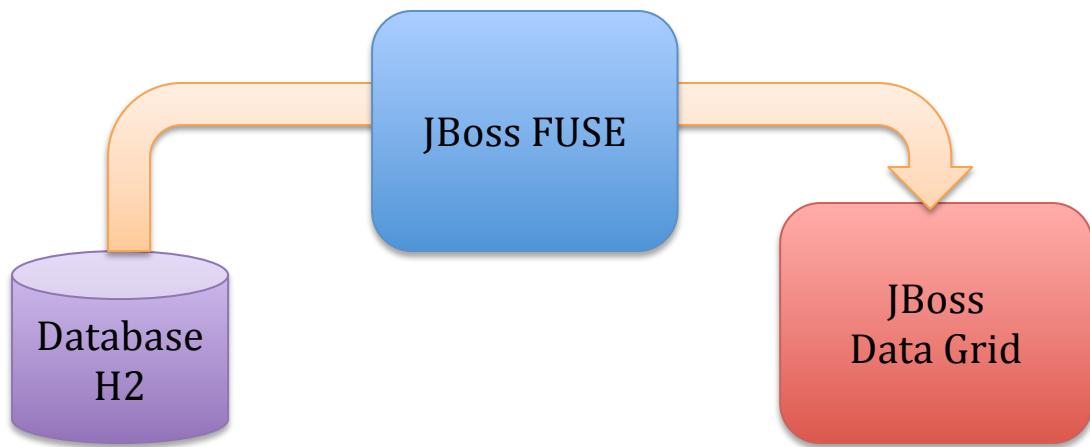
Index

Index	2
Introduction	3
Prerequisite	4
Installation	5
Install Red Hat JBoss Developer Studio	5
Install Red Hat JBoss Fuse	13
Lab 1	15
Lab 2	25

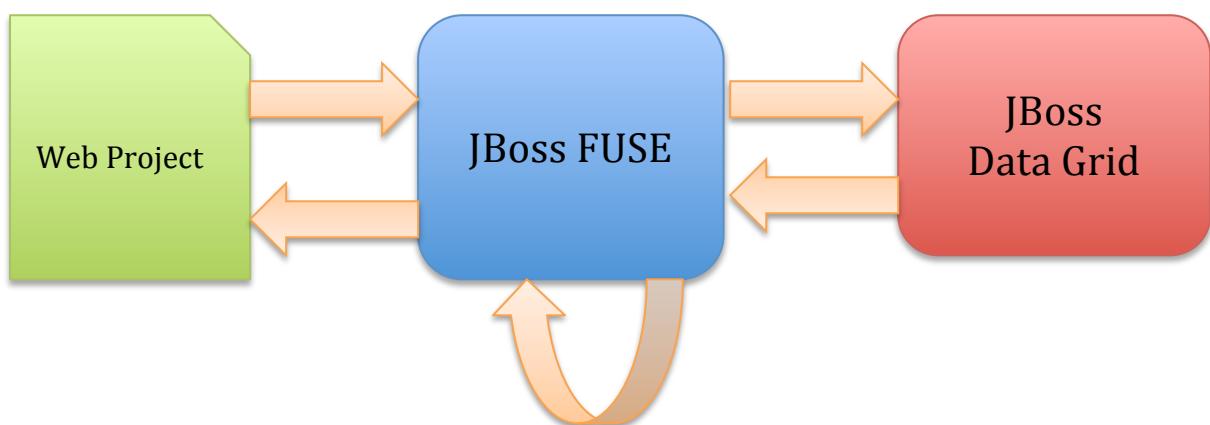
Introduction

Because we only have one hour, this lab is going to be a very simple demonstration to get you started on using several components and Enterprise Integration Pattern in Red Hat JBoss Fuse. This is a subset and simply version of a project call myWiri, which will be out shortly in July 2014 by Christina. Look forward to it! ☺

We are first going to load data from H2 database to Red Hat JBoss Data Grid, provide a faster elastic memory grid cache for user to look up.



Then we are going to build a web application with a interface for user to lookup data from the cache we have created. The communication will be synchronized with WebSocket protocol with Fuse Camel route. In the route, it will determine if the requested content can be found in the cache. We will use the content routing integration pattern to reply the content that we are going to reply back.



Prerequisite

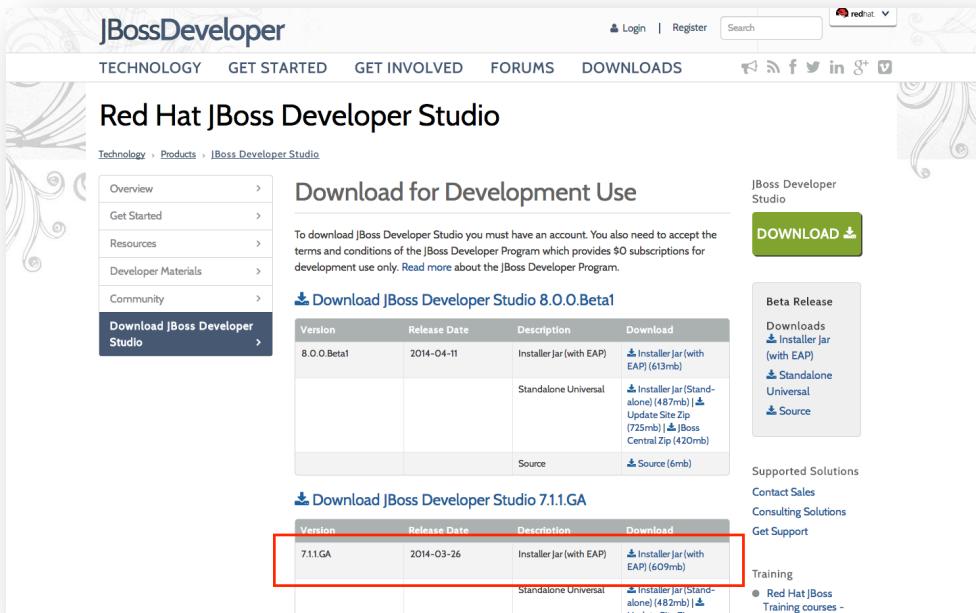
- This lab only runs in Linux and Mac. Please make sure you have the right platform.
(Or you are always welcome to contribute Window script to this project ☺!)
- Java Runtime Environment 6 or 7 and Maven 2.1+ should be already installed in your machine.

Installation

Install Red Hat JBoss Developer Studio

This lab will guide you through installing Red Hat JBoss Developer Studio and the tools for developing JBoss Fuse Integration Projects.

If you don't already have *Red Hat JBoss Developer Studio 7.1.1 GA*, download it from <https://www.jboss.org/products/jbds.html>.

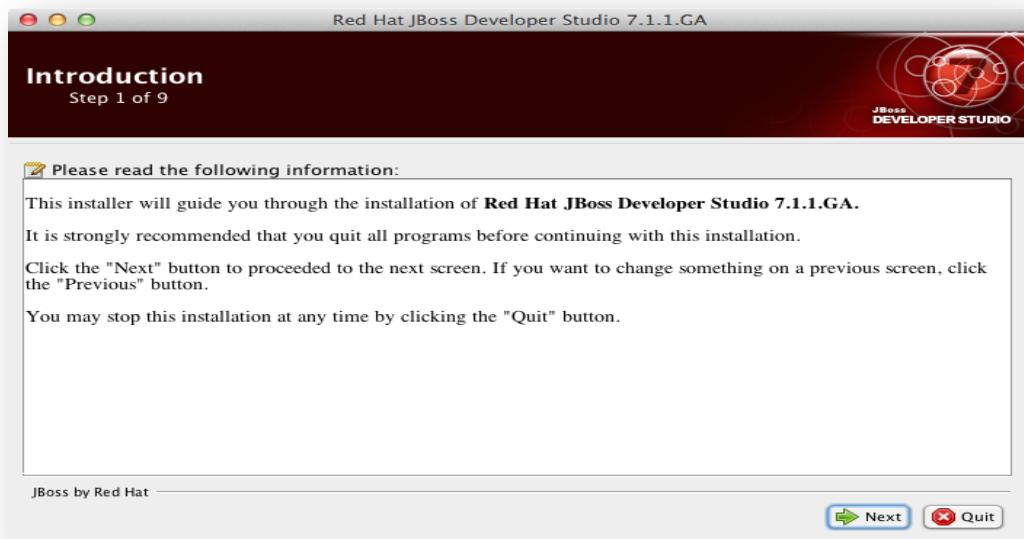


The screenshot shows the JBossDeveloper website's "Downloads" section for Red Hat JBoss Developer Studio. The "Download for Development Use" table lists two versions: 8.0.0.Beta1 and 7.1.1.GA. The 7.1.1.GA row is highlighted with a red box. The "Description" column for 7.1.1.GA indicates "Installer Jar (with EAP)" and "Standalone Universal". The "Download" column contains links to download files, including "Installer Jar (with EAP) (609mb)" and "Standalone Universal (482mb)". To the right of the table, there is a "Beta Release" sidebar with links to "Downloads" (including "Installer Jar (with EAP)", "Standalone Universal", and "Source"), "Supported Solutions" (Contact Sales, Consulting Solutions, Get Support), and "Training" (Red Hat JBoss Training courses).

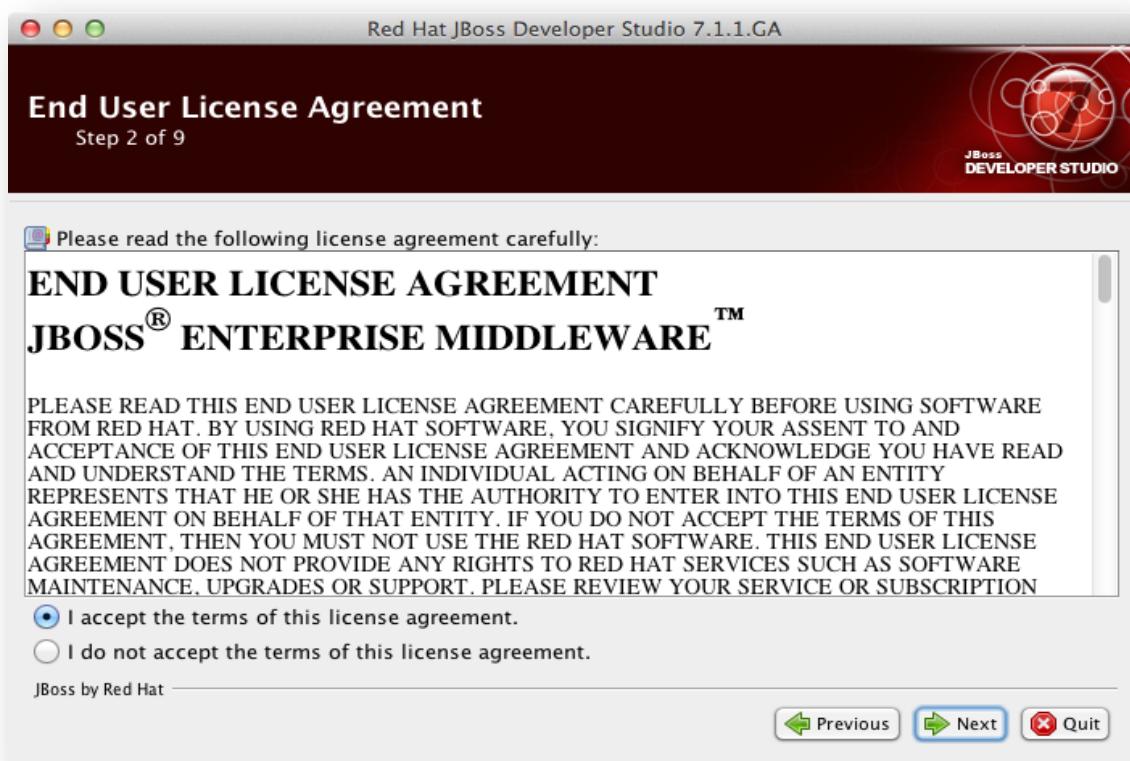
Version	Release Date	Description	Download
8.0.0.Beta1	2014-04-11	Installer Jar (with EAP) Standalone Universal	Installer Jar (with EAP) (613mb) Standalone Universal (487mb) Update Site Zip (725mb) JBoss Central Zip (420mb)
7.1.1.GA	2014-03-26	Installer Jar (with EAP) Standalone Universal	Installer Jar (with EAP) (609mb) Standalone Universal (482mb) JBoss Central Zip (420mb)
		Source	Source (6mb)

Run the JBDS installer using java at the command prompt or double click on the jar file.

