JBoss Data Virtualization Workshop

Cojan van Ballegooijen

Revision 1.0
May 14, 2014

Table of Contents

List of Figures

Chapter 1. Introduction

Red Hat JBoss Data Virtualization, formerly known as Red Hat JBoss Enterprise Data Services Platform (EDS), is a complete data provisioning, federation, integration and management solution that enables organizations to gain actionable and unified information. Red Hat JBoss Data Virtualization enables agile data utilization in three easy steps:

1. Connect: Access data from multiple, heterogeneous data sources.
2. Compose: Easily create reusable, business-friendly logical data models and views by combining and transforming data.
3. Consume: Make unified data easily consumable through open standard interfaces.

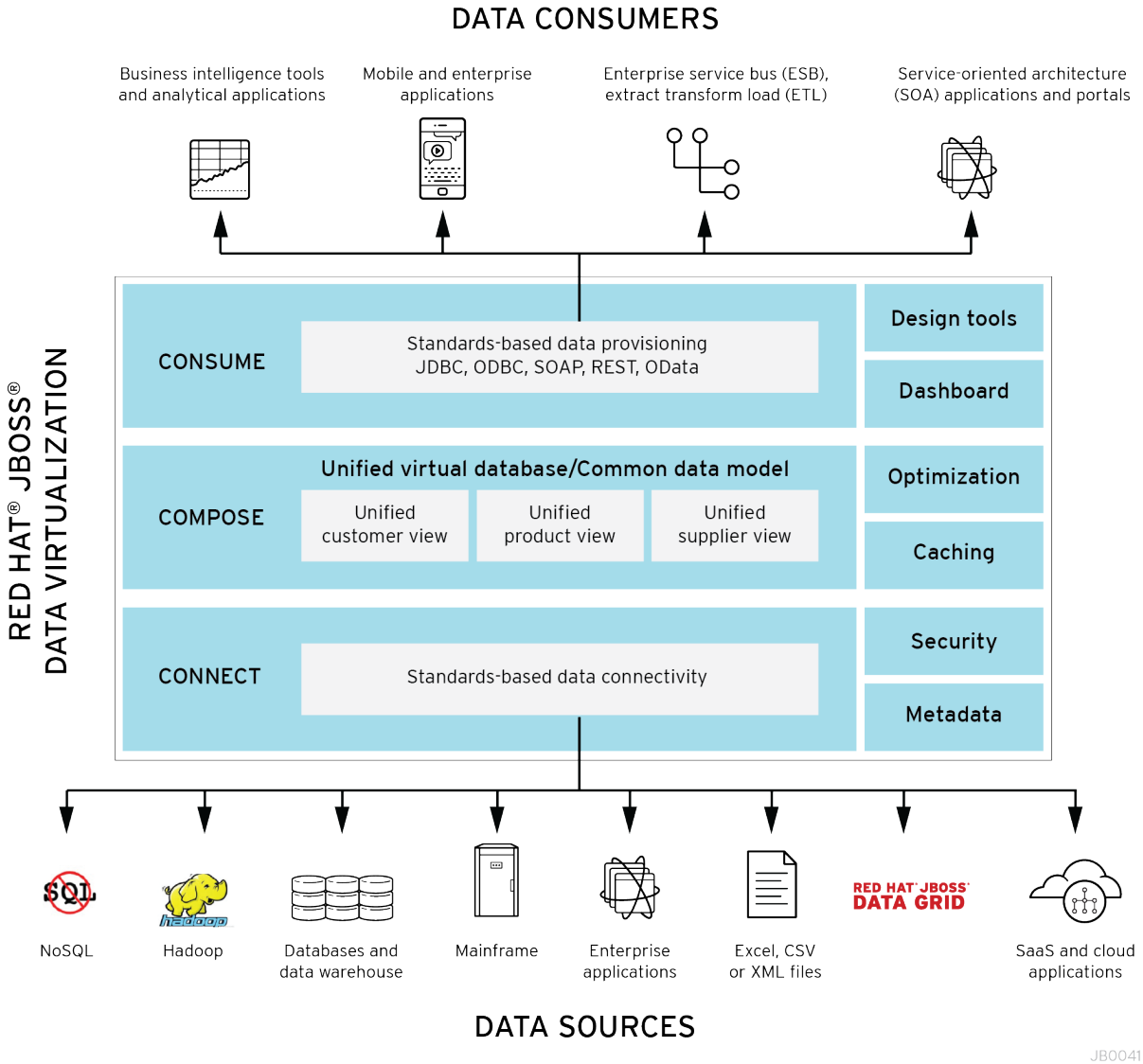

Figure 1.1.

Red Hat JBoss Data Virtualization includes:

- Tools for creating data views that are accessible through standard protocols (the Teiid Designer plug-in for Red Hat JBoss Developer Studio (JBDS)).
- A robust runtime environment that provides enterprise-class performance, data integrity, and security (the Red Hat JBoss Data Virtualization Server, which executes as a process within the Red Hat JBoss Enterprise Application Platform (EAP)).
- A repository for storing metadata (ModeShape)

Red Hat JBoss Data Virtualization is based on the following community projects:

- Teiid (http://www.jboss.org/teiid)
- Teiid Designer (http://www.jboss.org/teiiddesigner)
- Modeshape (http://www.jboss.org/modeshape)

The figure below depicts the architectural overview of Red Hat JBoss Data Virtualization:

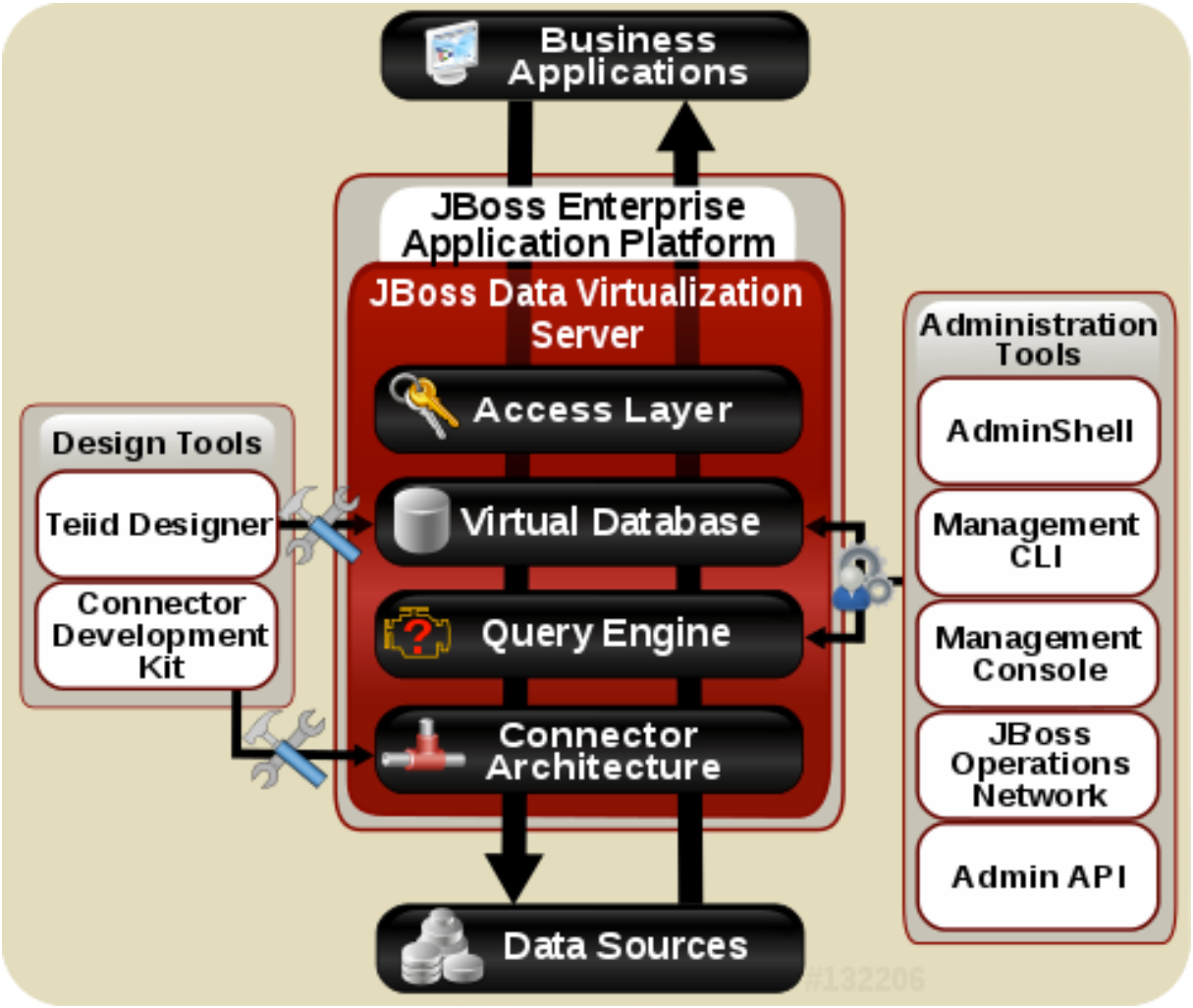

Figure 1.2.

Component	Description
Query Engine	The heart of Red Hat JBoss Data Virtualization Server is a high-performance query engine that processes relational, XML, XQuery and procedural queries from federated datasources. Features include support for homogeneous schemas, heterogeneous schemas, transactions, and user defined
Embedded	An easy-to-use JDBC Driver that can embed the Query Engine in any Java application.
Server	An enterprise ready, scalable, manageable, runtime for the Query Engine that runs inside JBoss EAP that provides additional security, fault-tolerance, and administrative features.

Component	Description
Connectors	Red Hat JBoss Data Virtualization Server includes a rich set of Translators and Resource Adapters that enable access to a variety of sources, including most relational databases, web services, text files, and ldap. Need data from a different source? A custom translators and resource adaptors can easily be developed.
Tools	Red Hat JBoss Data Virtualization Server includes development and administration tools • Create - Use Teiid Designer to define virtual databases containing views, procedures or even dynamic XML documents. • Monitor & Manage - Use the Management Console with JBoss EAP or use the JBoss Data Virtualization JBoss Operations Network (JON) plugin to control any number of servers. • Script - Use the AdminShell to automate administrative and testing tasks.

1.1. What is expected of you

Please feel free to raise your hands with any questions that you have about the lab; feel free to ask why it is you are doing something, or if something does not feel right. Please know that all care was made in creating this user guide, but all screen shots and steps along the way might be off by just a little so please be patient with any issues.

1.2. Prerequisites

- OpenJDK 1.6 or 1.7 or Oracle JDK 1.6, 1.7 installed
- Git installed
- PostgreSQL server and/or MySQL server running
- PostgreSQL and/or MySQL client tools installed if using a remote database

1.3. Setup demo environment

In order to use the labs we need to prepare the database. We provide database SQL scripts for PostgreSQL and MySQL to load initial data into the database. The next

paragraph will describe how to install the database and load the demo data into the database.

1.3.1. Setup PostgreSQL

Installation The easiest way to install PostgreSQL is to use the pre-built binary packages which are available for a number of operating systems. See http://www.postgresql.org/download/ for more information and downloads. Post-install steps:

1. If *nix or MacOSX switch to user postgres or other OS user who is able to use psql command to connect to the PostgreSQL database.

```
$ su postgres
```

1. Go to the DVWorkshop/dv-docker/demo directory and run the following command:

```
$ psql -a -f financials-psql.sql
```

1. If Step 2 is successfully executed the the PostgreSQL environment contains the following databases. Hint: start the psql command line utility and type the “\l” to list the databases in PostgreSQL database.

```
$ psql
psql (9.3.4, server 9.0.13)
Type "help" for help.

postgres=# \l
                                  List of databases
     Name      |  Owner   | Encoding  | Collate | Ctype |   Access
 privileges
---------------+----------+-----------+---------+-------
+-----------------------
 apaccustomers | postgres | SQL_ASCII | C       | C     |
 brokerinfo    | postgres | SQL_ASCII | C       | C     |
 eucustomers   | postgres | SQL_ASCII | C       | C     |
 postgres      | postgres | SQL_ASCII | C       | C     |
 products      | postgres | SQL_ASCII | C       | C     |
 rhq           | rhqadmin | SQL_ASCII | C       | C     |
 template0     | postgres | SQL_ASCII | C       | C     | =c/postgres
     +
               |          |           |         |       | postgres=CTc/
postgres
```

```
 template1     | postgres | SQL_ASCII | C       | C     | =c/postgres
     +
               |          |           |         |       | postgres=CTc/
postgres
 uscustomers   | postgres | SQL_ASCII | C       | C     |
(9 rows)

postgres=# \q
```

1.3.2. Setup MySQL

Installation The easiest way to install MySQL is to use the pre-built binary packages which are available for a number of operating systems. See http://dev.mysql.com/downloads/mysql/ for more information and downloads. Post-install steps

1. If *nix or MacOSX go to the /usr/local/mysql directory and start mysqld_safe

```
$ cd /usr/local/mysql
$ sudo ./bin/mysqld_safe
```

1. Go to the DVWorkshop/dv-docker/demo directory and tun the following command as depicted in the picture below.

```
$ sudo /usr/local/mysql/bin/mysql < financials-mysql.sql
```

1. If Step 2 is successfully executed the the MySQL environment contains the following databases.

```
sudo /usr/local/mysql/bin/mysql
Password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.6.17 MySQL Community Server (GPL)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights
 reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input
 statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| apaccustomers      |
| brokerinfo         |
| eucustomers        |
| mysql              |
| performance_schema |
| products           |
| test               |
| uscustomers        |
+--------------------+
9 rows in set (0.02 sec)

mysql> exit
Bye
```

The labs will use the following databases:

- apaccustomer
- brokerinfo
- eucustomers
- products
- uscustomers

Congratulations, you have now completed this lab.

Chapter 2. Install Red Hat JBoss Data Virtualization

There are three different ways to install Red Hat JBoss Data Virtualization.

1. Graphical mode: Graphical mode launches a graphical wizard which provides step-by-step instructions for installing and configuring the Red Hat JBoss Data Virtualization. Additional setup, including the Quickstarts and Maven Repository, is also possible with the installer.
2. Text mode: You can launch the installer in the text mode as well. Text mode provides step-by-step instructions for installing and configuring the Red Hat JBoss Data Virtualization.
3. Automated script mode: You can install multiple identical instances of Red Hat JBoss Data Virtualization using the automated script. This automated script is generated after the first installation instance. We will explain the graphical mode and automated script mode in more detail in the following paragraphs.

2.1. Installing Red Hat JBoss Data Virtualization through graphical mode

Download the Red Hat JBoss Data Virtualization installer binary by clicking the green download button at http://www.jboss.org/products/datavirt.html.

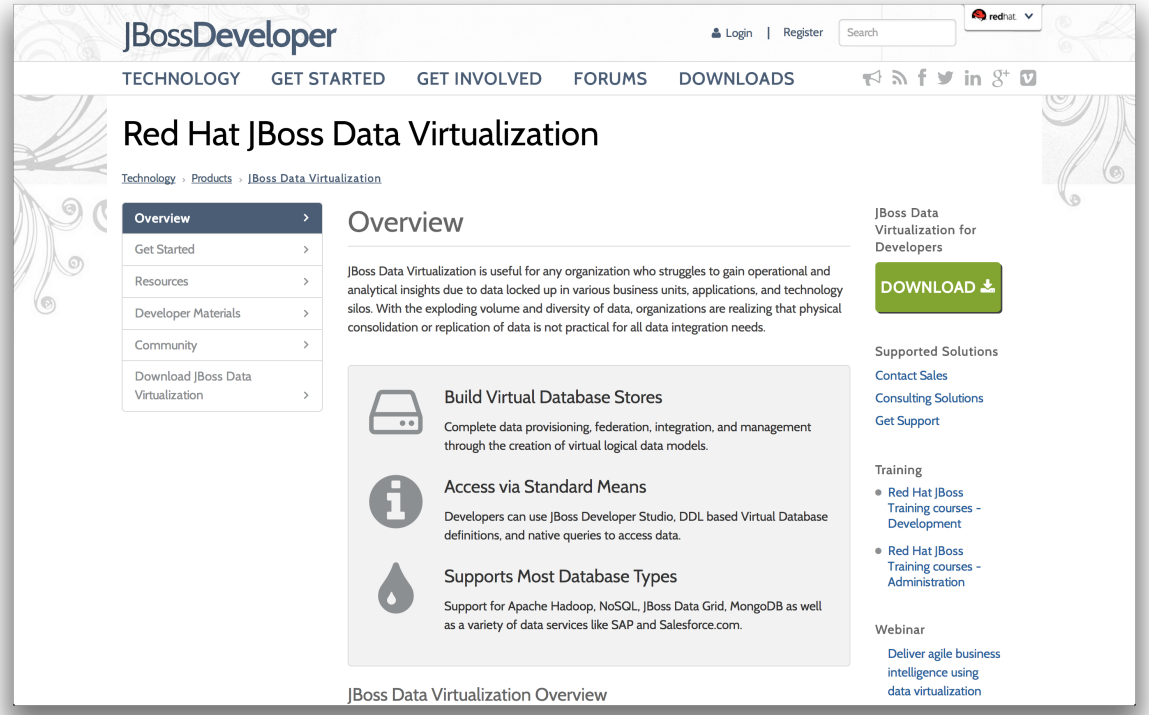

Figure 2.1.

Open a terminal window and navigate to the location where the GUI installer was downloaded. Run the installer using java at the command prompt: java -jar jboss-dv-installer-{version}.jar

The current available version is 6.0.0.GA-redhat-4 and to run the installer at the command prompt see below:

```
$ java -jar jboss-dv-installer-6.0.0.GA-redhat-4.jar
```

Follow the installer prompts to complete the installation process. See the Getting Started with Red Hat JBoss Data Virtualization Installation and Configuration video. See http://vimeo.com/76457404 for more details. A dialogue box will open followed by the End User License Agreement. If you accept the terms of the agreement, select I accept the terms of this license agreement and then click Next.

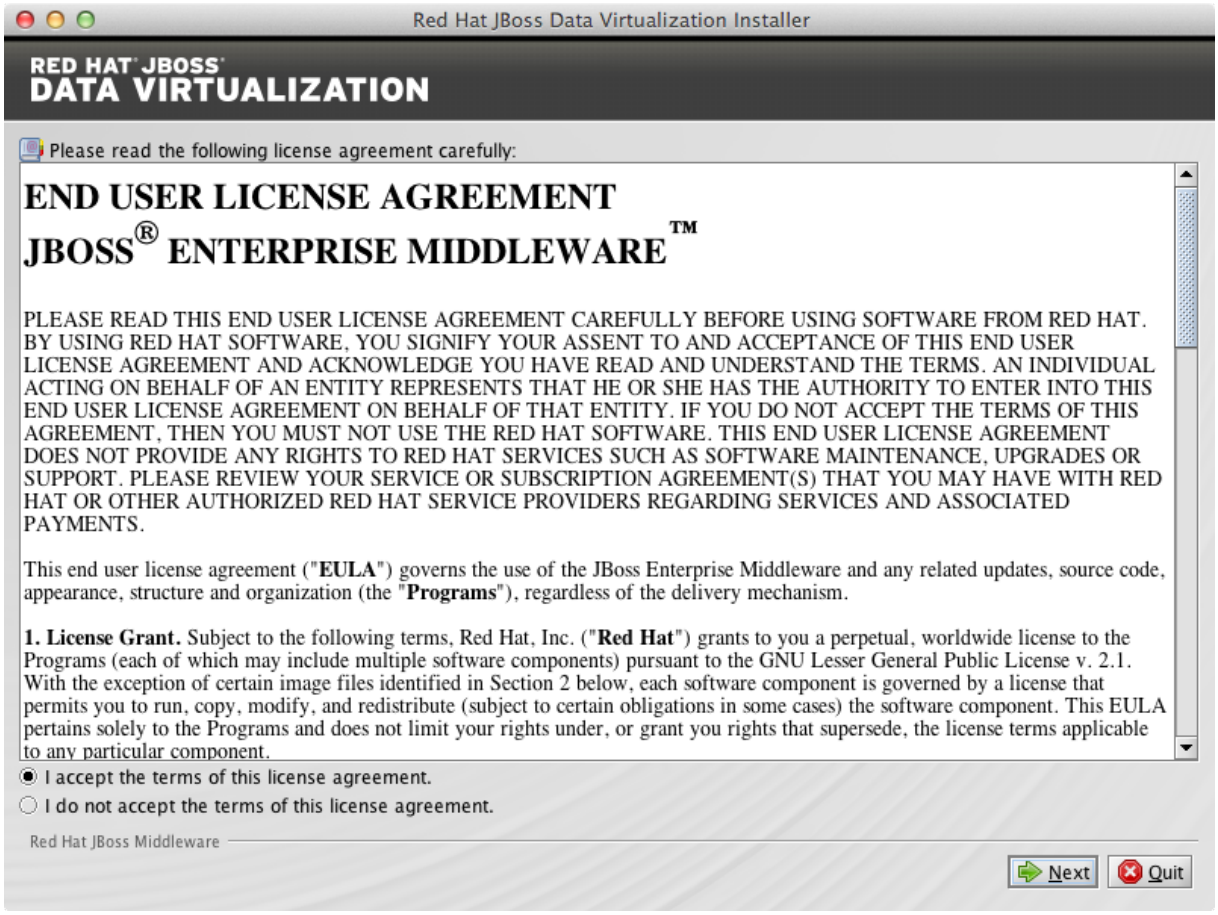

Figure 2.2.

A filepath confirmation dialogue box will appear. In the Select the installation path field, type the path where you want JBoss Data Virtualization to be installed or click Browse to navigate to the desired location. When the Select the installation path field shows the correct path, click Next.

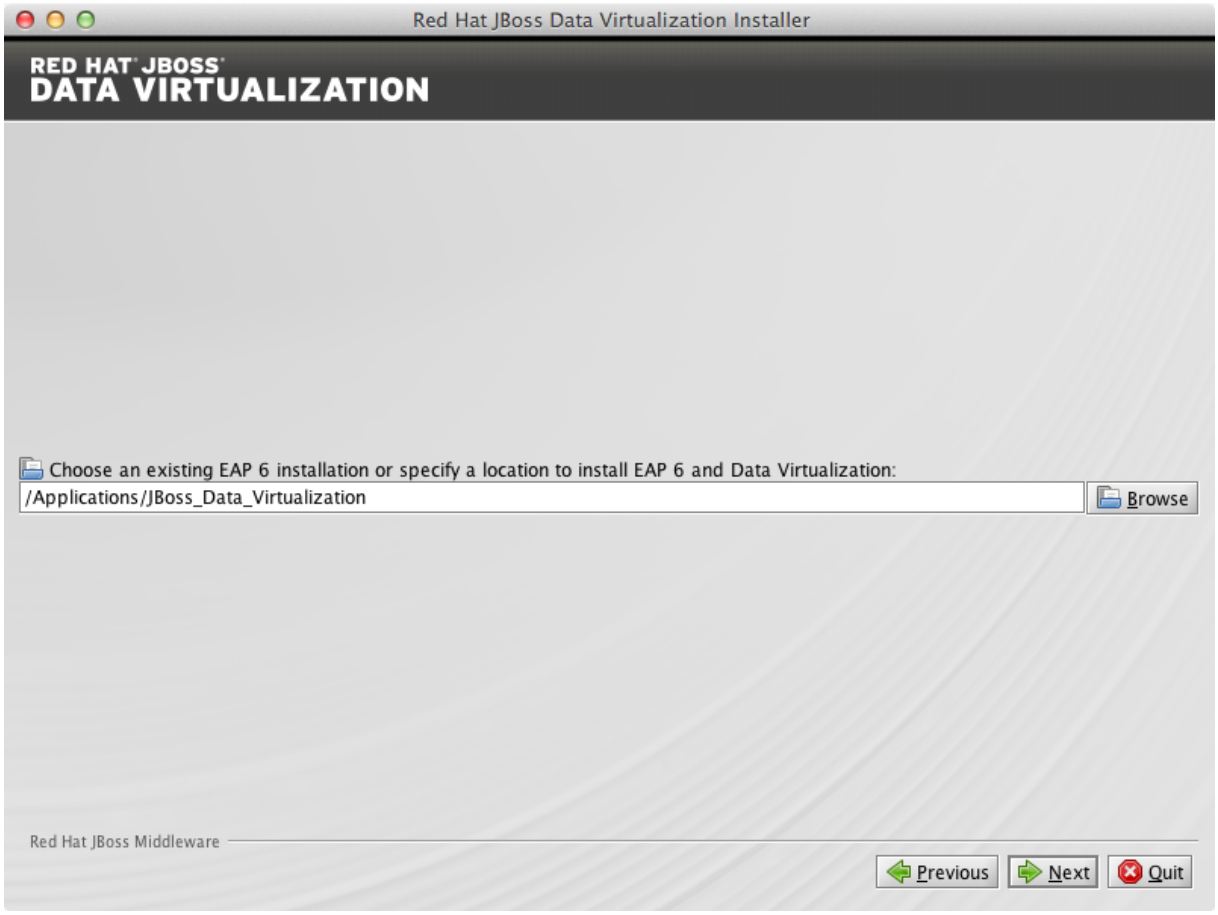

Figure 2.3.

When you are prompted about the specified location being created or overwritten, review the message and, if satisfied, click OK and then press Next.

You will be prompted to create a new admin username and password. Once created, it will be added to the ManagementRealm and can be used to access the Management Console and other applications secured using the ManagementRealm. Enter the new username and password in the appropriate fields and click Next.

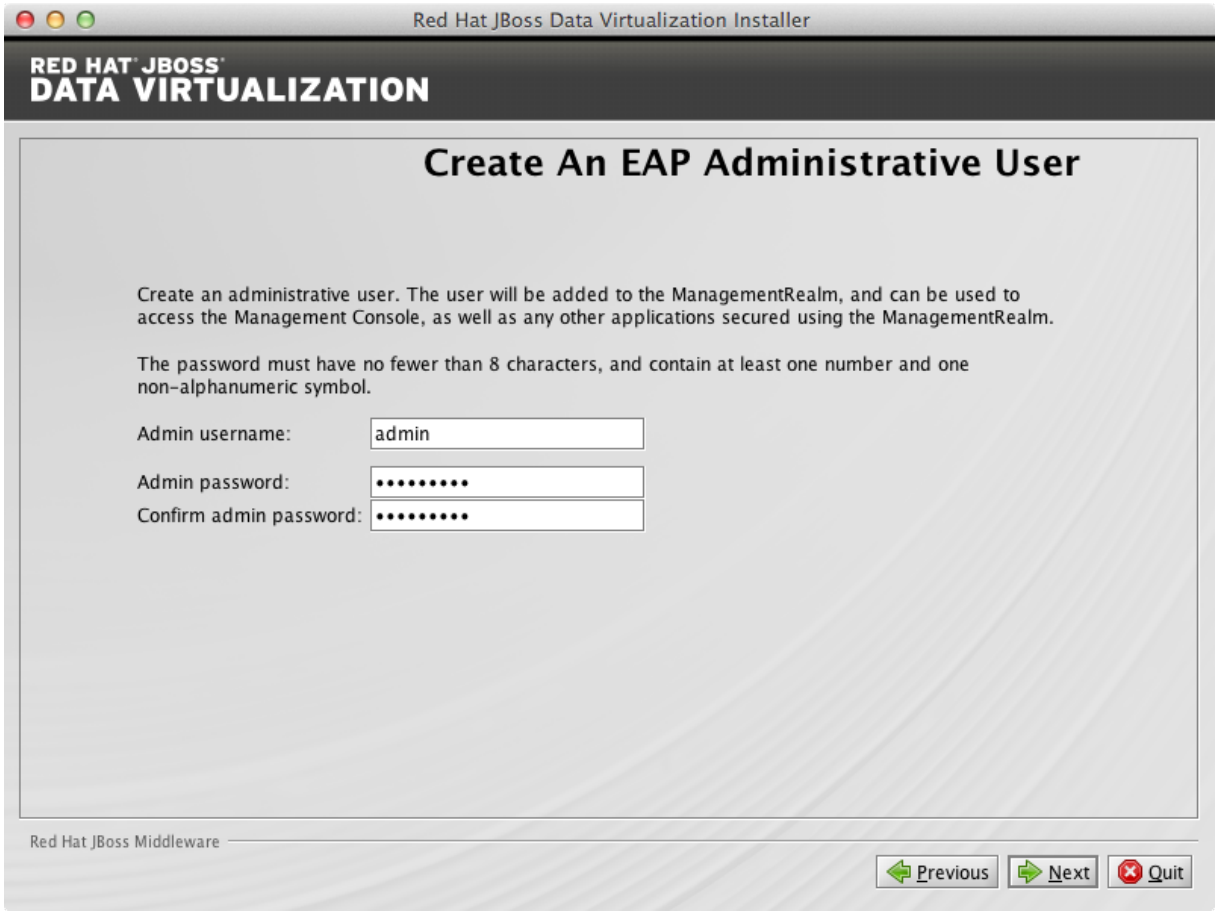

Figure 2.4.

