A Fuse/Data Virtualisation/BPMS Integrated Demo

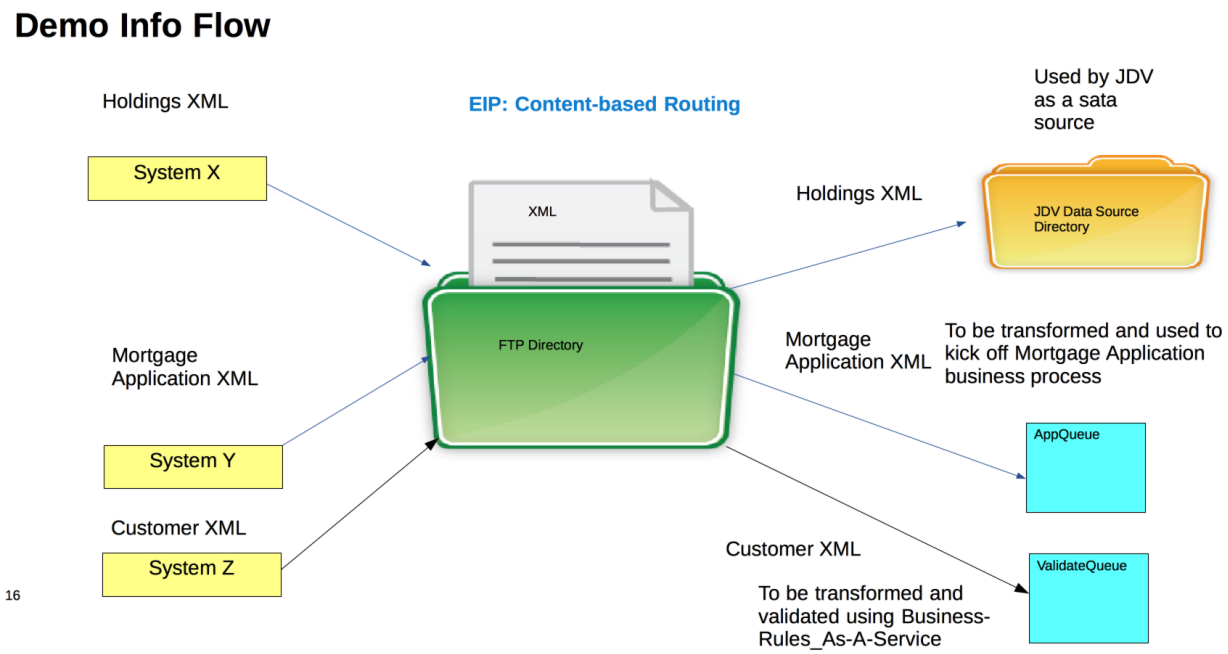
Need for an Integrated Demo

Red Hat has been using the phrase "Accelerate, Integrate and Automate", for sometime now, to explain its comprehensive middleware portfolio to its customers. Red Hat middleware does not work in isolation. It is only natural that we show our customers how the different middleware products can work together to achieve a business outcome.

In this article, I am going to show you a demo which involves the 2 "Integrate" products: JBoss Fuse (Fuse), JBoss Data Virtualisation (JDV) and the "Automate" products: JBoss Business Process Management Suite (BPMS)/JBoss Business Rules Management System (BRMS) together. Since BPMS/BRMS and JDV run on JBoss Application Server (EAP), we can even claim that the integrated demo involves middleware products of all 3 categories ie, Accelerate, Integrate and Automate. For this reason and the lack of a better name, I am going to name this demo the ***JBoss-Middleware-In-Action-Demo*** ;-)

The Business Scenario

The following diagram describes the business scenario this integrated demo implements:



A banks receives XML documents from its trading partners. These documents are placed in a specific server's directory. This demo does not cover how the documents get there (maybe via managed FTP or other unspecified means. In the demo, we shall drag-and-drop files into this directory). There are 3 types of XML documents:

1. A Holdings XML - this is an XML document that contains banking customers other personal holdings (eg, mortgages, cars, etc.) to complete the net-worth picture of customers. More on this later.
2. A Customer XML - this is an XML document containing customer details that require validation by using Business-Rules-As-A-Service.
3. A Mortgage Application XML - this is an XML document sent by a partner to start a customer mortgage application. The Mortgage Application business process is running on BPMS.

The Holding XML requires a bit more explanation. Like most organisations, the information regarding a customer and his net worth is not stored in a central place. Instead, one has to go get information in several places to get a complete picture. In this demo, the data can be obtained from 3 data sources:

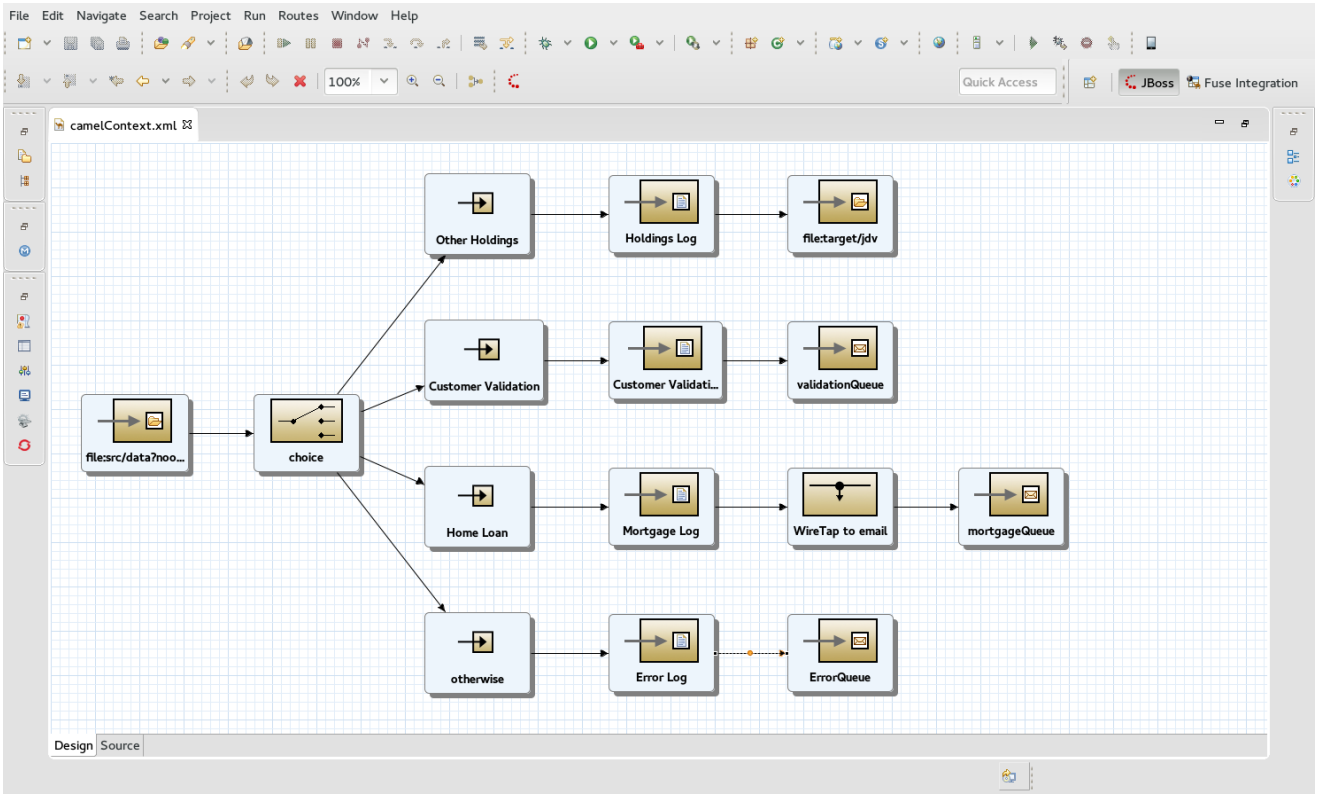
1. Account Information including shares ownership (stored in a MySQL database)
2. Market Data (stock price retrieved using a REST Web Service)
3. Other Personal Holdings (stored in an XML file which we refer to as a Holdings XML earlier)

The implementation of this business scenario illustrates the following:

- Use of JBoss Fuse to implement Enterprise Integration Patterns (EIPs) which serve as the linchpin of the solution
- Use of message queues
- Use of the graphical data mapper
- Invoke BPMS' Business-Rules-As-A-Service running on the Realtime Decision Server
- Start a Business Process running on the BPMS Execution server from JBoss Fuse
- Deliver the XML file needed as the 3rd data source (the Holdings XML file) by JBoss Data Virtualisation to provide a consolidated view of customer assets.