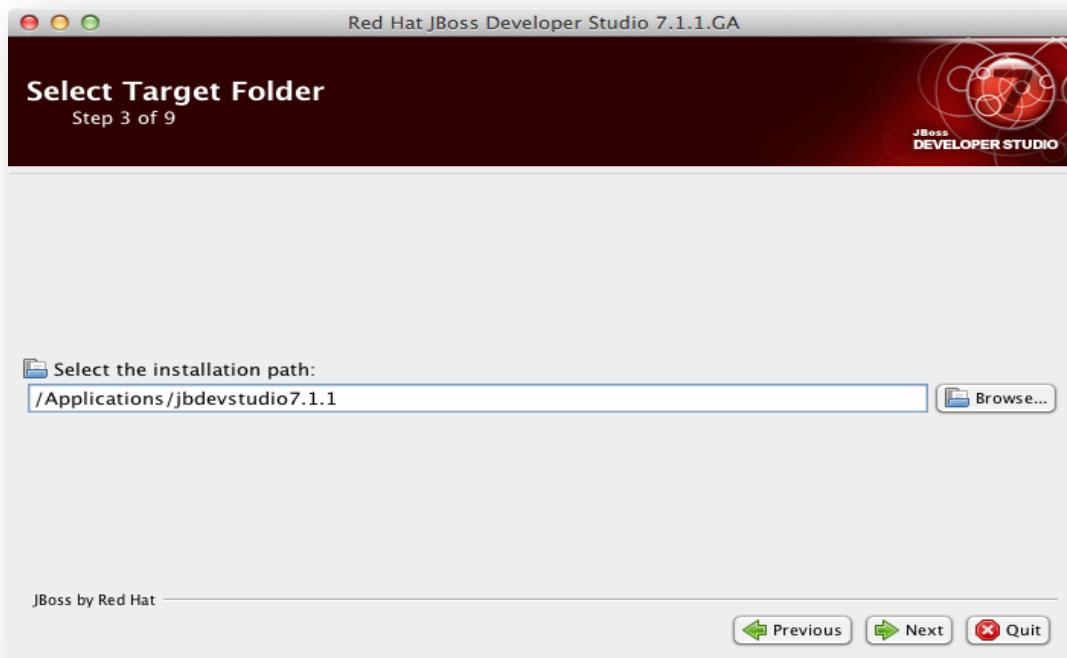
```
java -jar {jbdevstudio- installer.jar}
```



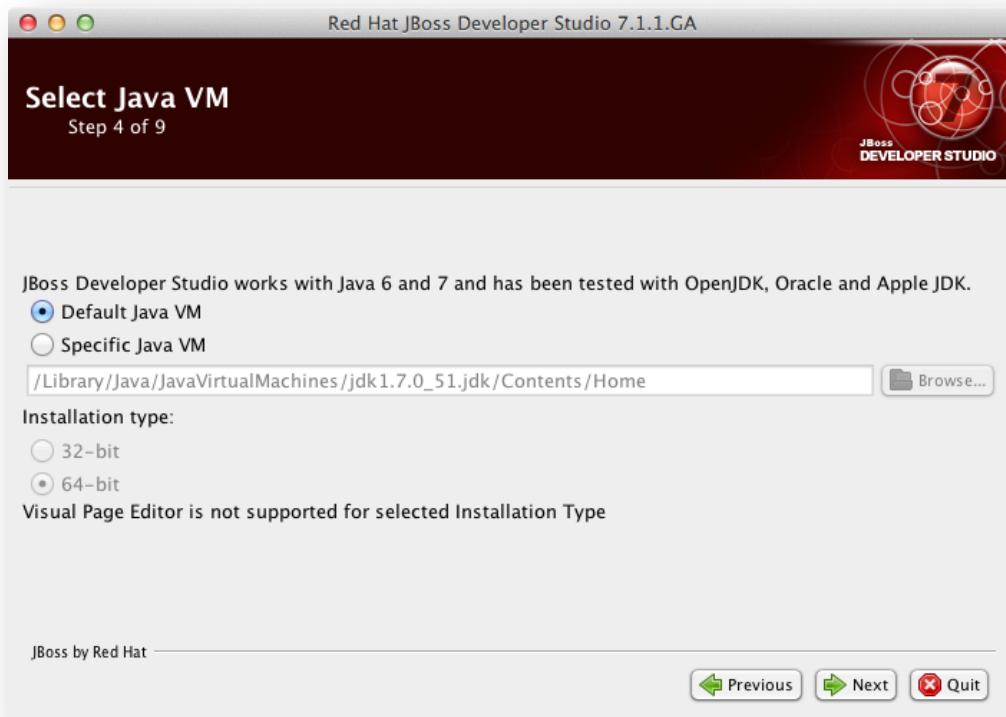
Follow the installer prompts to complete the installation process.



After reading and agreeing to the terms of the End User License Agreement, select **I accept the terms of this license agreement** and click **Next**.

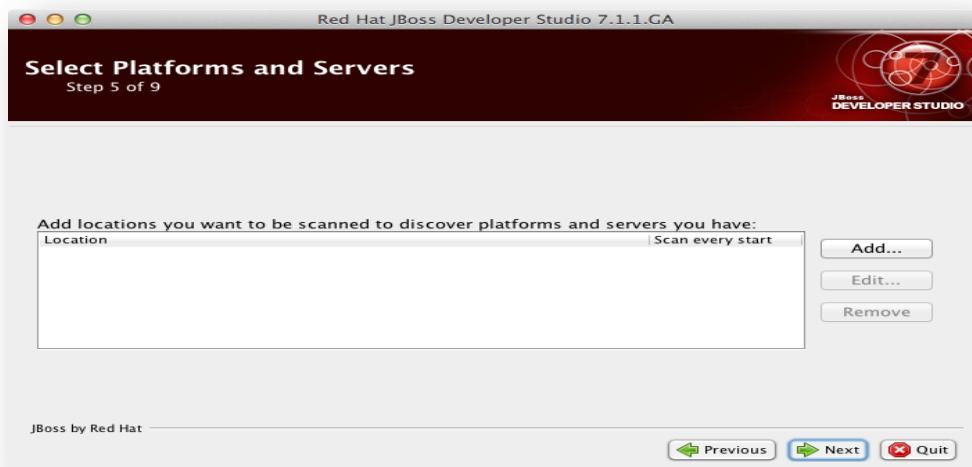


In the Select the installation path field, type the path where you want Red Hat JBoss Developer Studio to be installed or click Browse to navigate to the location. When the Select the installation path field shows the correct path, click Next.

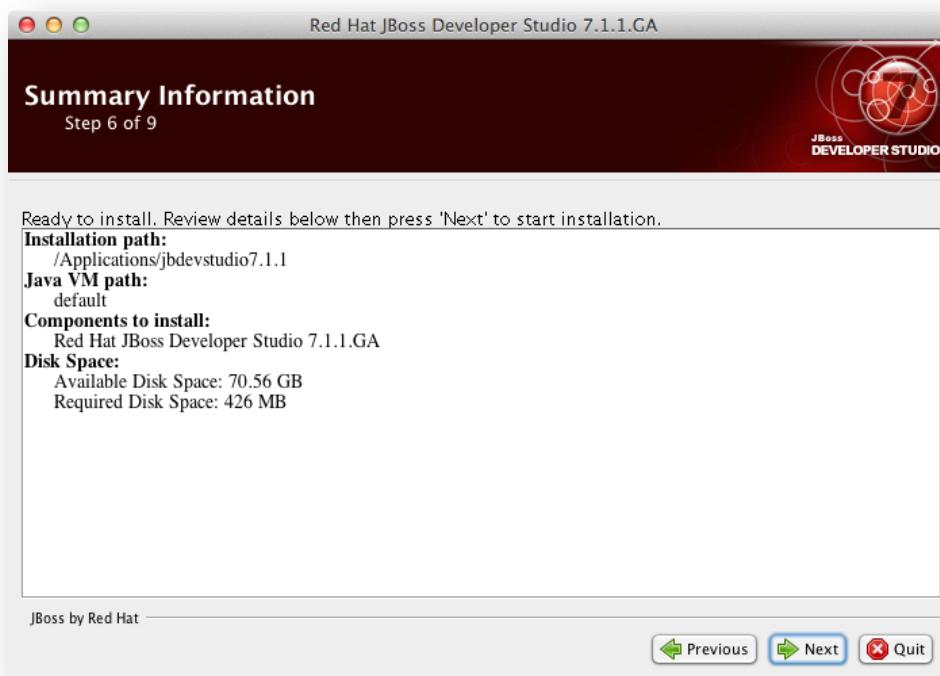


In the Select Java VM step, Default Java VM is automatically selected. Ensure that the

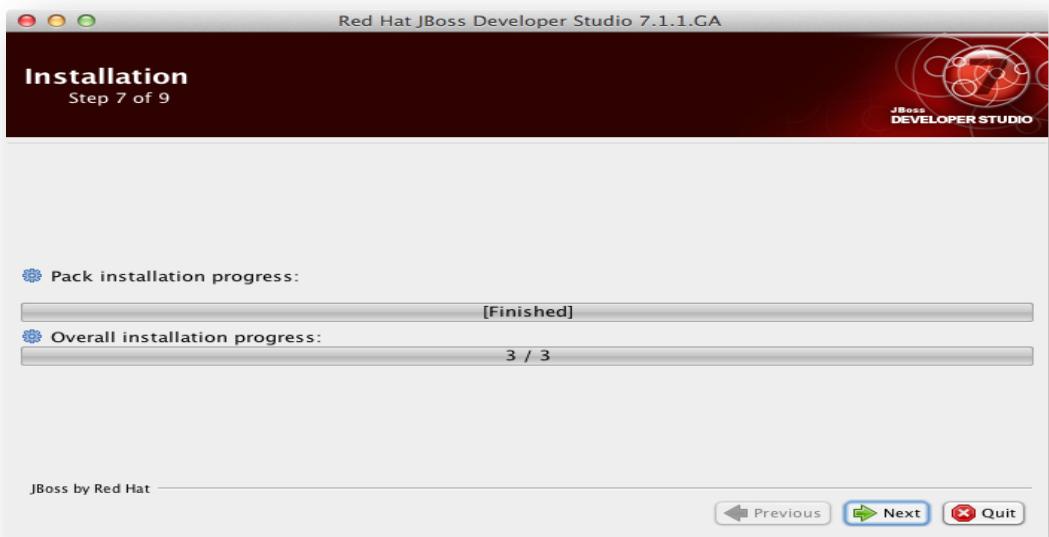
disabled text field contains the path of the Java developer kit you want to use. To change the specified Java developer kit, select Specific Java VM and type the path of the Java developer kit in the text field or use the Browse button to locate the Java developer kit. When the text field shows the correct Java developer kit path, click Next.



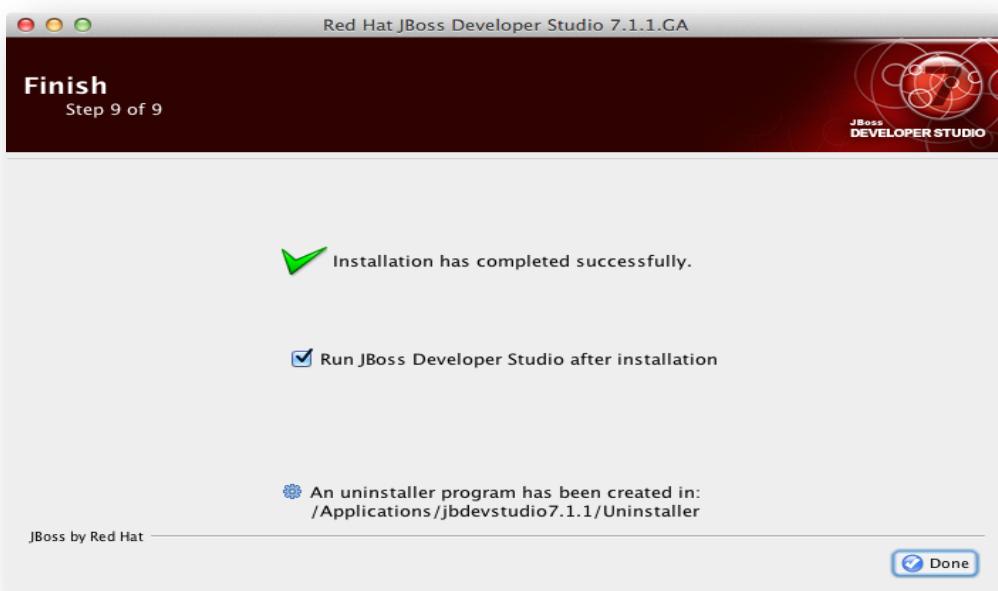
In the Select Platforms and Servers step one can add server to make use of automatic runtime detection for finding already installed application servers. Skip this for now since we will add servers later. Click Next to proceed.



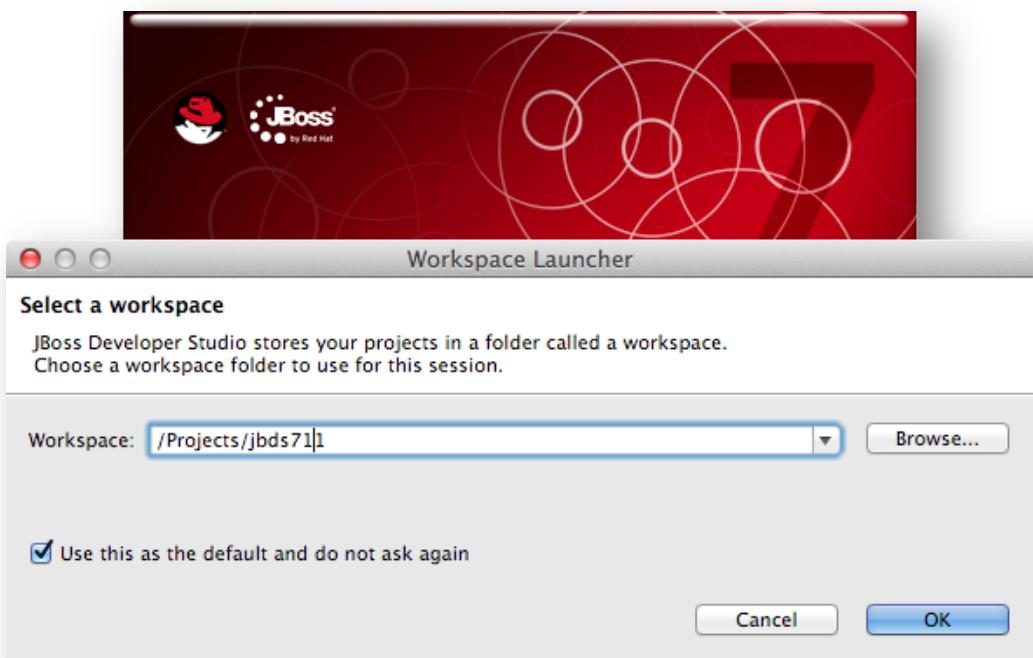
Review the details in the Summary Information window and, if they are correct, click Next. The installation commences.



When the installation progress bar shows Finished, click Next. The installation process is now complete.