The Maven Repository Setup window appears. You will need to provide Maven repository settings in order to build quickstarts provided in the Red Hat JBoss Data Virtualization installation. At this point, the installer can automatically configure your Maven settings to setup the online repository for remote access. To setup the Maven repository, select Specify the path (or URL)…. Enter the location of the Maven settings.xml file or select Browse to navigate to the file. Alternatively, you can choose to skip the Maven repository setup.

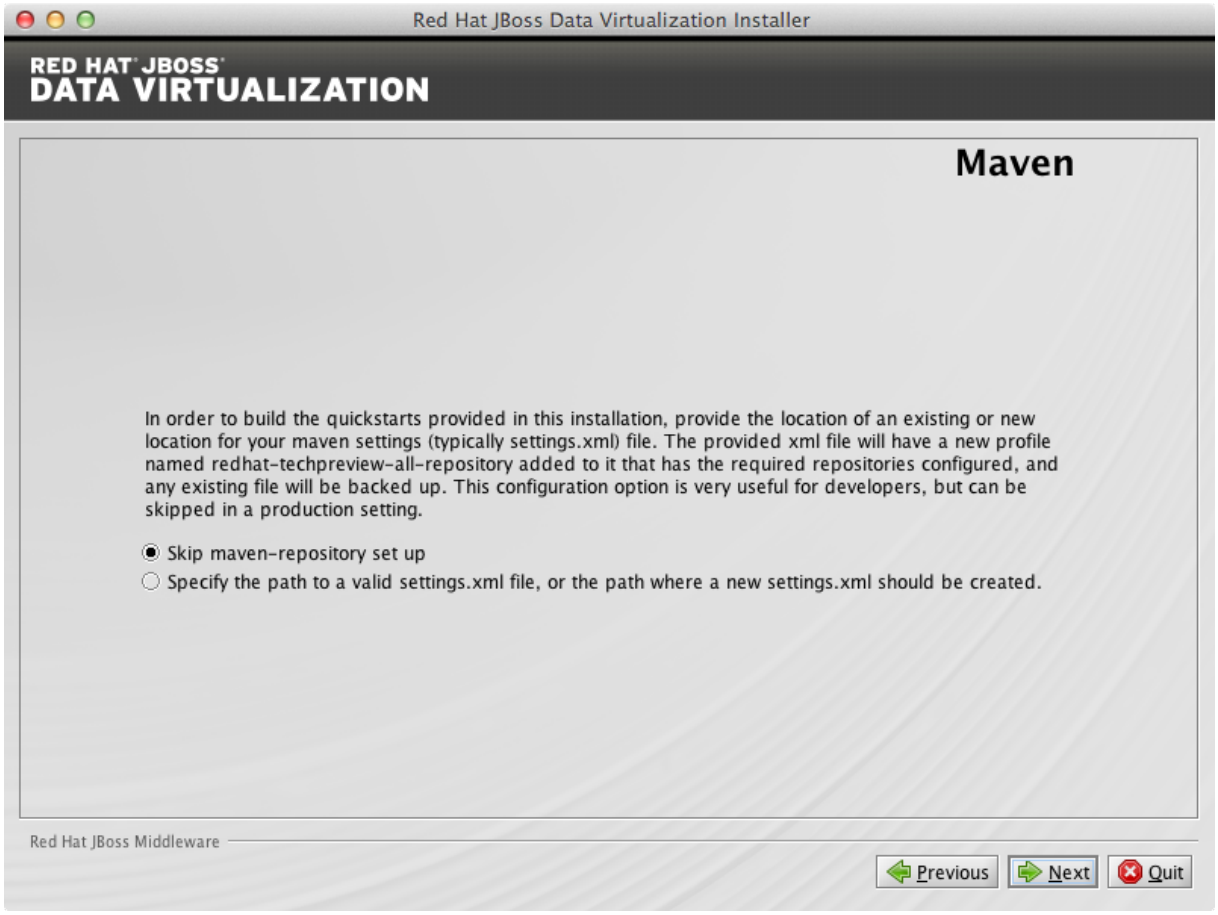

Figure 2.5.

Skip maven-repository set up for now. Click Next to proceed.

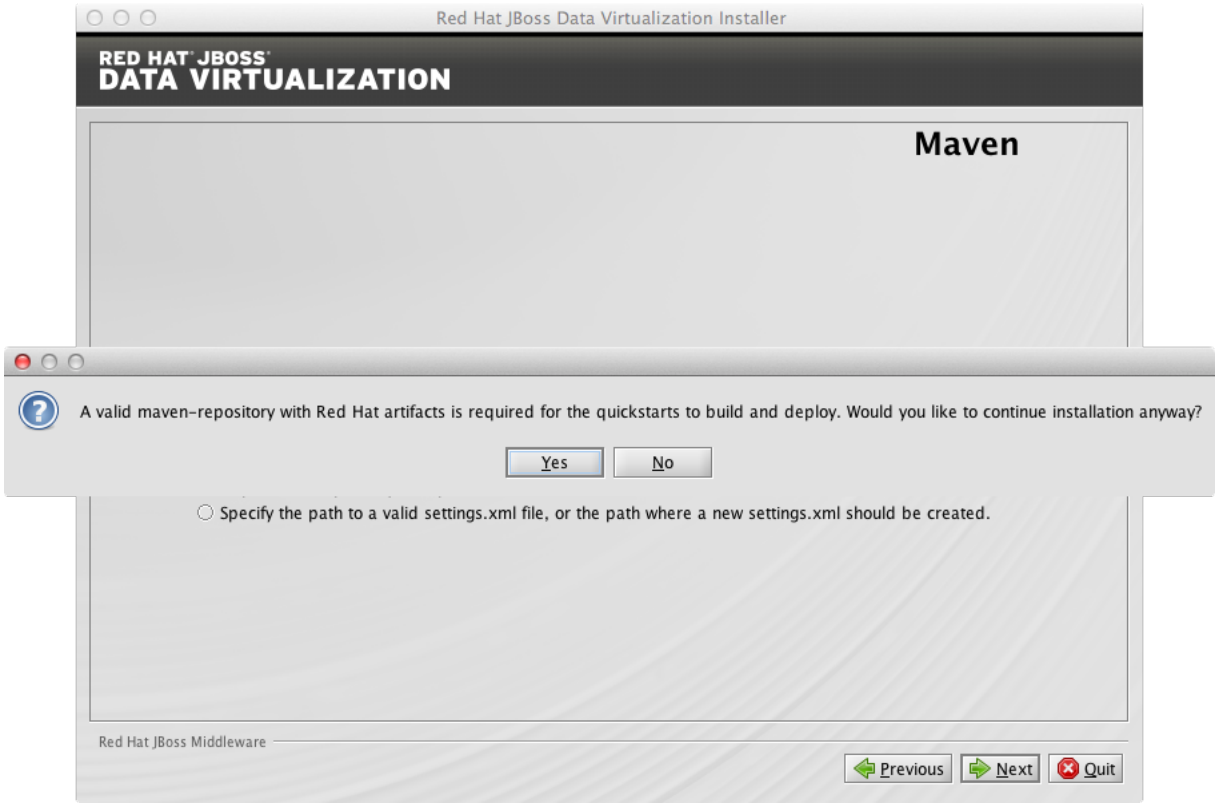

Figure 2.6.

Click Yes to proceed.

You can install Red Hat JBoss Data Virtualization either with default configuration or with additional configuration options. Select option Perform default configuration. Click Next to proceed.

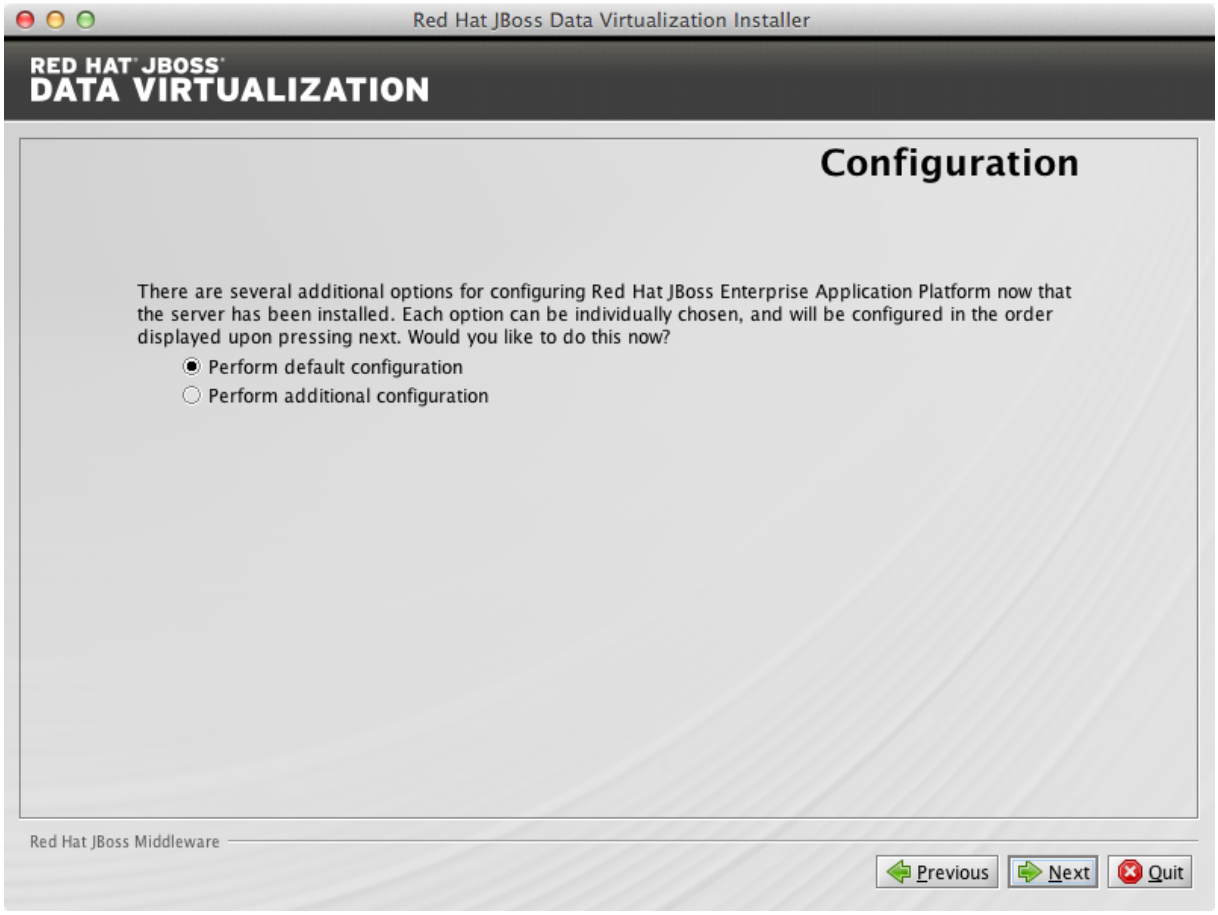

Figure 2.7.

A summary of the installation will be displayed, see below.

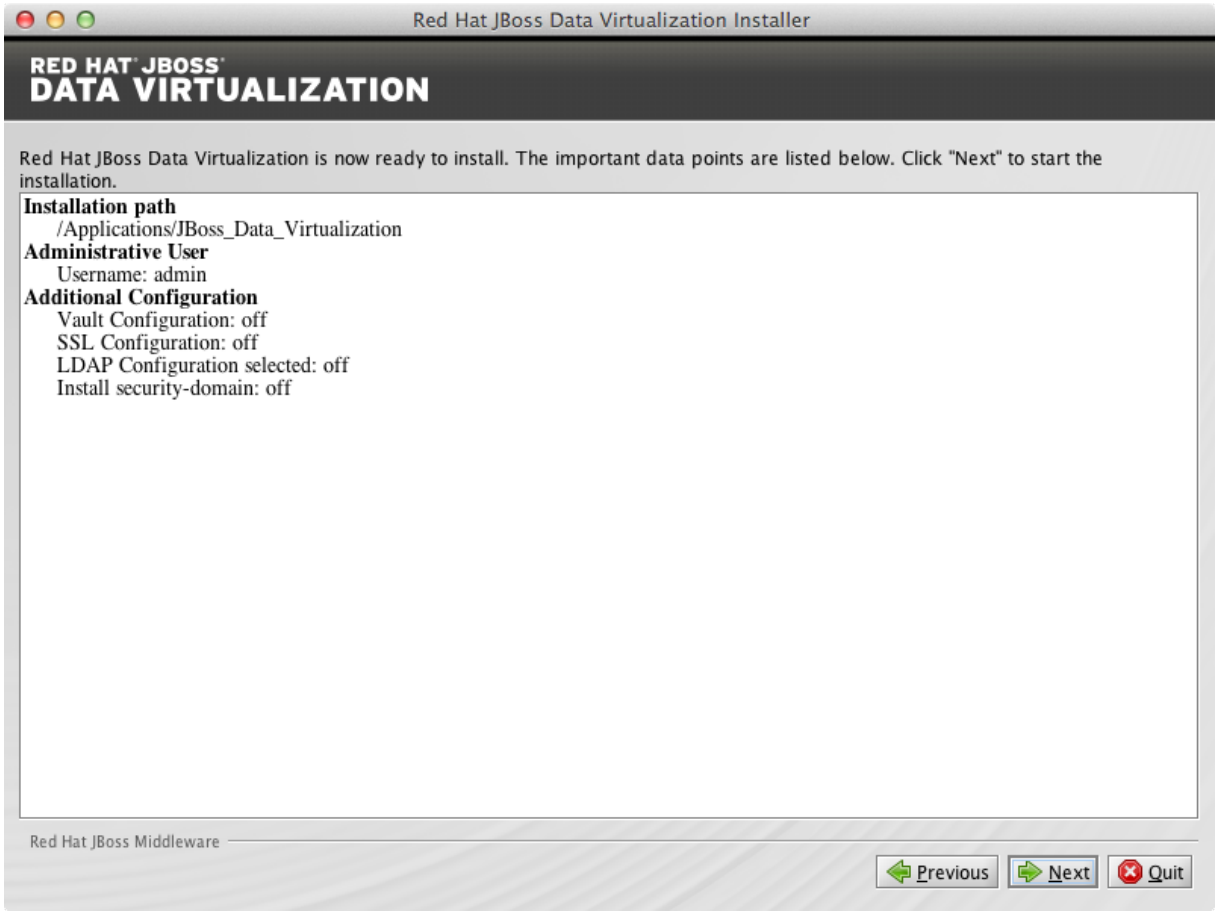

Figure 2.8.

Click Next for the installation to commence. This may take a minute.

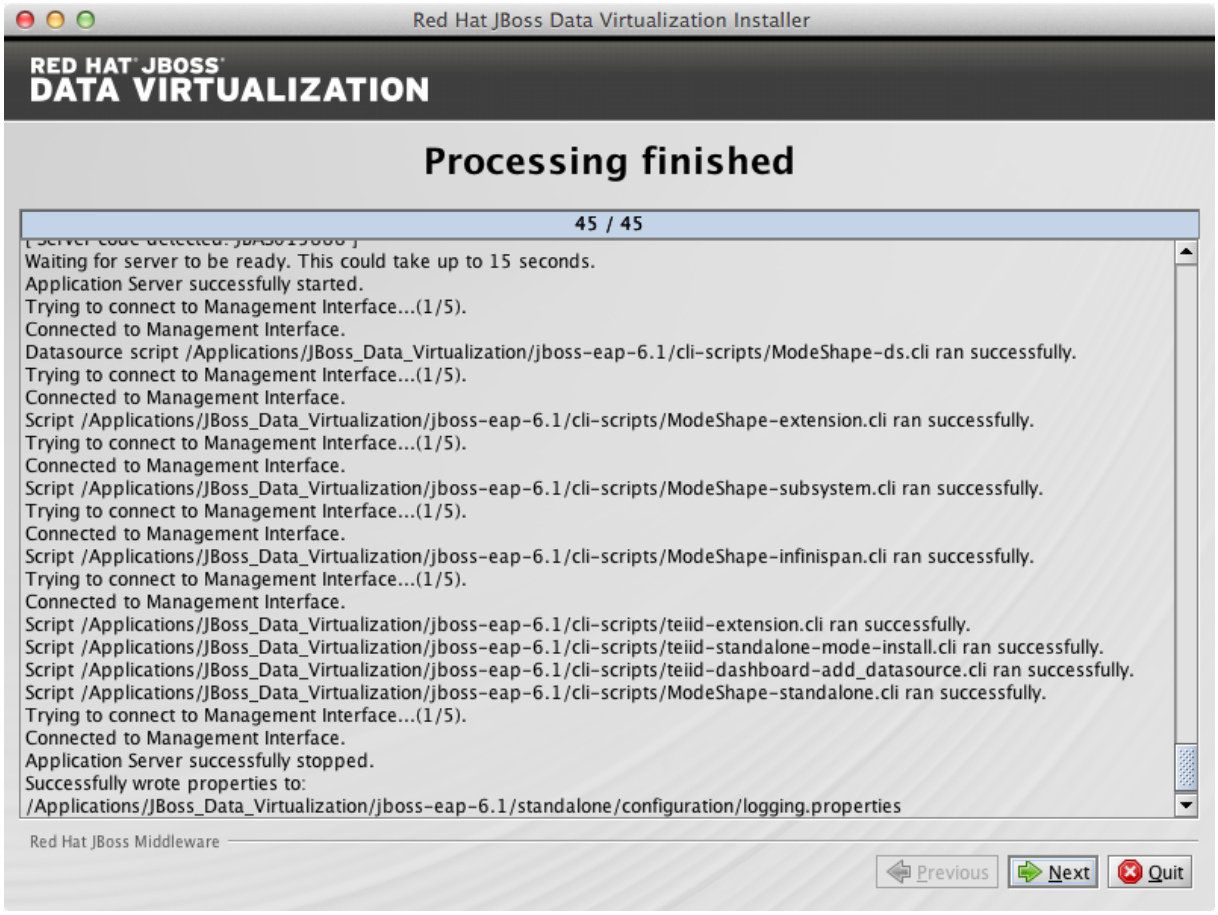

Figure 2.9.

Once all the components are installed, click Next.

Click Generate installation script and properties file if you wish to generate an automatic script and properties file.

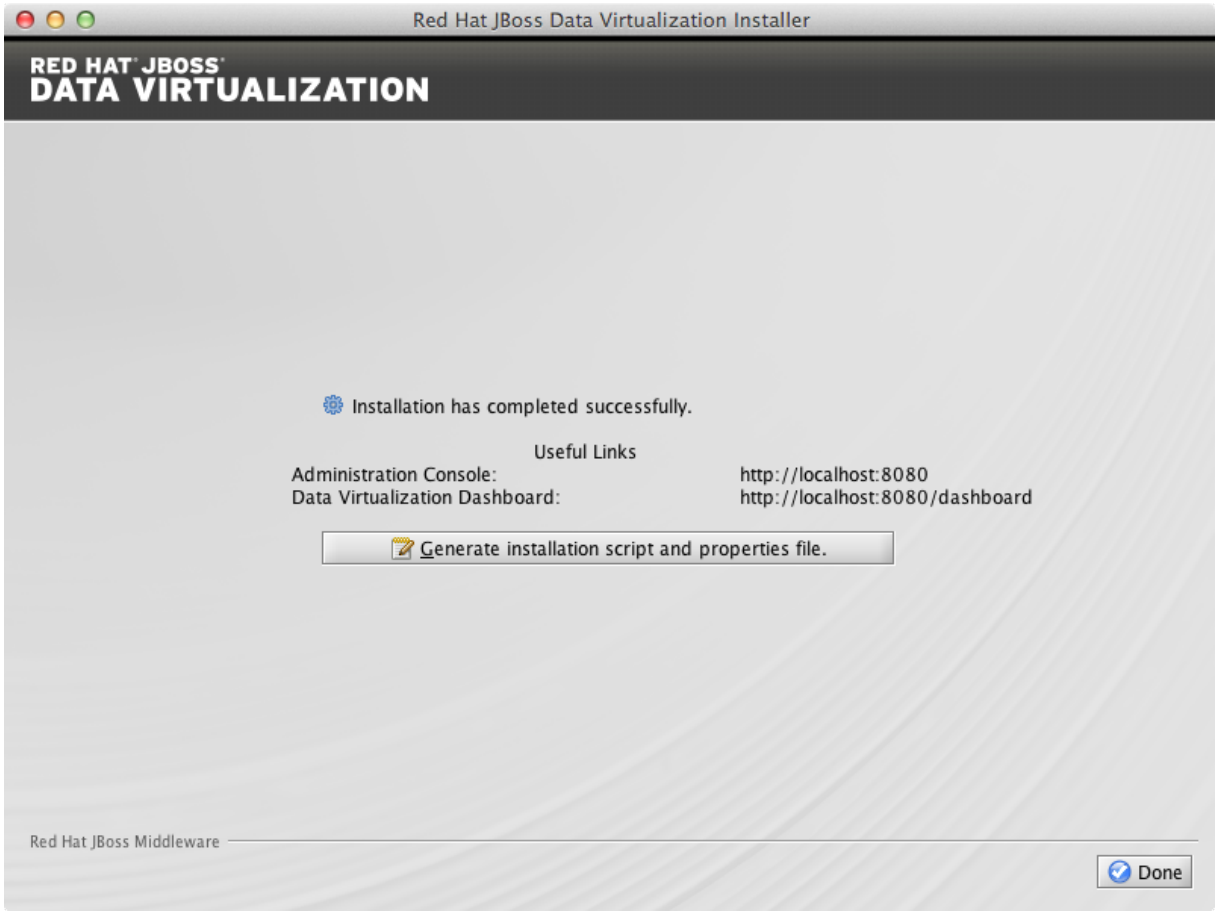

Figure 2.10.

For now click Done to complete the installation.

Red Hat JBoss Data Virtualization is now successfully installed and configured. When the installation is complete, navigate to

- Unix/Linux: EAP_HOME/bin and run the ./standalone.sh
- Windows: EAP_HOME\bin and run standalone.bat

to start the Red Hat JBoss Data Virtualization server.

2.2. Installing Red Hat JBoss Data Virtualization through automated script mode

Installing Red Hat JBoss Data Virtualization by using an automated script provides everything you need to get you started quickly. Clone the repository using git from https://github.com/kpeeples/simplified-dv-template.git

```
$ git clone https://github.com/kpeeples/simplified-dv-template.git
```

The following username/passwords will be installed automatically for access to

	Username	Password
JBoss EAP Administration console	admin	redhat1!
Teiid Server	user	user

Download the JBoss Data Virtualization installer binary by clicking the green download button at http://www.jboss.org/products/datavirt.html.

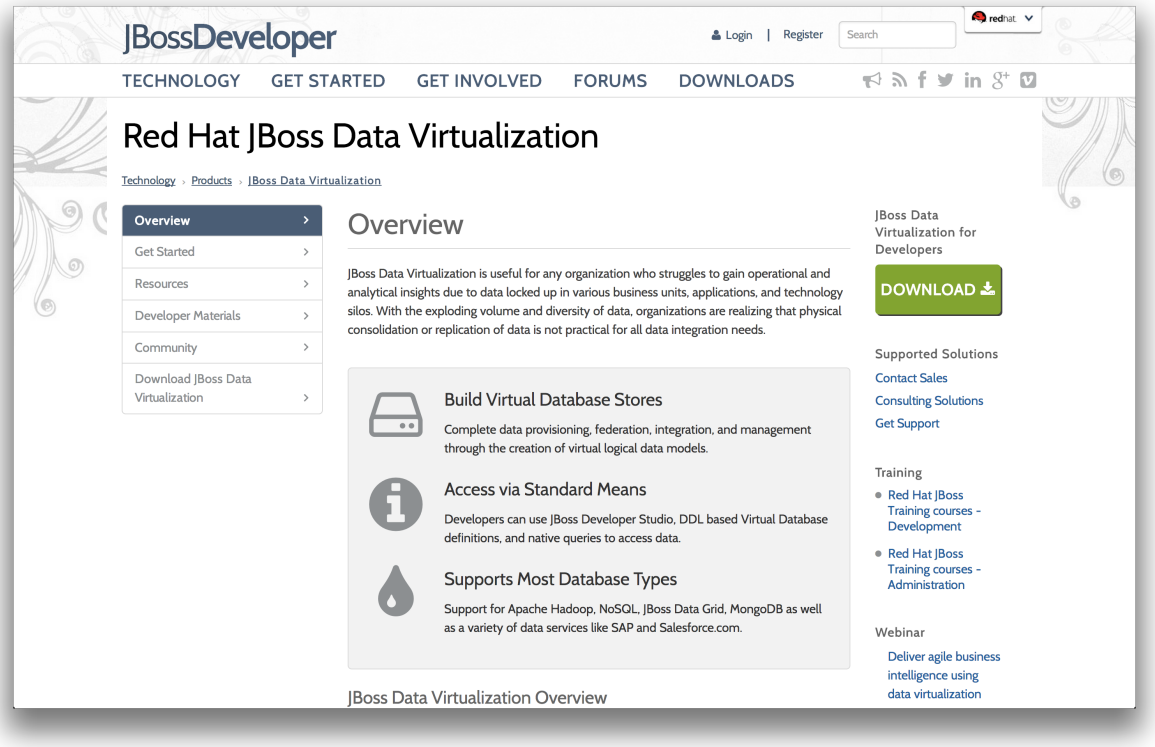

Figure 2.11.

Place the software in the distros subfolder of simplified-dv-template folder which was previously created when cloning the repository.