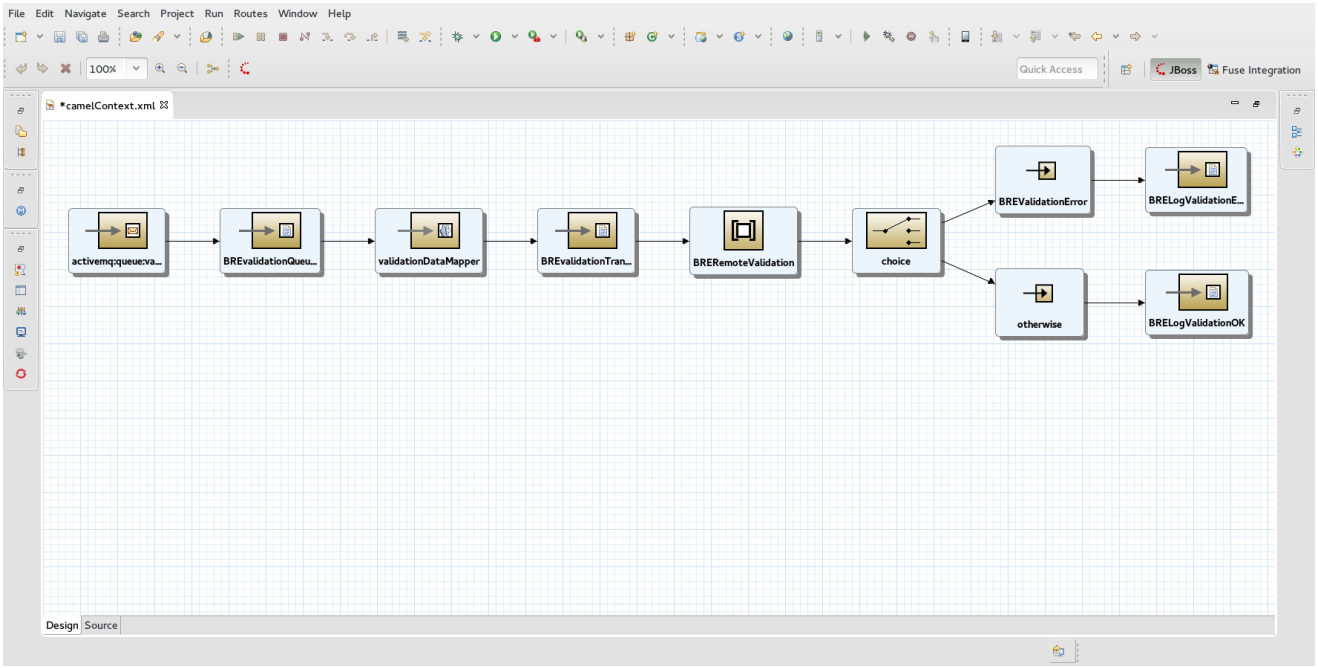
Fuse Camel Routes

As stated in the previous section, Fuse Camel routes implementing Enterprise Integration Patterns serve as the linchpin for the whole solution. The main route implementing EIP Content-based Routing is shown below:

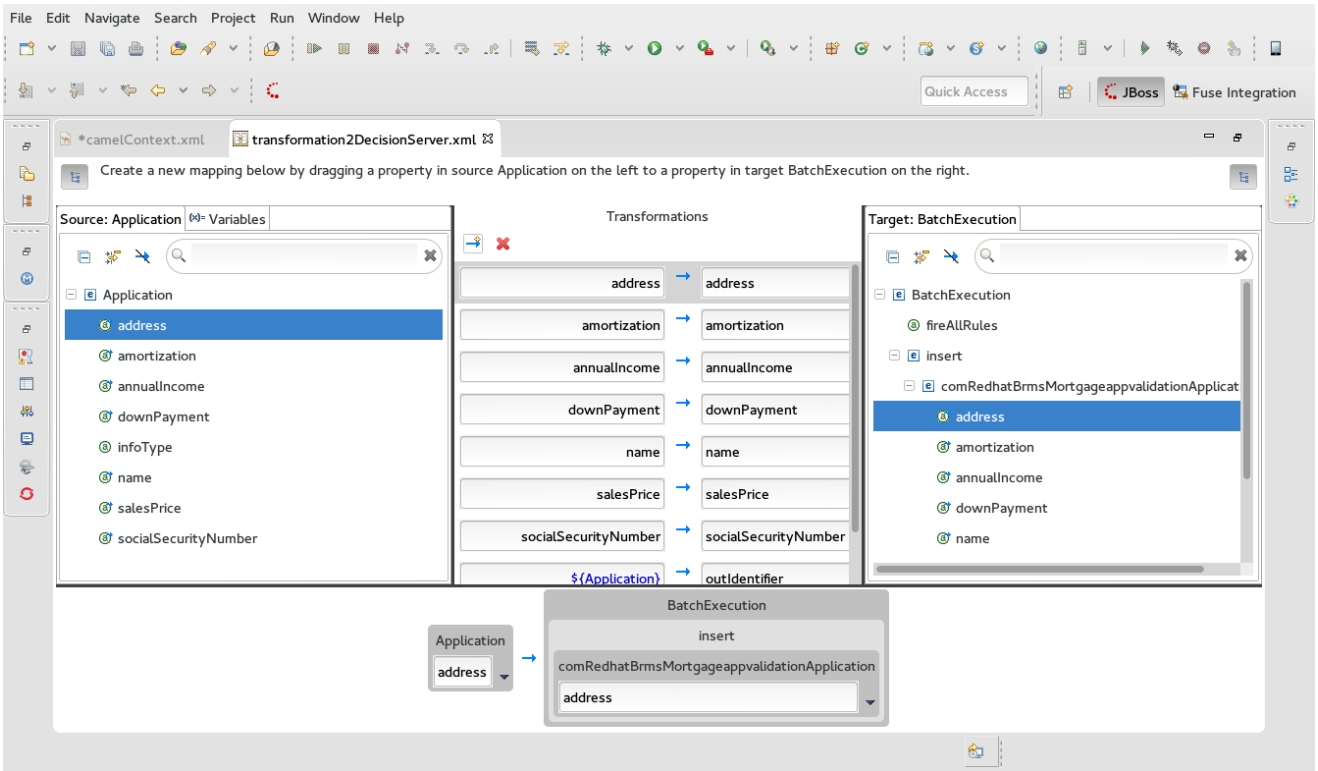


- Different types of XML documents are distinguished by the attribute named "infoType" of its root element. Here is a description of each route:
- Other Holdings Route - logs the holdings document and moves it to a target directory to be picked up by JDV
- Customer Validation Route - logs the xml and put the document in the validateionQueue (for further processing)
- Home Loan Route - logs the xml, use another EIP called Wire Tap which takes a copy of the message, sends it in an email to designated receivers while the original copy is put in the mortgageQueue (or further processing)
- Otherwise Route - the document type is not recognised, logs the message and put it in an errorQueue

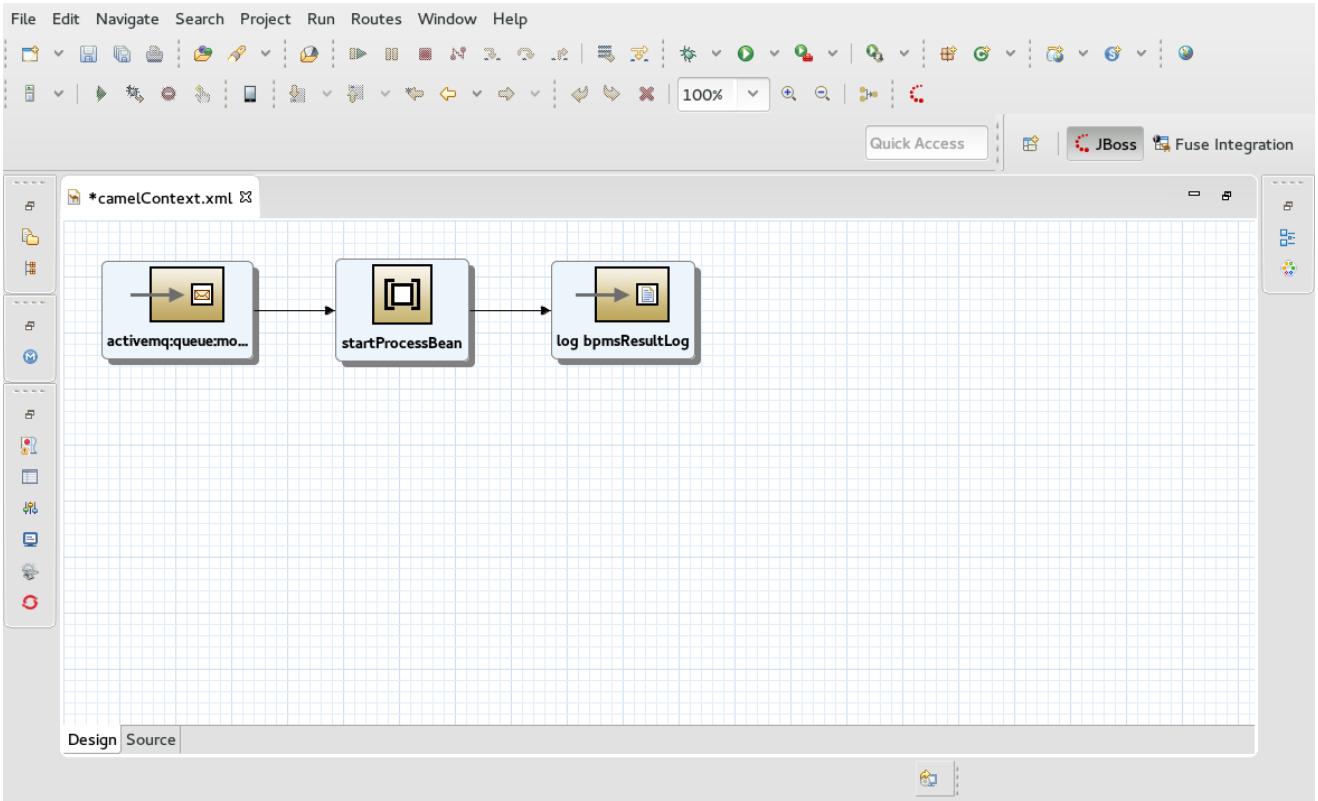
The Customer Validation Route continues as shown in the diagram below:



The other endpoint picks up the message in the validationQueue, passes the message to a data mapper which converts the xml to a format mandated by the Business-Rules-As-A-Service running on the BPMS Realtime Decision Server. The invocation of the Business-Rules-As-A-Service is performed by a custom bean (BRERemoteValidation task in the diagram). The validation result is logged. The graphical representation of the data mapper is shown below:



The Home Loan Route continues with the endpoint shown below:

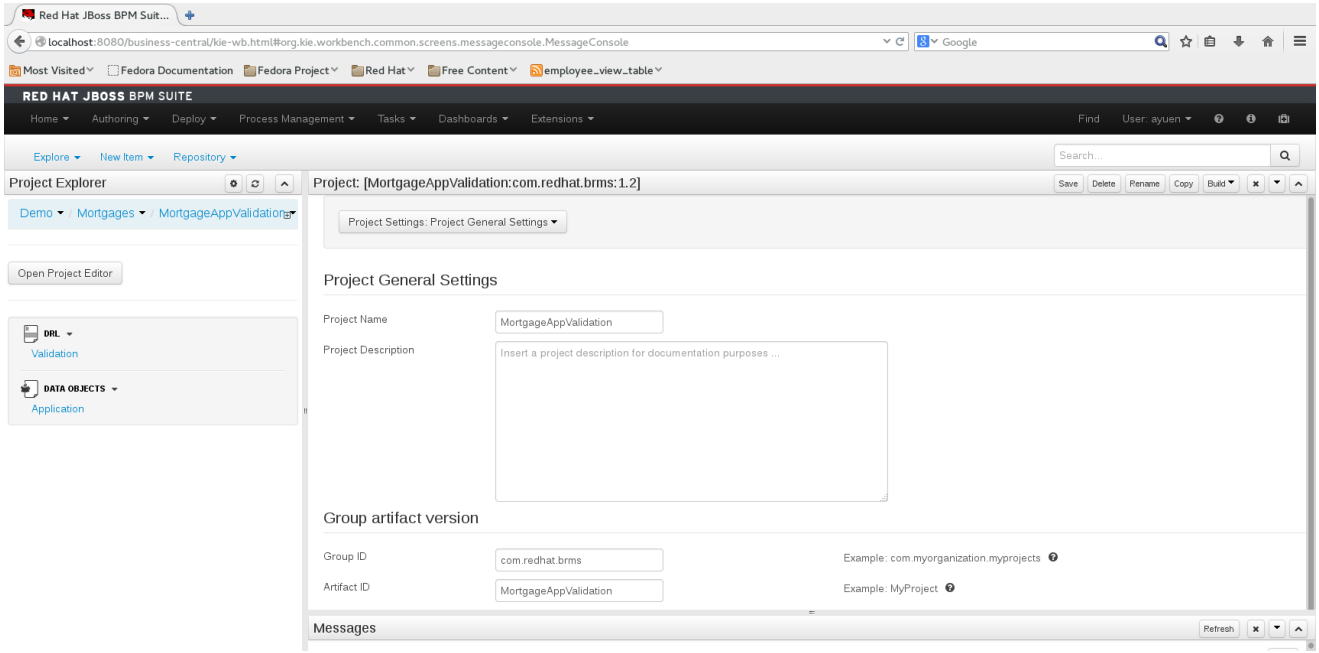


The endpoint picks up the message from the mortgageQueue and uses a custom bean which uses the BPMS Java Remote API to kick off an instance of the mortgage business process using the information contained in the xml.

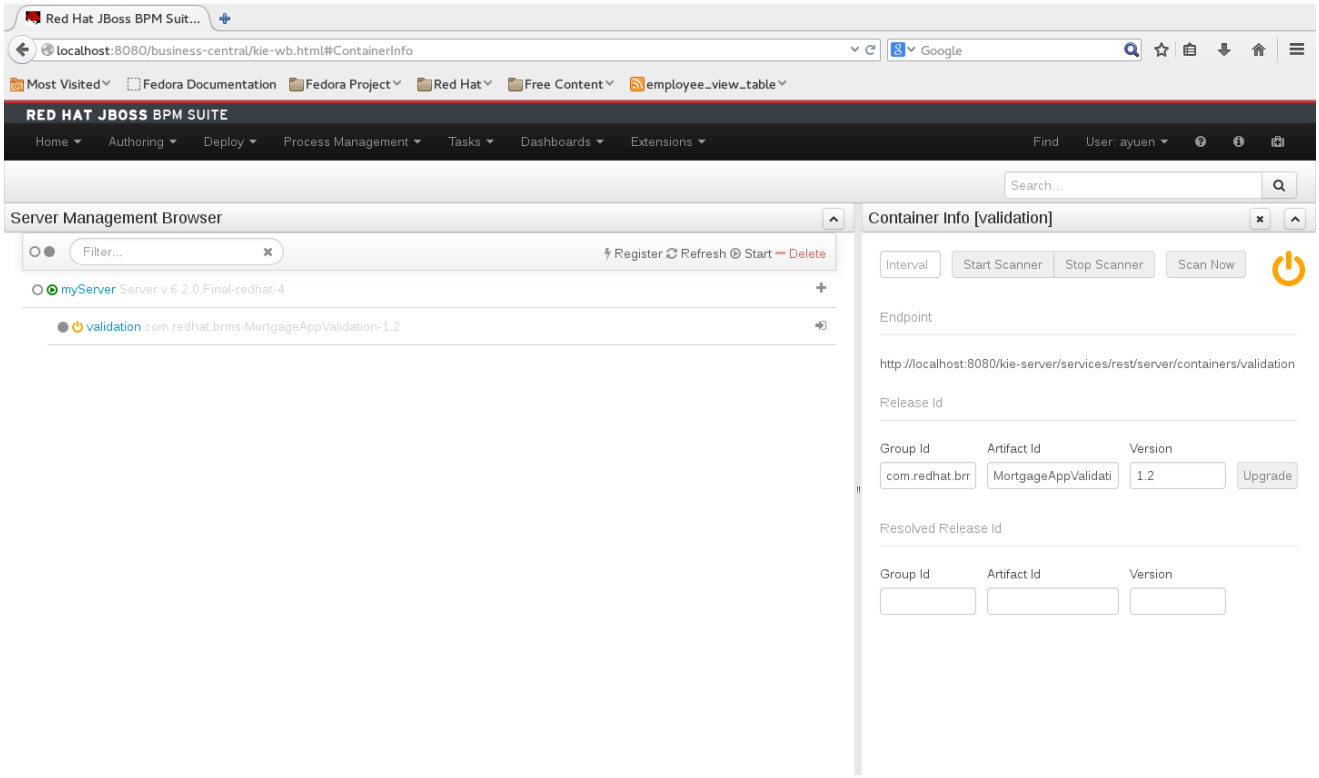
The following sections describe the other major components in the demo.

BRMS Realtime Decision Server Integration

For customer validation, a simple business rules application called MortgageAppValidation is used. It consists of only 1 DRL (Drools Rule Language file) and 1 Data Object. Both are created using Business Central.



And deployed in the Realtime Decision Server:



For those of you who want to migrate the application from the Realtime Decision Server to BPMS 6.3's Intelligent Process Server, please refer to my other article (link provided below) to avoid frustration:

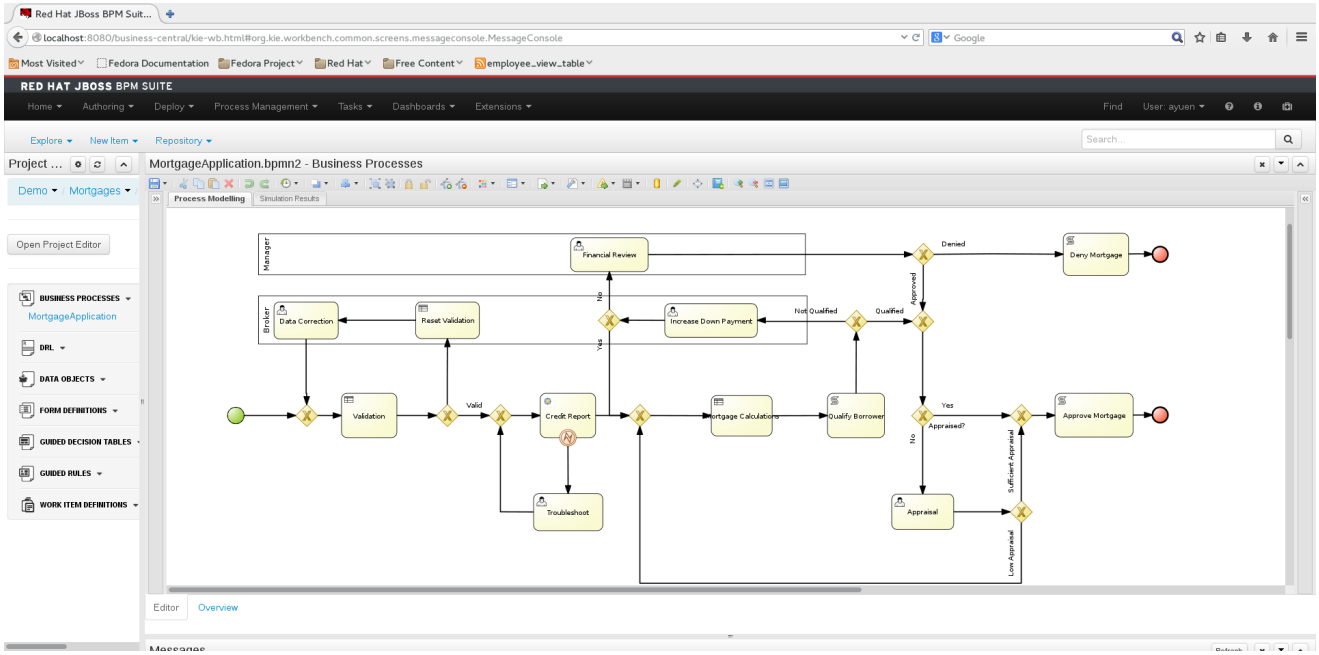
BPMS/BRMS 6.3: An Intelligent Process Server Odyssey

As mentioned earlier, the integration is done using a custom bean named RemoteBrmsValidation which uses Apache HTTP client to post the transformed (by the data mapper) XML to the Realtime Decision Server using HTTP Post.