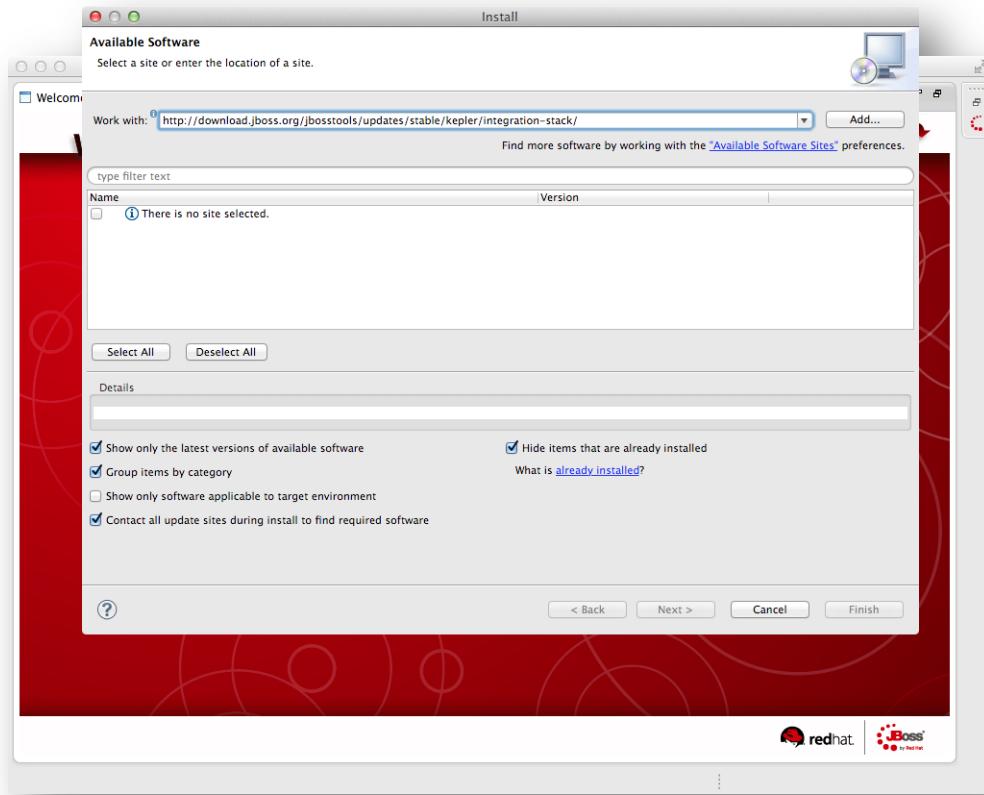
Click Done to close the Installer window. When the installation is completed, run Red Hat JBoss Developer Studio 7.1.1. When the Red Hat JBoss Developer Studio starts, you are asked to choose the workspace folder for the session. The workspace is where your projects are stored.



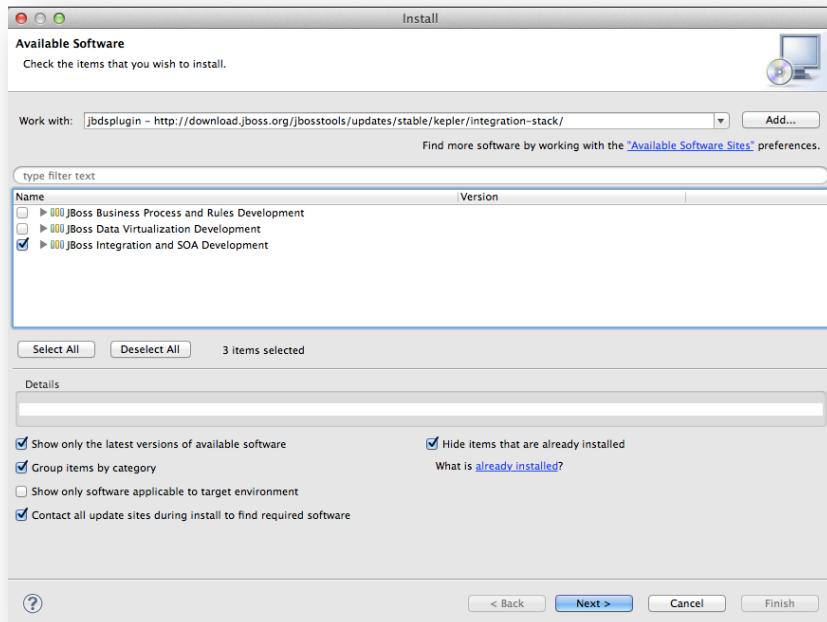
To set the workspace location, follow these steps:

1. In the Workspace field, type the path for a new or existing workspace or use Browse to navigate to the workspace location.
2. If you do not want to be asked to choose a workspace folder each time the IDE starts, select the Use this as the default and do not ask again check box.
3. Click OK.

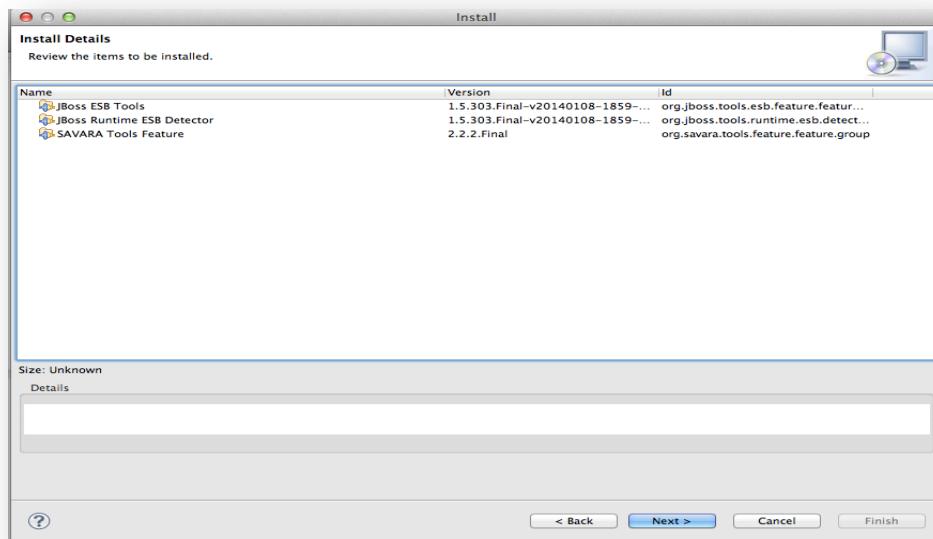
The workspace location prompting behavior can be altered at any time by clicking Window → Preferences. Expand General → Startup and Shutdown → Workspace. Select or clear the Prompt for workspace on startup check box to alter the behavior as appropriate. From the JBDS menu bar, choose Help → Install New Software...



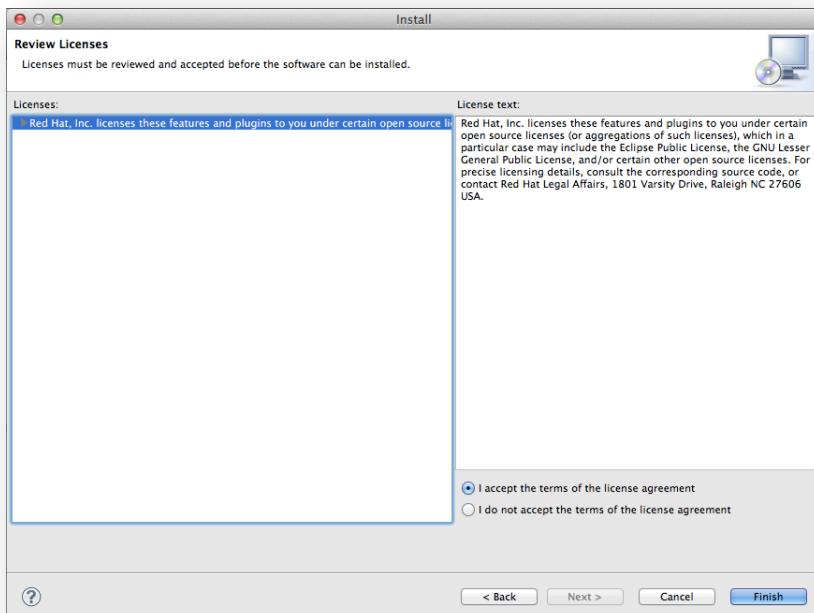
In the "Work with:" field on the Install wizard, paste the following link:
<http://download.jboss.org/jbosstools/updates/stable/kepler/integration-stack/> and click the Add... button.



Select JBoss Integration and SOA Development option in the list and click Next.



Review Install Details and click the Next > button.



The Security Warning Window will appear and click the OK button to proceed.



The Software Updates window appears. Press the Yes button to restart Red Hat JBoss Developer Studio to apply the changes to take effect.

Install Red Hat JBoss Fuse

If you don't already have **Red Hat JBoss Fuse 6.1.0 GA**, download it from <https://www.jboss.org/products/fuse.html>.

JBossDeveloper

TECHNOLOGY GET STARTED GET INVOLVED FORUMS DOWNLOADS 

Red Hat JBoss Fuse

Technology > Products > JBoss Fuse

[Overview](#) >
[Get Started](#) >
[Resources](#) >
[Developer Materials](#) >
[Community](#) >
[Download JBoss Fuse](#) >

Download for Development Use

To download JBoss Fuse you must have an account. You also need to accept the terms and conditions of the JBoss Developer Program which provides \$0 subscriptions for development use only. [Read more about the JBoss Developer Program.](#)

Download JBoss Fuse 6.1.0.GA

Version	Release Date	Description	Download
6.1.0.GA	2014-04-14	Zip Files	Full Zip (601mb) Medium Zip (162mb)
		Release Notes	Release Notes

[View Older Downloads](#) ▾

JBoss xPaaS Services for OpenShift

You can also deploy applications using JBoss Fuse on OpenShift, meaning you don't even need to download the bits! Try JBoss Fuse and other middleware products on OpenShift Online to get your applications up and running even faster.

[Use JBoss Fuse on OpenShift](#)

JBoss Fuse for Developers

DOWNLOAD

Supported Solutions
[Contact Sales](#)
[Consulting Solutions](#)
[Get Support](#)

Training

- [Red Hat JBoss Training courses - Development](#)
- [Red Hat JBoss Training courses - Administration](#)

Webinars
[Getting Started with](#)

The installation will be done for you in the lab. But if you would like further installation guideline, please go to https://access.redhat.com/site/documentation/en-US/Red_Hat_JBoss_Fuse/6.1/html/Installation_Guide/files/front.html, for more details.

Install Red Hat JBoss Data Grid

If you don't already have **Red Hat JBoss Data Grid 6.2.0 GA**, download it from <https://www.jboss.org/products/datagrid/download/>.

JBossDeveloper

TECHNOLOGY GET STARTED GET INVOLVED FORUMS DOWNLOADS 

Red Hat JBoss Data Grid

Technology > Products > JBoss Data Grid

[Overview](#) >
[Get Started](#) >
[Resources](#) >
[Developer Materials](#) >
[Community](#) >
[Download JBoss Data Grid](#) >

Download for Development Use

To download JBoss Data Grid you must have an account. You also need to accept the terms and conditions of the JBoss Developer Program which provides \$0 subscriptions for development use only. [Read more about the JBoss Developer Program.](#)

Download JBoss Data Grid 6.3.0 Beta

Version	Release Date	Description	Download
6.3.0.Beta	2014-05-28	CPP Client	CPP Client Win32 (3.5mb) CPP Client Win64 (3.5mb) CPP Client RH-EL6_64 (1.3mb) CPP Client RH-ELS_64 (1mb)
		Zip (Data Grid Library)	Zip (Data Grid Library) (3.9mb)
		Zip (Data Grid Server)	Zip (Data Grid Server) (142mb)
		Release Notes	Release Notes
		Sources	Sources (13mb)
		Quickstarts	Quickstarts (214kb)

JBoss Data Grid for Developers

DOWNLOAD

Beta Release
Features expanded security with role-based authorization and access control as well as secured data within clusters. Also, modules for JBoss EAP users, a new IP cache store, and a C# Hot Rod client in technology

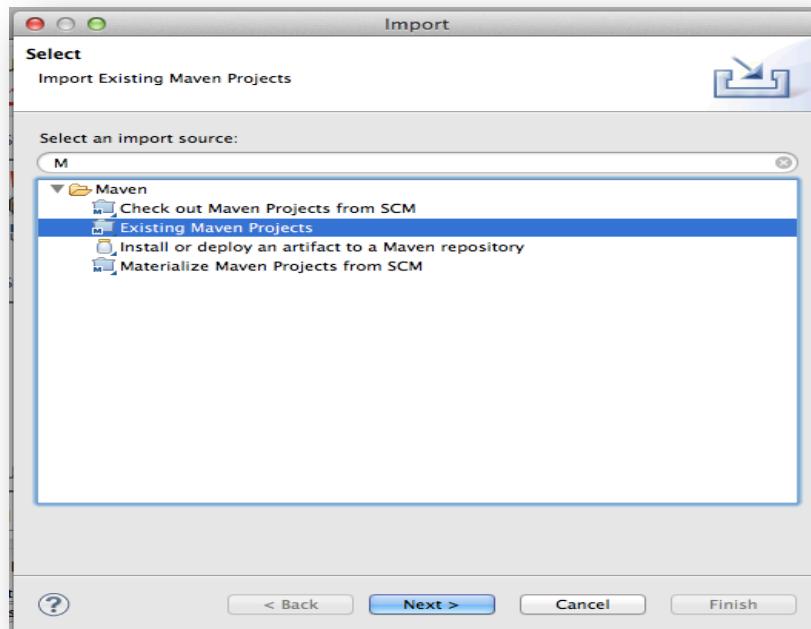
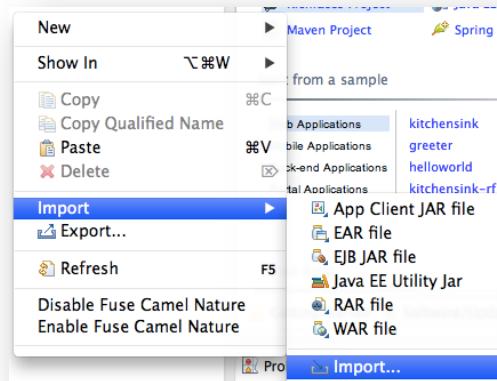
The installation will be done for you in the lab. But if you would like further installation guideline, please go to https://access.redhat.com/site/documentation/en-US/Red_Hat_JBoss_Data_Grid/6.2/html/Getting_Started_Guide/, for more details.

Lab 1

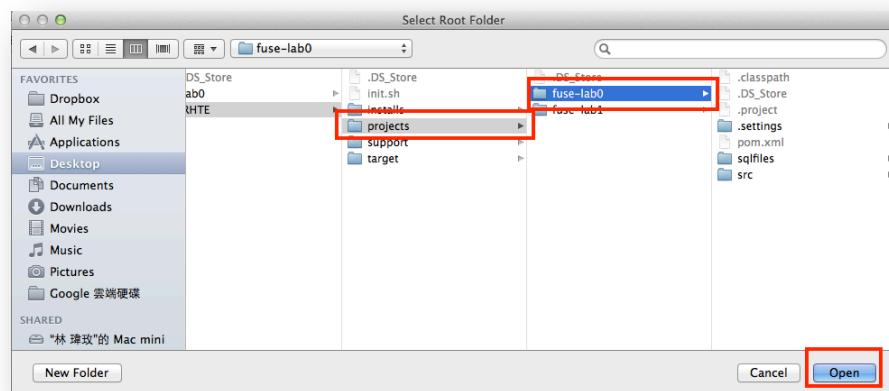
Environment

We will be working under fuse-lab0 directory.