Modify the simplified-dv-template/support/InstallationScript.xml file to contain the full path to your installed/dv directory.

```
<installpath>/home/kpeeples/demos/dv-install-script-demo/installed/dv</
installpath>
```

Make sure to leave the installed/dv directory. The script performs the automated install of JBoss Data Virtualization v6.0.0.GA. Run the install-run.sh script to install JBoss Data Virtualization and the server will be started automatically as shown below.

```
$ ./install-run.sh
```

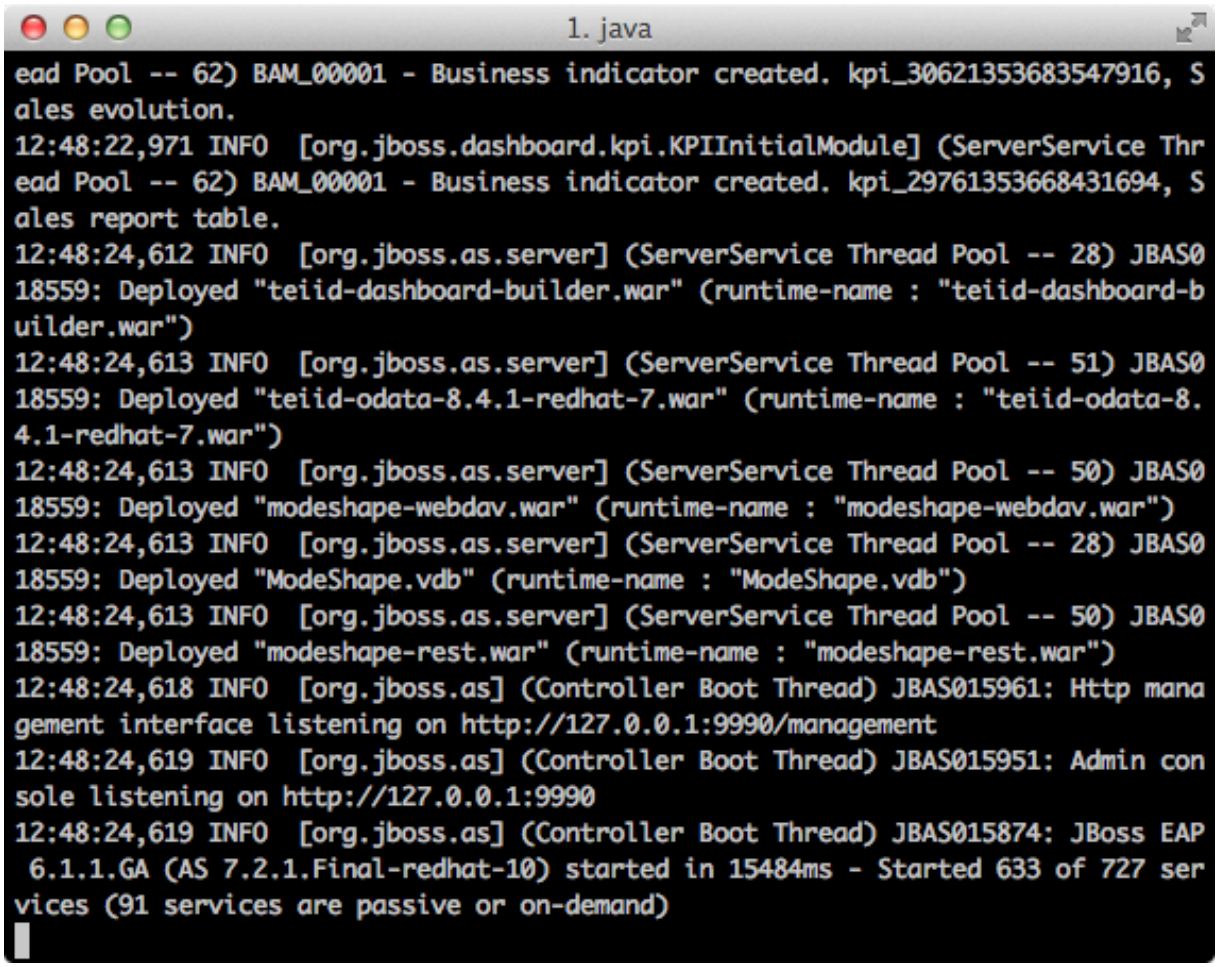

Figure 2.12.

In case you want to change the password of the admin user, go to simplified-dv-template/installed/dv/jboss-eap-6.1/bin and type the following command and inputs as shown below.

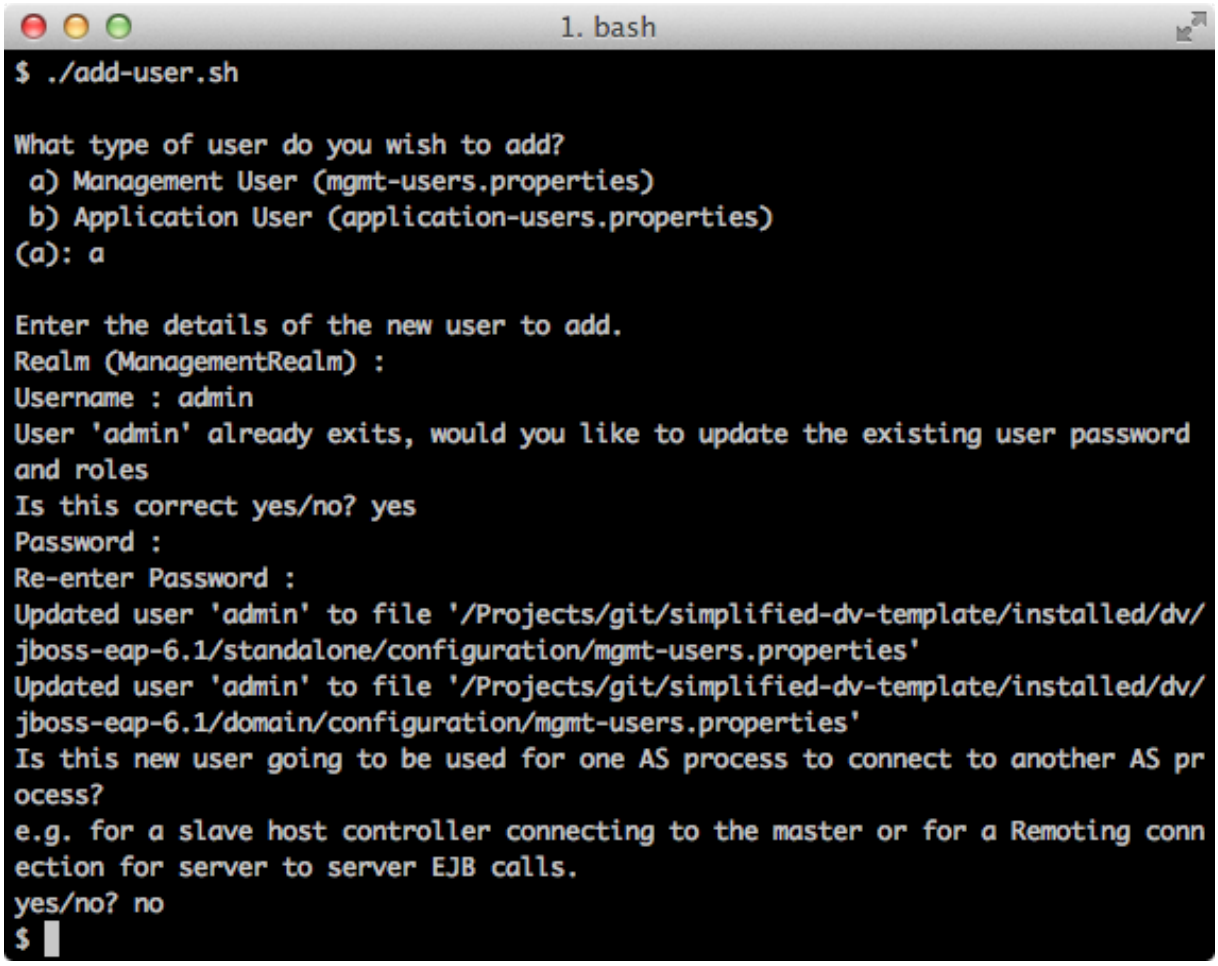

Figure 2.13.

Browse to http://localhost:8080/dashboard for the Red Hat JBoss Data Virtualization Dashboard to verify the installation and use user/user as the credentials that were installed as default and click Log In.

Red Hat JBoss Data Virtualization is now successfully installed, configured and started using the automated script mode.

2.3. Provision Red Hat JBoss Data Virtualization on OpenShift Online

With OpenShift you can easily deploy and run JBoss Data Virtualization in minutes to connect your applications to data from many different sources. JBoss Data Virtualization on OpenShift Online is available as a Developer Preview to allow you to explore the capabilities of the technology running on OpenShift Online.

Get your free OpenShift Online account Sign up for your free account OpenShift Online account at https://www.openshift.com/app/account/new and you should see the screen below.

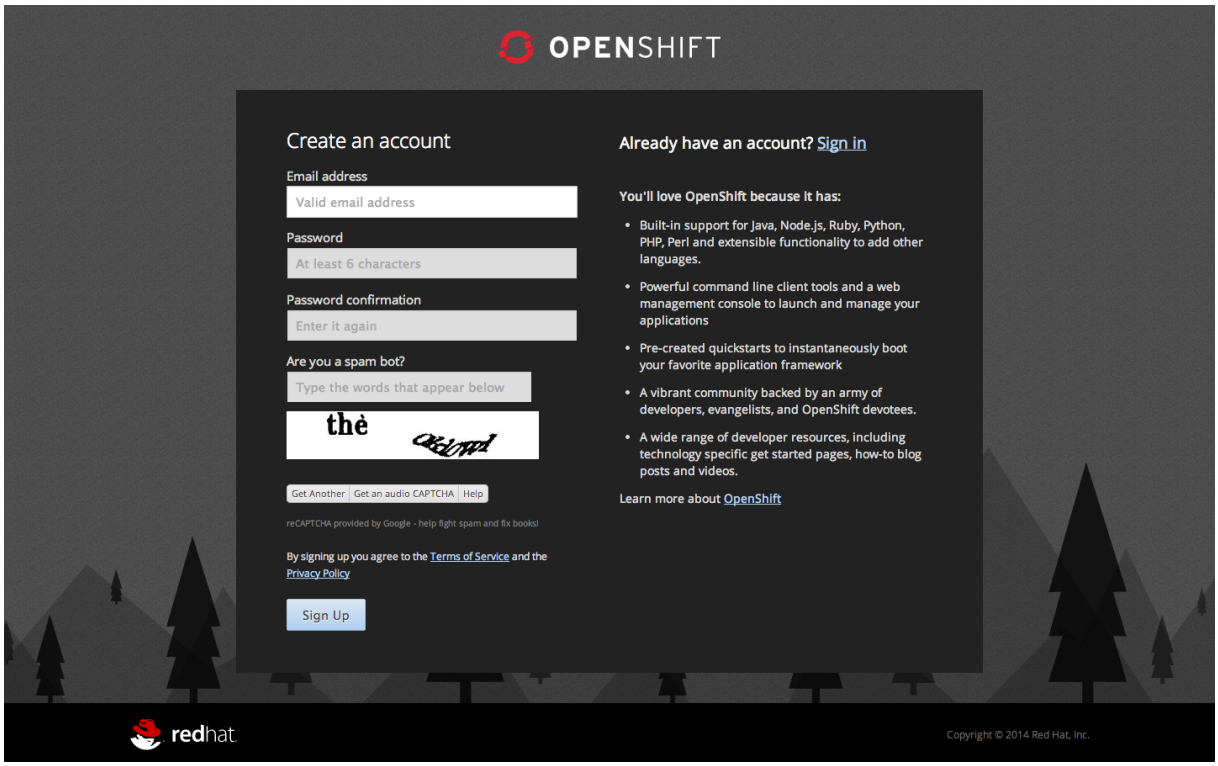

Figure 2.14.

If you already have an OpenShift Online account please sign in with your known OpenShift Online username password combination.

Create a new application If this is your first login into OpenShift Online click at the "→ Create your first application now" link

If you already have an OpenShift Online account click Add Application below your list of applications. Alternatively, you can deploy the DataVirtualization cartridge using the OpenShift RHC Client Tools. Using the rhc client tools type:

```
$ rhc app create dv jboss-dv-6.0.0
```

Choose a type of applications You can either scroll down to the list of quick links and click the JBoss Data Virtualization 6 button under "xPaaS" or search for "Data".

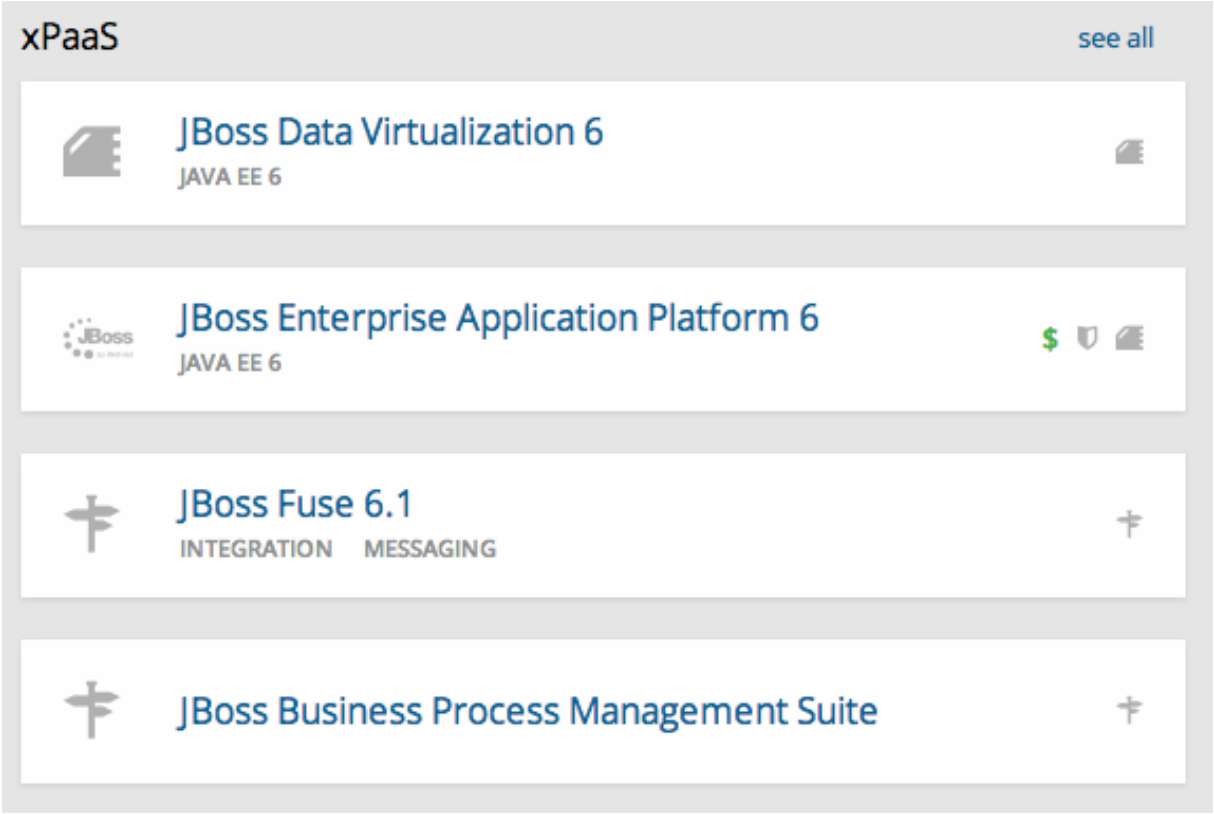

Figure 2.15.

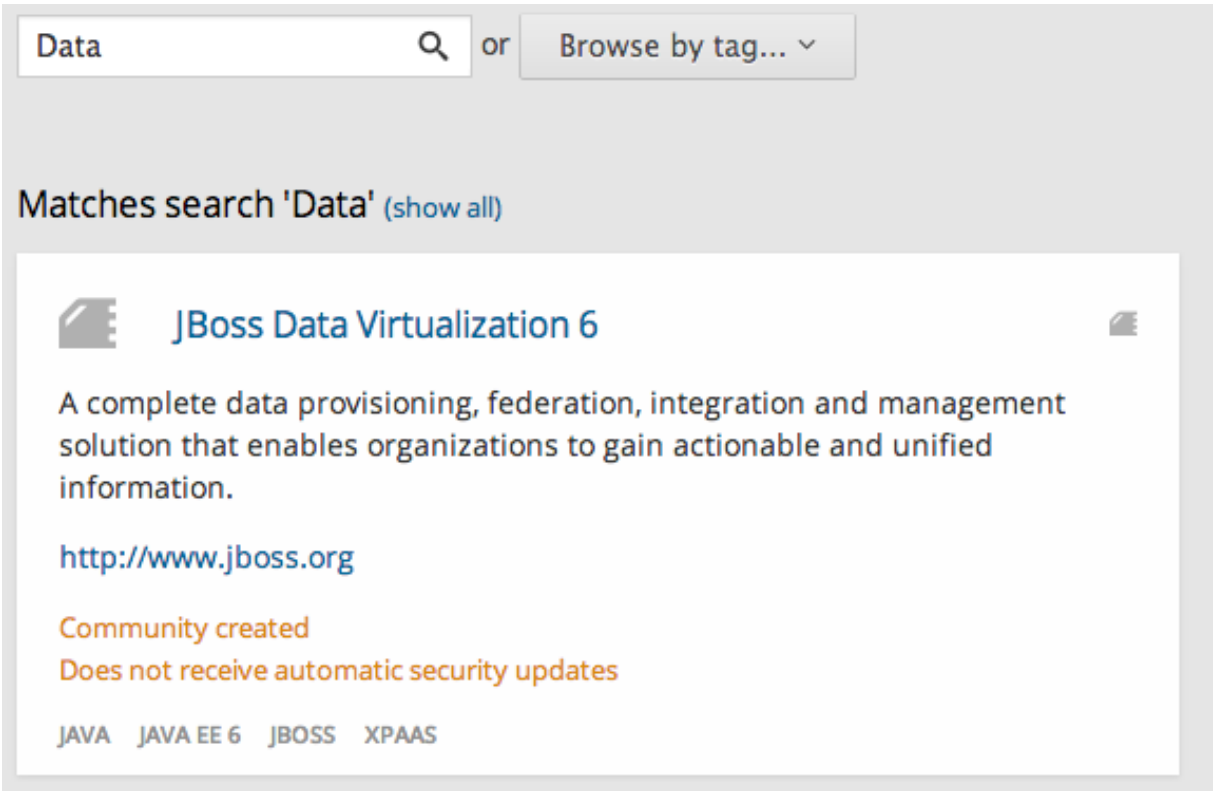

Figure 2.16.

Configure Application Name your application in your domain, scroll down and click the Create Application button.

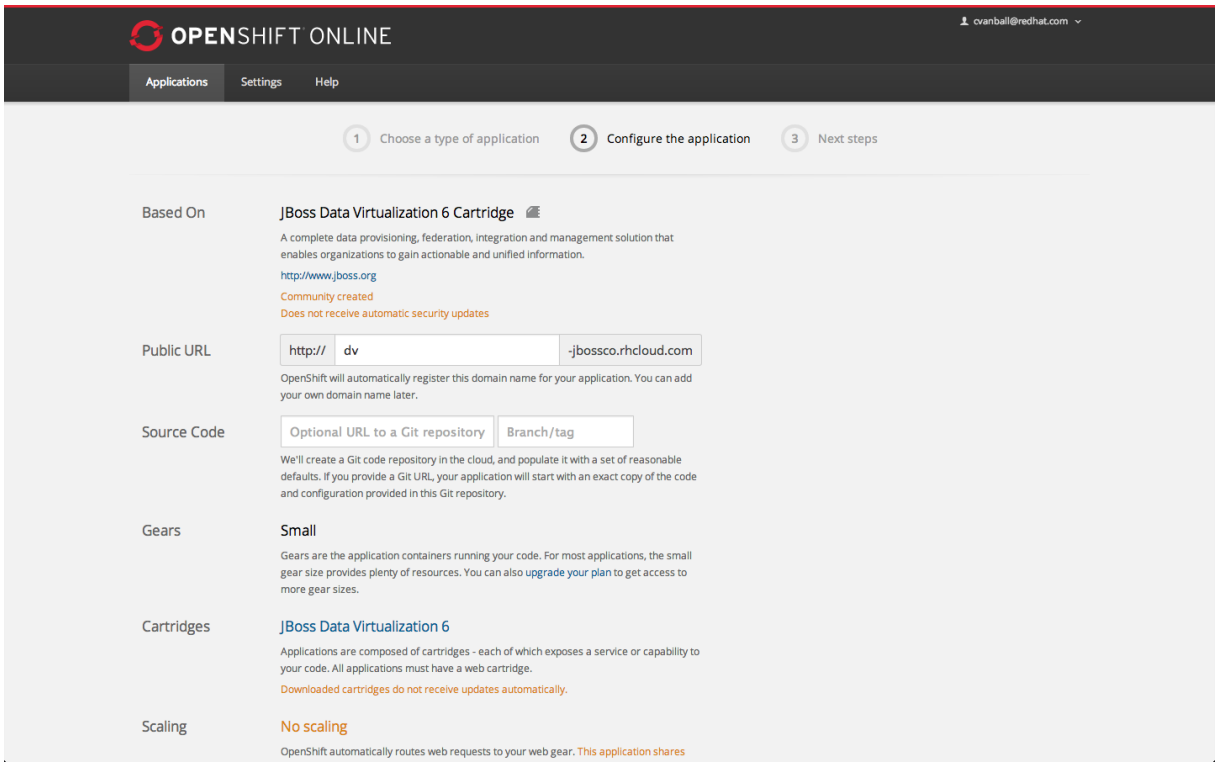

Figure 2.17.

Next steps In the Next steps we would like a PostgreSQL database to the application previously created on OpenShift Online. The figure below is shown when the application is successfully created in your domain.

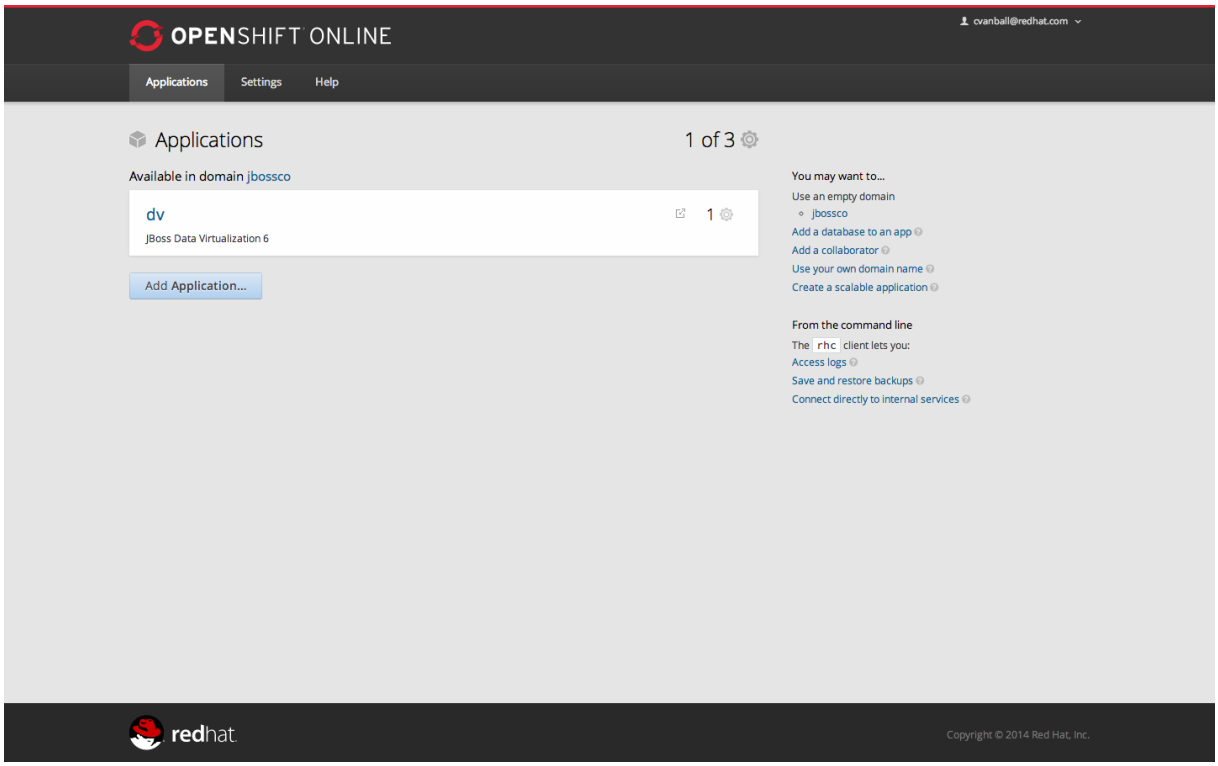

Figure 2.18.

Now we would like to add a PostgreSQL database to the application. Click the Application name link, in the above figure it's called "dv".

The following screen should appear.

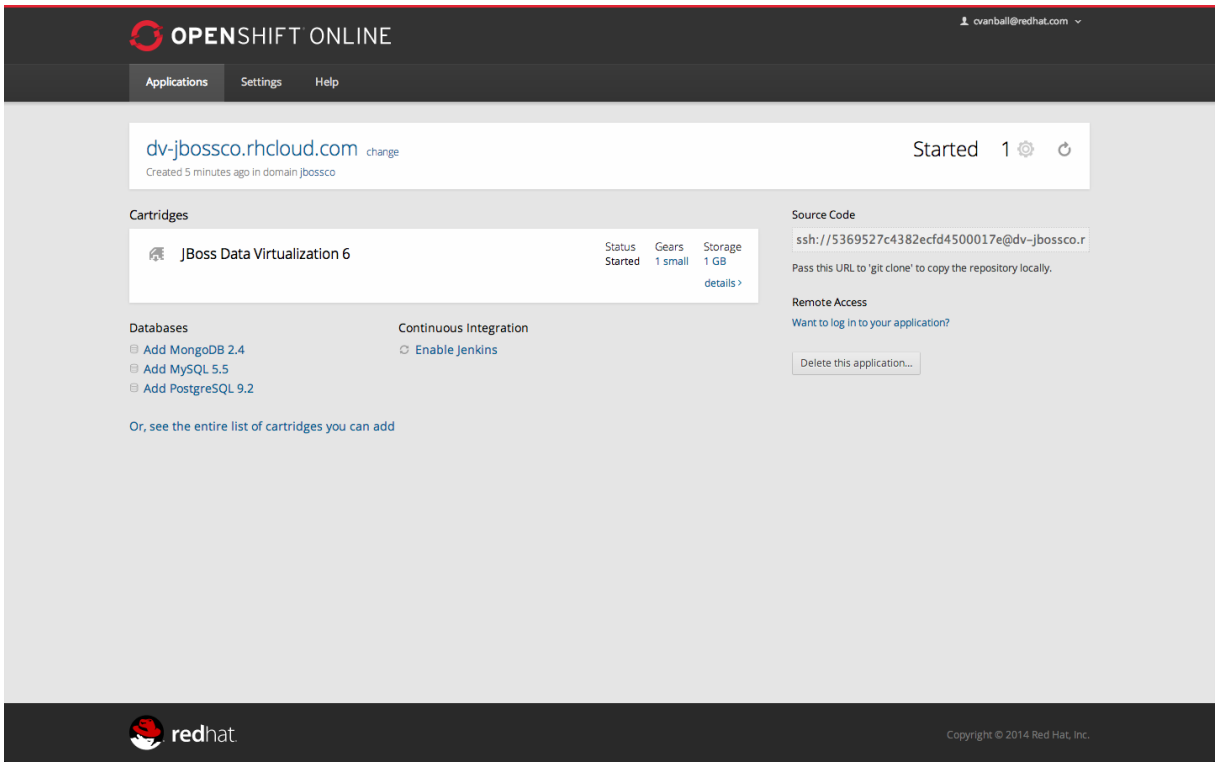

Figure 2.19.

Click “Add PostgreSQL 9.2” and click at the next appearing screen “Add Cartridge”. You have now a successfully created a Red Hat JBoss Data Virtualization environment with a PostgreSQL 9.2 database in just a matter of seconds.

At the moment you need a local installation of Red Hat JBoss Data Virtualization in order to deploy Data Virtualization projects to the OpenShift environment. This will be addressed in a newer version of JBoss Developer Studio.

Congratulations, you have now completed this lab.

Chapter 3. Install Red Hat JBoss Developer Studio

This lab will guide you how to install Red Hat JBoss Developer Studio and the tools for creating data views that are accessible through standard protocols (the Teiid Designer plug-in for Red Hat JBoss Developer Studio (JBDS) and connect Red Hat JBoss Developer Studio to the Red Hat JBoss Data Virtualization server.