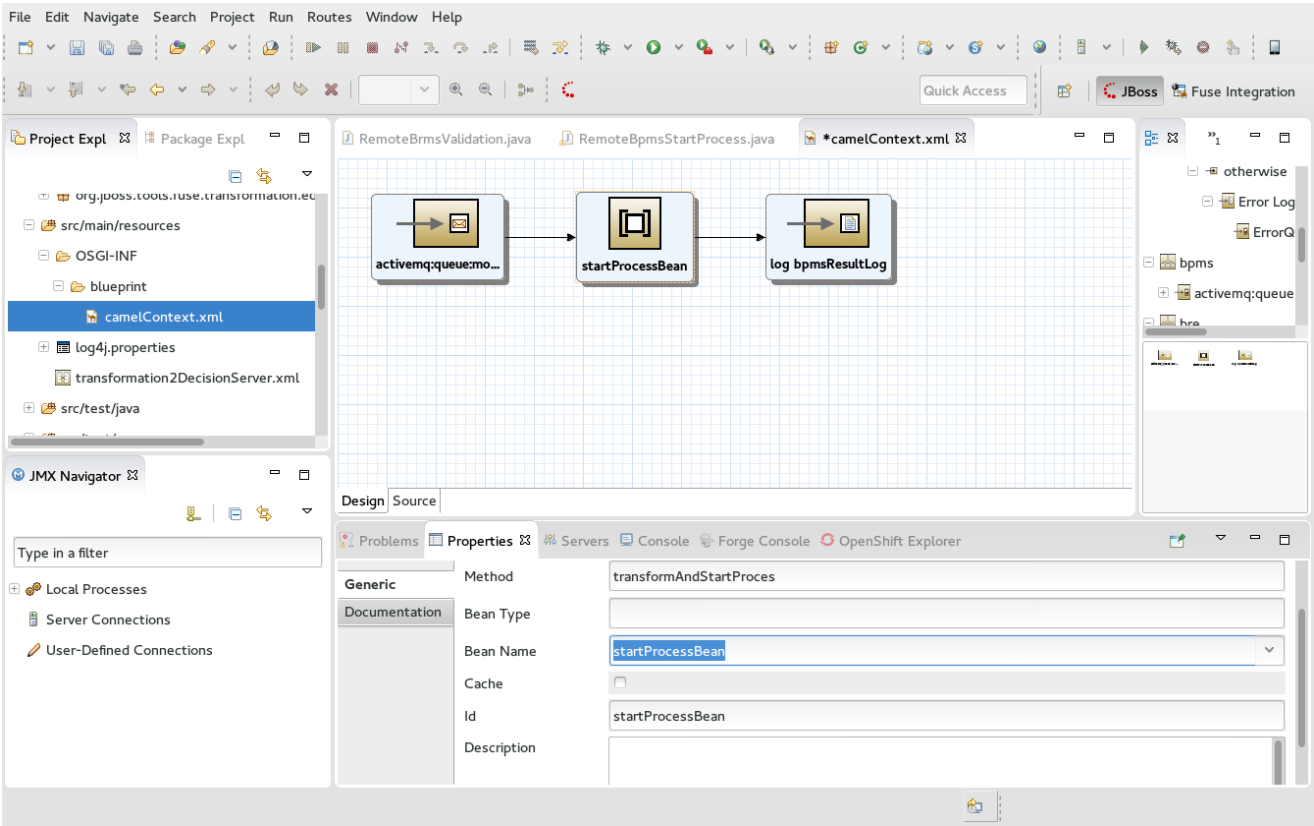
To learn how to build a rules application and deploy in the Realtime Decision Server, please refer to my blog: http://mrdreambot.ddns.net/building-a-jboss-bpms-rules-application-without-writing-code/#more-91

BPMS Business Process Integration

The mortgage business process is the stock version that comes with the jboss-bpm-example-dist-${version}.zip. The business process diagram is shown below:



Fuse uses a custom bean named startProcessBean to kick off an instance of the mortgage application business process utilising the BPMS Java Remote API.



You can reference my blog on using the BPMS Java Remote API here. http://mrdreambot.ddns.net/building-a-bpms-web-application-part-2-remote-java-api/

JBoss Data Virtualisation Integration

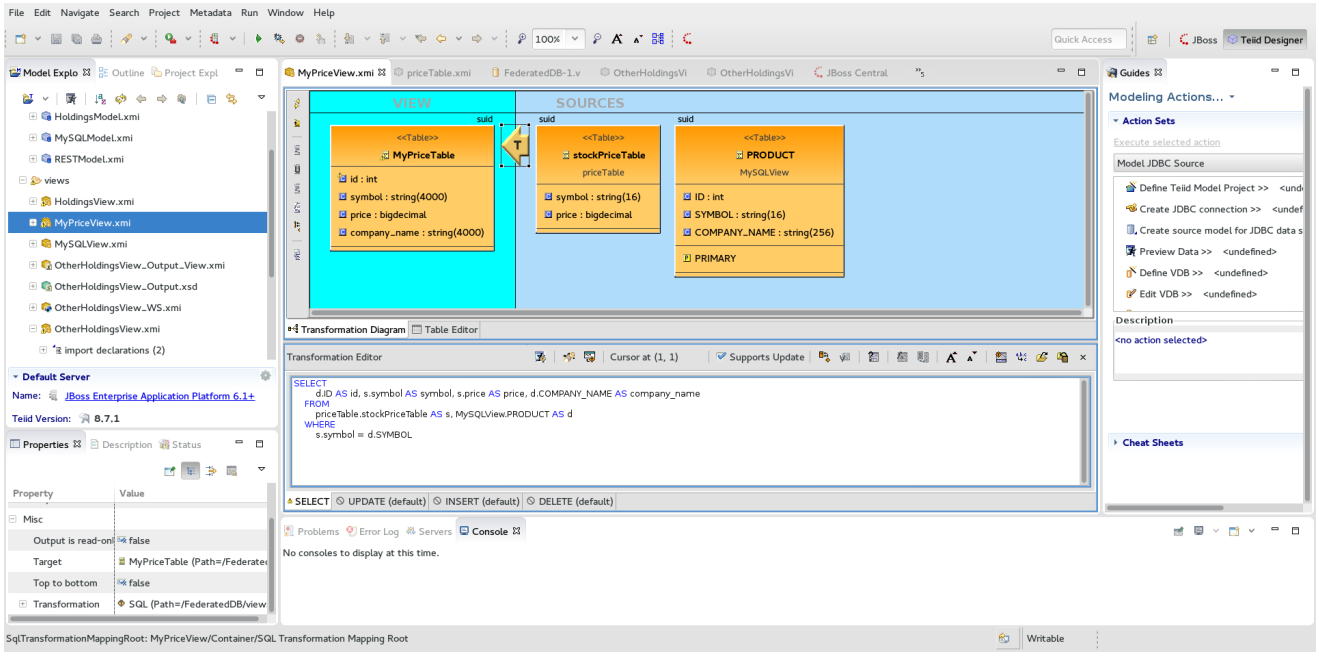
As described earlier, the virtual database is created using 3 data sources. They are:

1. Account Information including shares ownership (MySQL database)
2. Market Data (stock price retrieved using a REST Web Service)
3. Other Personal Holdings (a Holdings XML)

To grasp the concept on data virtualisation, please refer to my blog: http://mrdreambot.ddns.net/jboss-data-virtualization-part-1-concept/#more-269

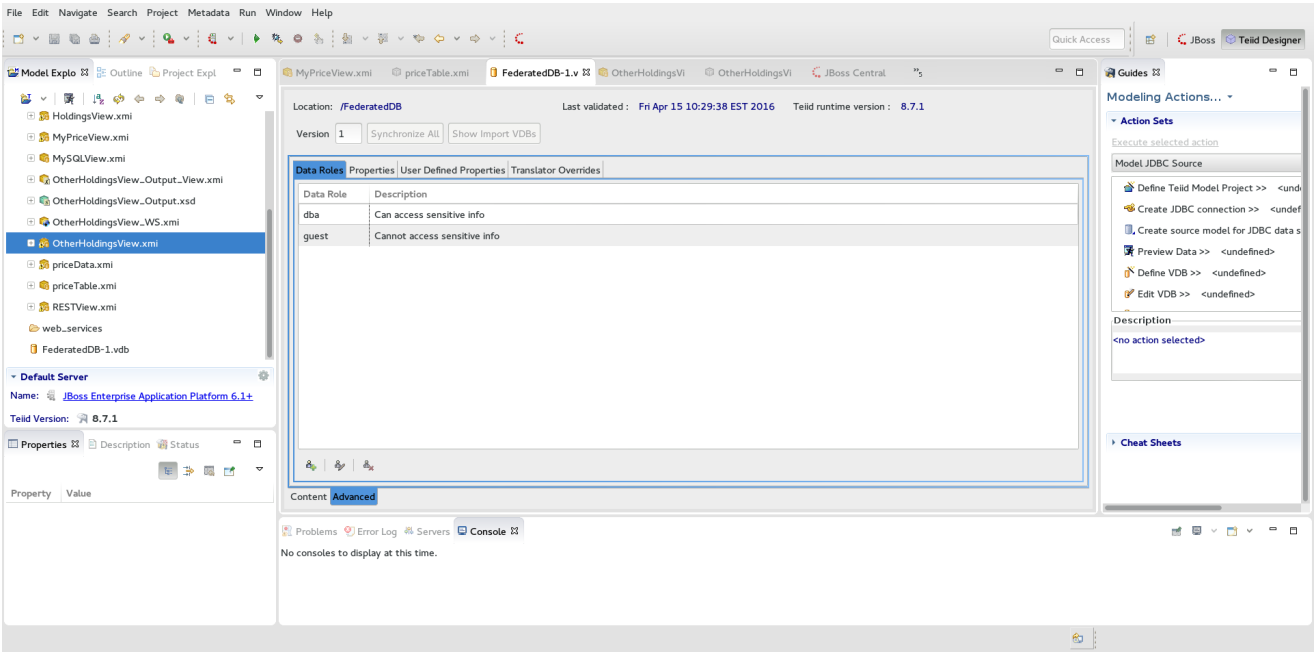
The MySQL database has been set up and populated with data and the web service has been deployed in EAP running JDV.
Applying the JDV best practice, for each data source, we create a source model as well as a view model. Additional virtual layers are created using the view models and not directly built on the source models. The

source model and view model for each data source are created using the import wizard and Teiid Meta Models wizard respectively.
The MyPriceTable is composed of data from 2 views: stockPriceTable from priceTable view (from a web service) and the PRODUCT from MySQLView.