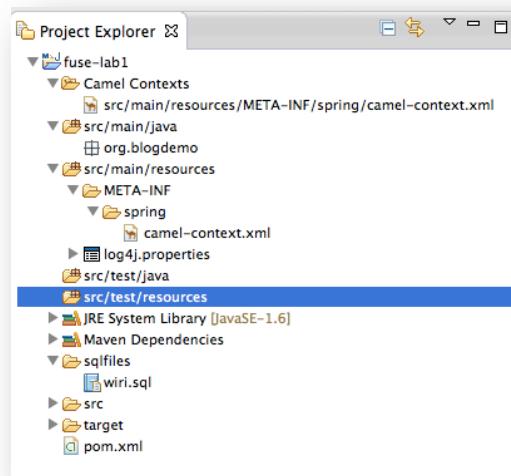
1. First we are going to load project into JBDS. Open up JBDS, and import an existing maven project.



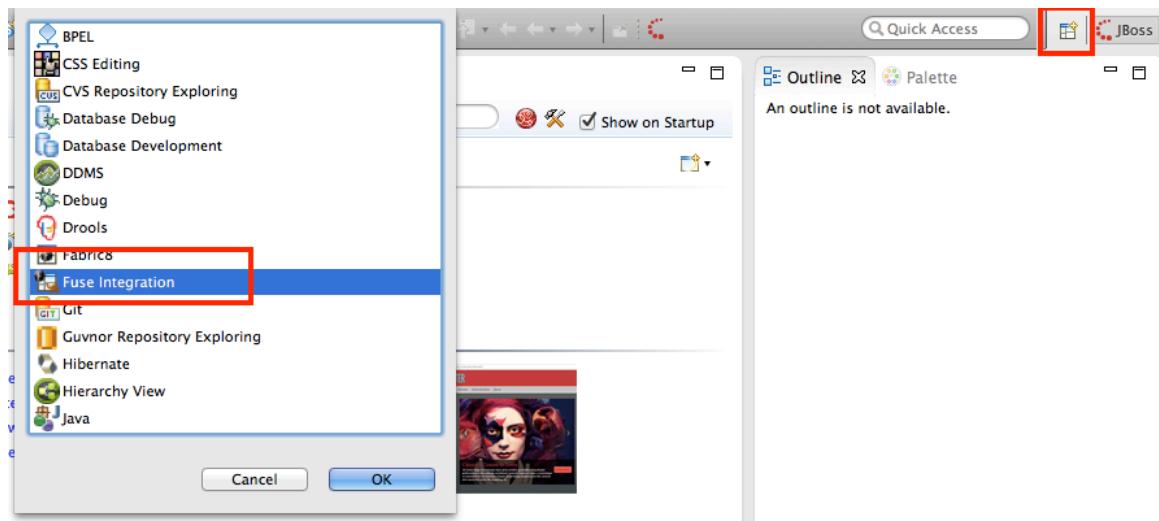
Go to the first project directory, fuse-lab0



You will see the project loaded into the console.

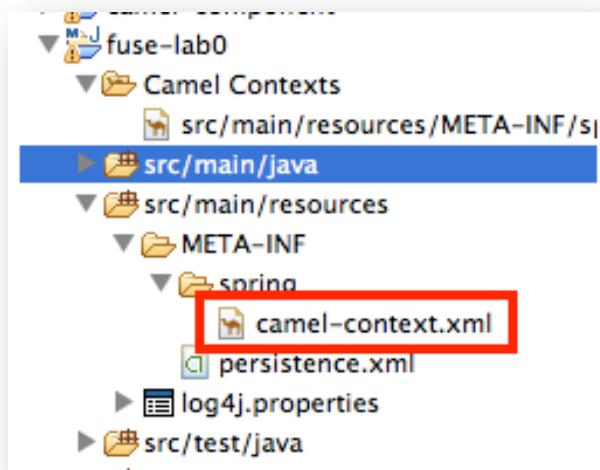


If your JBDS does not go to FUSE perspective, go to the upper right corner, click on the +, then select Fuse Integration.

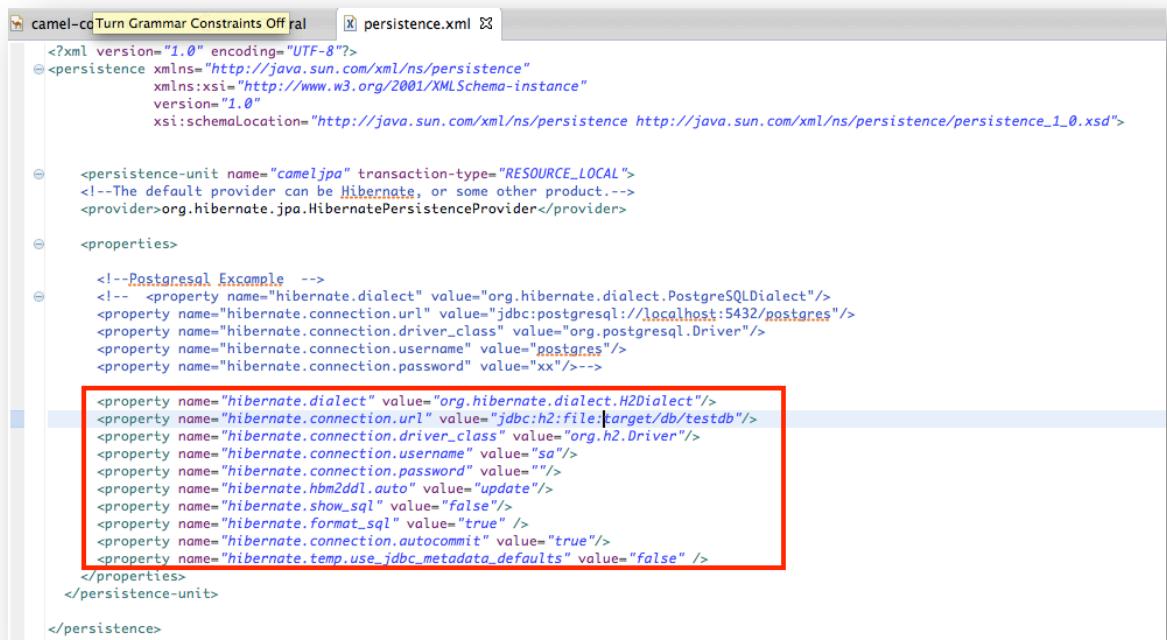


- Look at the Camel context, I have already added the JPA related setting for you. But see the “entityManagerFactory” is connected to a **cameljpa**, so we need to specify Database setting for JPA. We are going to add a persistence.xml to our project.

Copy from **support/persistence.xml**
to project’s **src/main/resources/persistence.xml**



The **persistence.xml** contains all the database setting.



```

<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  version="1.0"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence_1_0.xsd">

  <persistence-unit name="cameljpa" transaction-type="RESOURCE_LOCAL">
    <!--The default provider can be Hibernate, or some other product.-->
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>

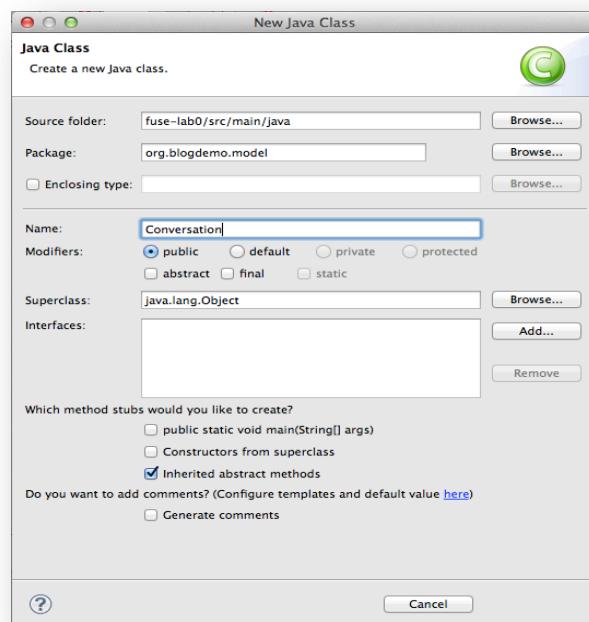
    <properties>
      <!--Postgres Example -->
      <!--<property name="hibernate.dialect" value="org.hibernate.dialect.PostgreSQLDialect"/>
      <property name="hibernate.connection.url" value="jdbc:postgresql://localhost:5432/postgres"/>
      <property name="hibernate.connection.driver_class" value="org.postgresql.Driver"/>
      <property name="hibernate.connection.username" value="postgres"/>
      <property name="hibernate.connection.password" value="xx"/>-->
      <property name="hibernate.dialect" value="org.hibernate.dialect.H2Dialect"/>
      <property name="hibernate.connection.url" value="jdbc:h2:file:/target/db/testdb"/>
      <property name="hibernate.connection.driver_class" value="org.h2.Driver"/>
      <property name="hibernate.connection.username" value="sa"/>
      <property name="hibernate.connection.password" value="" />
      <property name="hibernate.hbm2ddl.auto" value="update"/>
      <property name="hibernate.show_sql" value="false"/>
      <property name="hibernate.format_sql" value="true" />
      <property name="hibernate.connection.autocommit" value="true"/>
      <property name="hibernate.temp.use_jdbc_metadata_defaults" value="false" />
    </properties>
  </persistence-unit>
</persistence>

```

- Before we get started constructing the route, we need to create the bean for the data we are going to send, Conversation Bean.

Under package : `org.blogdemo.model`

Class Name: Conversation



Paste the following code into Conversation.java

```
package org.blogdemo.model;

import static javax.persistence.LockModeType.PESSIMISTIC_WRITE;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.NamedQuery;
import javax.persistence.Table;

@Entity
@Table(schema="fusedemo", name="Conversation")
@NamedQuery(name="getAll", query="SELECT conversationitem from Conversation conversationitem", lockMode = PESSIMISTIC_WRITE)
public class Conversation implements Serializable{

    private static final long serialVersionUID = 113744899661593117L;

    @Id
    @Column(name = "conininput")
    String conininput;

    @Column(name = "conresponse")
    String conresponse;

    public String getConininput() {
        return conininput;
    }

    public void setConininput(String conininput) {
        this.conininput = conininput;
    }
}
```

```

    }

    public String getConresponse() {
        return conresponse;
    }

    public void setConresponse(String conresponse) {
        this.conresponse = conresponse;
    }

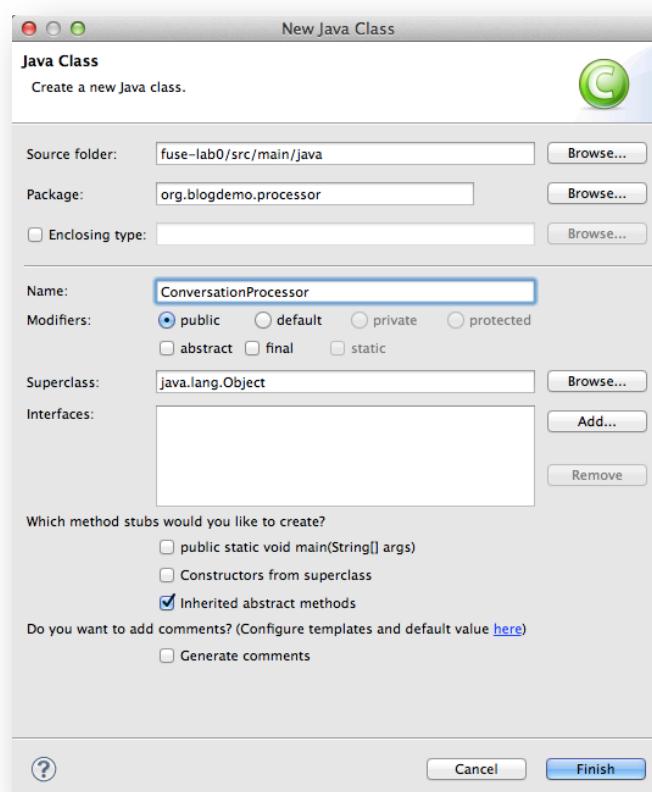
}

```

4. We are also going to add a bean processor, which in this lab, only print out the content of the bean. Create Processors.

Under package : org.blogdemo.processor

Class Name: ConversationProcessor



Paste the following code into ConversationProcessor.java

```

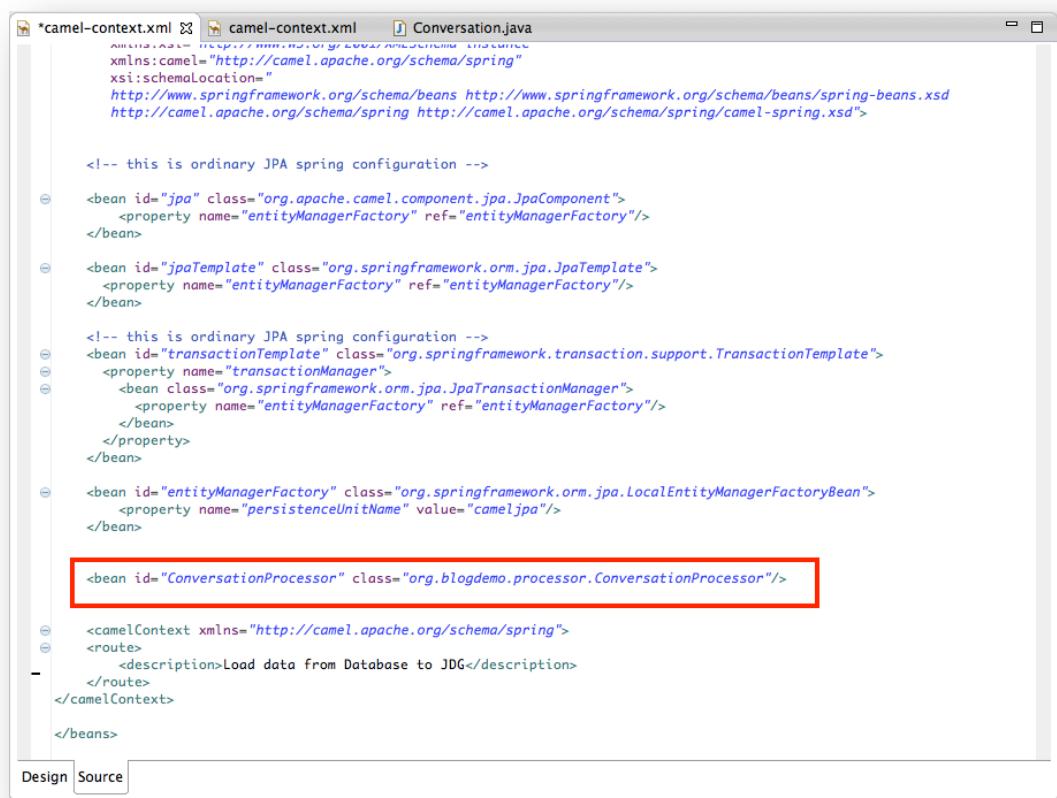
package org.blogdemo.processor;
import org.blogdemo.model.Conversation;

public class ConversationProcessor {
    public Conversation processConversation(Conversation conversation) {
        System.out.println("Input: ["+conversation.getConininput()+"]");
        Response: [" +conversation.getConresponse()+"];
        return conversation;
    }
}