If you don't already have Red Hat JBoss Developer Studio 7.1.1, download it from https://www.jboss.org/products/jbds.html by clicking the green download.

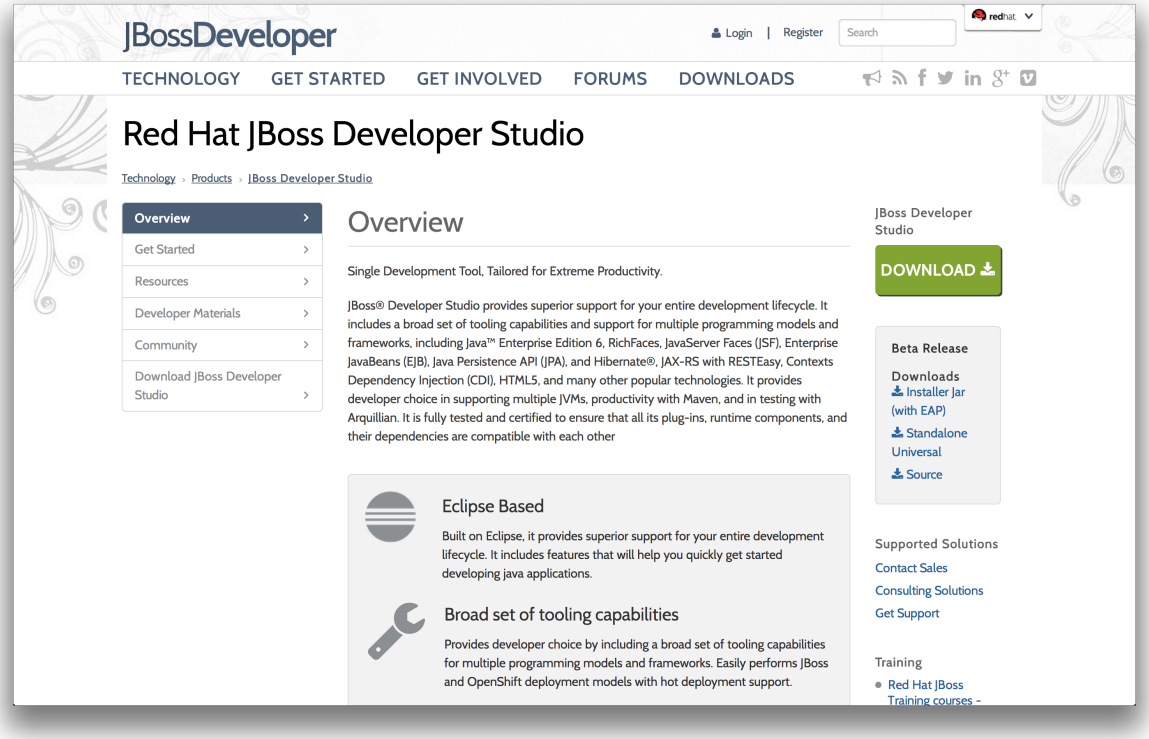

Figure 3.1.

Run the JBDS installer using java at the command prompt: java -jar {jbdevstudio-installer.jar} The current available version is 7.1.1.GA, and to run the installer at the command prompt see below.

```
$ java -jar jbdevstudio-product-universal-7.1.1.GA-v20140314-2145-B688.jar
```

Follow the installer prompts to complete the installation process.

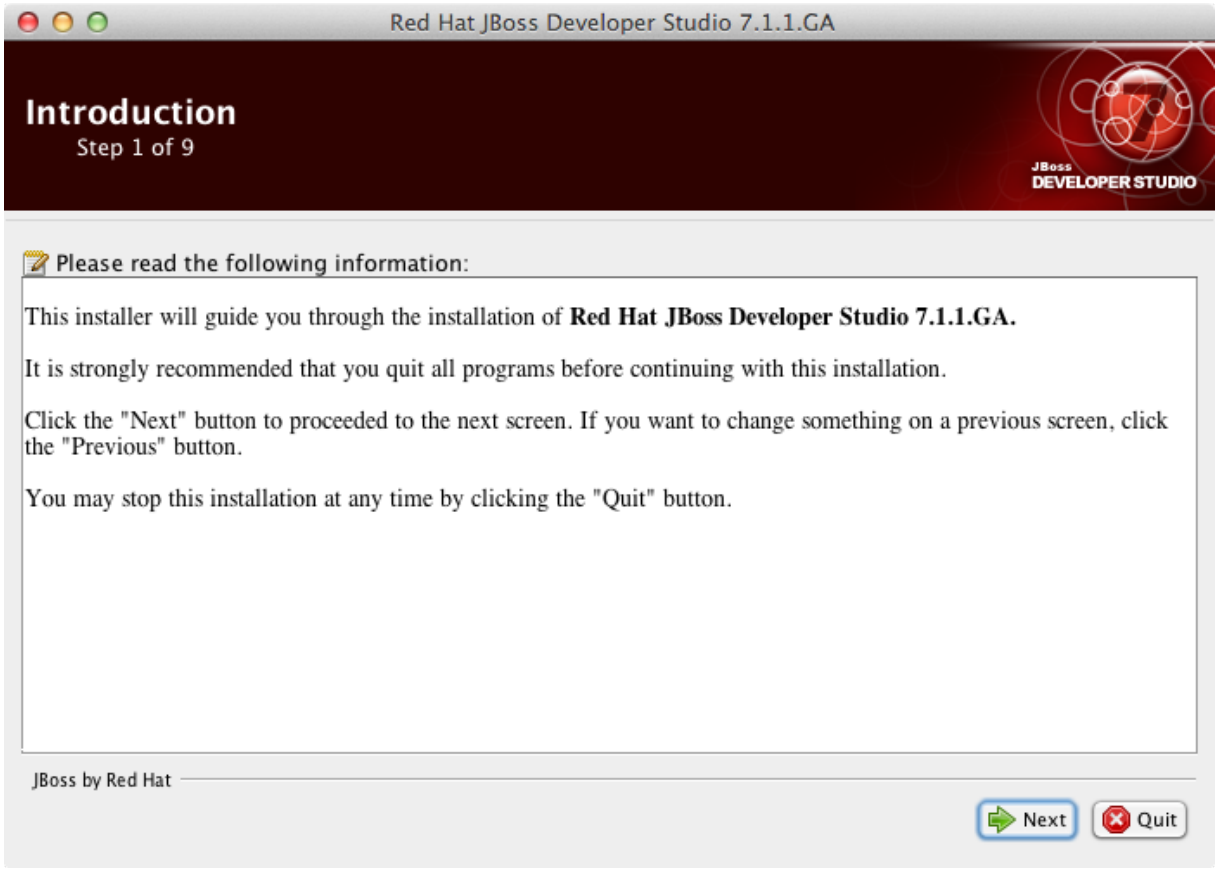

Figure 3.2.

When the Installer window opens, click Next.

Figure 3.3.

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.

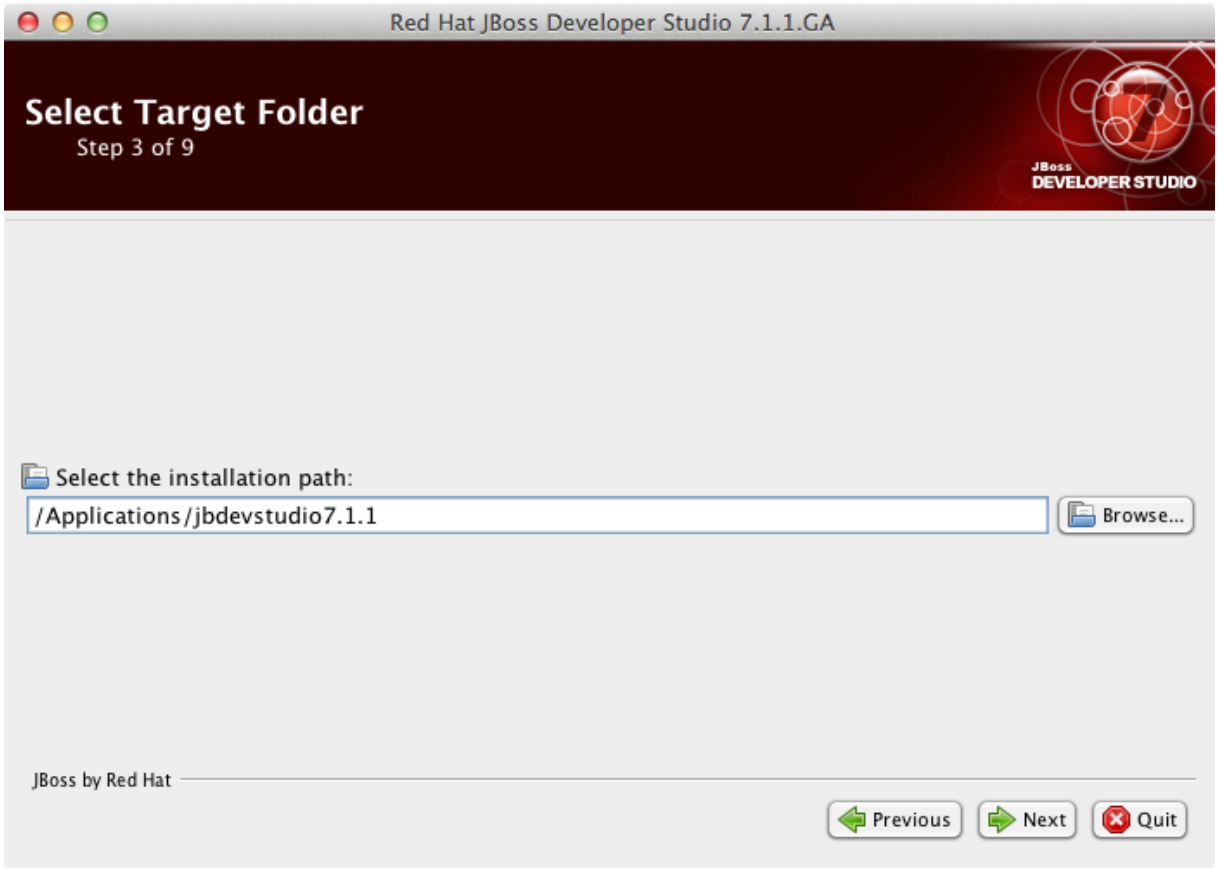

Figure 3.4.

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.

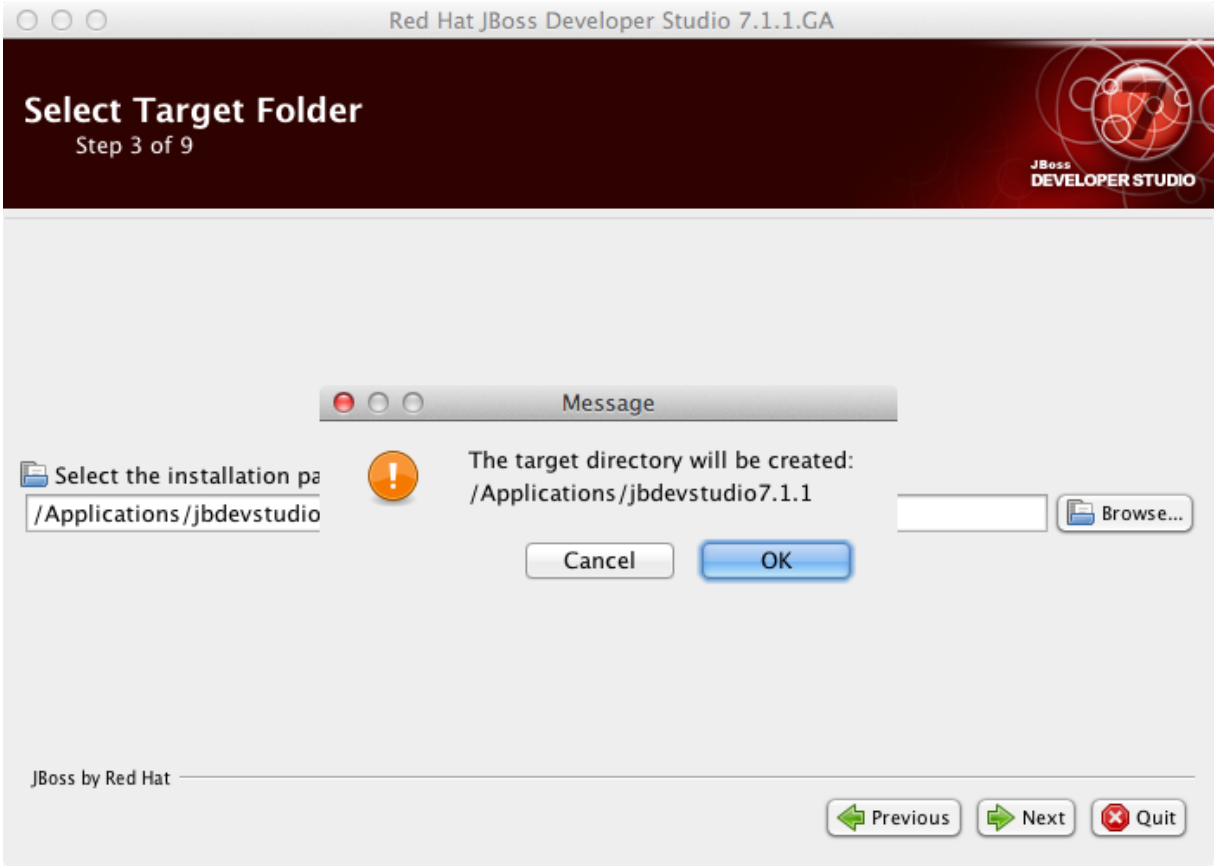

Figure 3.5.

When you are prompted about the specified location being created or overwritten, review the message and, if satisfied, click OK to proceed and click Next to continue.

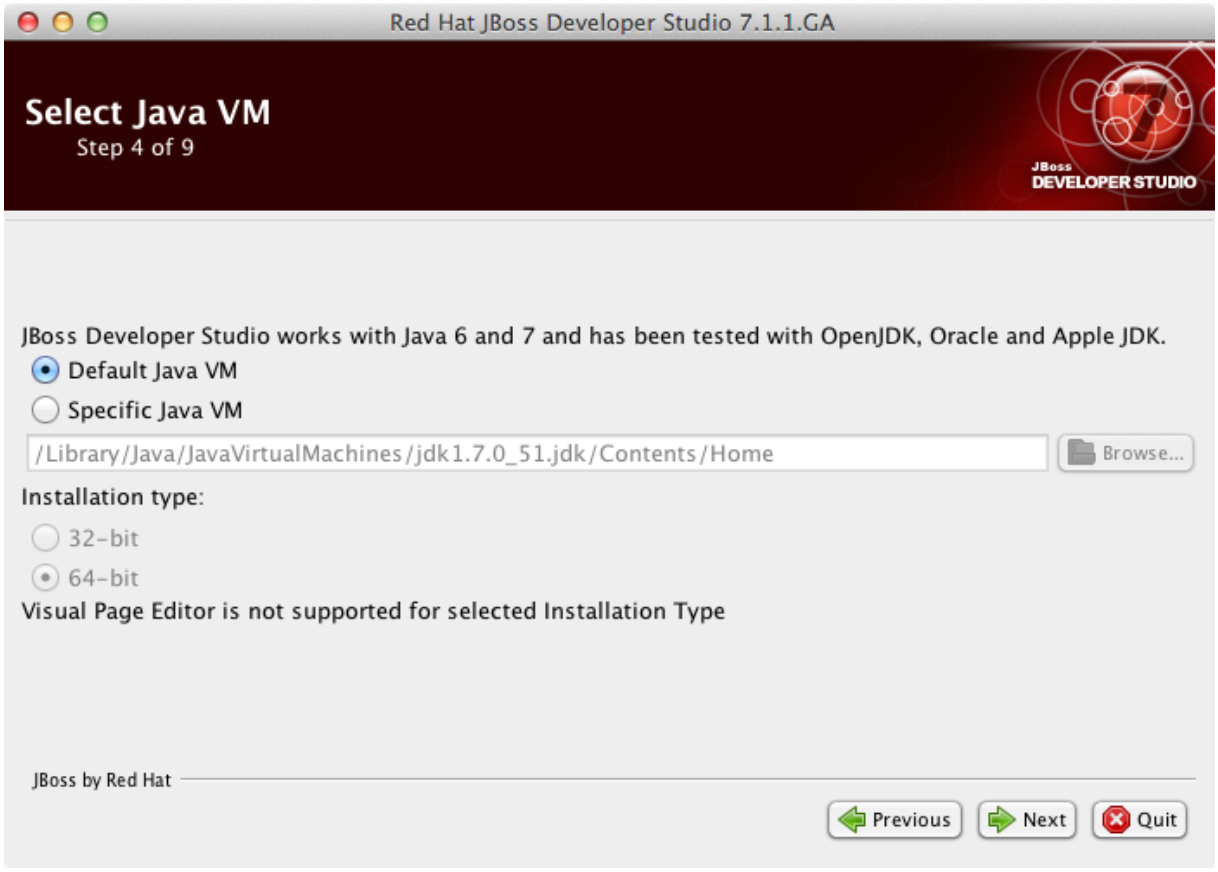

Figure 3.6.

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. This is based on the default Java developer kit of your system. 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.

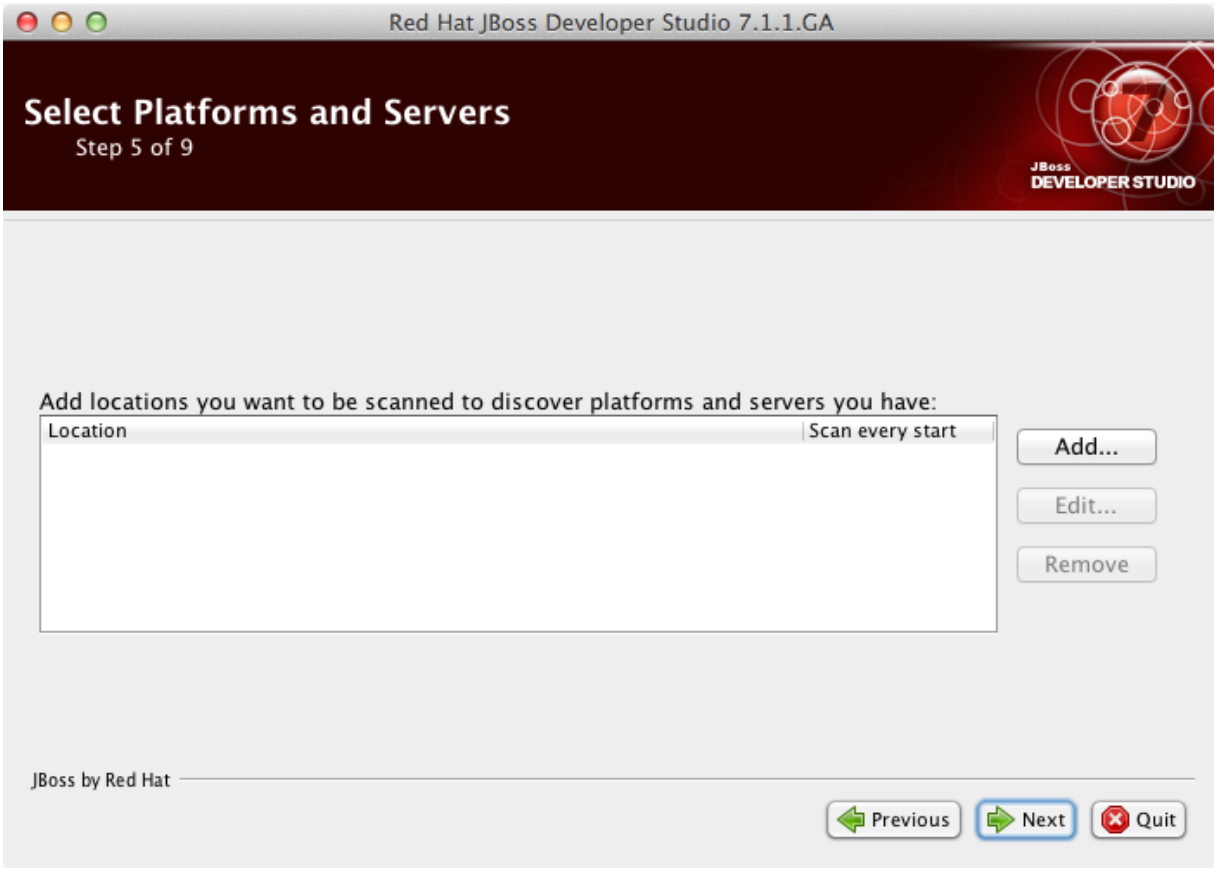

Figure 3.7.

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.

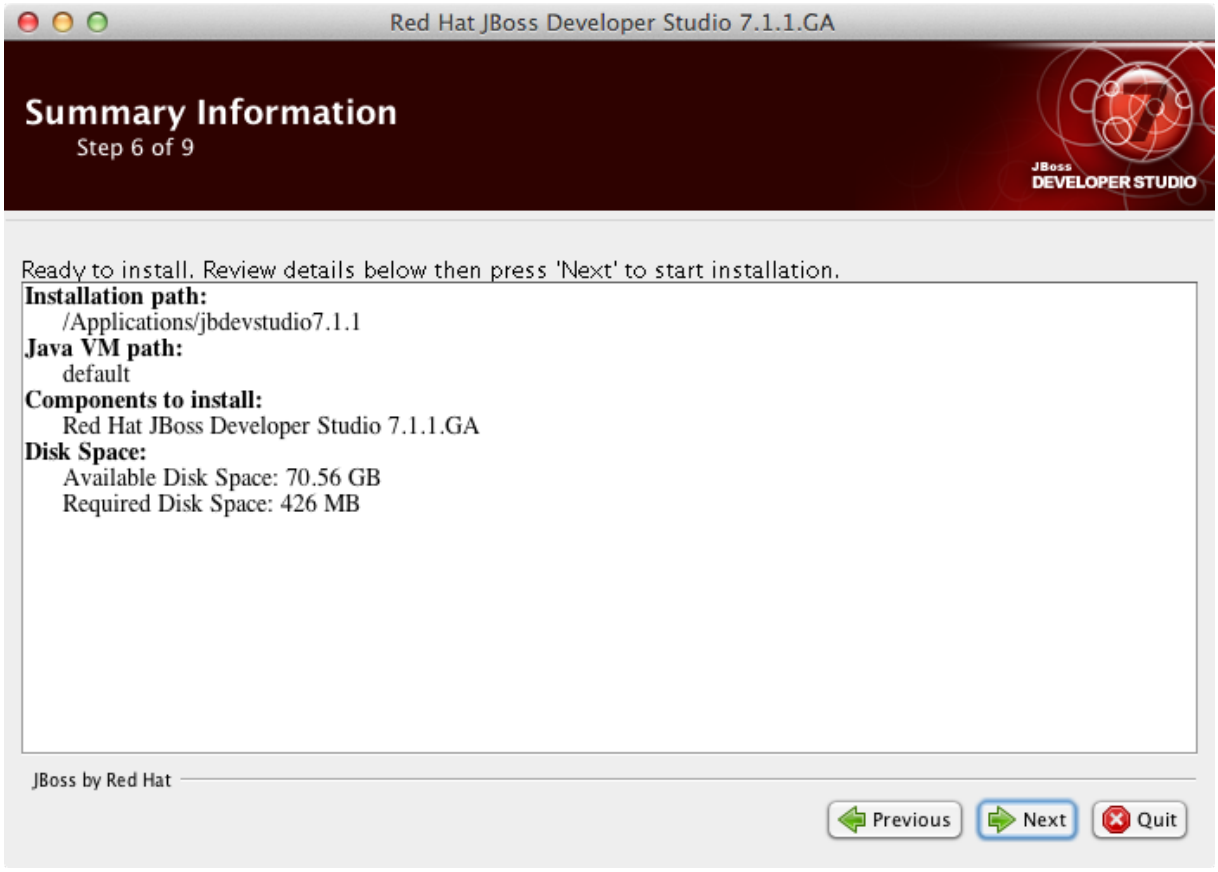

Figure 3.8.

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

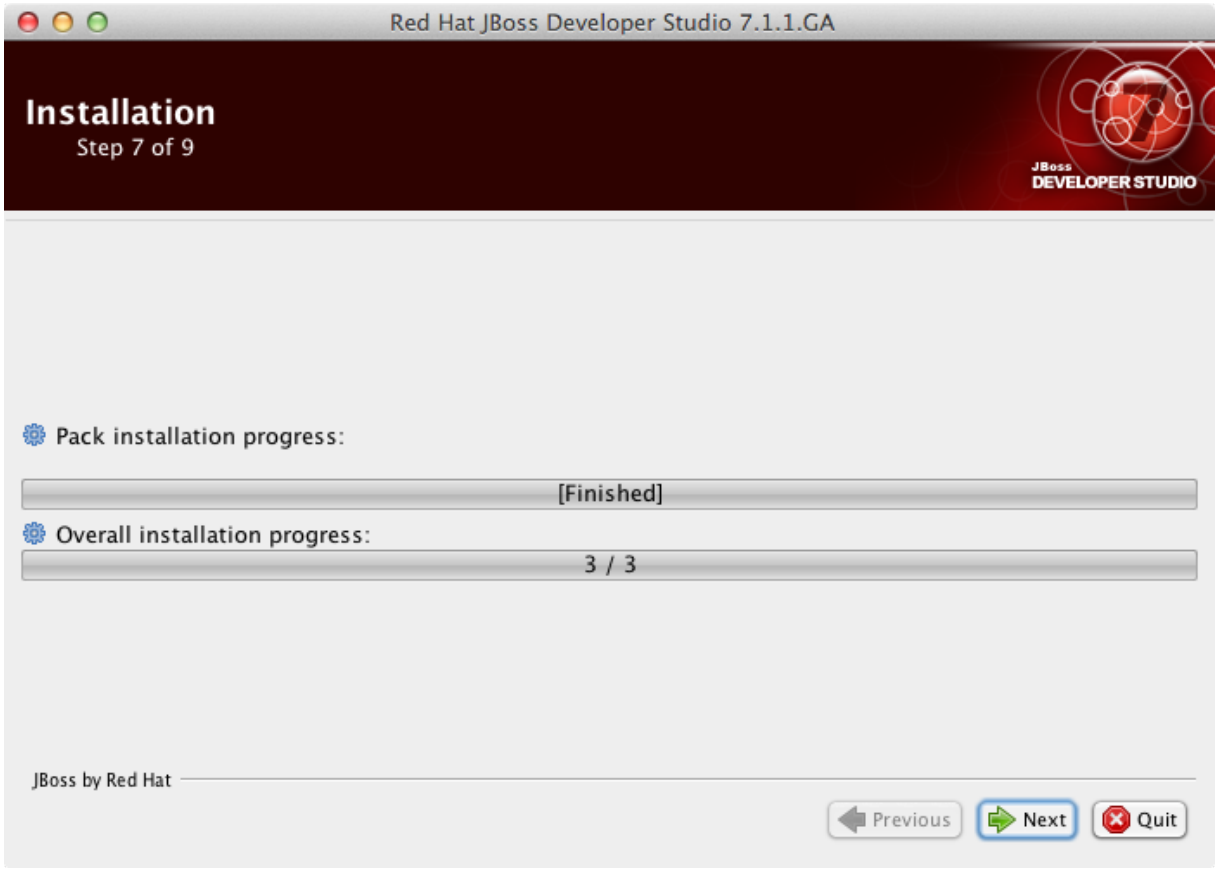

Figure 3.9.

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

To create shortcuts for starting JBoss Developer Studio, select the Create shortcuts in the Start-Menu and Create additional shortcut on the desktop check boxes and click Next. To automatically start JBoss Developer Studio when the Installer window closes, select the Run JBoss Developer Studio after installation check box.