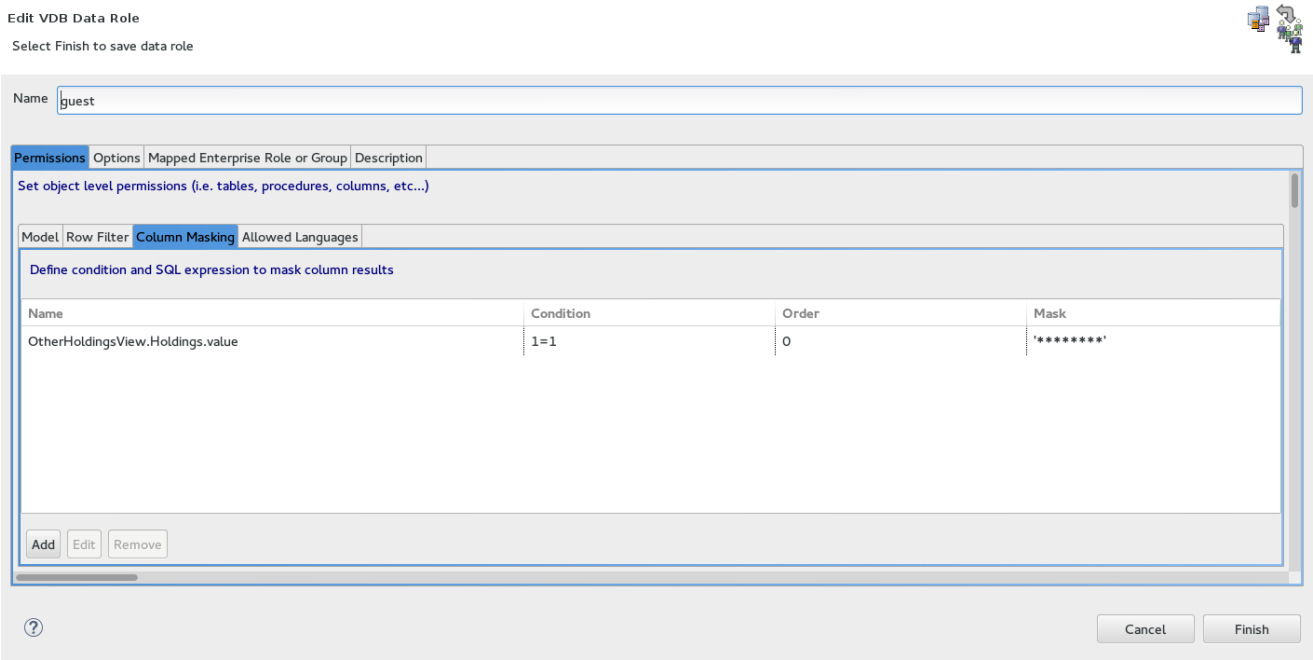


The virtual Holdings table in OtherHodingsView is created from the Holding table from the HoldingsView. There is no transformation. However, note that the Materialized property in the Properties tab has been set to true. Also notice that in the Transformation Editor, a cache hint has been added: /*+ cache(ttl:10000) */. It specifies that the cache (materialized view) has a time-to-live (ttl) duration of 10000 milliseconds or 10 seconds. This means that after 10 seconds if a query on this table takes place, it will trigger the reloading of the materialized view. Remember the OtherHoldings XML described earlier? This file is updated in the target directory by Fuse when a new document is sent by its Partner. Since the Holdings table will be refreshed every 10 seconds, this means the data in this table will be stale not more than t10 seconds (ttl value set).

Looking at the FederatedDB-1.vdb ie, the virtual database created for the demo, you will see that 2 roles have been defined: dba and guest. They are mapped to the login credentials.



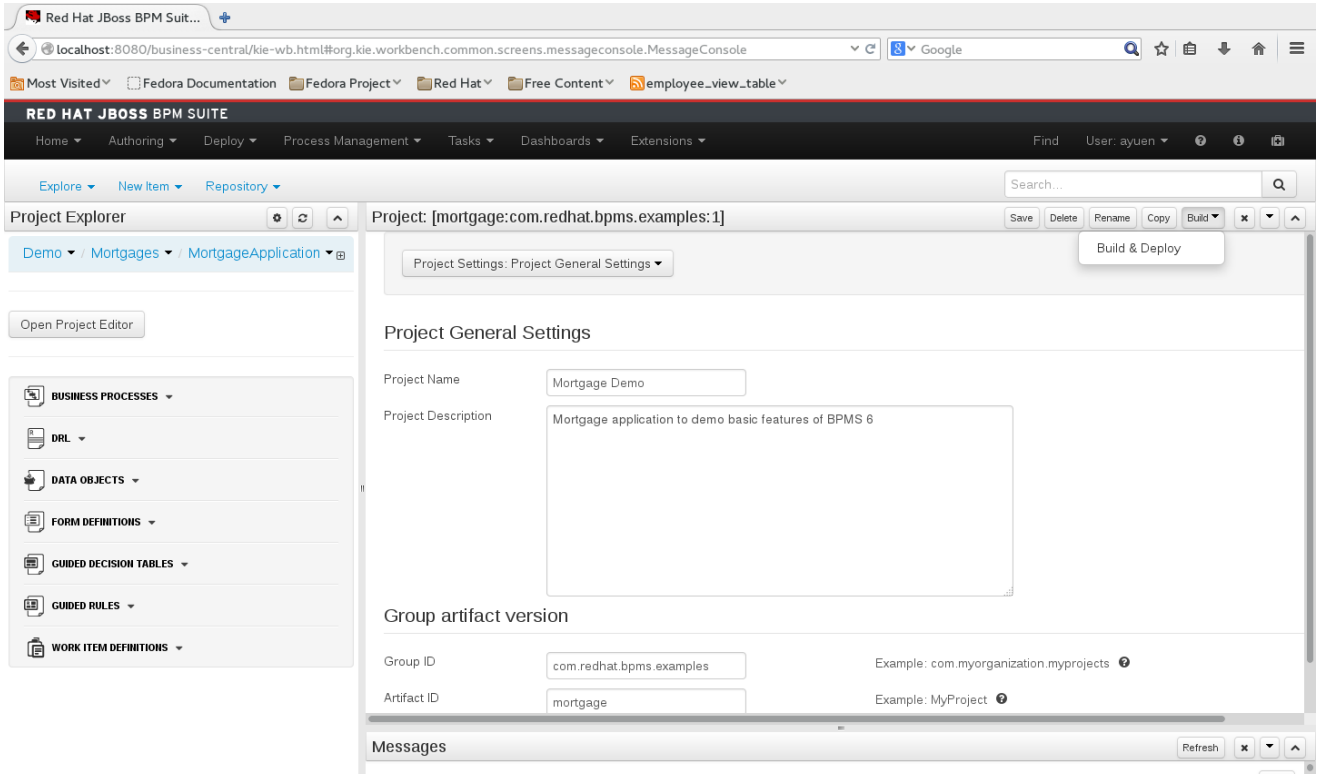
For the guest data role, Column Masking has been set for the column "value" in the otherHoldingdView.Holdings table. This means that when queried by users with quest role, this column will return 8 stars as the column's value. For dba data role, there is no such restriction.