```

```
}
```

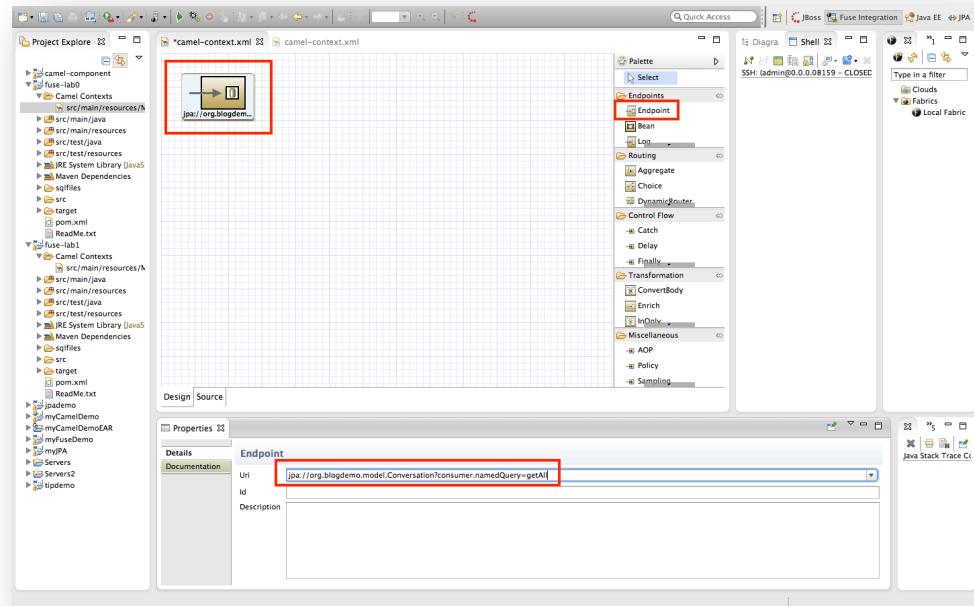
And add bean declaration on the route (camel-context.xml).



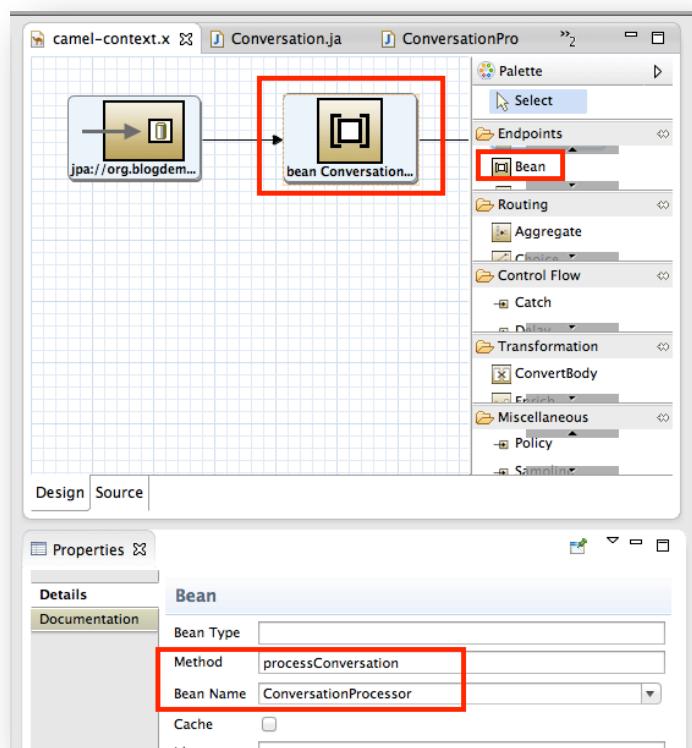
```
<bean id="ConversationProcessor" class="org.blogdemo.processor.ConversationProcessor"/>
```

5. Open up and edit Camel Route (camel-context.xml) and choose design tab.
- a. Add JPA endpoint

Uri: `jpa://org.blogdemo.model.Conversation?consumer.namedQuery=getAll`

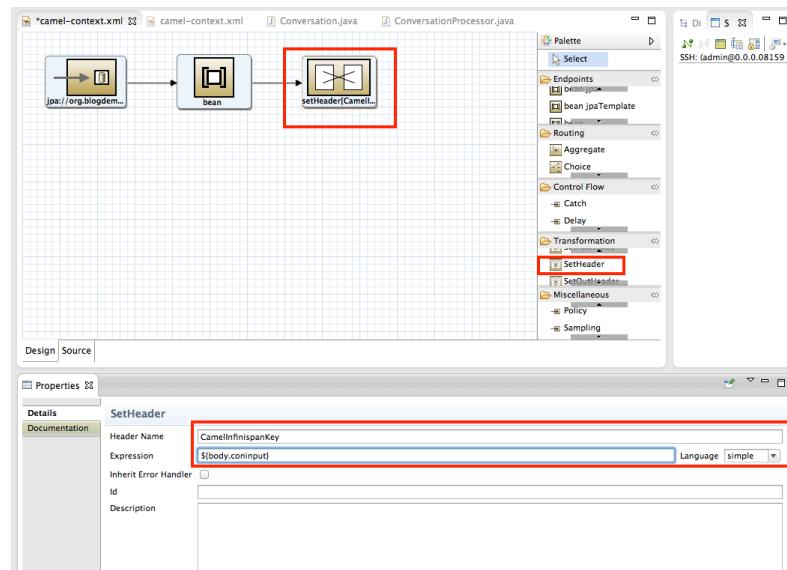


- b. Add Processor Bean to print out the content from database before we insert it into JDG Cache.



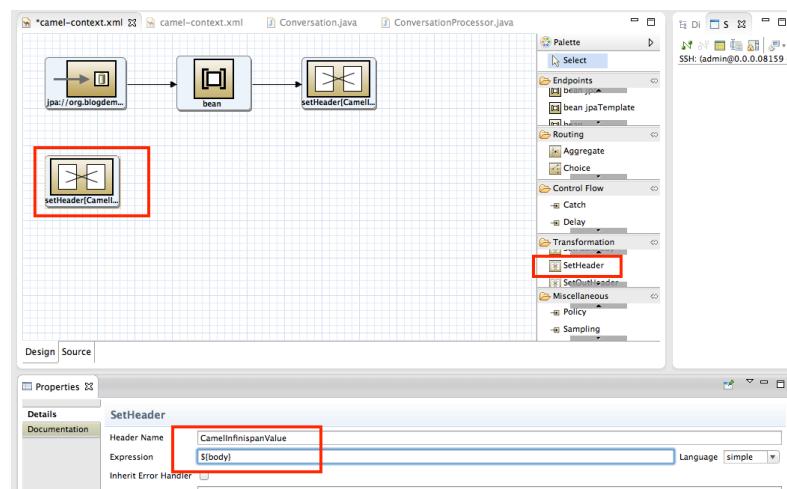
Method: *processConversation*
Bean Name: *ConversationProcessor*

- c. All the content needs to be set in the header for it to put inside the cache. That's why we need content transformation adding the right key and value.



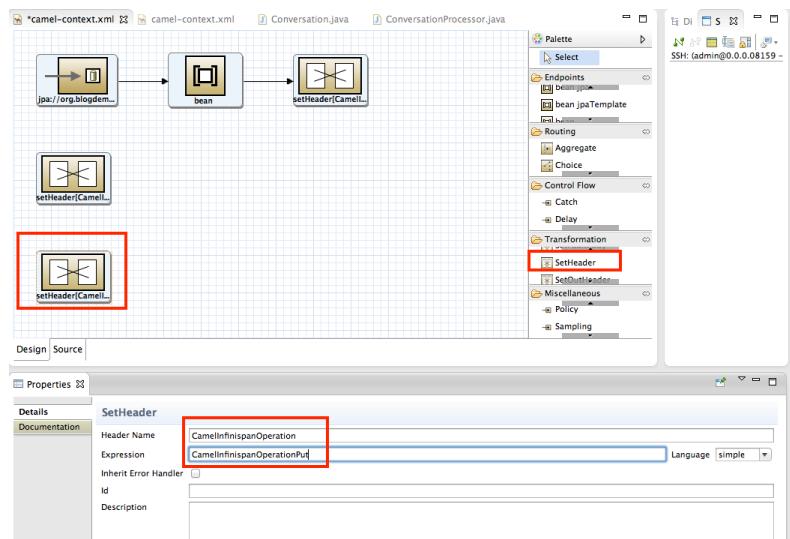
HeaderName : CamelInfinispanKey

Expression : \${body.coninput} (Note this is the key in the Conversation Bean)



HeaderName : CamelInfinispanValue

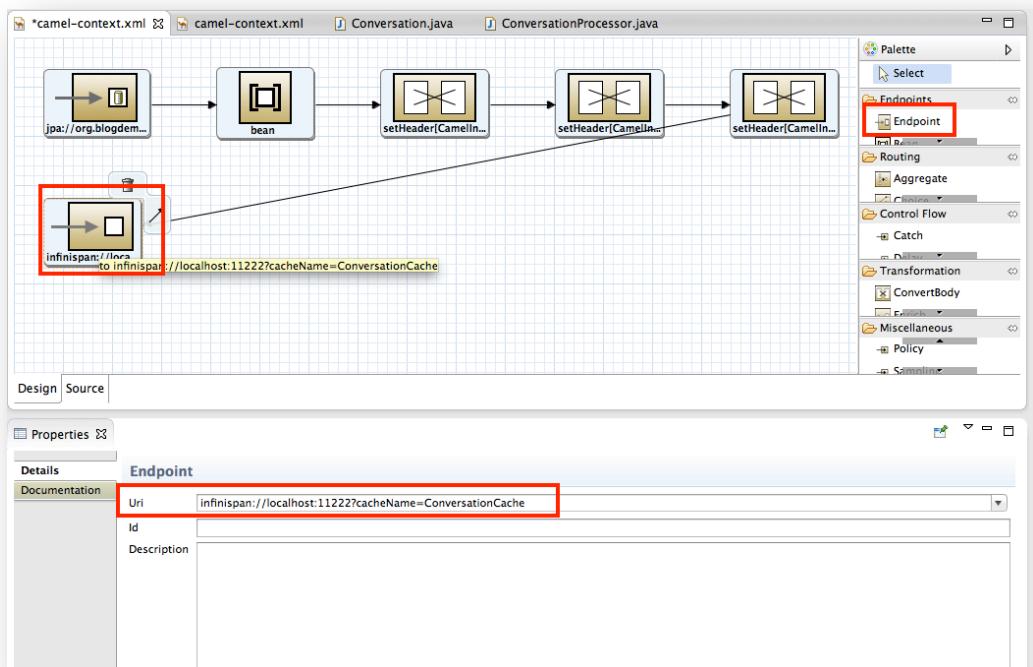
Expression : \${body } (Note this is the Conversation Bean)