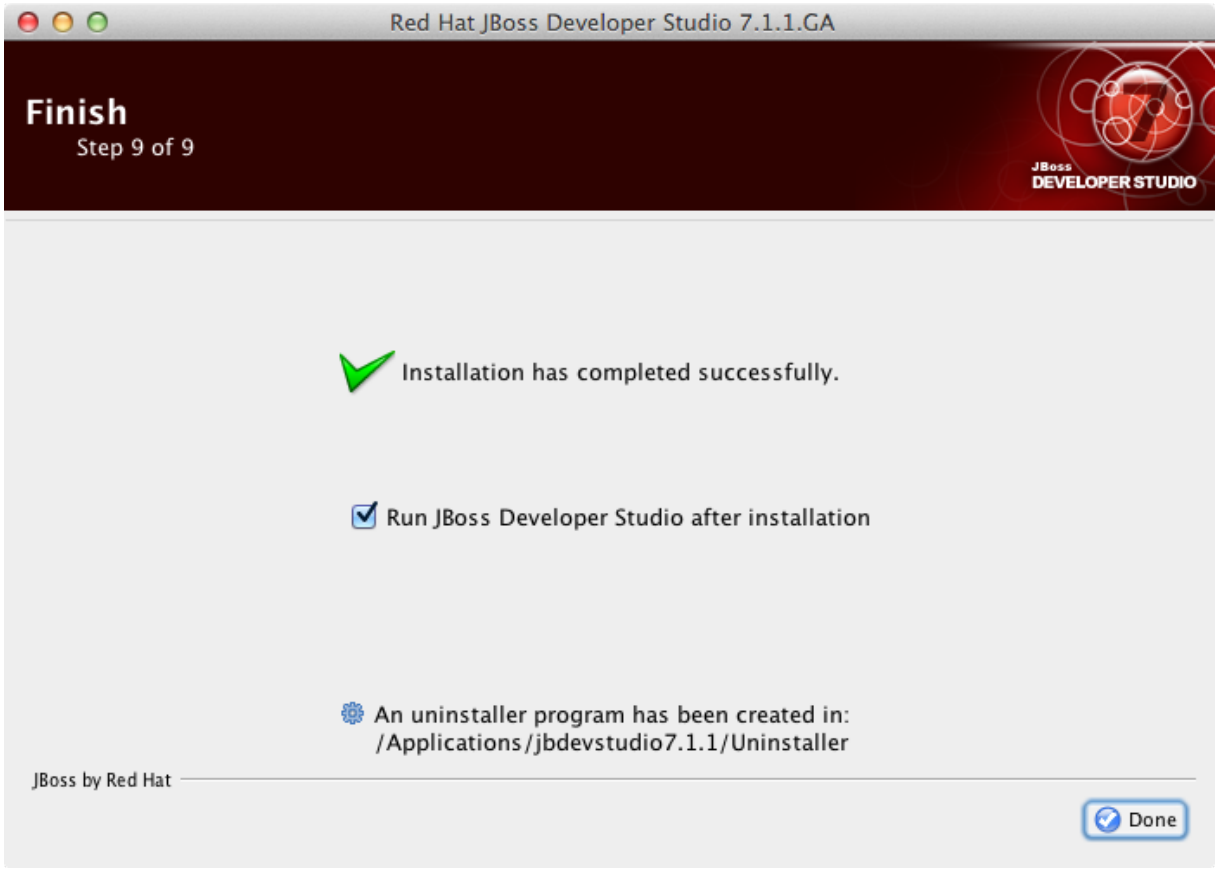

Figure 3.10.

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.

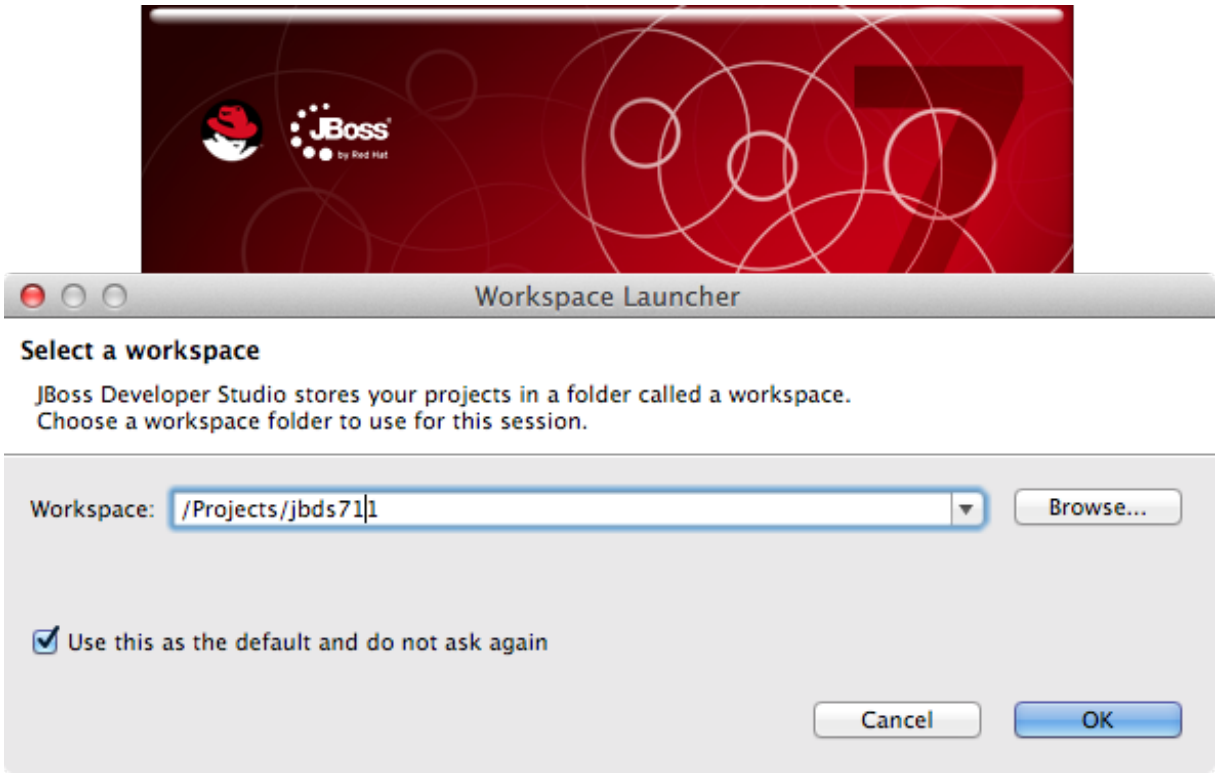

Figure 3.11.

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…

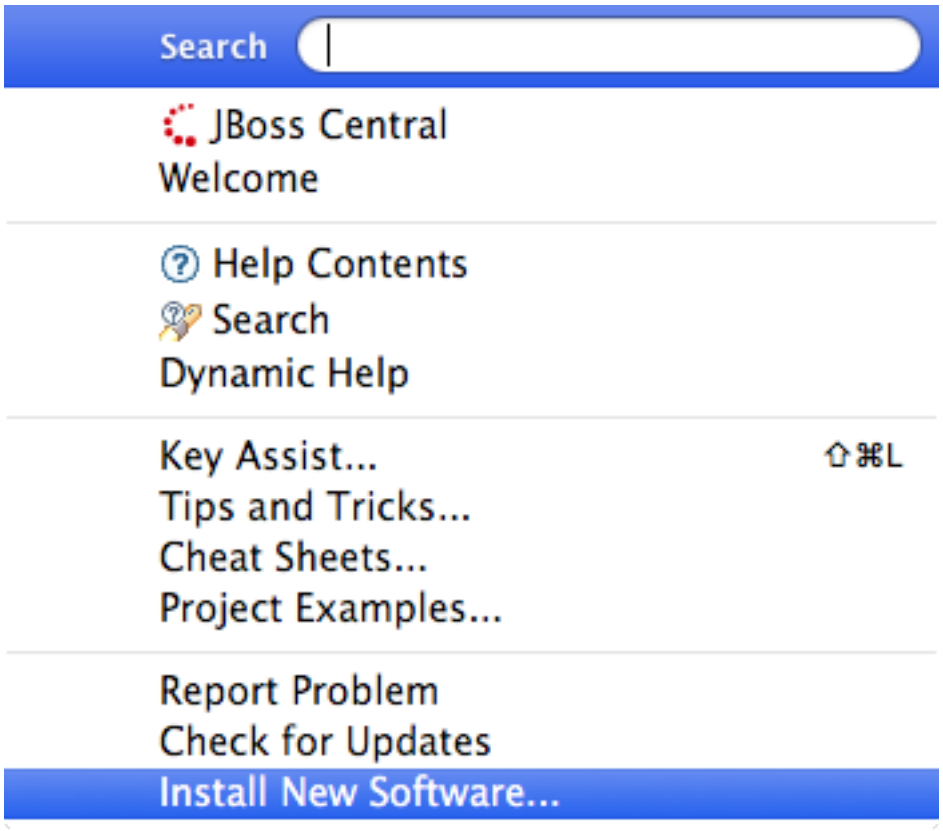

Figure 3.12.

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.

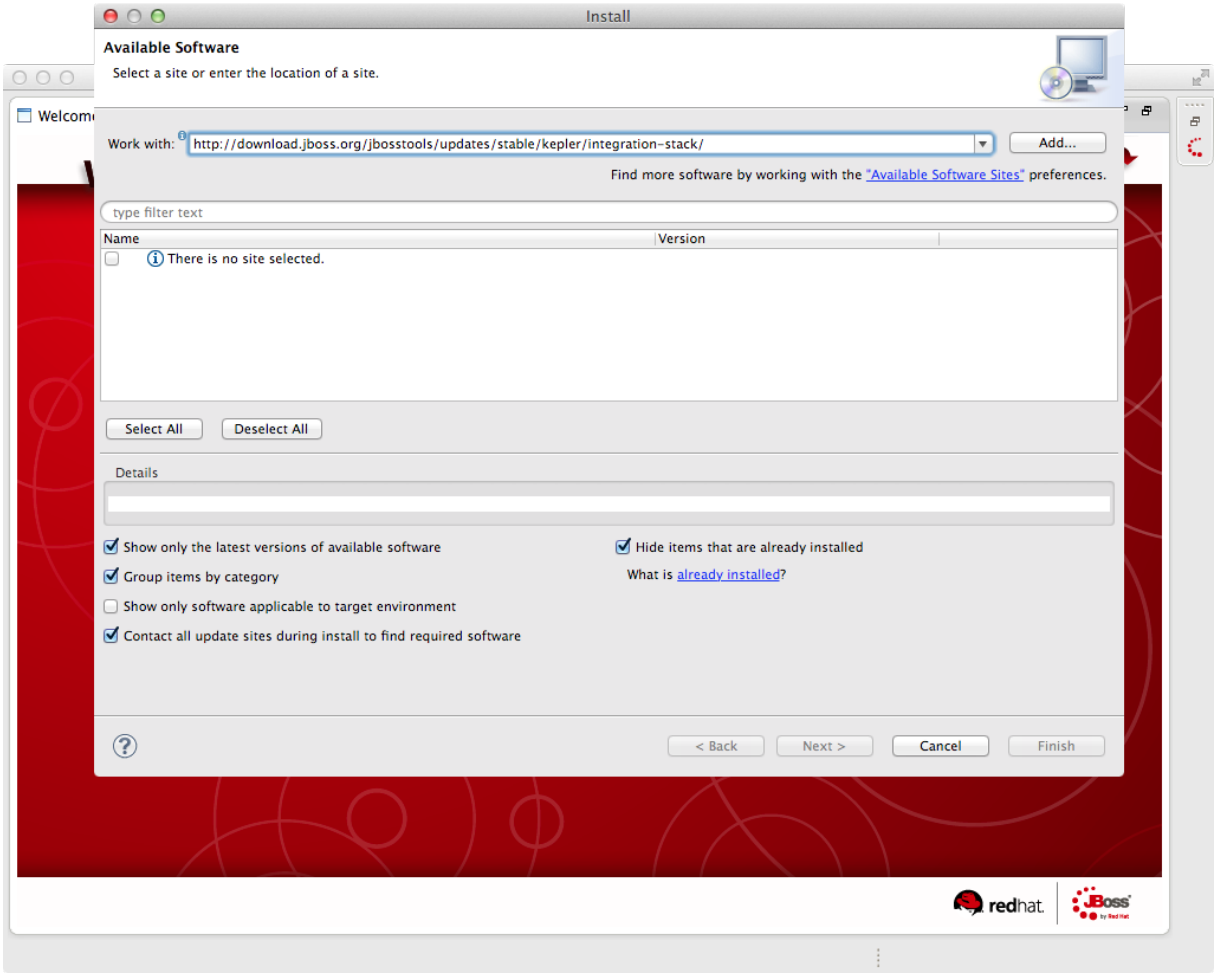

Figure 3.13.

Select JBoss Data Virtualization Development option in the list and click Next.

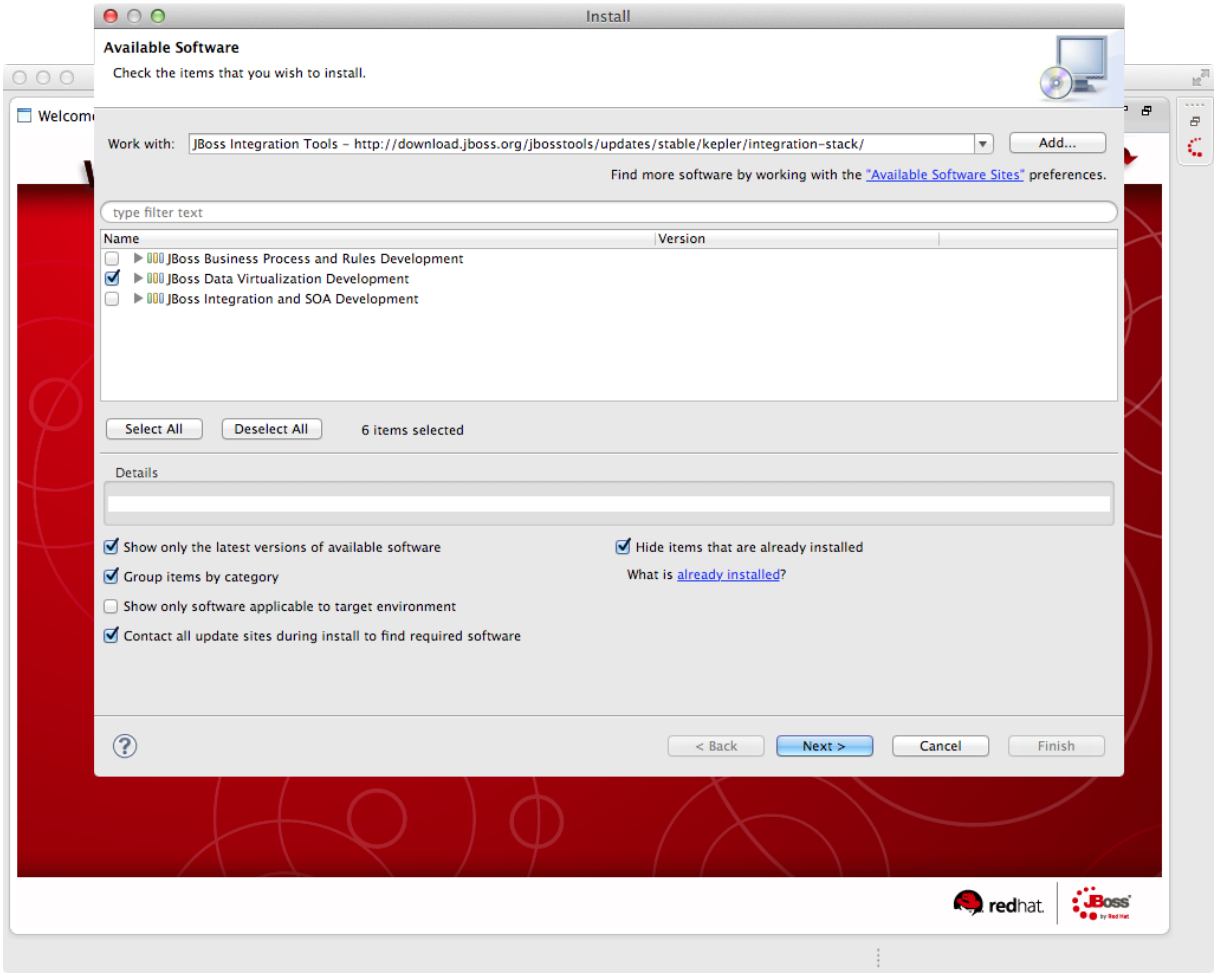

Figure 3.14.

Review Install Details and click the Next > button.

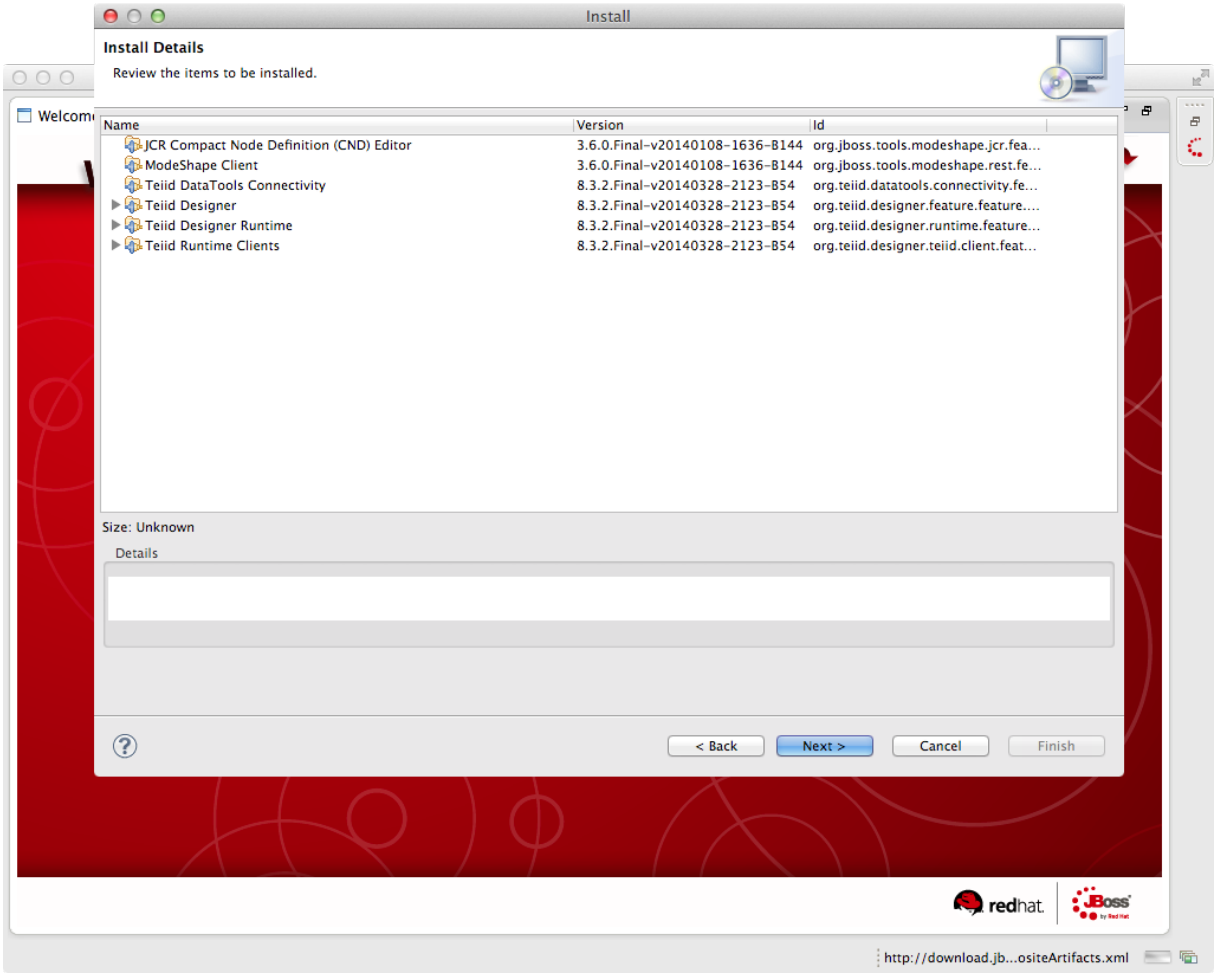

Figure 3.15.

The Security Warning window appears and click the OK button to proceed.

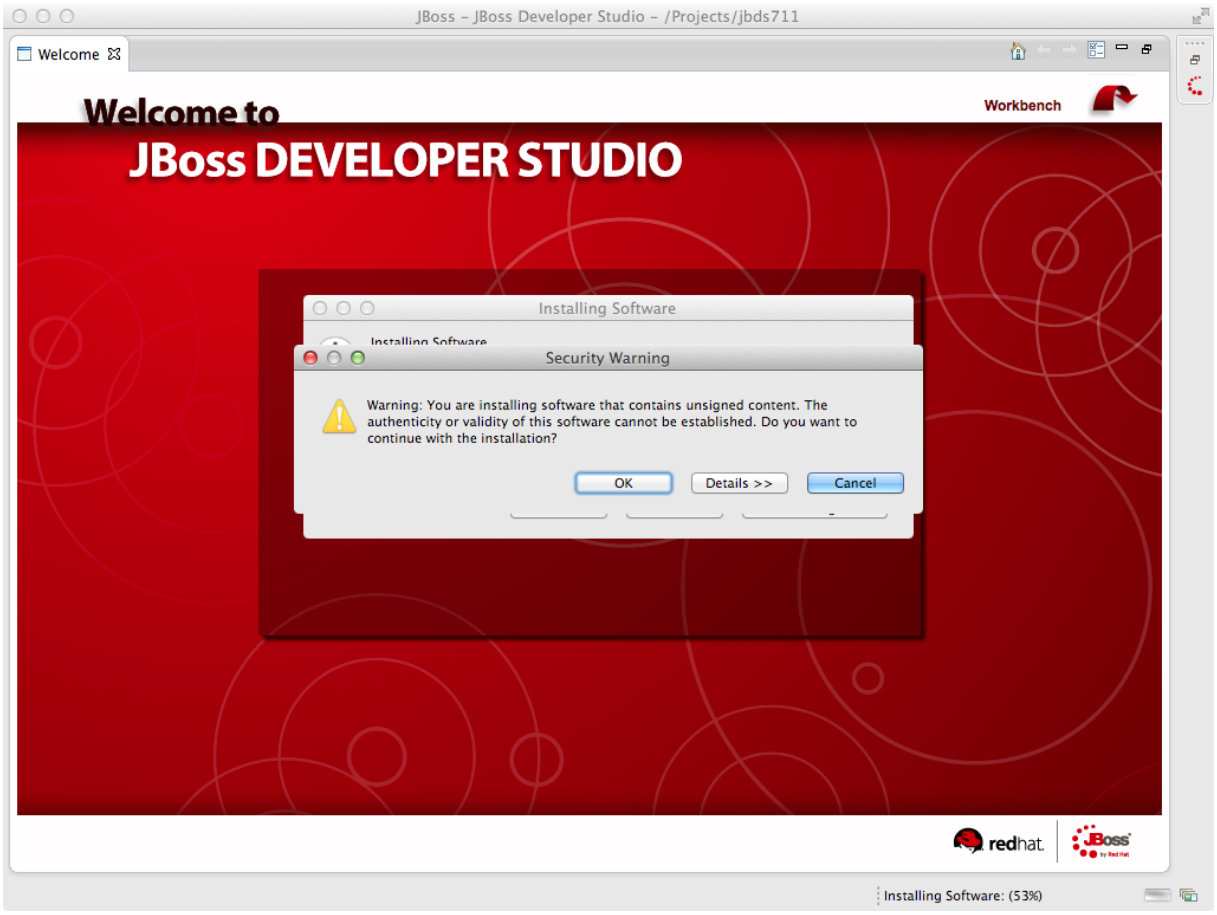

Figure 3.16.

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

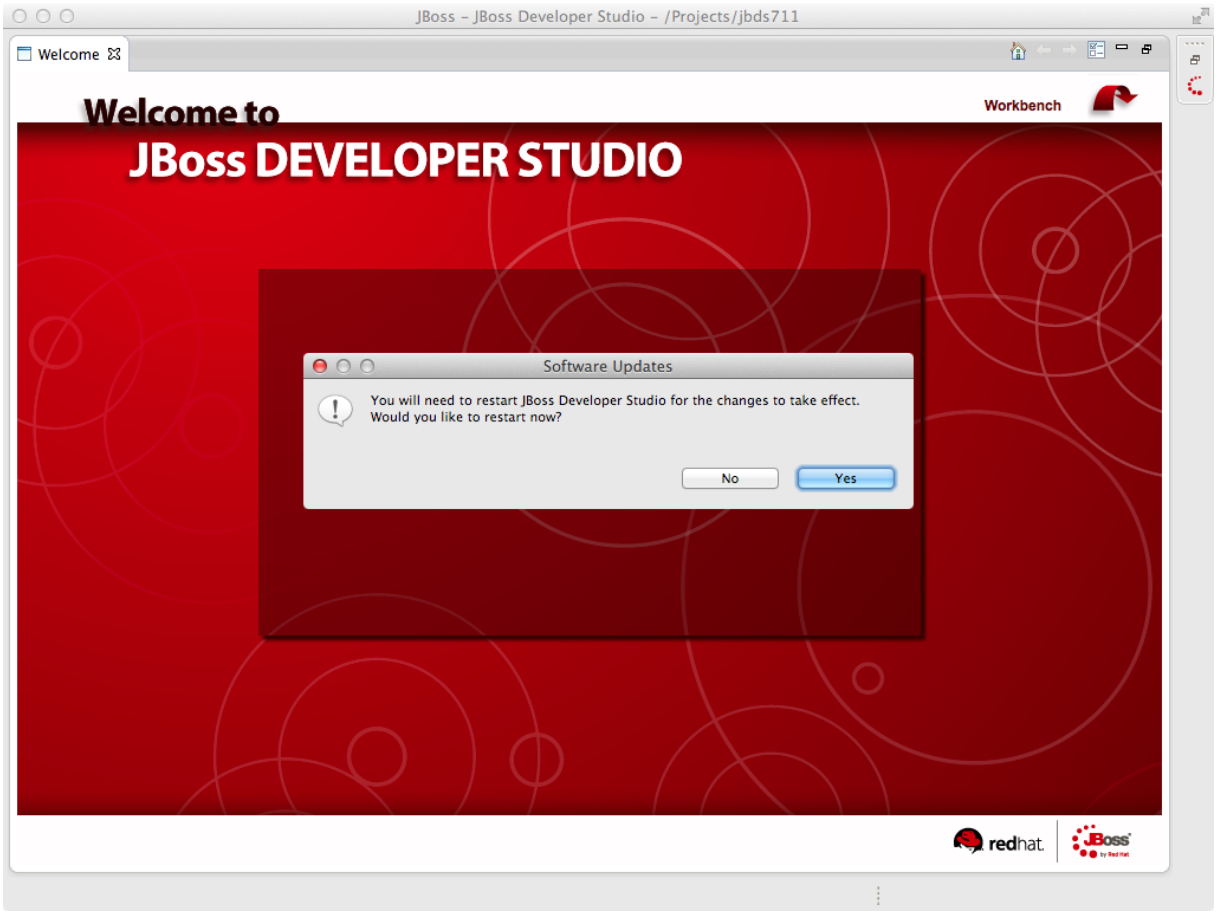

Figure 3.17.

Now that you have JBoss Data Virtualization and JBoss Developer Studio successfully installed, it is time to “hook up” JBoss Developer Studio to the JBoss Data Virtualization server instance.