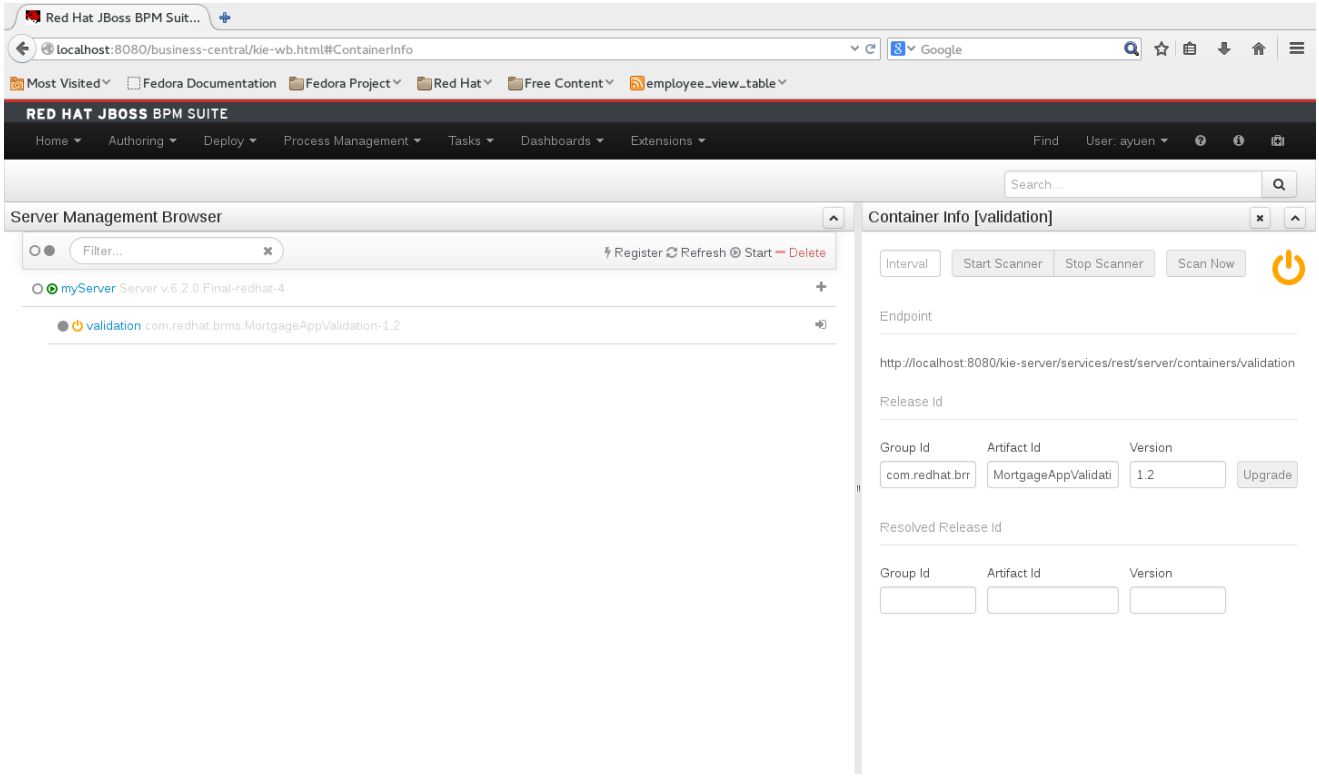
How to Demo

At the moment, the demo is set up on 2 virtual machines. One is used to run JBoss Developer Studio (JBDS), Fuse Camel routes and BPMS servers. The other is used to demonstrate JBDS together with JDV. There is no reason why they cannot be combined into one virtual machine. I just happened to develop different demos on different virtual machines and did not have time to consolidate them together.
First of all, explain the business scenario the demo is fulfilling. Make sure the customers know what to expect before the demo and make sure they understand what you are showing them during the demo.
VM1: JBDS, Fuse, BPMS
Start BPMS Server

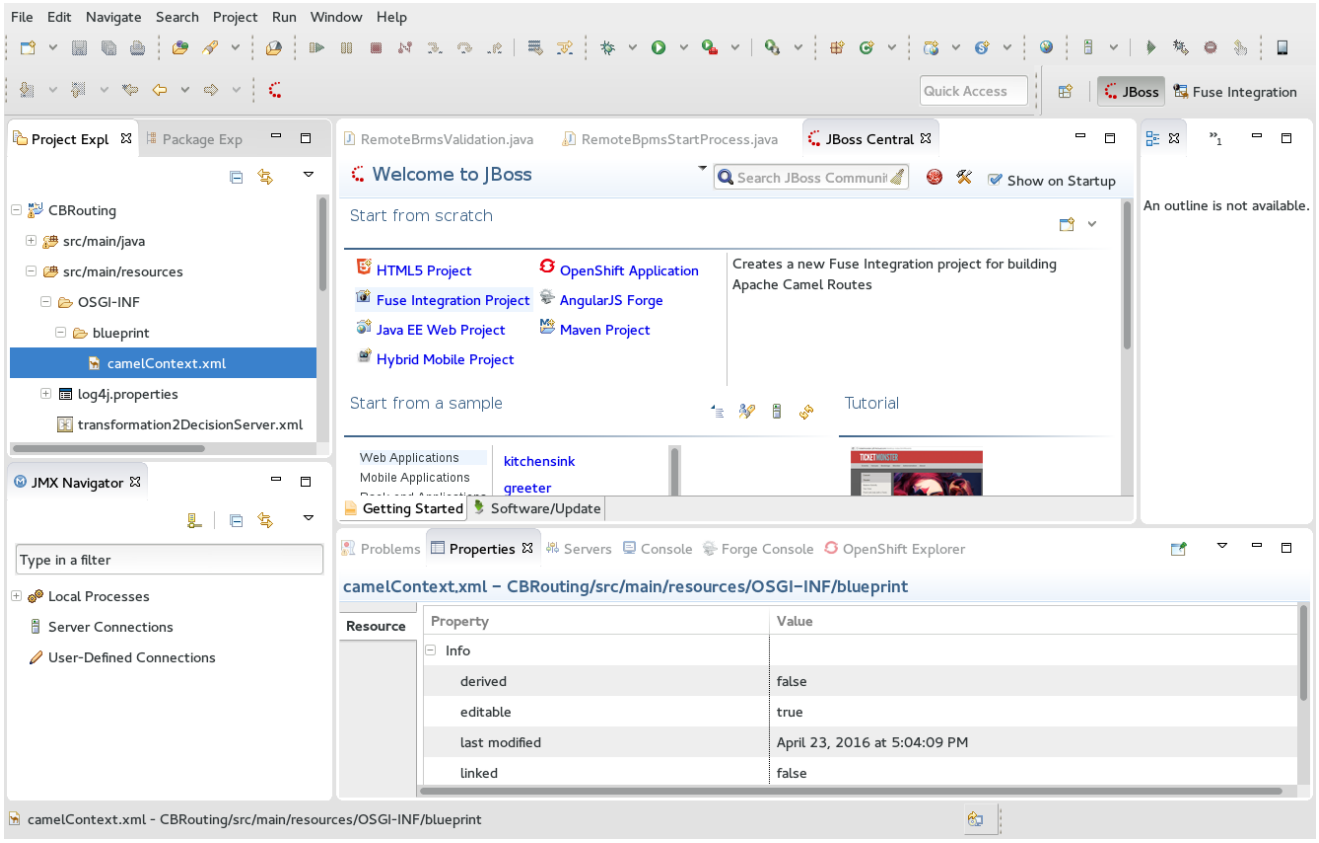
- Log in to Business Central. Build and deploy the Mortgage application. See screen shot.



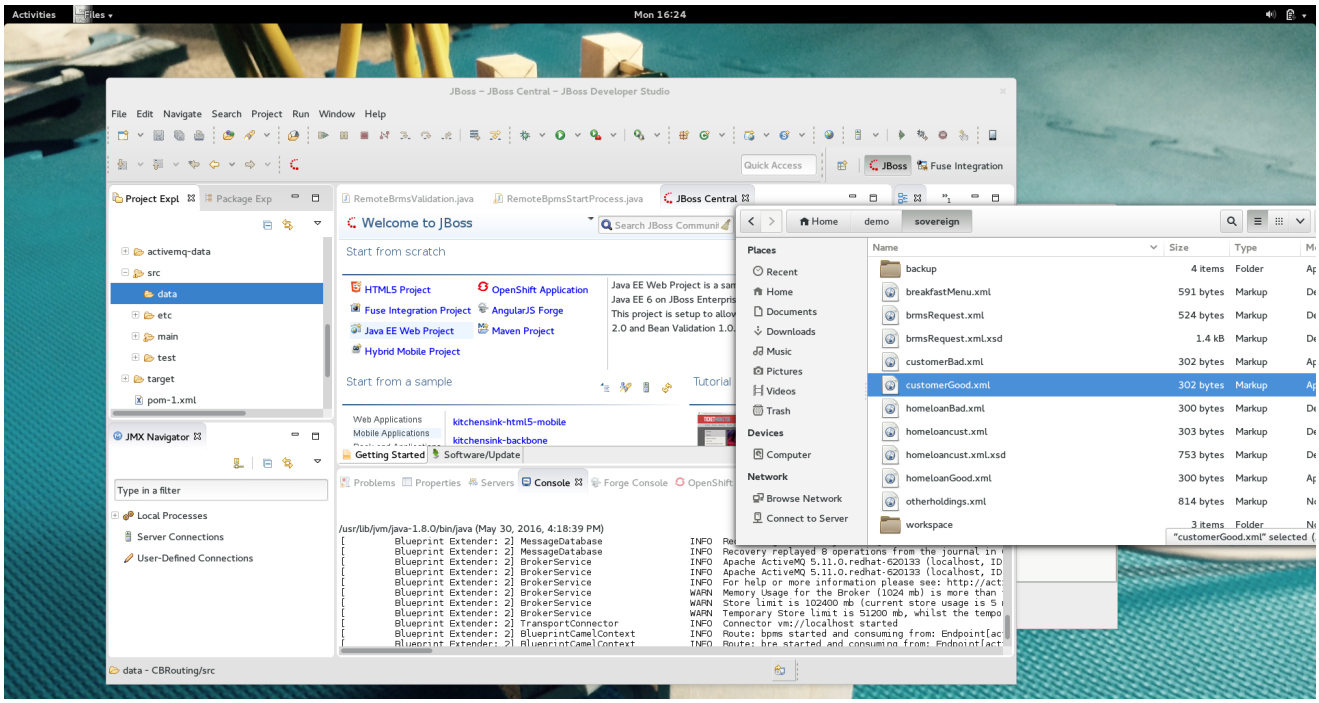
- Again from Business Central: start the Realtime Decision Server. See screen shot.