HeaderName : CamelInfinispanOperation
Expression : CamelInfinispanOperationPut

d. Add InfiniSpan Endpoint

Uri: `infinispan://localhost:11222?cacheName=ConversationCache`



6. Don't forget to link all the components together and save.

7. Go to command line load data to H2 database. Please run
`mvn process-test-resources`
 in the project directory.

8. Compile camel project

`mvn compile`

9. Start up JDG, go to target/jboss-datagrid-6.2.0-server/bin
run ./standalone.sh

```
christinasteck-MacBook-Air:bin christinasteck$ ./standalone.sh
=====
JBoss Bootstrap Environment

JBOS_HOME: /users/christina/Desktop/FUSELab/lab08/target/jboss-datagrid-6.2.6-server
JAVA: /Library/Java/JavaVirtualMachines/jdk1.7.0_50.jdk/Contents/Home/bin/java
JAVA_OPTS: -server -XX:UseCompressedOops -Xms64m -Xmx512m -XX:MaxPermSize=256m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.modules -Djava.awt.headless=true

23:12:01,516 INFO [org.jboss.modules] (main) JBoss Modules version 1.2.2.Final-redhat-1
23:12:01,743 INFO [org.jboss.msc] (main) JBoss MSC version 1.0.4.GA-redhat-1
obj:[B@4304: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtualMachines/jdk1.7.0_50.jdk/Contents/Home/bin/java and /Library/Java/JavaVirtualMachines/jdk1.7.0_50.jdk/Contents/Home/jre/lib/libinstrumentation.dylib. One of the two will be used. Which one is undefined.
23:12:02,064 INFO [org.xnio] (MSC service thread 1-8) XNIO Version 3.8.7.GA-redhat-1
23:12:02,069 INFO [org.xnio.nio] (MSC service thread 1-8) XNIO Implementation Version 3.8.7.GA-redhat-1
23:12:02,073 INFO [org.xnio.server] (Controller Boot Thread) JBA5015888: Creating http management service using socket-binding (management-http)
23:12:02,085 INFO [org.jboss.remoting] (MSC service thread 1-8) JBoss Remoting version 3.2.16.GA-redhat-1
23:12:02,091 INFO [org.jboss.solder.starter.infinispan] (ServerService Thread Pool -- 21) JBA5013171: Activating Infinispan subsystem.
23:12:02,147 INFO [org.jboss.as.naming] (MSC service thread 1-4) JBA5013171: Activating Naming Subsystem
23:12:02,162 INFO [org.jboss.as.naming] (ServerService Thread Pool -- 21) JBA5018800: Activating Naming Subsystem
23:12:02,170 INFO [org.jboss.as.naming] (MSC service thread 1-4) JBA5013170: Current PicketBox version=4.0.17.SP2-redhat-2
23:12:02,182 INFO [org.jboss.as.jsf] (ServerService Thread Pool -- 27) JBA5012605: Activated the following JSF Implementations: [main, 1.2]
23:12:02,186 INFO [org.jboss.as.connector.logging] (MSC service thread 1-4) JBA5013170: Starting Subsystem [IronJacamar 1.0.19.Final-redhat-2]
23:12:02,310 INFO [org.jboss.as.connector] (MSC service thread 1-4) JBA5013170: Started IronJacamar 1.0.19.Final-redhat-2
23:12:02,894 INFO [org.apache.coyote.ajp13] (MSC service thread 1-4) JWBEB003046: Starting Coyote AJP/1.3 on ajp://127.0.0.1:8009
23:12:02,895 INFO [org.apache.coyote.http11] (MSC service thread 1-2) JWBEB003001: Coyote HTTP/1.1 initializing on : http://127.0.0.1:8088
23:12:02,897 INFO [org.apache.coyote.http11] (MSC service thread 1-7) JWBEB003100: Coyote HTTP/1.1 starting on http://127.0.0.1:8088
23:12:04,326 INFO [org.jboss.as.clustering.infinispan] (MSC service thread 1-2) JBA5017100: Started jboss-wb-policy cache from security container
23:12:04,327 INFO [org.jboss.as.clustering.infinispan] (MSC service thread 1-2) JBA5017100: Started jboss-wb-policy cache from local container
23:12:04,561 INFO [org.infinispan.server.endpoint] (MSC service thread 1-1) JDSGS010000: HotRodServer starting
23:12:04,561 INFO [org.infinispan.server.endpoint] (MSC service thread 1-1) JDSGS010001: HotRodServer listening on 127.0.0.1:11222
23:12:04,814 INFO [org.infinispan.factories.GlobalComponentRegistry] (MSC service thread 1-8) ISPN000128: Infinispan version: Infinispan 'Infinispan' 6.0.1.Final-redhat-1
23:12:05,194 INFO [org.infinispan.jmx.CacheJmxRegistration] (MSC service thread 1-2) ISPN000031: MBeans were successfully registered to the platform MBean server.
23:12:05,196 INFO [org.infinispan.jmx.CacheJmxRegistration] (MSC service thread 1-2) ISPN000031: MBeans were successfully registered to the platform MBean server.
23:12:05,196 INFO [org.infinispan.jmx.CacheJmxRegistration] (MSC service thread 1-2) ISPN000031: MBeans were successfully registered to the platform MBean server.
23:12:05,196 INFO [org.infinispan.jmx.CacheJmxRegistration] (MSC service thread 1-2) ISPN000031: MBeans were successfully registered to the platform MBean server.
23:12:05,196 INFO [org.infinispan.jmx.CacheJmxRegistration] (MSC service thread 1-2) ISPN000031: MBeans were successfully registered to the platform MBean server.
23:12:05,196 INFO [org.infinispan.jmx.CacheJmxRegistration] (MSC service thread 1-2) ISPN000031: MBeans were successfully registered to the platform MBean server.
23:12:05,199 INFO [org.infinispan.jmx.CacheJmxRegistration] (MSC service thread 1-2) ISPN000031: MBeans were successfully registered to the platform MBean server.
23:12:05,200 INFO [org.infinispan.jmx.CacheJmxRegistration] (MSC service thread 1-2) ISPN000031: MBeans were successfully registered to the platform MBean server.
23:12:05,261 INFO [org.jboss.as.clustering.infinispan] (MSC service thread 1-6) JBA5010281: Started namedCache Cache from local container
23:12:05,283 INFO [org.jboss.as.clustering.infinispan] (MSC service thread 1-8) JBA5010281: Started other Cache from security container
23:12:05,286 INFO [org.infinispan.server.endpoint] (MSC service thread 1-7) JDSGS010001: ManagedCacheCache starting
23:12:05,286 INFO [org.infinispan.server.endpoint] (MSC service thread 1-7) JDSGS010001: ManagedCacheCache listening on 127.0.0.1:1211
23:12:05,288 INFO [org.jboss.as.clustering.infinispan] (MSC service thread 1-5) JBA5010281: Started default Cache from local container
23:12:05,215 INFO [org.infinispan.server.endpoint] (MSC service thread 1-8) JDSGS010000: REST starting
23:12:05,559 INFO [org.jboss.as] (Controller Boot Thread) JBA5015961: Http management interface listening on http://127.0.0.1:9990/management
23:12:05,611 INFO [org.jboss.as] (Controller Boot Thread) JBA5015961: Main context listening on http://127.0.0.1:9990
23:12:05,615 INFO [org.jboss.as] (Controller Boot Thread) JBA5015874: 0bss Data Grid 6.0.8 (AS 7.2.1.Final-redhat-10) started in 4594ms - Started 93 of 132 services (39 services are passive or on-demand)
```

10. Run Camel Route, if you see the data loaded, then congratulation, you have done it!

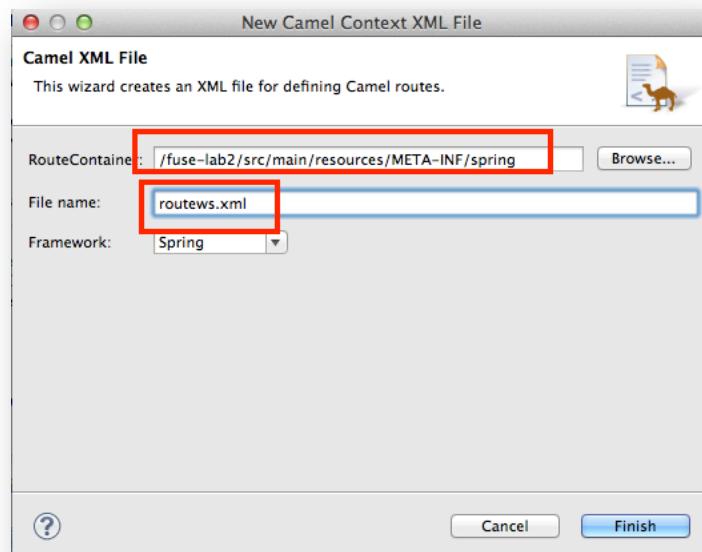
```
mvn camel:run
```

bash	bash	bash	java	bash
[INFO] Executing file: /var/folders/q4/0z50qnf7q0cylgk6z5tt_00000gn/T/wiri.1521411161isql				
[INFO] 10 of 10 SQL statements executed successfully				
[INFO] --- maven-compiler-plugin:2.5.1:testCompile (default-testCompile) @ fuse-lab1 ---				
[INFO] Nothing to compile - all classes are up to date				
[INFO]				
<<< camel-maven-plugin:2.12.0.redhat-610379:run (default-cli) @ fuse-lab1 <<<				
[INFO] --- camel-maven-plugin:2.12.0.redhat-610379:run (default-cli) @ fuse-lab1 ---				
[INFO] Using org.apache.camel.spring.Main to initiate a CamelContext				
[INFO] Starting Camel				
[apache.camel.spring.Main.main()] MainSupport	INFO	Apache Camel 2.12.0.redhat-610379 starting		
[apache.camel.spring.Main.main()] HibernatePersistence	WARN	HHHHH01016: Encountered a deprecated javax.persistence.spi.PersistenceProvider [org.hibernate.ejb.HibernatePersistence]; use [org.hiber-		
name.jpa.HibernatePersistenceProvider] instead.		nate.jpa.HibernatePersistenceProvider] instead.		
[apache.camel.spring.Main.main()] HibernatePersistence	WARN	HHHHH01016: Encountered a deprecated javax.persistence.spi.PersistenceProvider [org.hibernate.ejb.HibernatePersistence]; use [org.hiber-		
name.jpa.HibernatePersistenceProvider] instead.		nate.jpa.HibernatePersistenceProvider] instead.		
[apache.camel.spring.Main.main()] HibernatePersistence	WARN	HHHHH01016: Encountered a deprecated javax.persistence.spi.PersistenceProvider [org.hibernate.ejb.HibernatePersistence]; use [org.hiber-		
name.jpa.HibernatePersistenceProvider] instead.		nate.jpa.HibernatePersistenceProvider] instead.		
[apache.camel.spring.Main.main()] LogHelper	INFO	HHHHH000204: Processing PersistenceUnitInfo [
name: camelpj				
[apache.camel.spring.Main.main()] Version	INFO	HHHHH000412: Hibenate Core (4.3.5.Final)		
[apache.camel.spring.Main.main()] Environment	INFO	HHHHH000306: hibernate.properties not found		
[apache.camel.spring.Main.main()] Environment	INFO	HHHHH000421: Bytecode provider name : javassist		
[apache.camel.spring.Main.main()] Version	INFO	HCANN000001: Hibenate Commons Annotations {4.0.4.Final}		
[apache.camel.spring.Main.mainImpl]	INFO	HHHHH000402: Using Hibenate built-in connection pool (not for production use!)		
[apache.camel.spring.Main.mainImpl]	INFO	HHHHH000401: using driver [org.h2.Driver] at URL [jdbc:h2:file:target/db/testdb]		
[apache.camel.spring.Main.mainImpl]	INFO	HHHHH000403: using dialect [org.hibernate.dialect.H2Dialect]		
[apache.camel.spring.Main.mainImpl]	INFO	HHHHH000405: Autocommit mode: true		
[apache.camel.spring.Main.mainImpl]	INFO	HHHHH000115: Hibenate connection pool size: 20 (min=1)		
[apache.camel.spring.Main.mainImpl]	INFO	HHHHH000406: Using dialect: org.hibernate.dialect.H2Dialect		
[apache.camel.spring.Main.mainImpl]	INFO	HHHHH000422: Disabling contextual LOB creation as connection was null		
[apache.camel.spring.Main.mainImpl]	INFO	HHHHH000200: Registering BeanFactory [org.springframework.beans.factory.support.DefaultListableBeanFactory]		
[apache.camel.spring.Main.mainImpl]	INFO	HHHHH000220: Running hibernate schema update		
[apache.camel.spring.Main.mainImpl]	INFO	HHHHH000187: Fetching database metadata		
[apache.camel.spring.Main.mainImpl]	INFO	HHHHH000396: Updating schema		
[apache.camel.spring.Main.mainImpl]	INFO	HHHHH000261: Table found: TESTDB.FUSEDEMO_CONVERSATION		
[apache.camel.spring.Main.mainImpl]	INFO	HHHHH000837: Columns: [coninput, conresponse]		
[apache.camel.spring.Main.mainImpl]	INFO	HHHHH000104: Foreign keys: []		
[apache.camel.spring.Main.mainImpl]	INFO	HHHHH000126: Indexes: [primary_key_b]		
[apache.camel.spring.Main.mainImpl]	INFO	HHHHH000232: Schema update complete		
[apache.camel.spring.Main.mainImpl]	INFO	Apache Camel 2.12.0.redhat-610379 (CamelContext: camel-1) is starting		
[apache.camel.spring.Main.mainImpl]	INFO	JMX is enabled		
[apache.camel.spring.Main.mainImpl]	INFO	Using StreamCaching for converters		
[apache.camel.spring.Main.mainImpl]	INFO	Using EntityManagerFactory configured: org.springframework.orm.jpa.LocalEntityManagerFactoryBean@57a5a0f2		
[apache.camel.spring.Main.mainImpl]	INFO	Using TransactionManager found in registry with id [transactionTemplate] org.springframework.orm.jpa.JpaTransactionManager@6c7d778		
[apache.camel.spring.Main.mainImpl]	INFO	AllowsOriginalMessage is enabled. If access to the original message is not needed, then its recommended to turn this option off as it may improve performance.		
[apache.camel.spring.Main.mainImpl]	INFO	StreamCaching is not in use. If using streams then its recommended to enable stream caching. See more details at http://camel.apache.org/stream-caching.html		
[apache.camel.spring.Main.mainImpl]	INFO	ISPM0004021: Infinispan version: 6.0.1.Final		
[apache.camel.spring.Main.mainImpl]	INFO	ROUTE1 started and consuming From: Endpoint[jpa://org.blogdemo.model.Conversation?consumer.namedQuery=getAll]		
[apache.camel.spring.Main.mainImpl]	INFO	Total 1 routes, of which 1 is started.		
[apache.camel.spring.Main.mainImpl]	INFO	Apache Camel 2.12.0.redhat-610379 (CamelContext: camel-1) started in 0.618 seconds		
Input:[how are you Response:[so so]				
Input:[what are you doing Response:[NITE ofcourse]				
Input:[how old are you Response:[1 week old]				
Input:[what is JBoss Fuse Response:[JBoss Fuse is an open source Enterprise Service Bus (ESB).]				
Input:[what is JBoss AMQ Response:[JBoss A-MQ is a high performance, flexible, cost-effective messaging platform.]				
Input:[what is JBoss A-MQ Response:[JBoss A-MQ is a high performance, flexible, cost-effective messaging platform.]				

Lab 2

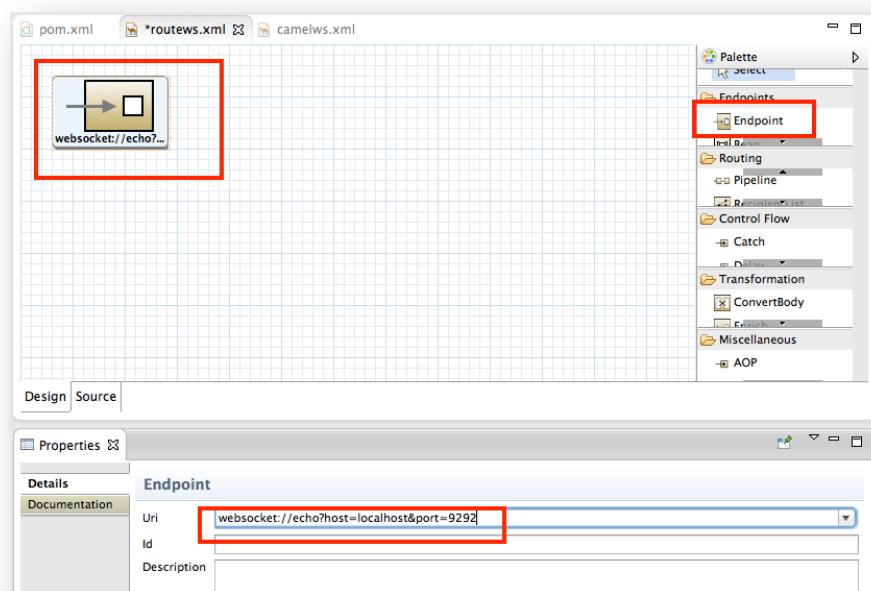
If you did not complete the last lab or having trouble with it. Go to Download the completed lab to start with lab2.