If the Servers pane is not already visible in JBoss Developer Studio, you can open it by Window → Show View → Other → Server → Servers. The Show View window is presented below.

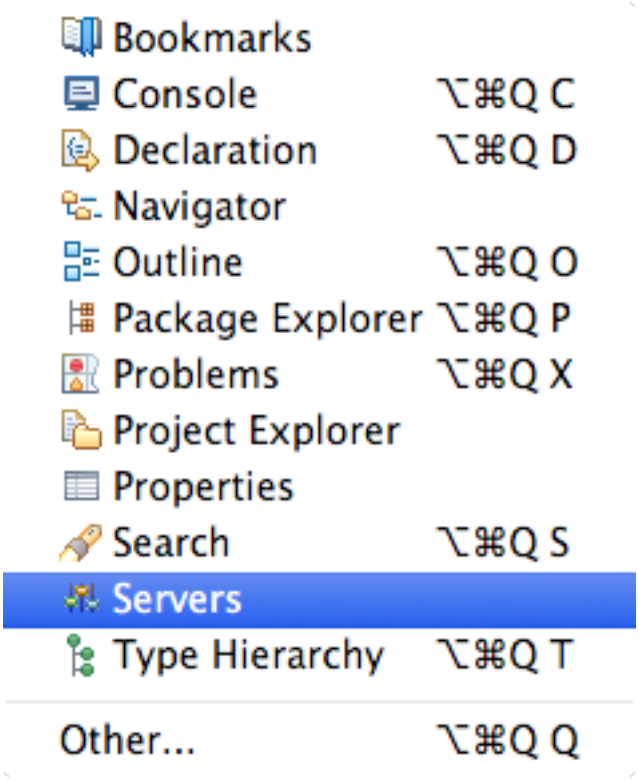

Figure 3.18.

Now, click the OK button. The Servers pane will now be visible in the lower portion of JBoss Developer Studio and is displayed below.

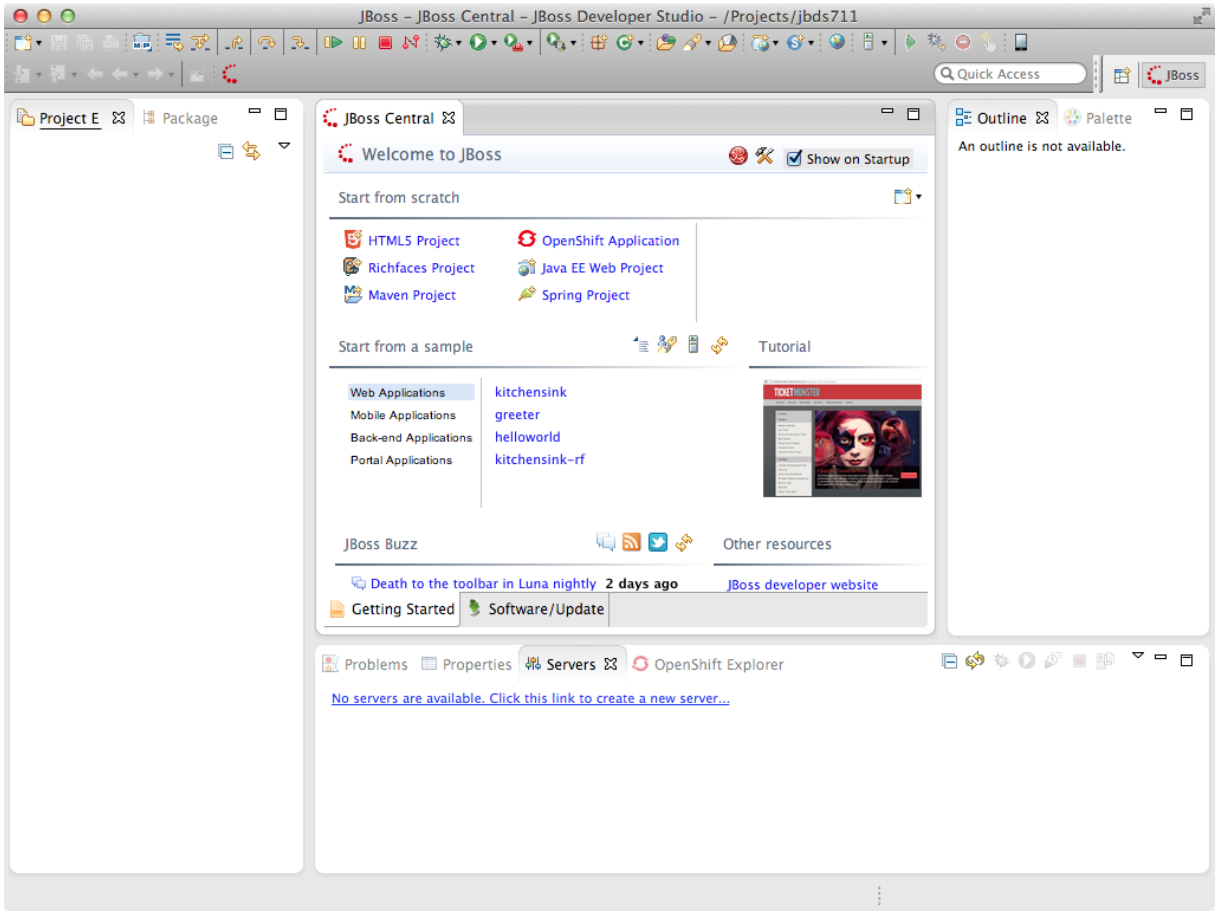

Figure 3.19.

Now, it is a matter of clicking through several screens to add the JBoss Data Virtualization server instance that was installed as part of Lab 1. Click the link “No servers are available. Click this link to create a new server…” and following window will appear:

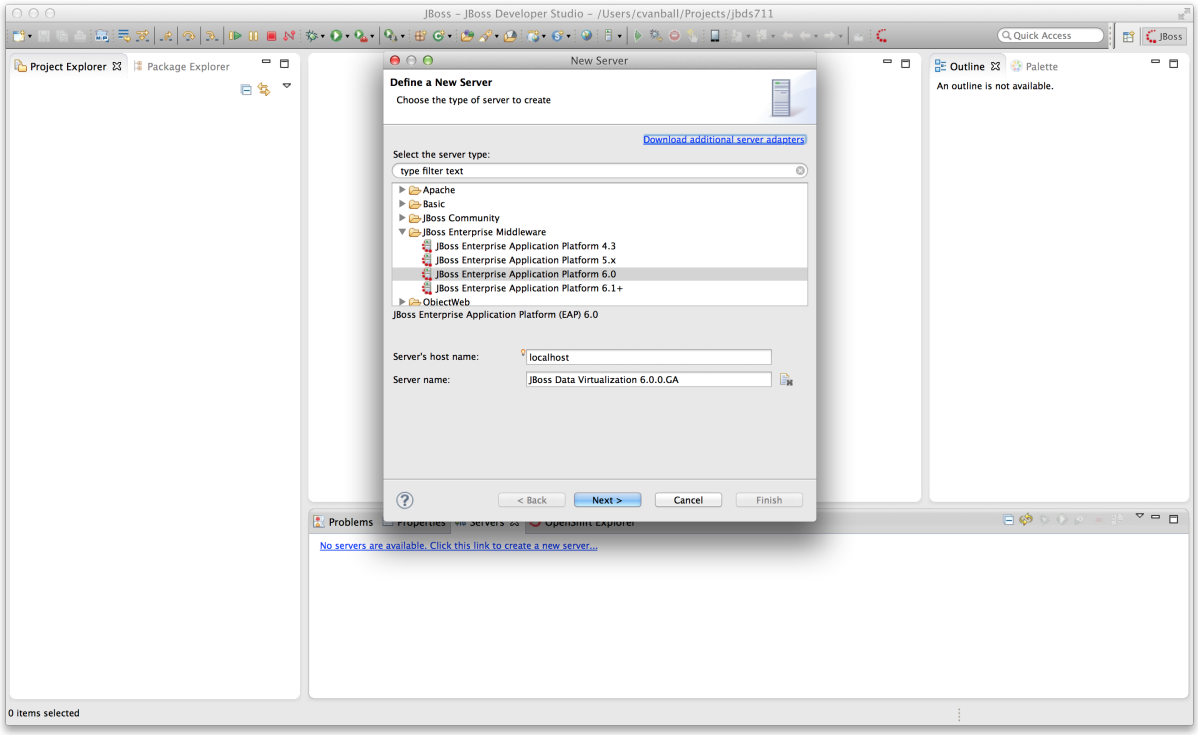

Figure 3.20.

With the New Server wizard enabled, be sure you entries look like those above. Select JBoss Enterprise Middleware → JBoss Enterprise Application Platform 6.1+ as the server type. You can keep the defaults that are selected or enter the values appropriate for your workstation. In this case, “localhost” is entered for Server’s host name and “JBoss EAP 6.1+ Runtime Server” is entered for the Server name. Change the Server name into a meaningful name. Click the Next button. Next, the JBoss Runtime will need to be defined. Essentially, this is selecting the “home” directory for the JBoss Data Virtualization instance that was installed as part of Lab 1. The values to select are illustrated below. Select the “home” directory for the JBoss Data Virtualization instance. This will be <path to installed instance>/jboss-eap-6.1. Once this runtime is selected, the available configurations are available. To keep things simple, select the “default” profile.

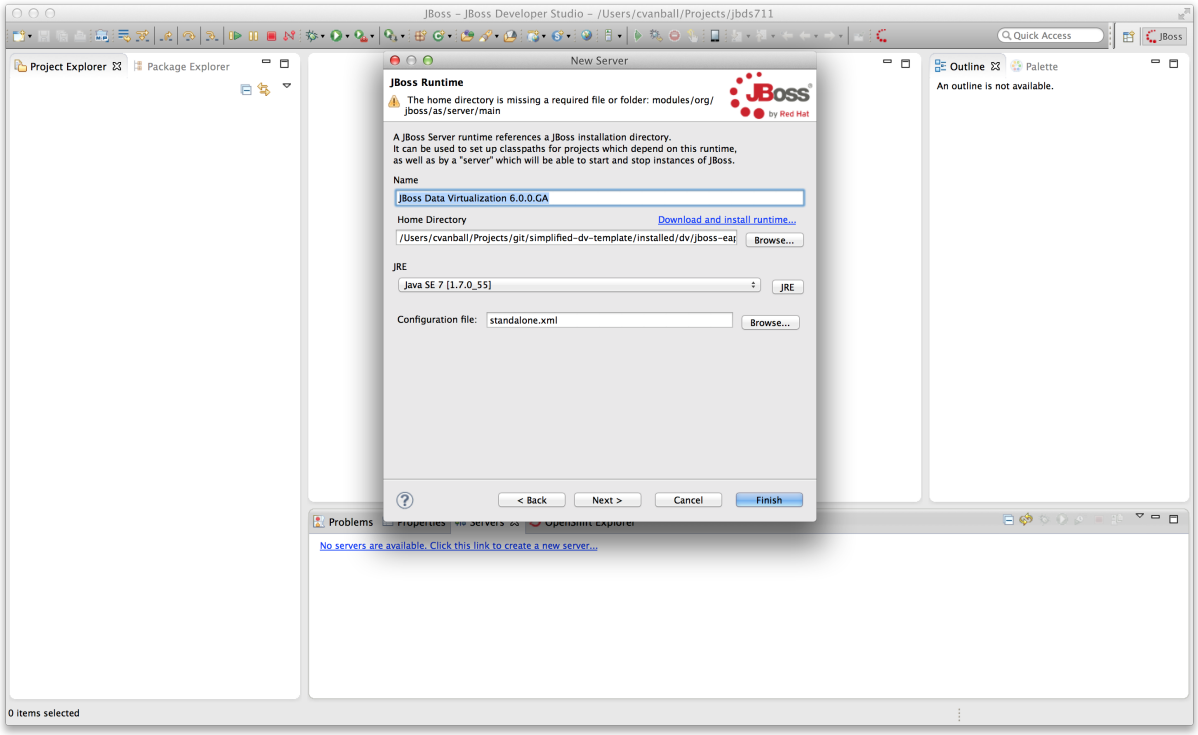

Figure 3.21.

Click Finish.

The Servers pane will now have the available server available as indicated below.

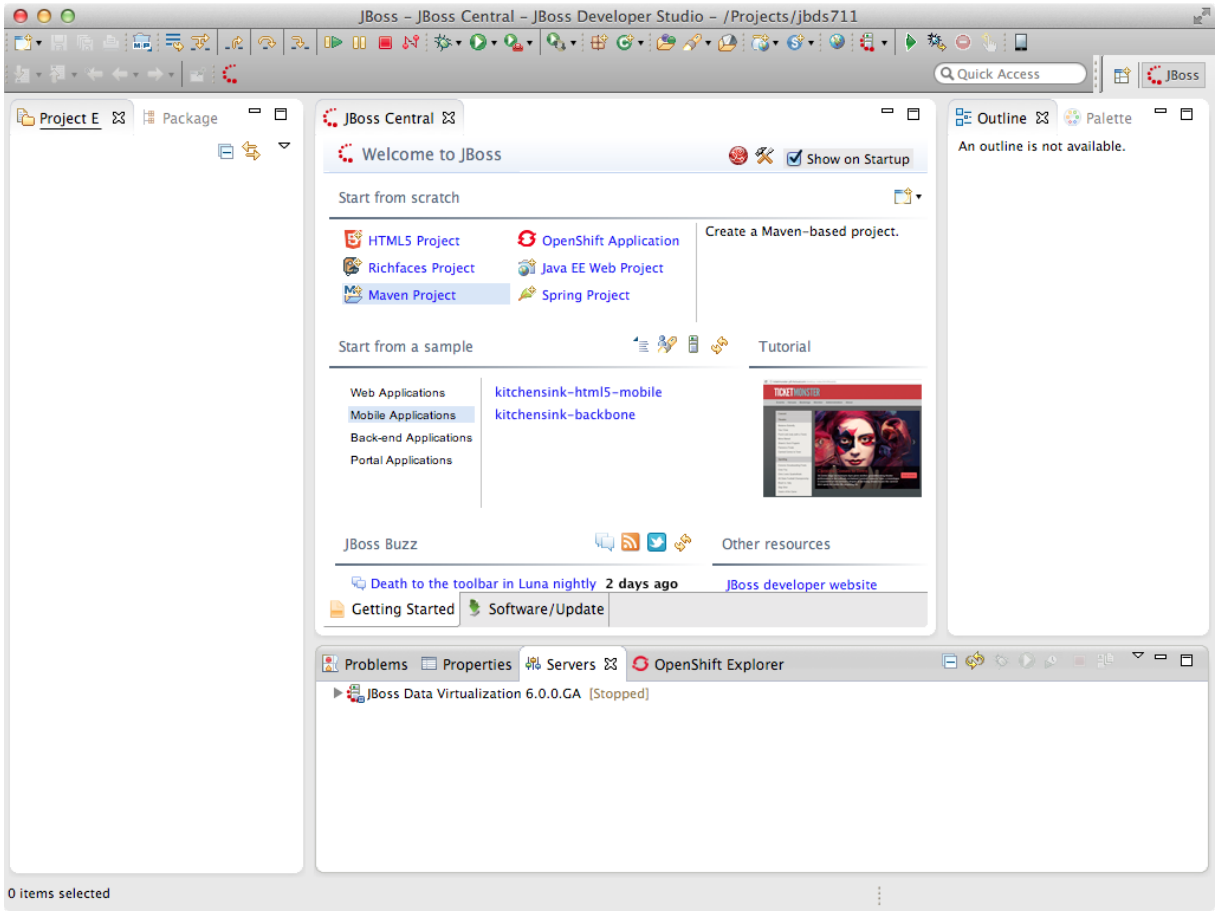

Figure 3.22.

At this point, you can right-click on the server and there is a list of available options. Click Start from the available options and the server will start up.

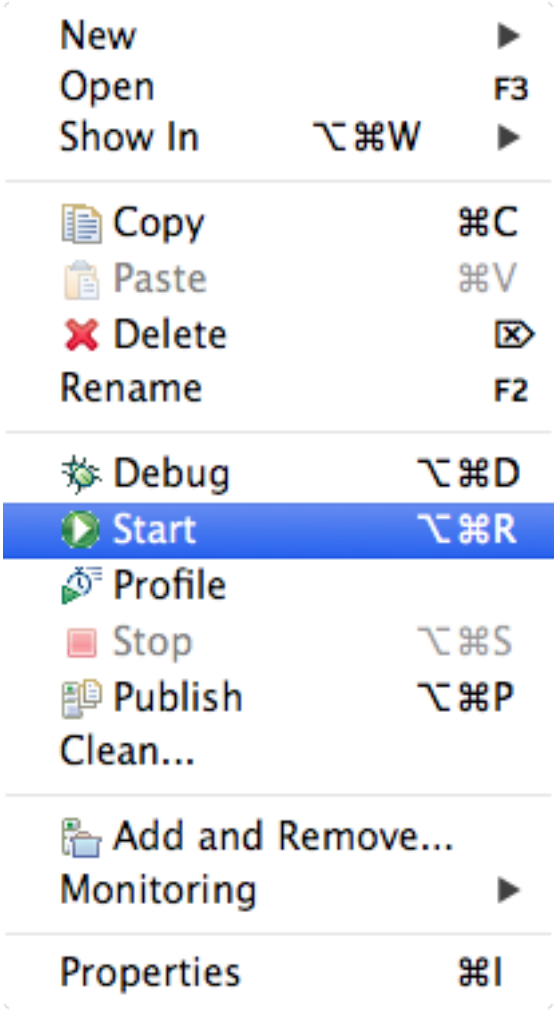
New
Open F3
Show In ⌥⌘W
Copy ⌘C
Paste ⌘V
Delete
Rename F2
Debug ⌥⌘D
Start ⌥⌘R
Profile
Stop ⌥⌘S
Publish ⌥⌘P
Clean...
Add and Remove...
Monitoring
Properties ⌘I

Figure 3.23.

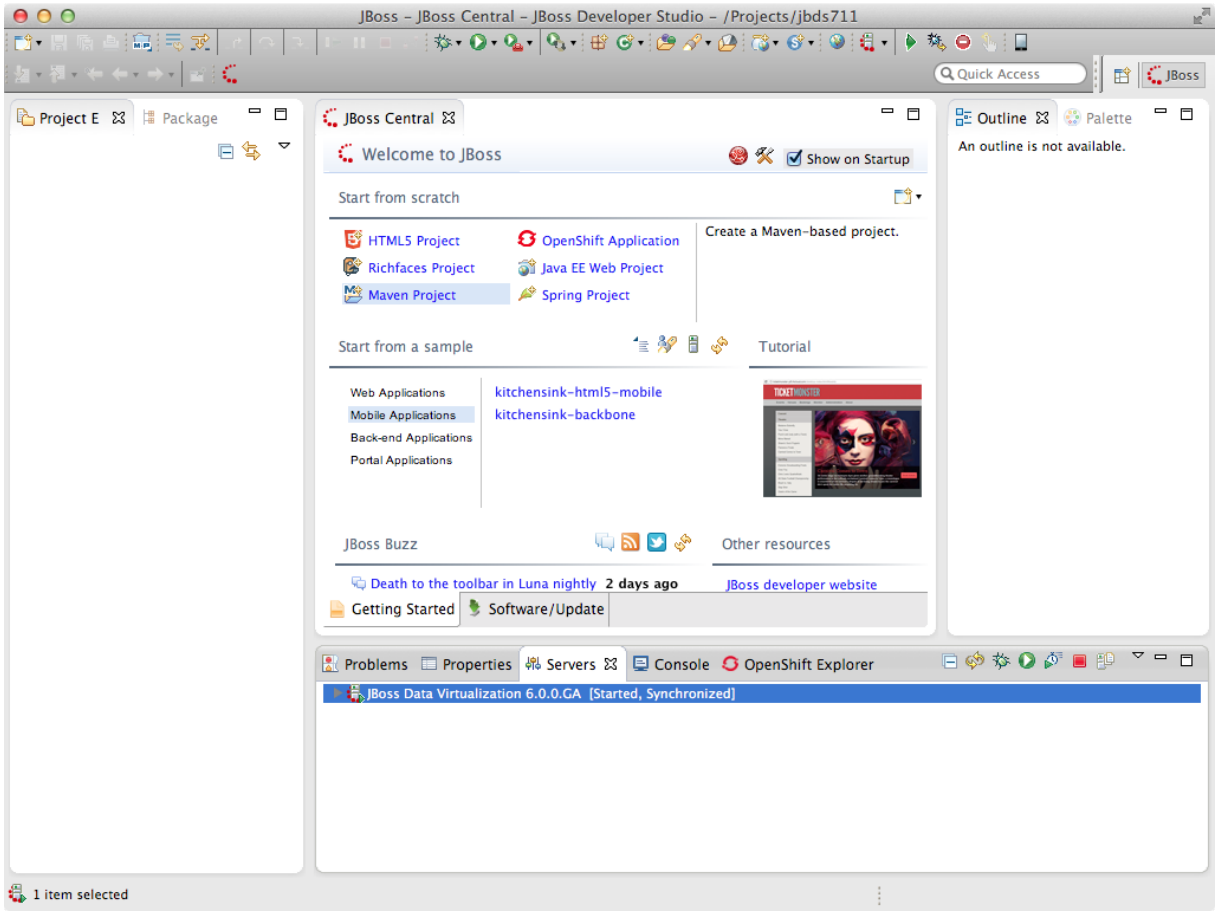

Figure 3.24.

You can now begin creating your own Red Hat JBoss Data Virtualization projects.

Congratulations, you have now completed this lab.

Chapter 4. Create a Teiid project and Import Data Source

Make sure that the previous labs have been completed so that a JBoss Data Virtualization instance is running and JBDS is connected to the JBoss Data Virtualization server.

4.1. Open the Teiid Perspective

To begin this exercise, launch JBDS (if it is not already open), and open the “Teiid Designer” perspective. This is because the JBoss perspective is the default perspective. To open the “Teiid Designer” perspective, first select Window → Open Perspective → Other… in order for the full list of perspectives to be displayed and the “Teiid Designer” perspective to be selectable.

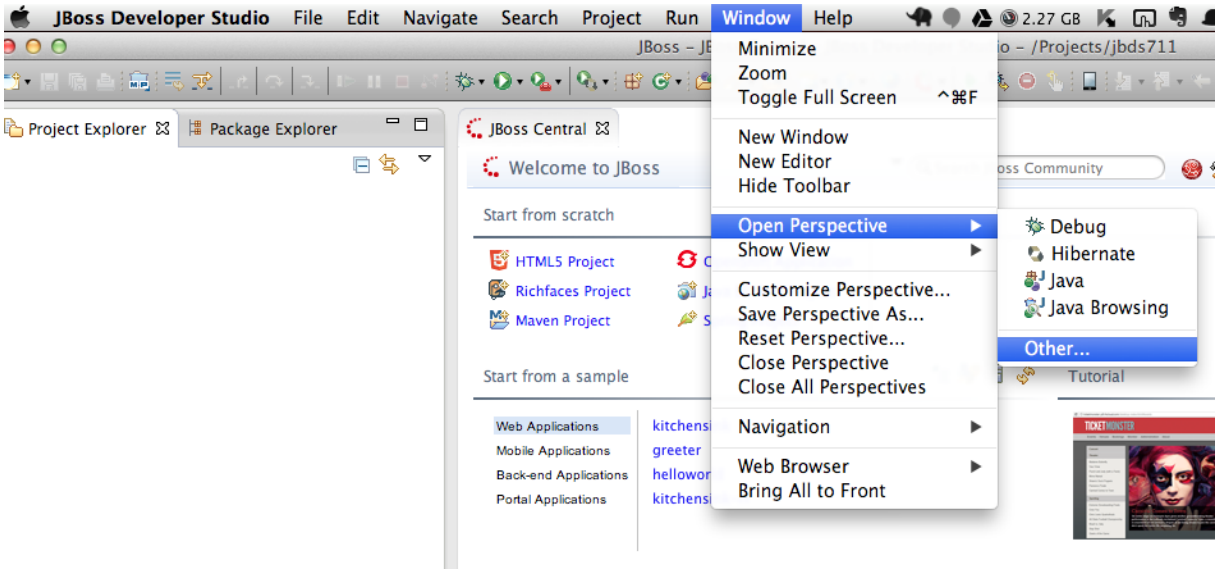

Figure 4.1.

Select Teiid Designer from the perspective list as shown below.

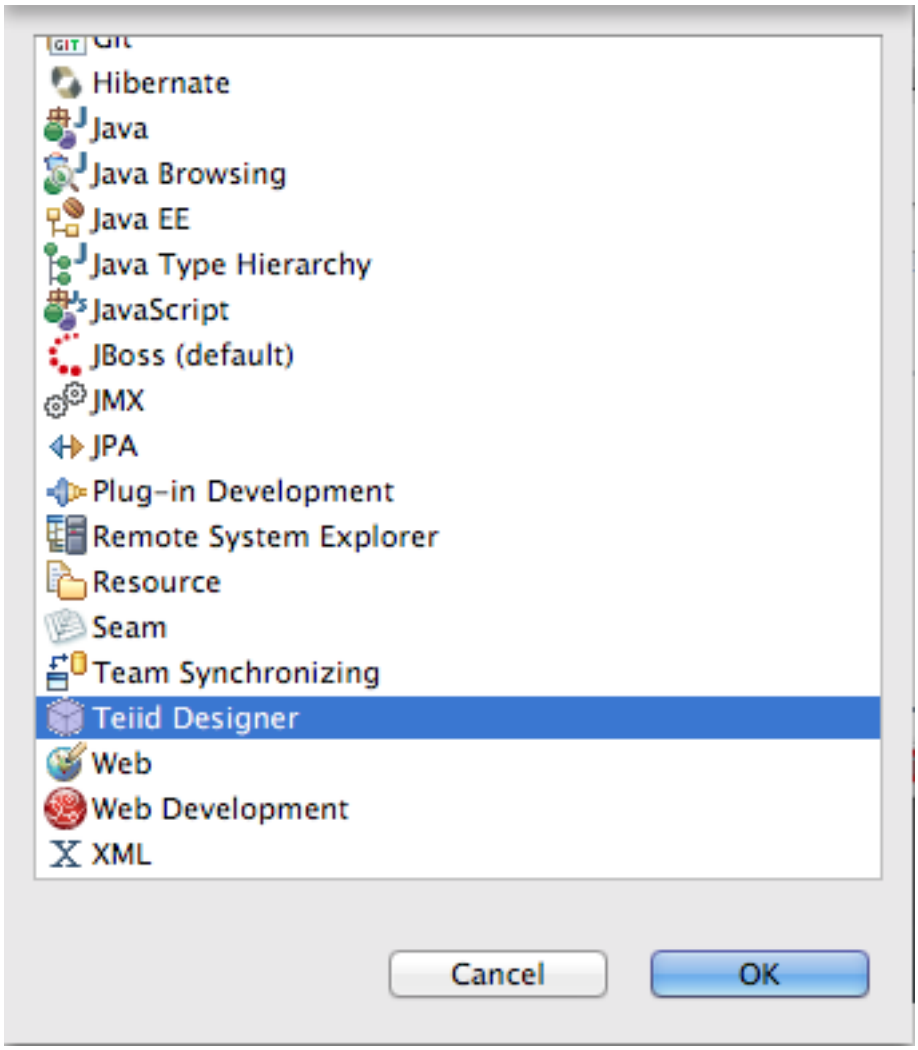

Figure 4.2.