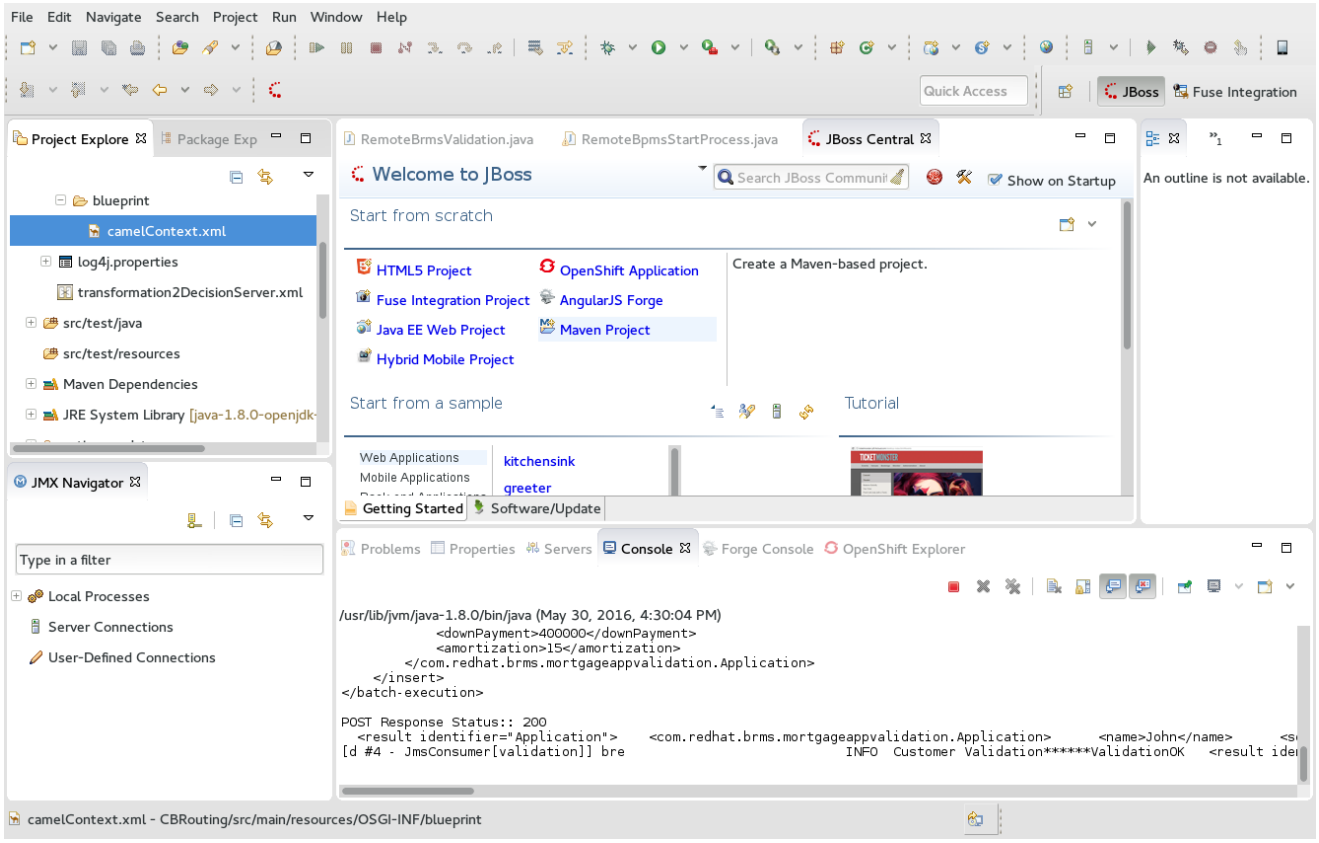


- Start JBDS and choose the demo workspace. Instead of deploying the Fuse Camel routes to the OSGi Karaf container, I prefer to demonstrate its execution using JBoss Developer Studio. Run the camelContext.xml as Local Camel Context (with tests). See screen shot below.

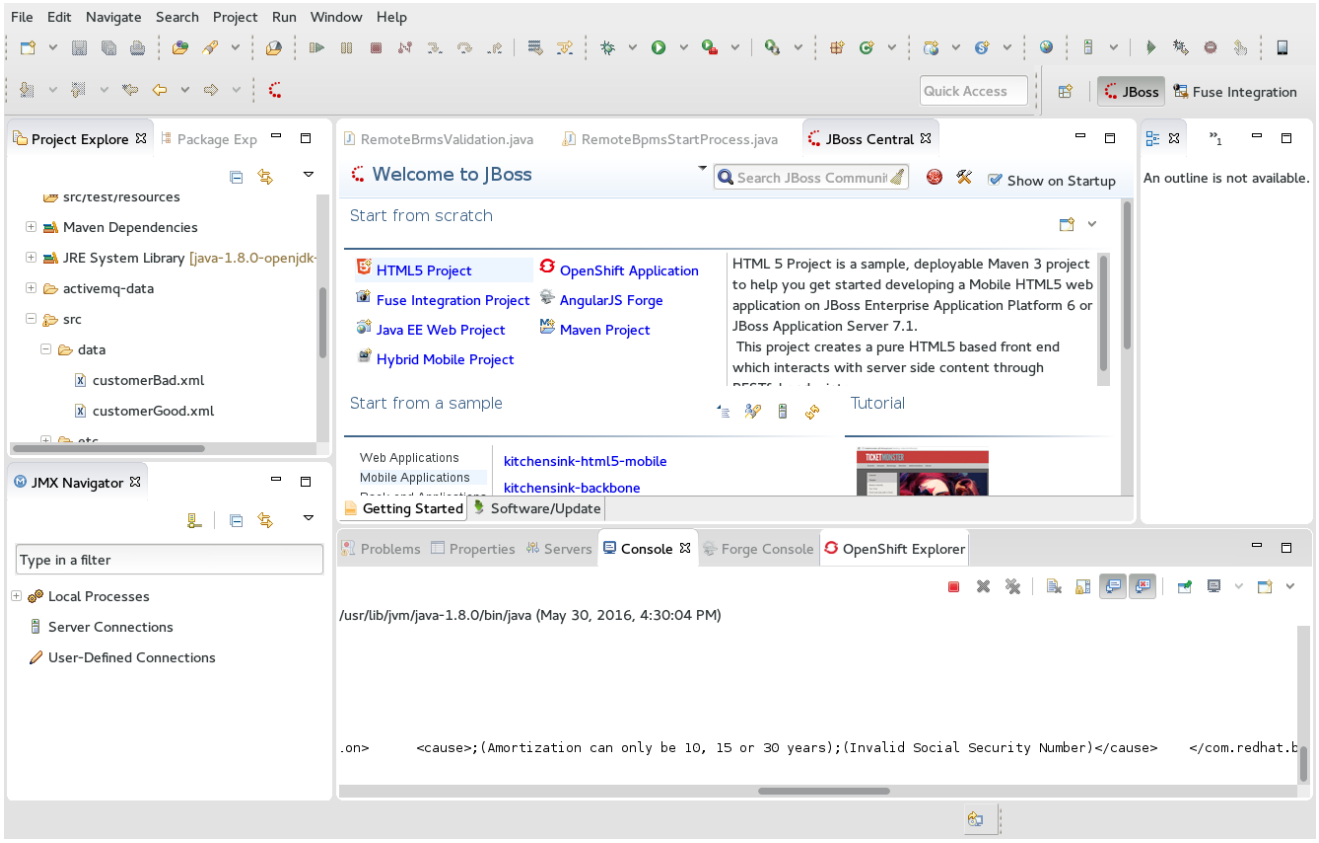


- Drag customerGood.xml from FileManager to JBDS' src/data directory and you will see the validationOK message.