1. Create new route, under the same directory as your other route, name it routews.xml



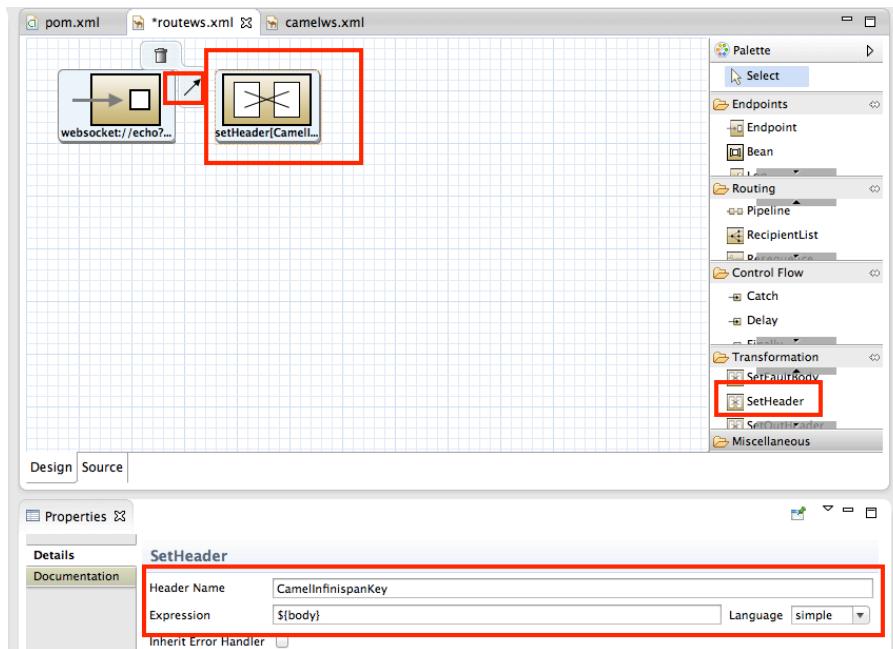
2. Since we are listening through WebSocket, we need to add WebSoket endpoint . Drag Endpoint to the left canvas .

Uri: `websocket://echo?host=localhost&port=9292`



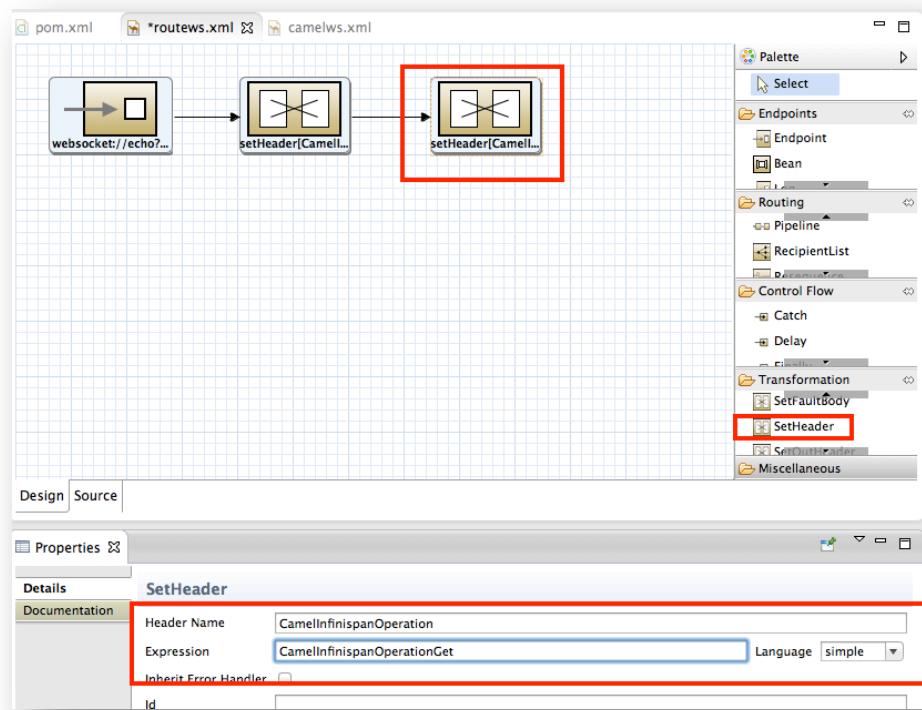
3. We want to get content out from the JDG cache, needs to set the key to look up in the header and also specify the action GET.

4.



Method: *CamelInfinispanKey*

Bean Name: *\${body}*

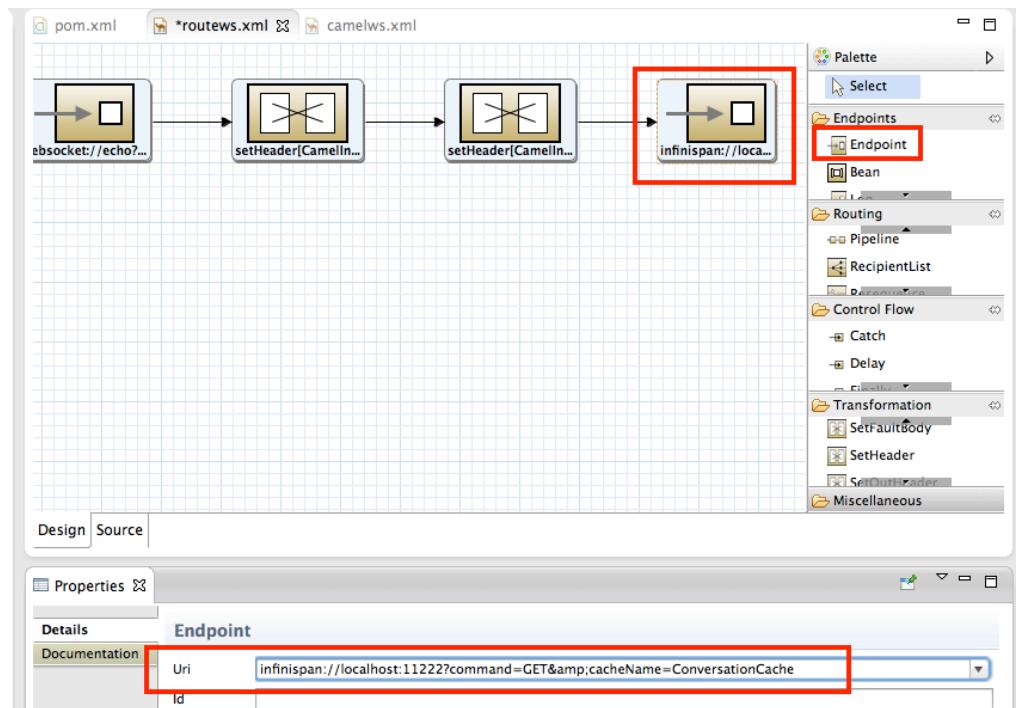


Method: *CamelInfinispanOperation*

Bean Name: *CamelInfinispanOperationGet*

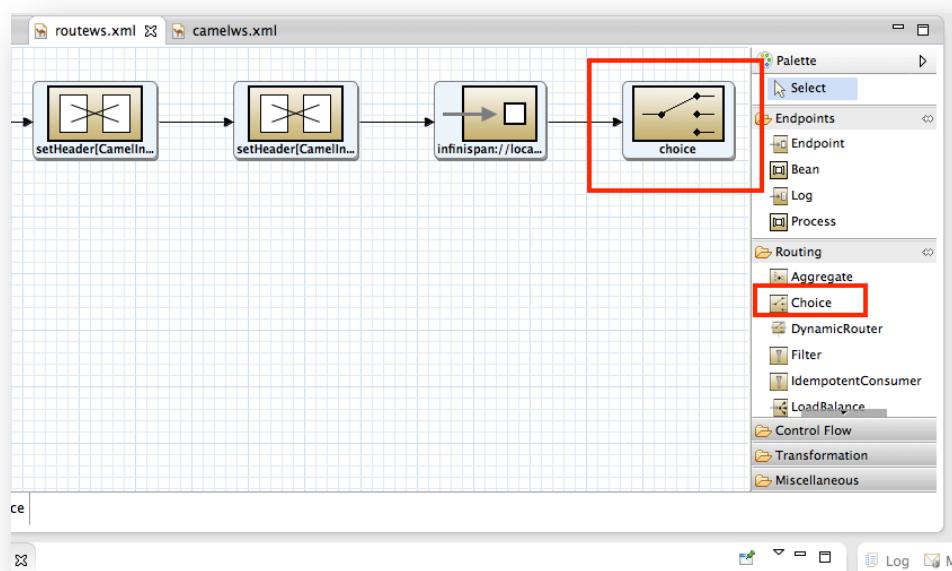
Add the JDG endpoint.

URI: `infinispan://localhost:11222?command=GET&cacheName=ConversationCache`

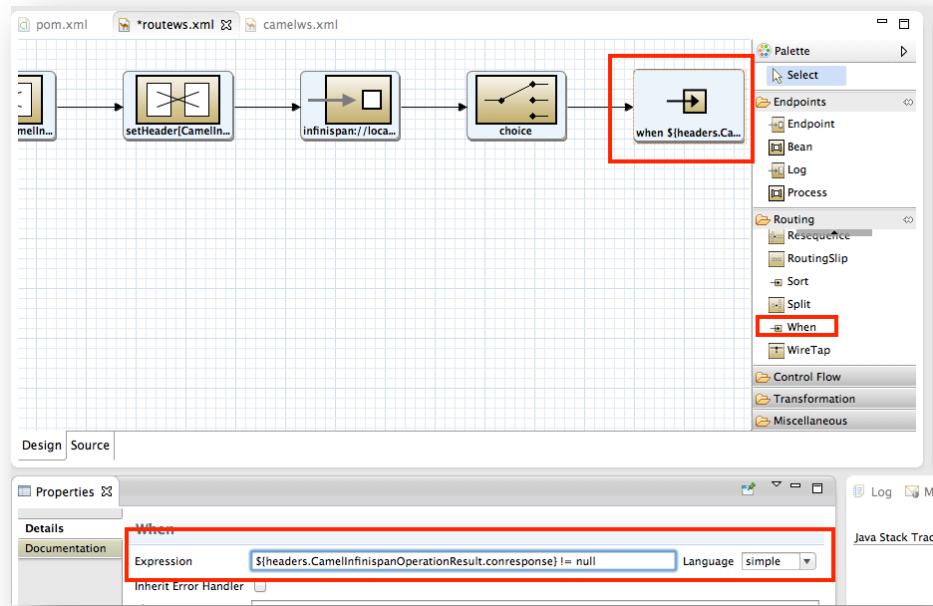


5. Base on the content we retrieve from JDG, determine what we reply back to webpage.

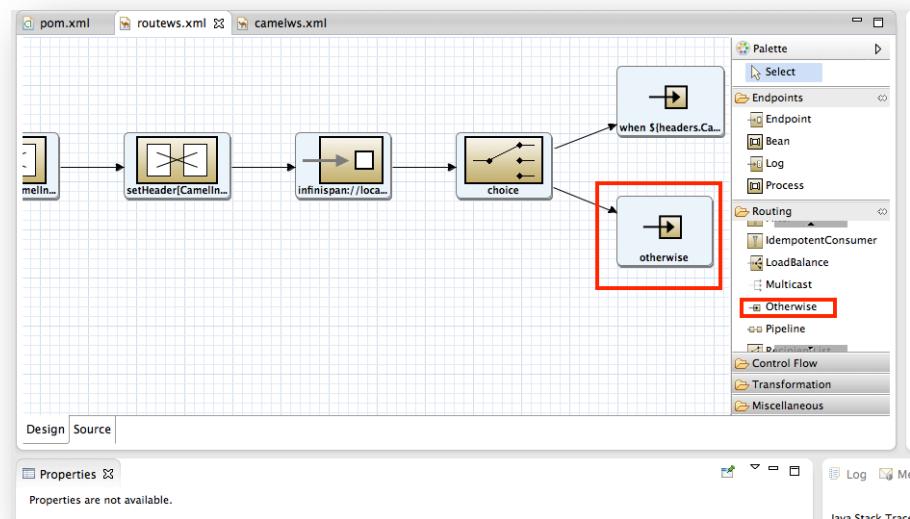
We are going to use Content Switching, drag Choice to the left corner.



Add when component, we want to make sure it's found in JDG, so we add
 `${headers.CamelInfinispanOperationResult.conresponse} != null`



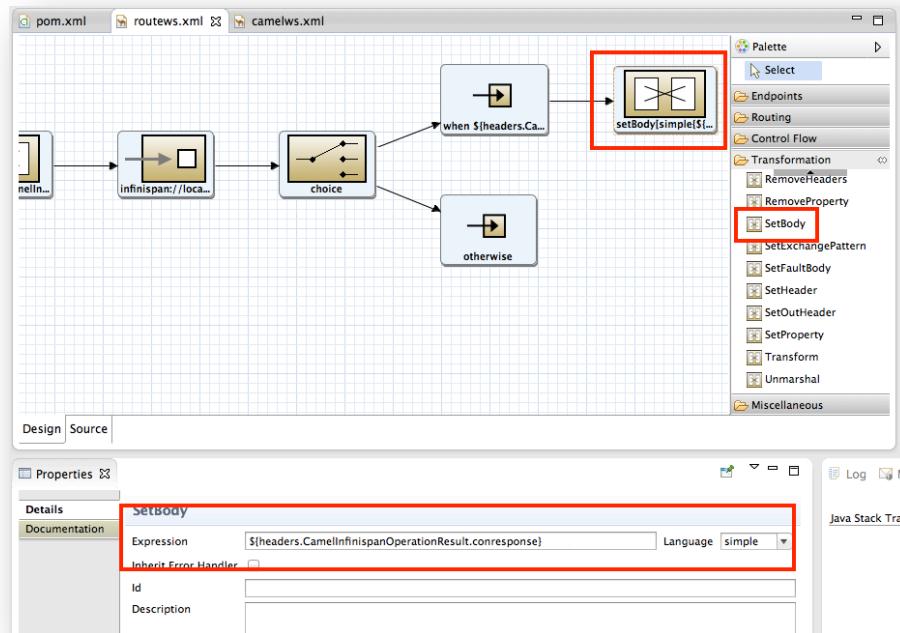
We need also to have a response if the content was not found in JDG, so, add Otherwise.