Click OK. This will bring you to a screen that looks like this:

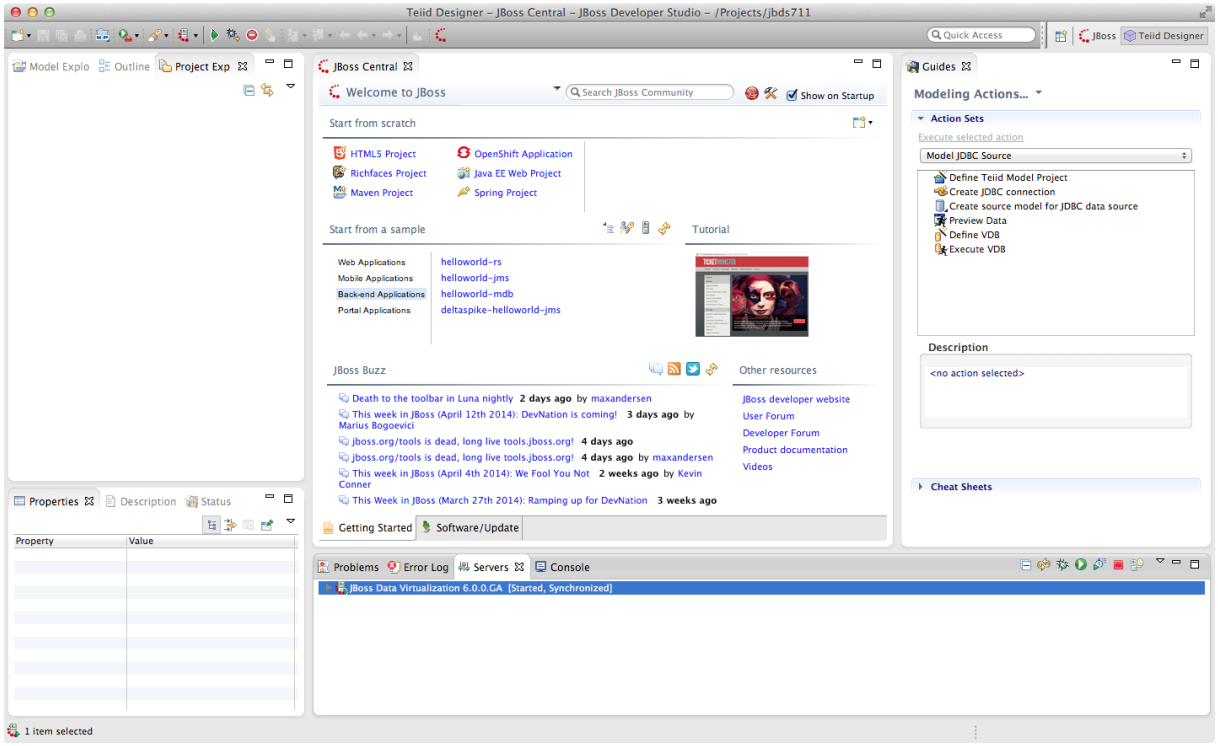

Figure 4.3.

Connecting to a running Server instance is necessary to execute previews of the data services that we will create. In the Teiid Guides window select Teiid. Select the option "Set the Default JBoss / Teiid instance" and double-click on the JBoss Data Virtualization 6.0.0.GA server in the Server pane. For the name simply enter “MyTeiidInstance”. For the Teiid JDBC Connection Info, enter “localhost” for the host and enter user / user for the username / password. Keep the default port number. Also, be sure that the “Save” checkbox is marked. The “SSL” box should not be marked. For the Teiid Admin Connection Info, keep the defaults that are listed and use admin / admin for the username / password combination. Keep the default port number. Both “Save” and “SSL” checkboxes should be checked. When complete, your Teiid Server Connection Information should look like the illustration below. As a “sanity” check, be sure to click the Test button. You should get a "OK" message. If you do not, please raise your hand. If it failed, it may be necessary to cancel and retry the steps again.

The Teiid View along the bottom of JBDS should look like the following illustration.

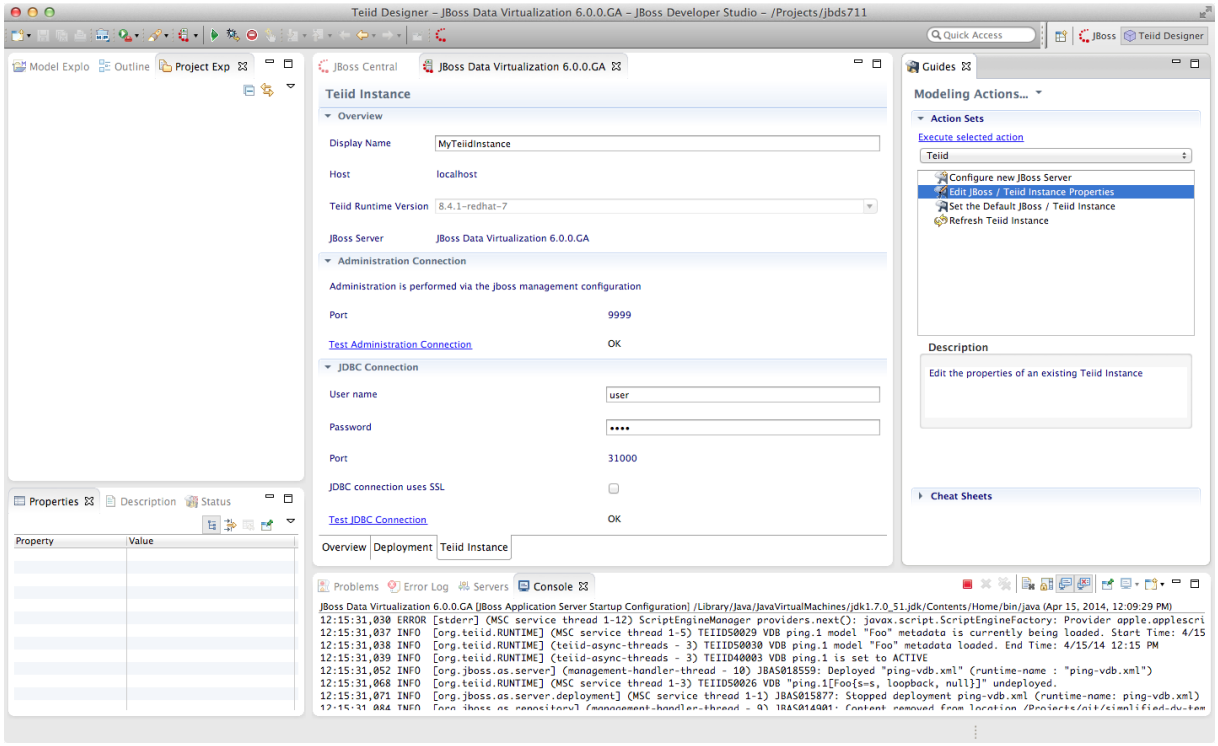

Figure 4.4.

4.2. Creating a Teiid Project

Before you can create models of how your data will be exposed or used, you must first create a project. For the purposes of these labs, we will create a project named Financials. This Financials project will be where we create all of our source and view models and Virtual Database (VDB) files. To create the project, from the menu bar on JBDS, select File → New → Teiid Model Project. The New Model Project wizard will be displayed as below.

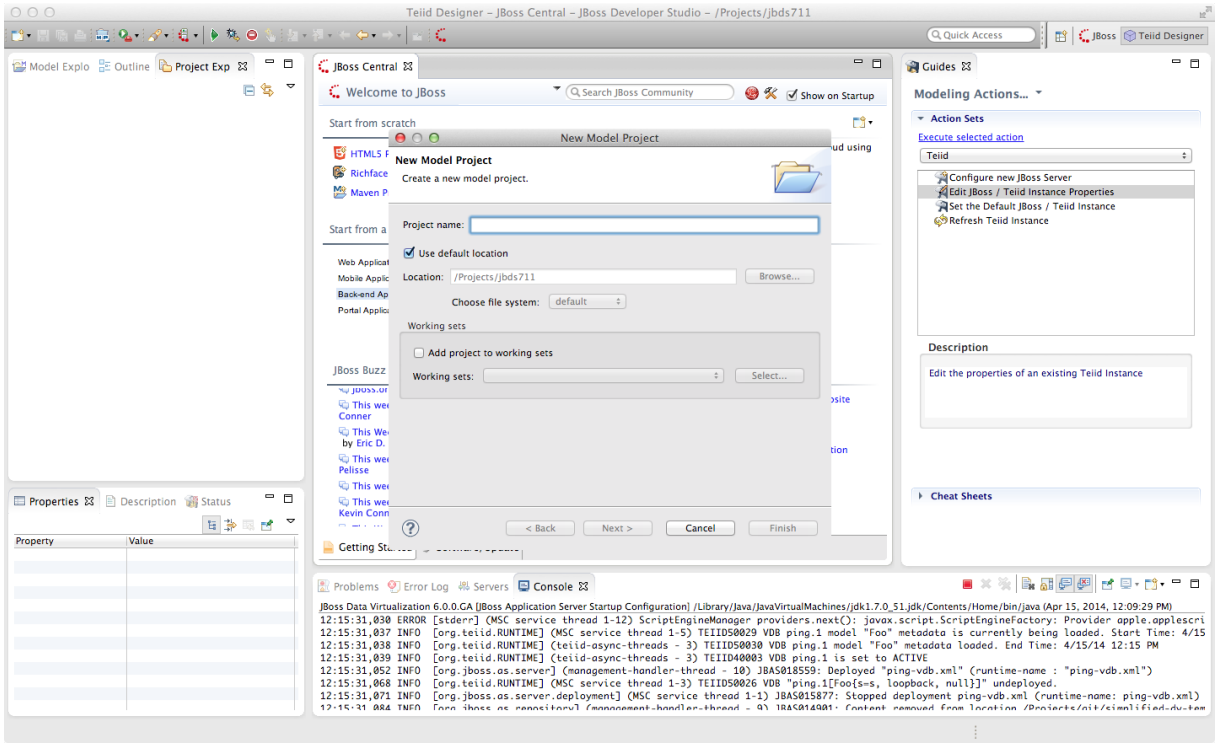

Figure 4.5.

In the project name, enter “Financials” and keep the remaining defaults. Select the Next > button.

After clicking Next, the following window is presented for Project References. Keep the default and click Next > again.

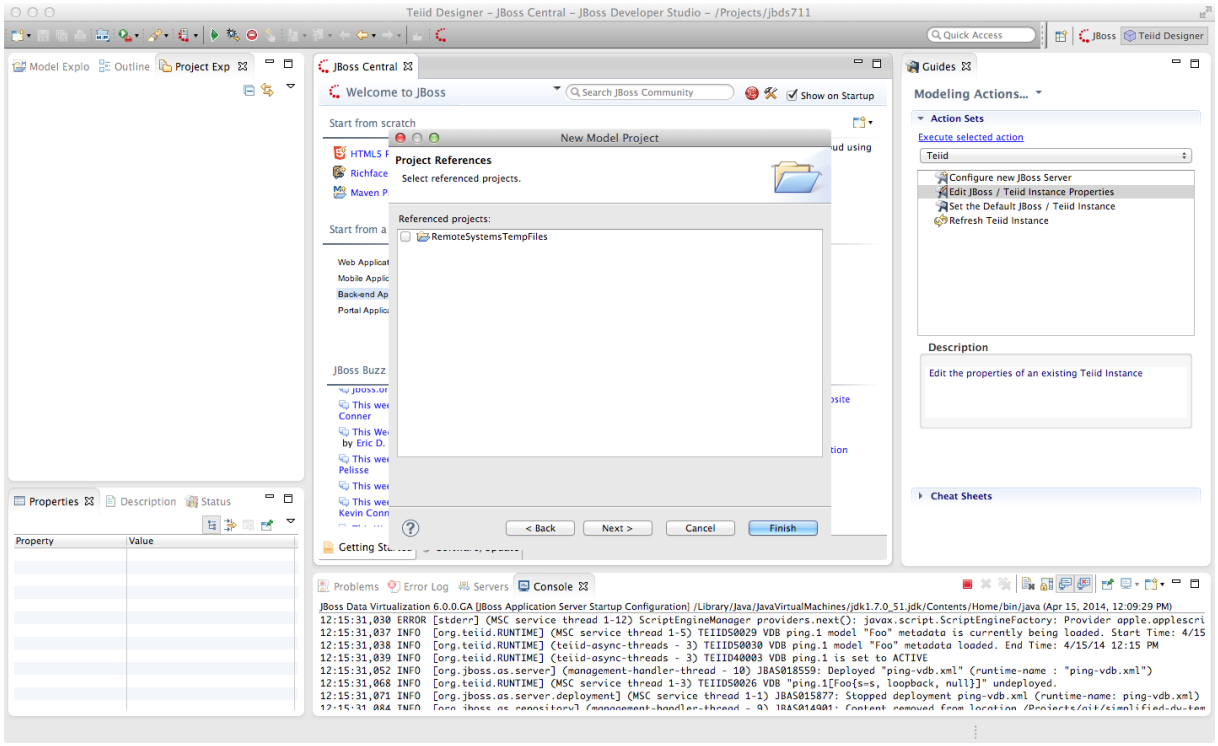

Figure 4.6.

The next window that is presented is the Model Project Options. These are the folders that we will use during the course of building our Financials project. For this lab we will enter DataSources, EnterpriseDataLayer, Schemas, VirtualBaseLayer and WebServices. The Model Project Options window should look like that below.

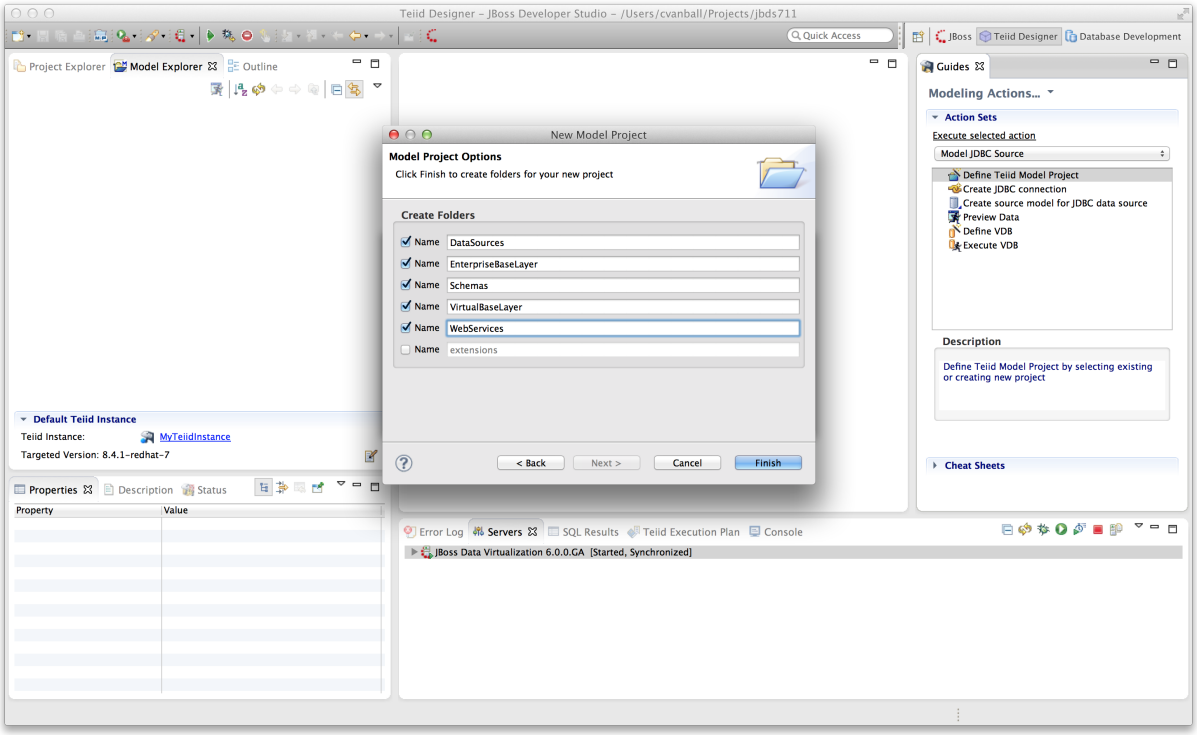

Figure 4.7.

At this point, you can click the Finish button.

After expanding the Financials Project, the JBDS Teiid Designer Perspective should look similar to the figure below below.

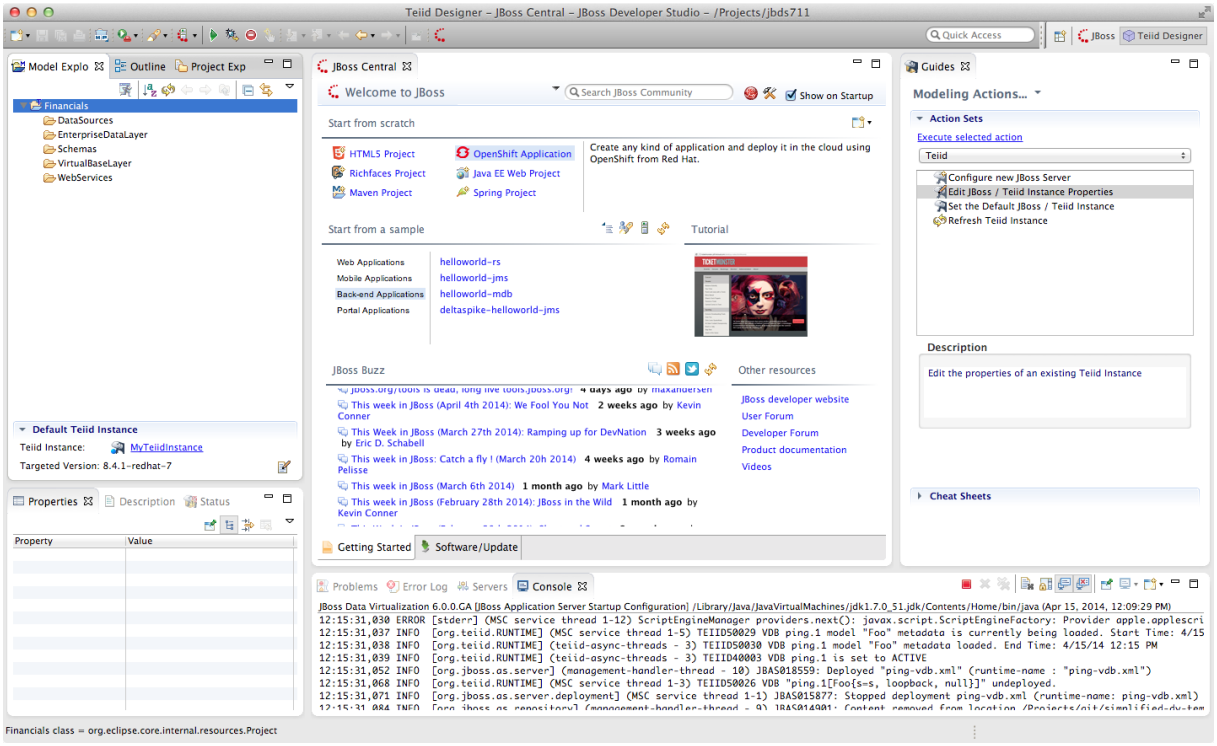

Figure 4.8.

4.3. Creating a Source Model

We must create a source model in order to access physical data or information from a source. The source model (also referred to as the physical model) contains all the metadata necessary for a Virtual Database (VDB) and its associated connectors to access or query data from a target source. There are a few different ways of creating source models. We will first go over the process of creating a source model using the Metadata Import Wizard. The Metadata Import Wizard helps you create new models in the workspace by importing metadata information from a physical enterprise information system or other data source. When you import metadata, the Designer creates a new metadata model for you. Once you have created this metadata model, you can alter it as you would any other. Keep in mind that any changes you make to an imported metadata model do not impact the underlying structure of the enterprise information system the model represents. In some cases you can also use the Metadata Import Wizard to update the information within the models based on changes to the underlying data source. The Teiid Designer comes with a number of plug-ins to import metadata from sources such as JDBC-compliant databases, text files, Salesforce.com, WSDL's, XML Schemas, and DDL files. More information on the Import Wizard (and all of the features in the Teiid Designer) is available in the "Designer Users Guide".

4.4. Importing Metadata from the Product Database

Right-click on the “DataSources” folder and select Import…. In the Import wizard dialog, select the arrow next to “Teiid Designer” to expand the import options. Now, select “JDBC Database >> Source Model” and click Next >.

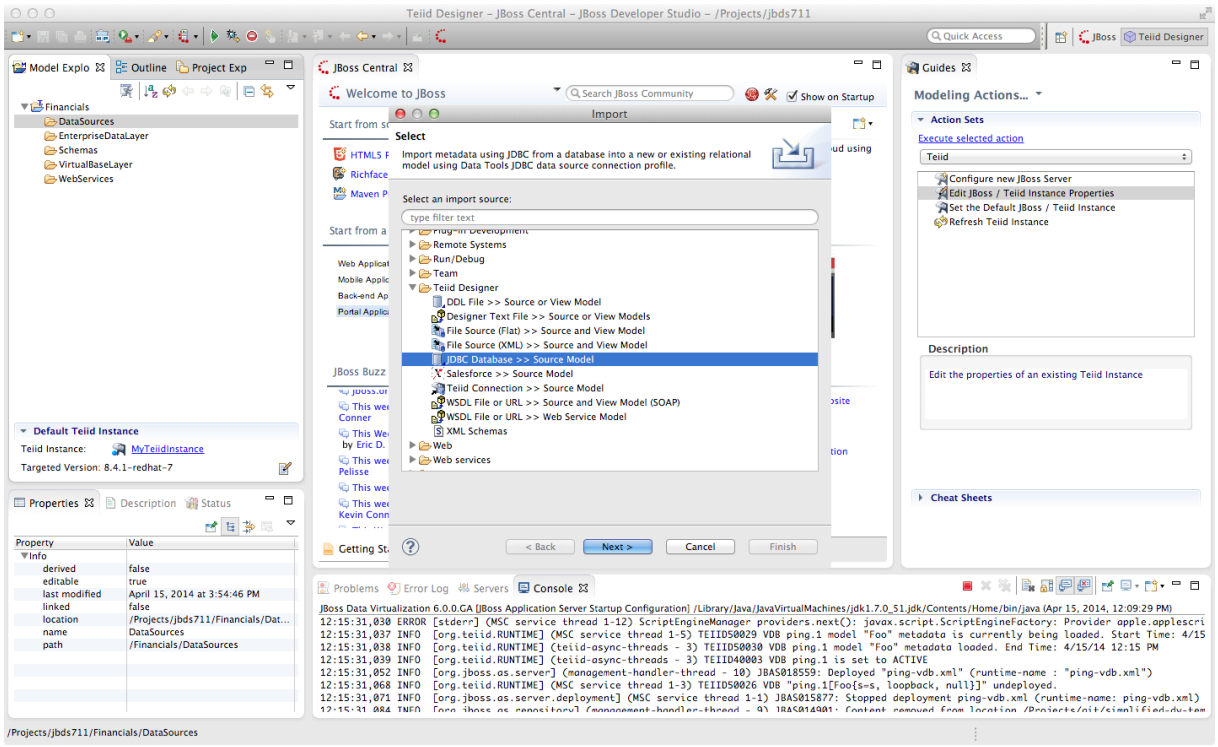

Figure 4.9.

In the “Import JDBC Database >> Source Model” wizard, you will need to select a Connection Profile. If a connection profile does not exist for the database that contains the Product Schema, then select the New… button to create it. The Connection Profile Wizard will come up. Scroll through the list to see the supported databases, then choose “PostgreSQL” for the connection profile type. Enter “Products” for the Name and click Next >.

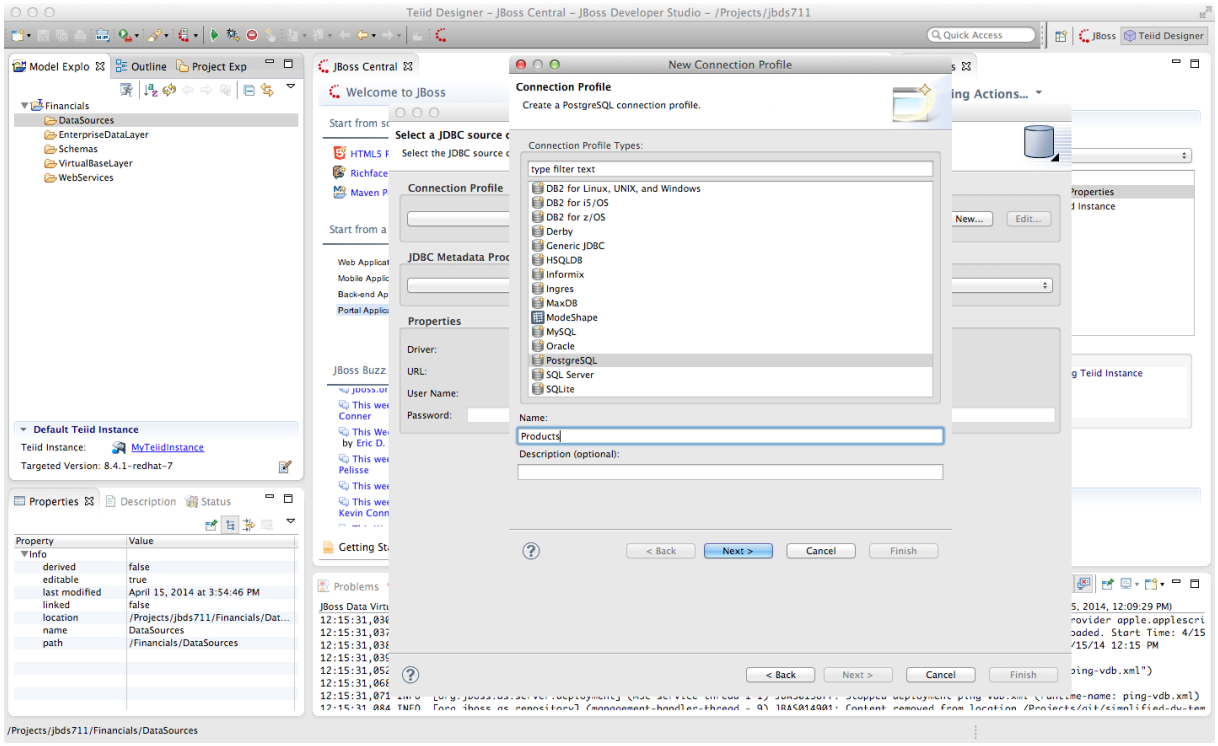

Figure 4.10.

The next step of setting up the connection profile is selecting the driver to use. If the driver you need is not listed in the drop-down list of Drivers (and it should not be if this is your first time through these steps).