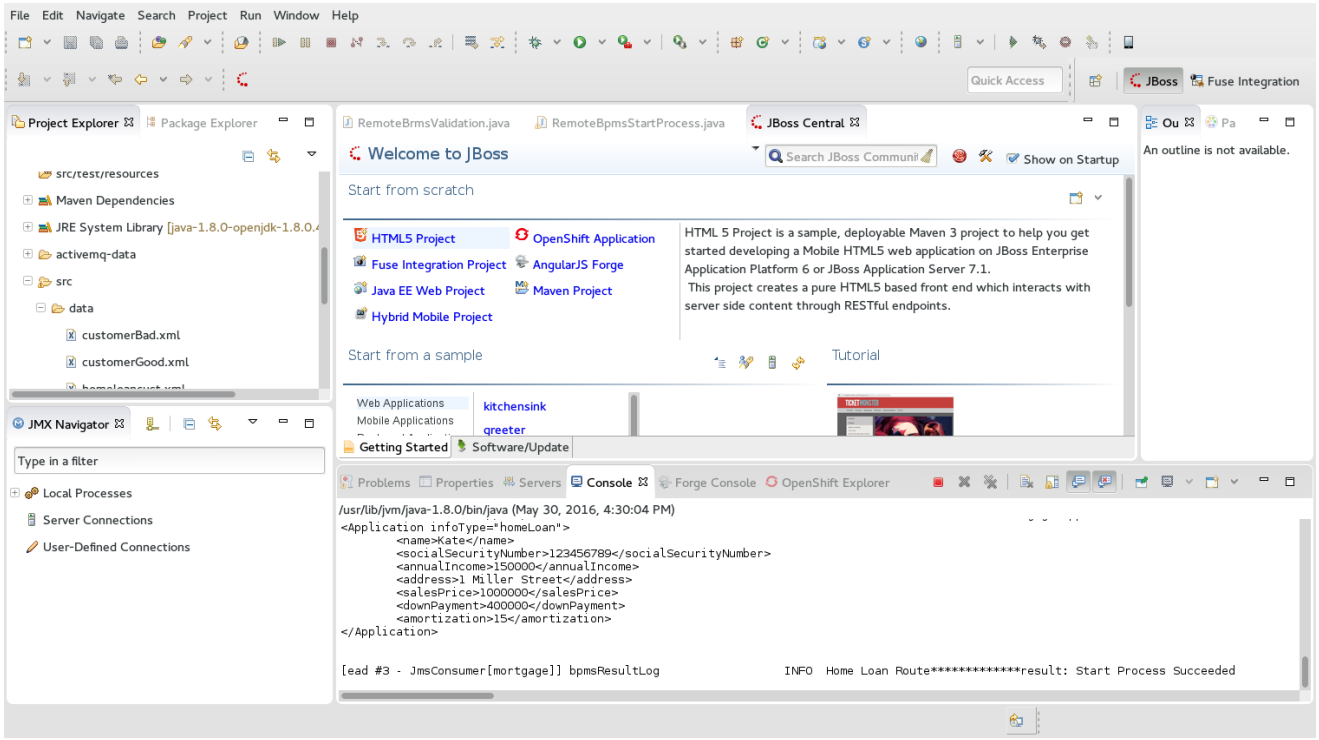


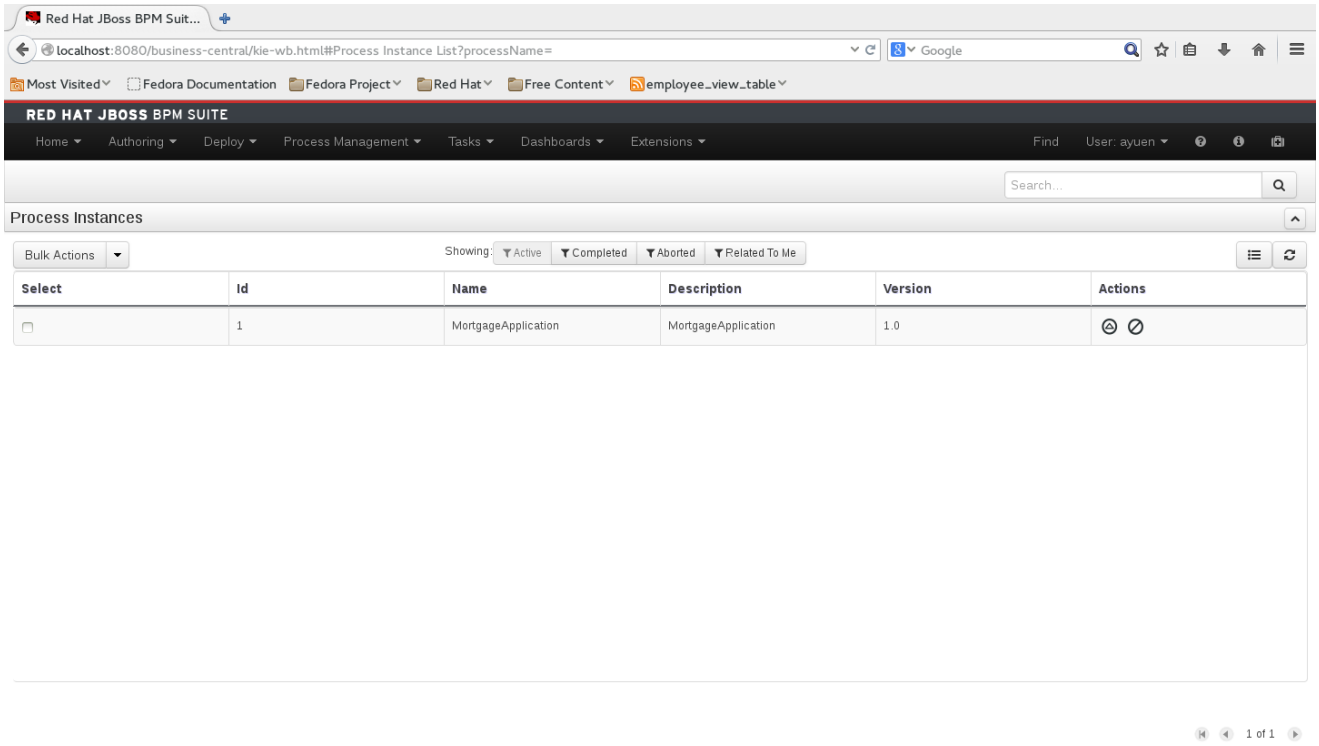


- Drag customerBad.xml from FileManager to JBDS' src/data directory and you will see the validationError message with the reason(s) why it failed validation.

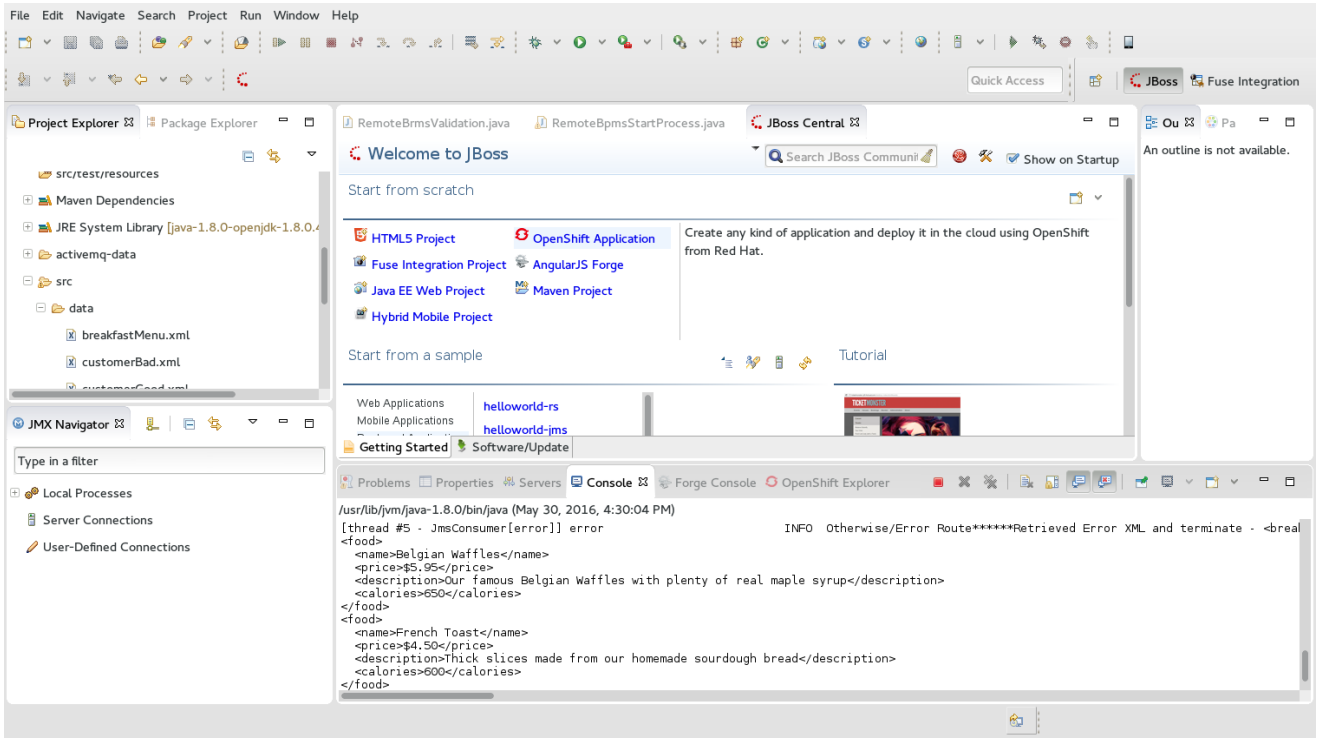


- Drag homeloancust.xml from FileManager to JBDS' src/data directory and you will see Start process OK message. You will also see in Business Central that there is a new Mortgage business process instance.





- Drag breakfastMenu.xml from FileManager to JBDS' src/data directory and you will see an error message as it format is not recognised.



You will also receive an email for every mortgage application received. Here is a sample. Note that the demo has been configured to send emails to me. You have to change it to send emails to you.

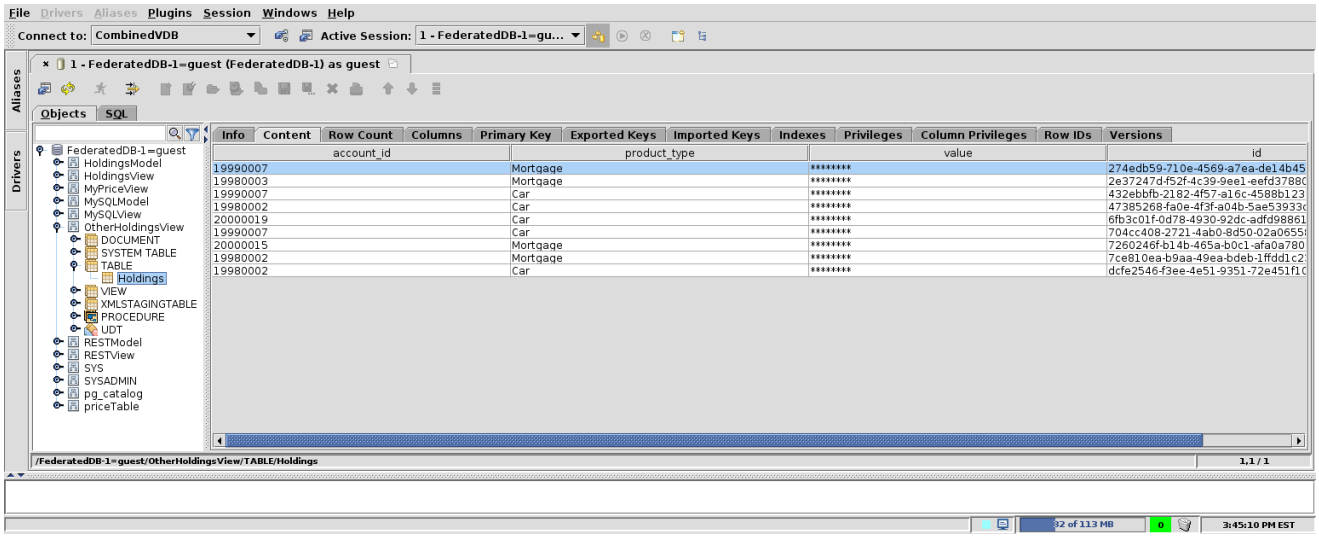
From: andyyuen105@gmail.com

Date: 30/05/2016 4:34 PM (GMT+10:00)
To: andyyuen105@gmail.com
Subject: RE: Mortgage Application Recevied
<Application infoType="homeLoan">
<name>Kate</name>
<socialSecurityNumber>123456789</socialSecurityNumber>
<annualIncome>150000</annualIncome>
<address>1 Miller Street</address>
<salesPrice>1000000</salesPrice>
<downPayment>400000</downPayment>
<amortization>15</amortization>
</Application>

VM2: JBDS, JDV

- Start JBDS with the correct workspace
- Start the server from JBDS
- Start Squirrel SQL client to connect to JDV FederatedVDB-1

- Use the Squirrel client to show the virtual tables and the column masking that was set up for the otherHoldingdView.Holdings.value table (see diagram below)



TODO

- Combine the 2 virtual machine into 1 and reduce its size
- Make it available for download so that others can use it for demo purposes