When the content is found we want to reply the content that we found in JDG. So we will set the result to body

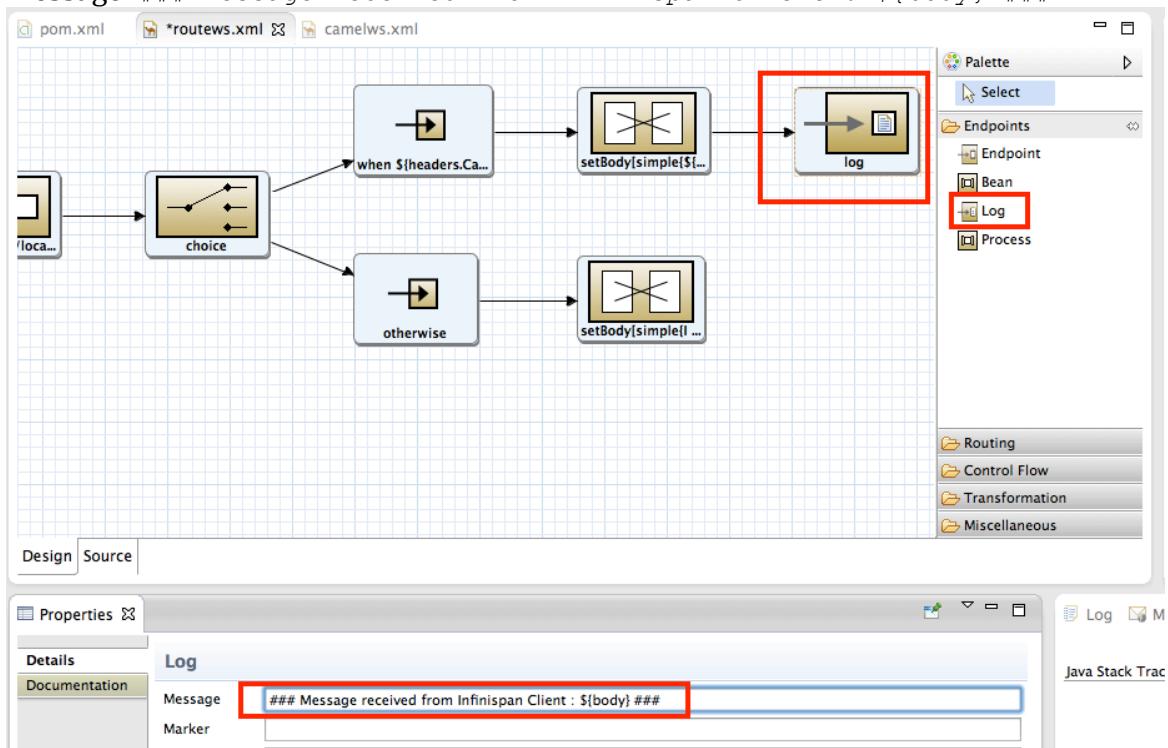
Expression:

`$headers.CamelInfinispanOperationResult.conresponse}`



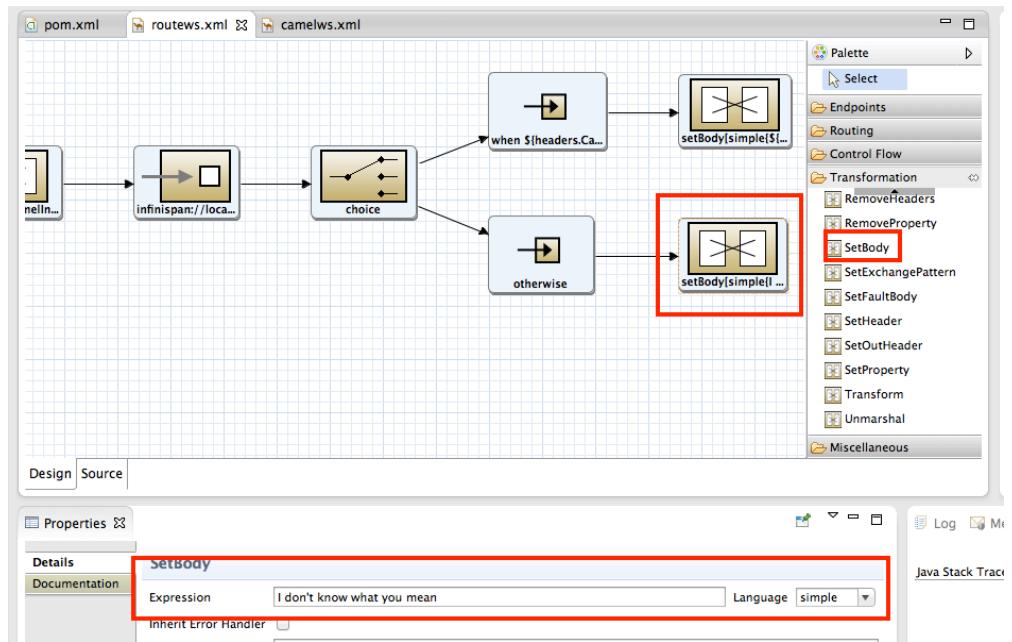
Add log,

Message: ### Message received from Infinispan Client : \${body} ###



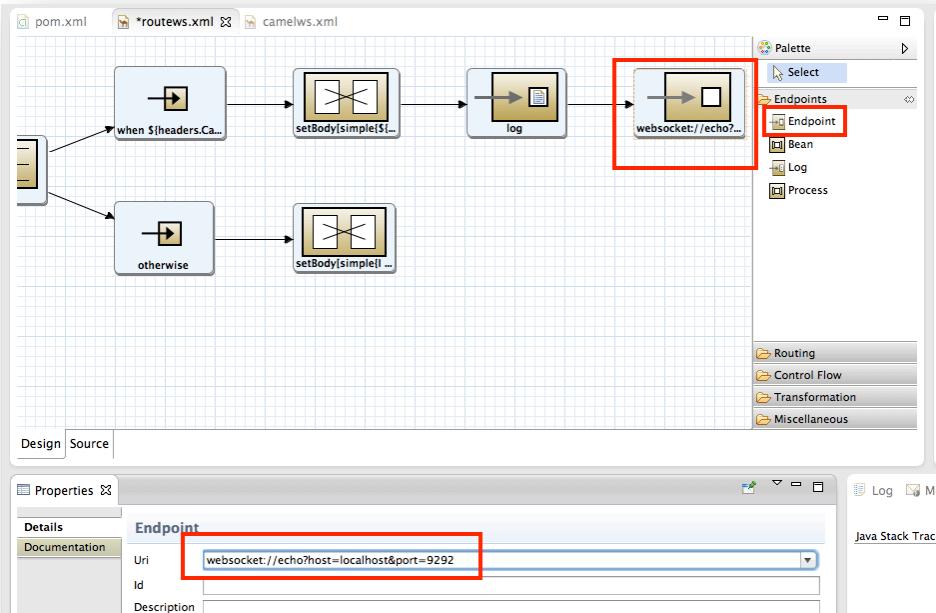
And if there are nothing found in JDG, we need a notify message:

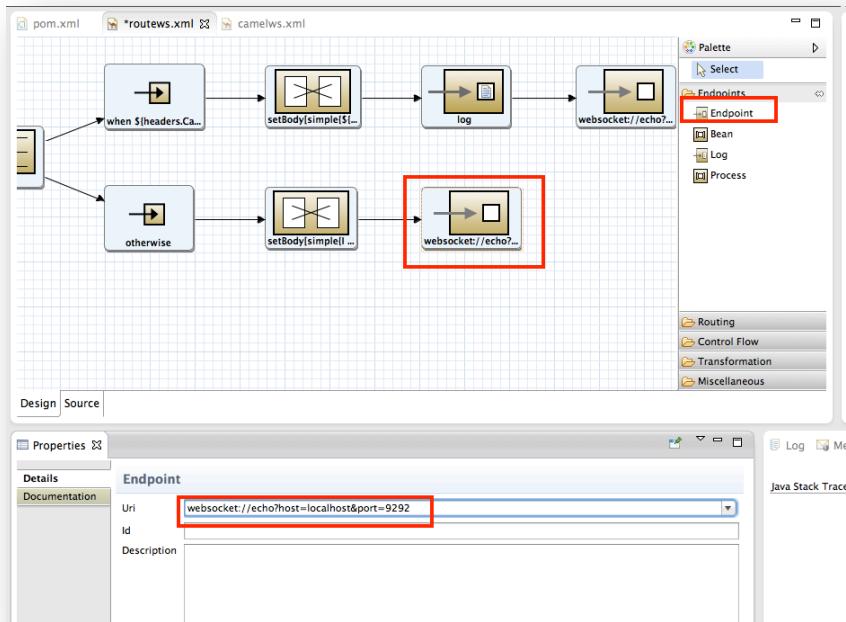
Expression: I don't know what you mean



- Lastly we need these result to be send back to the webpage, Send data back to WebSocket endpoint
URI: `websocket://echo?host=localhost&port=9292`

Connect to both route.





That's it for development.

We can now start running the example:

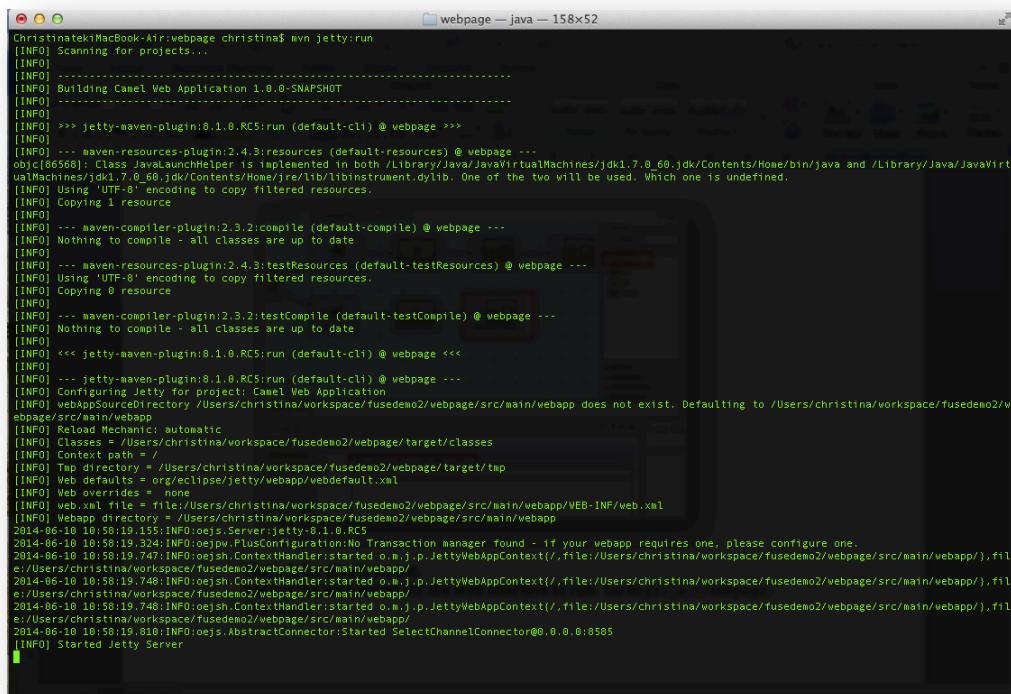
Frist make sure your JDG is still running.

Start up the camel route by running under /project/labx/directory

```
mvn camel:run
```

Startup the web server for the web interface to run. Go to `project/webpage/` directory.

mvn jetty:run



```
[INFO] Scanning for projects...
[INFO]
[INFO] Building Camel Web Application 1.0.0-SNAPSHOT
[INFO]
[INFO] --- jetty-maven-plugin:8.1.0.RC5:run (default-cli) @ webpage ---
[INFO]
[INFO] --- maven-resources-plugin:2.4.3:resources (default-resources) @ webpage ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO]
[INFO] --- maven-compiler-plugin:2.3.2:compile (default-compile) @ webpage ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.4.3:testResources (default-testResources) @ webpage ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 8 resource
[INFO]
[INFO] --- maven-compiler-plugin:2.3.2:testCompile (default-testCompile) @ webpage ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] <<< jetty-maven-plugin:8.1.0.RC5:run (default-cli) @ webpage <<<
[INFO]
[INFO] --- jetty-maven-plugin:8.1.0.RC5:run (default-cli) @ webpage ---
[INFO] Configuring Jetty for project: Camel Web Application
[INFO] webappSourceDirectory /Users/christina/workspace/fusedemo2/webpage/src/main/webapp does not exist. Defaulting to /Users/christina/workspace/fusedemo2/webpage/src/main/webapp
[INFO] Reload Mechanic: automatic
[INFO] Classes = /Users/christina/workspace/fusedemo2/webpage/target/classes
[INFO] ContextHandler directory = /Users/christina/workspace/fusedemo2/webpage/target/tmp
[INFO] Web defaults = org/eclipse/jetty/webapp/webdefault.xml
[INFO] Web overrides = none
[INFO] web.xml file = file:/Users/christina/workspace/fusedemo2/webpage/src/main/webapp/WEB-INF/web.xml
[INFO] Webapp directory = /Users/christina/workspace/fusedemo2/webpage/src/main/webapp
2014-06-10 10:58:19.155:INFO:oejs.Server:jetty-8.1.0.RC5
2014-06-10 10:58:19.324:INFO:oejw.PlusConfiguration:No Transaction manager found - if your webapp requires one, please configure one.
2014-06-10 10:58:19.747:INFO:oeish.ContextHandler:started o.m.j.p.JettyWebAppContext(/,file:/Users/christina/workspace/fusedemo2/webpage/src/main/webapp/),fil
e:/Users/christina/workspace/fusedemo2/webpage/src/main/webapp/
2014-06-10 10:58:19.748:INFO:oeijs.ContextHandler:started o.m.j.p.JettyWebAppContext(/,file:/Users/christina/workspace/fusedemo2/webpage/src/main/webapp/),fil
e:/Users/christina/workspace/fusedemo2/webpage/src/main/webapp/
2014-06-10 10:58:19.748:INFO:oeish.ContextHandler:started o.m.j.p.JettyWebAppContext(/,file:/Users/christina/workspace/fusedemo2/webpage/src/main/webapp/),fil
e:/Users/christina/workspace/fusedemo2/webpage/src/main/webapp/
2014-06-10 10:58:19.810:INFO:oejs.AbstractConnector:Started SelectChannelConnector@0.0.0.0:8585
[INFO] Started Jetty Server
```

Go to <http://localhost:8585/fuseworkshop/demo.html>

You will see the console, try "hi", you should see the built in response.



Type yellow , you should see the not found response.