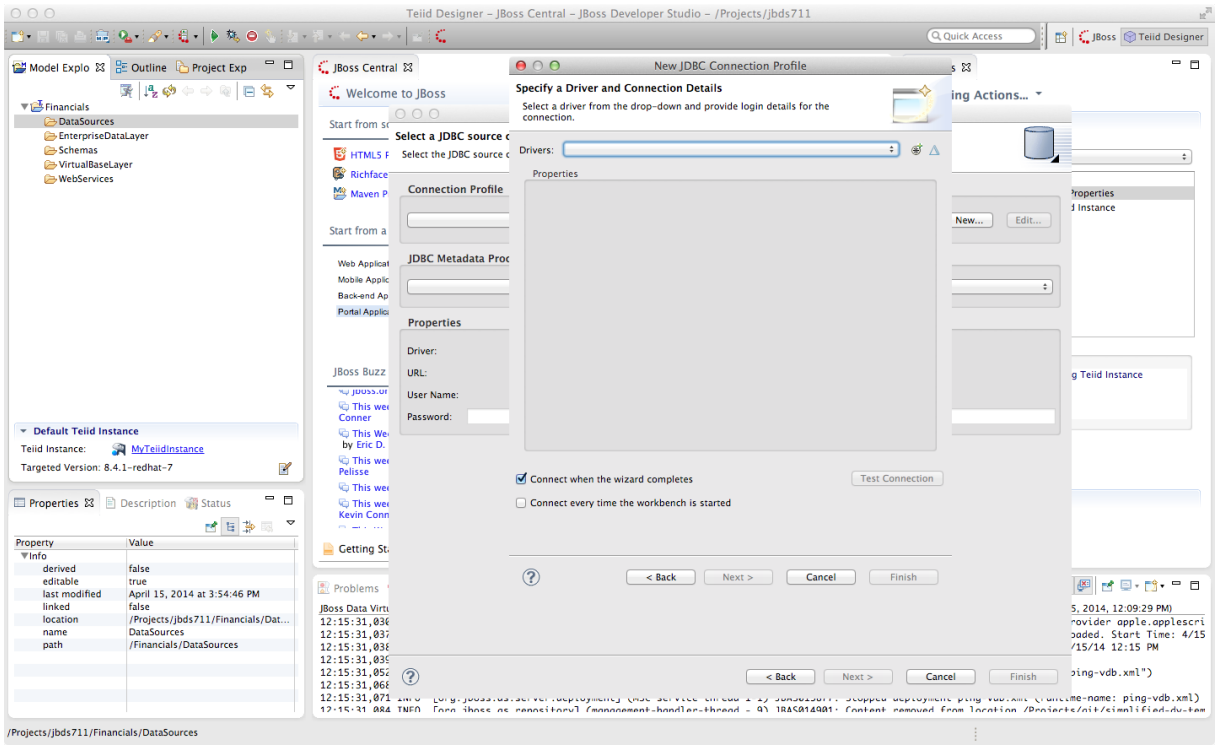

Figure 4.11.

Then select the cross-haired icon “New Driver Definition” which is to the right of Drivers.

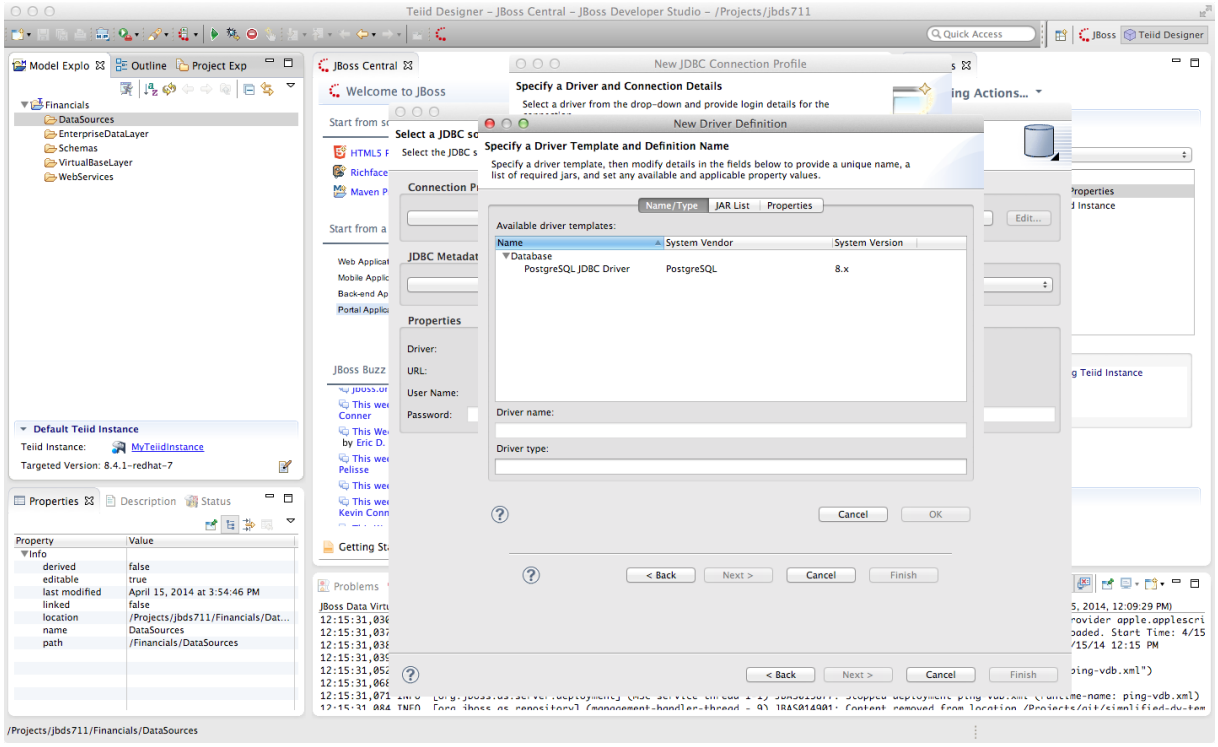

Figure 4.12.

In the “New Driver Definition” dialog, select the PostgreSQL JDBC Driver. It will indicate that the driver JAR is not found. Click on the “JAR List” tab in the New Driver Definition wizard and select the postgresql-8.1-404.jdbc2.jar and click Remove JAR/Zip and then click the Add JAR/Zip button to select the JDBC driver file to use to access the PostgreSQL jar that is part of your student drive. Select the postgresql-9.3-1101.jdbc41.jar file from folder where this file is saved after download. Click OK. After clicking OK, the warning that the JAR file could not be found will go away. Click OK to return to the previous wizard.

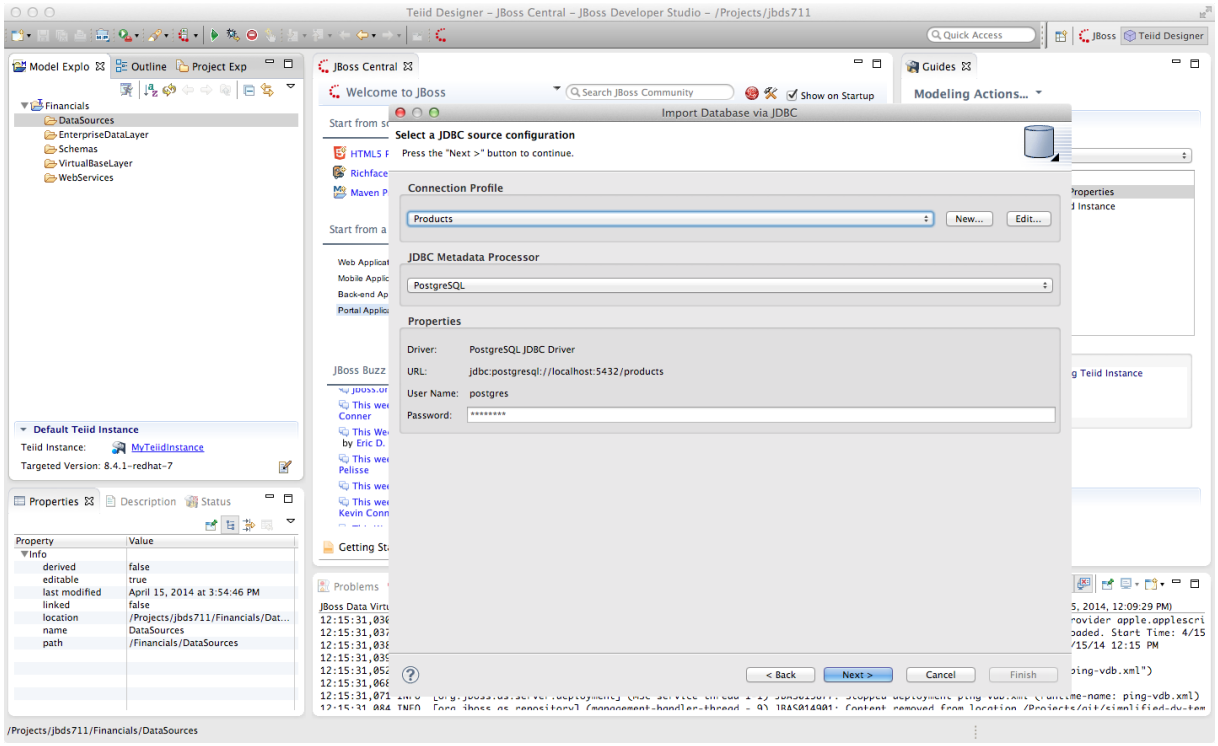

Figure 4.13.

Now that the driver has been selected, you can fill in the database, url, username, and password for the connection profile. Use the following values:

Database	URL	Username	Password
products	jdbc:postgresql://localhost:5432/products	postgres	postgres

Go ahead and check the “Save password” checkbox. Your JDBC Connection Profile wizard should resemble that below. Click Test Connection. A successful ping should return. If it does not, please raise your hand. Click Finish. After clicking “Finish”, your Import Database via JDBC wizard should look like that below. From this point, clicking Next > will take you to the dialog to select the metadata types that will be included when imported. The metadata that will be selected is indicated in the illustration below.

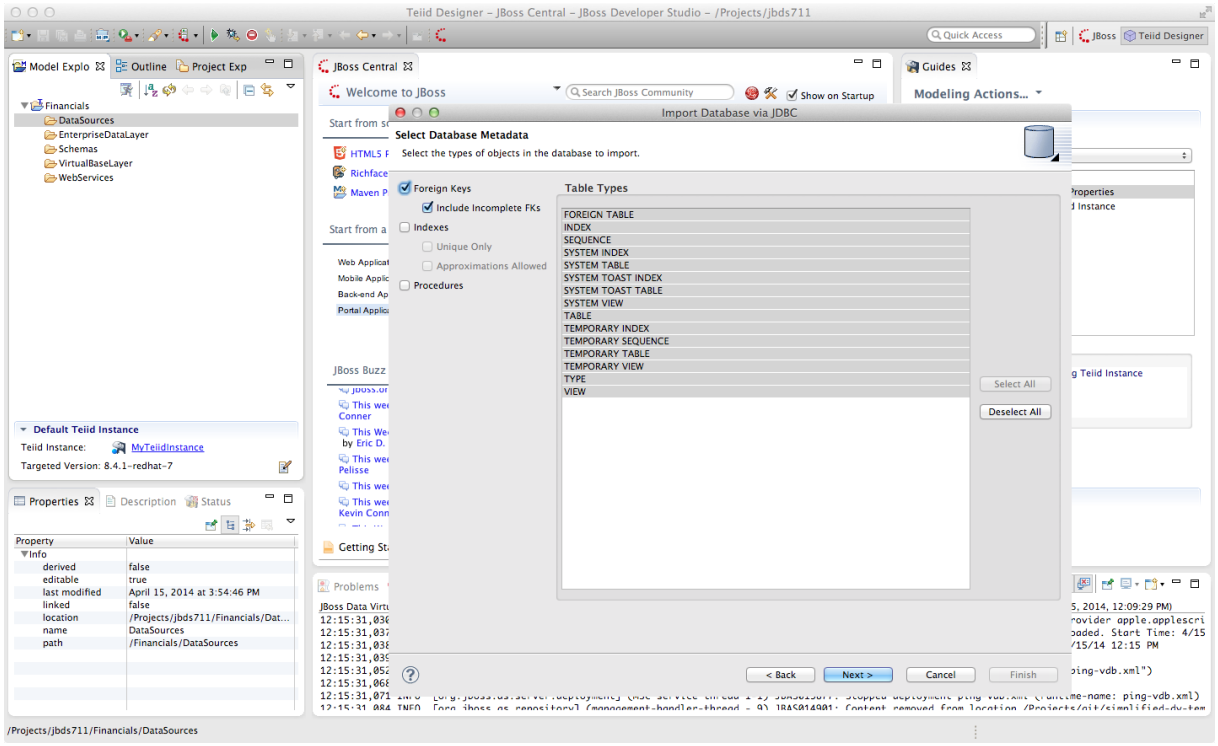

Figure 4.14.

Once your database metadata selections have been made, click the Next button. This will bring up the following dialogue. Be sure to click the arrow icon next to “public” in the Products database. This will select the two tables that we want to import. Specifically, your dialogue for selecting database objects should look like that below.

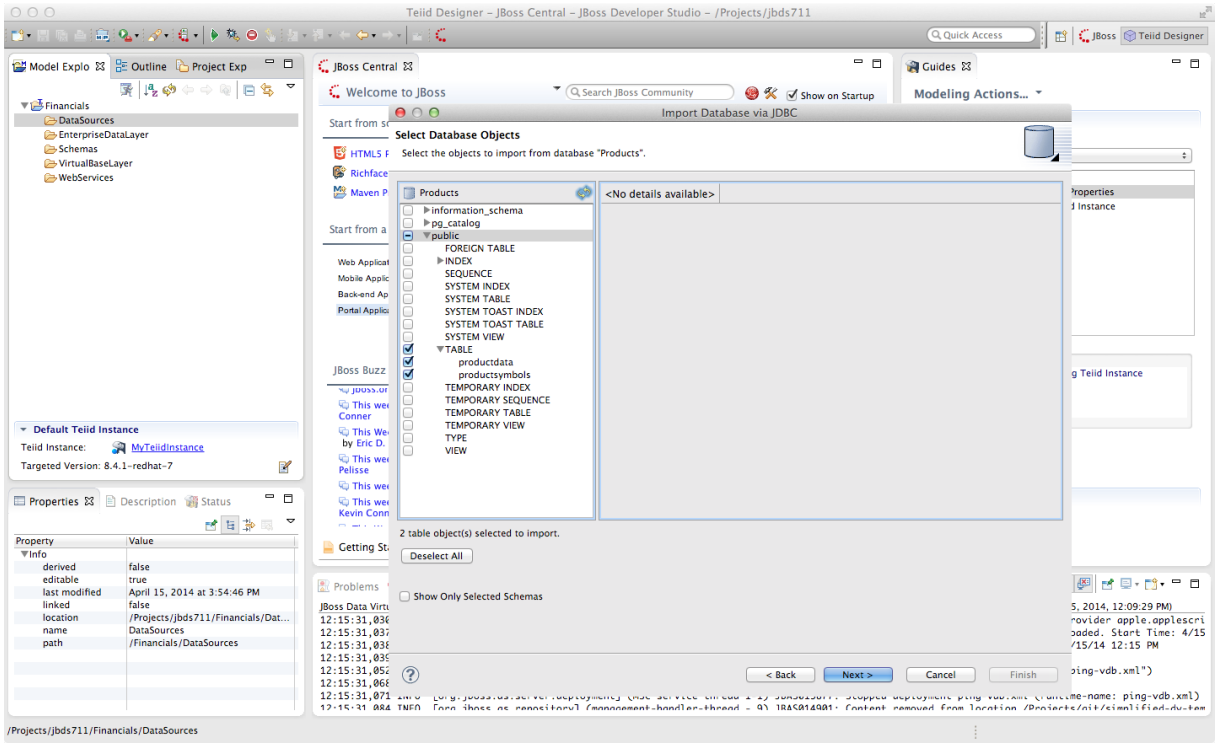

Figure 4.15.

If your dialogue looks like that above, click the Next > button. This will bring up the final screen of the JDBC Import Wizard as indicated below. Notice that there is a requirement to select which folder this model should be created in. To the right of the “Into Folder” attribute, there is a button with “…” on it. Click this button and the following screen will be shown.

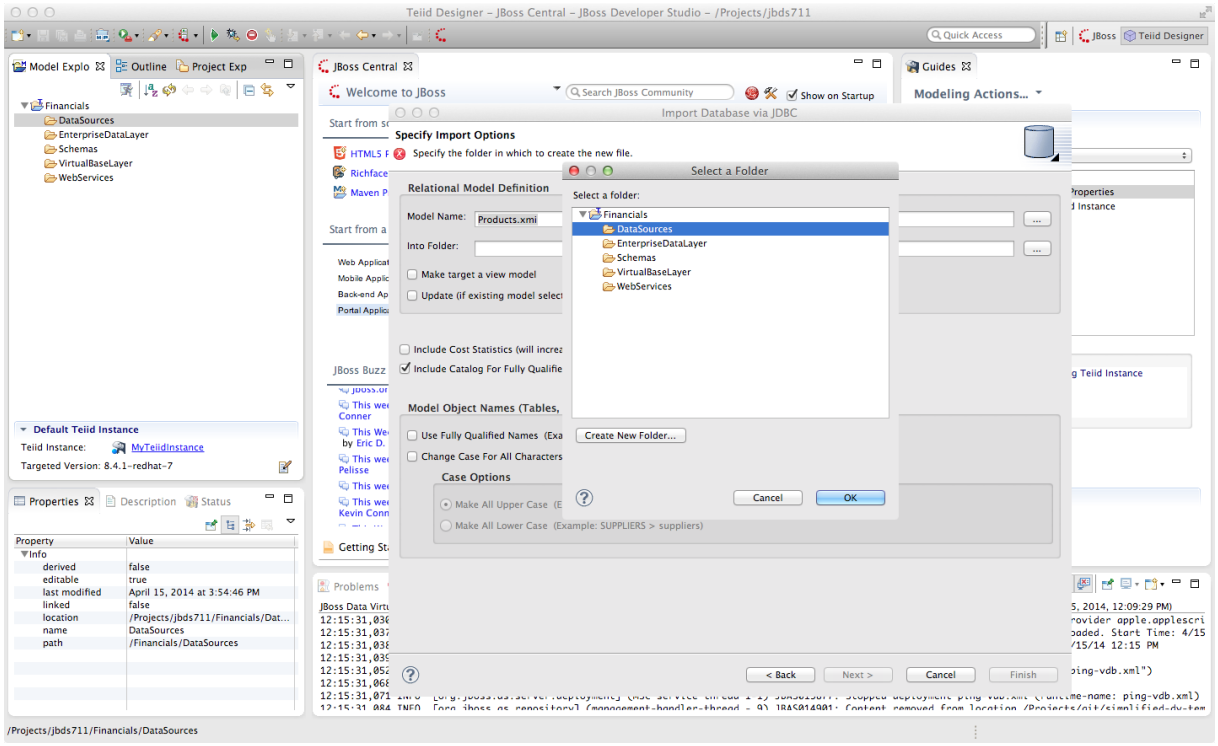

Figure 4.16.

Select the DataSources folder as indicated above. Once the folder has been selected, click the OK button. This will bring us back to the final screen of the Import Database via JDBC wizard. Your screen should look like the one below.

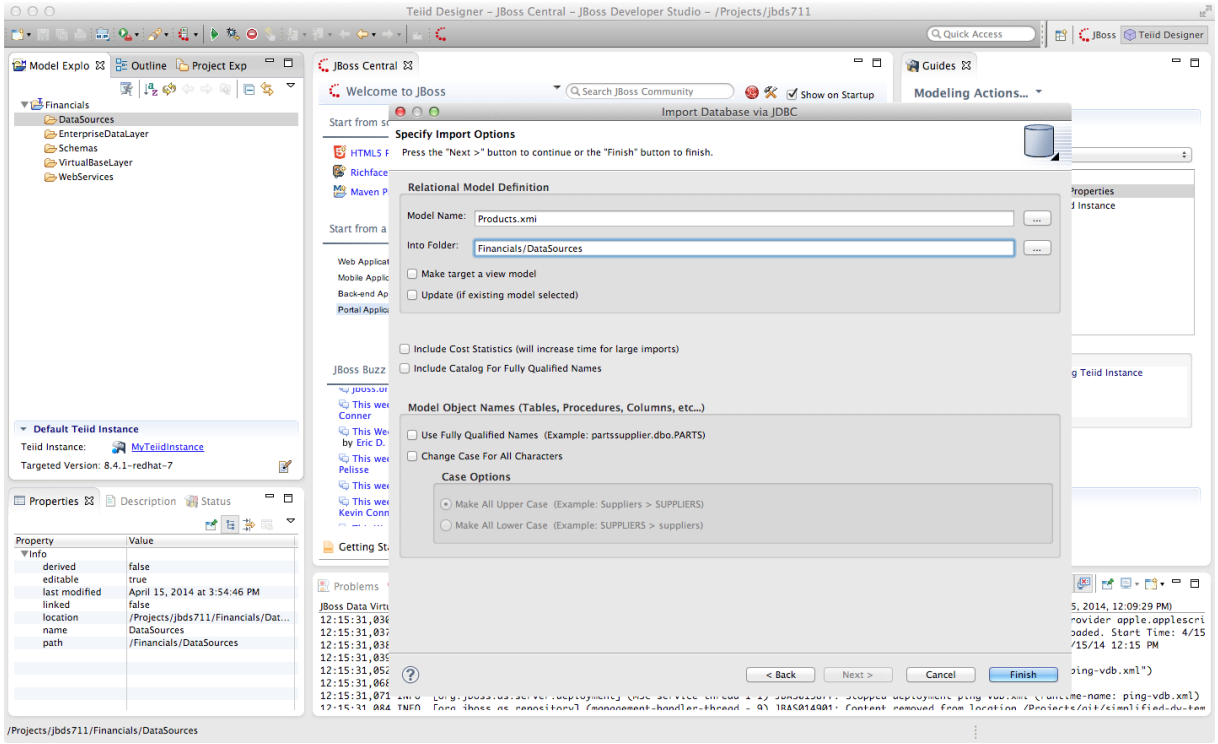

Figure 4.17.

To keep the table names simple, make sure the “Use Fully Qualified Names” checkbox is unchecked. After verifying it matches, click the Finish button. You will now see the Products.xmi source model was opened and its Package Diagram can be seen in the model view area. Click on productdata_pkey (the primary key of the productdata table at the bottom) and note that the Primary Key (productid) in productdata and the Foreign Key in the productssymbols table are highlighted. This is because Teiid Designer knows via the metadata that all of these elements are related.

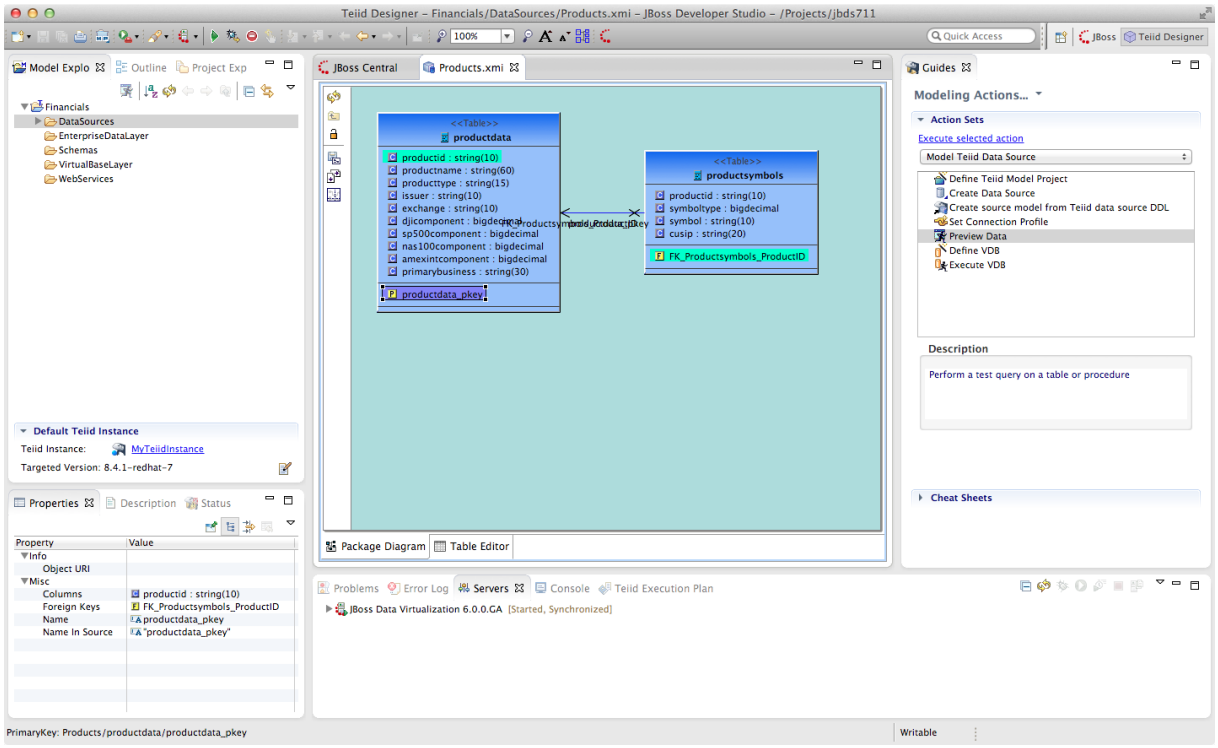

Figure 4.18.

4.5. Preview Data via the Teiid Server

With an active Teiid Server connection, all physical models that have been imported, along with any virtual models that are built on top of them, can be sampled (previewed) with the simple click of a button. To do this, let’s utilize the Modeling Actions palette on the right-hand side of the Designer. Select Model JDBC Source and to Preview data, double-click on the Preview Data action. This will bring up the Preview Data dialogue as indicated below.

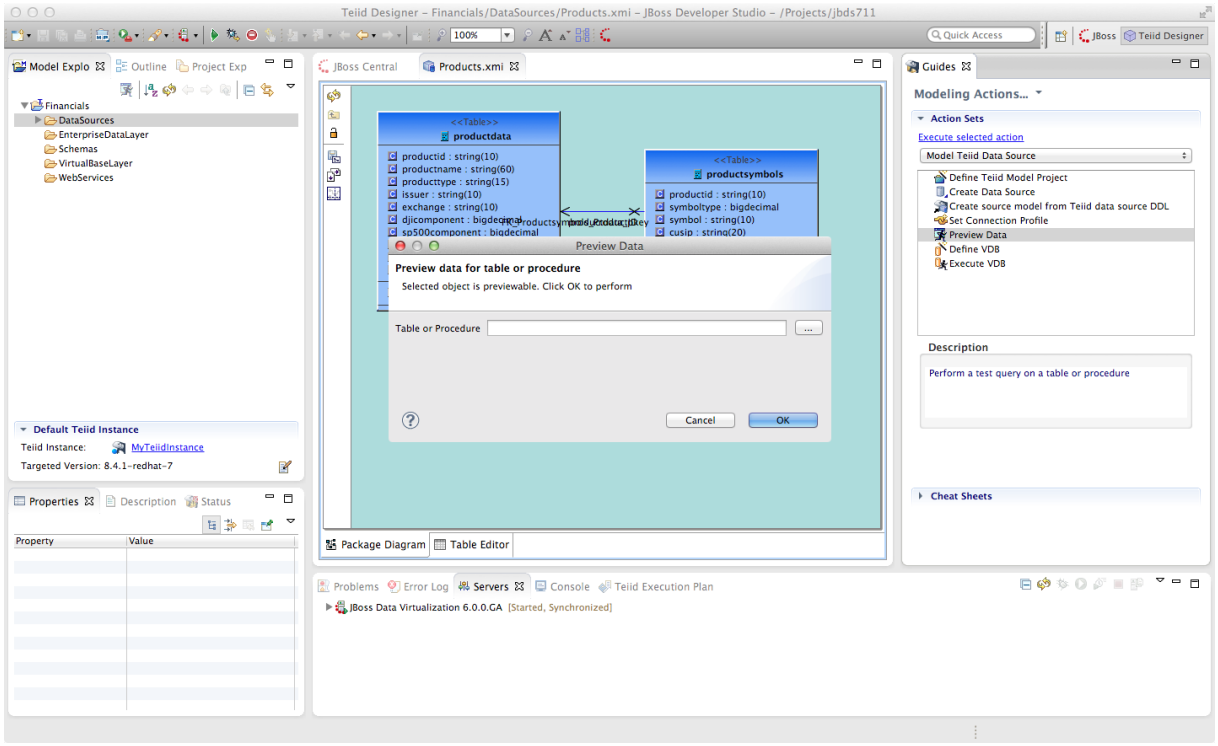

Figure 4.19.

Click the … button to open up a Table or Procedure Selection window. This allows us to drill-down into the tables that we wish to preview data for. For this lab, simply expand Financials, DataSources, and Products.xmi in order to select the productdata table as indicated in the illustration below.

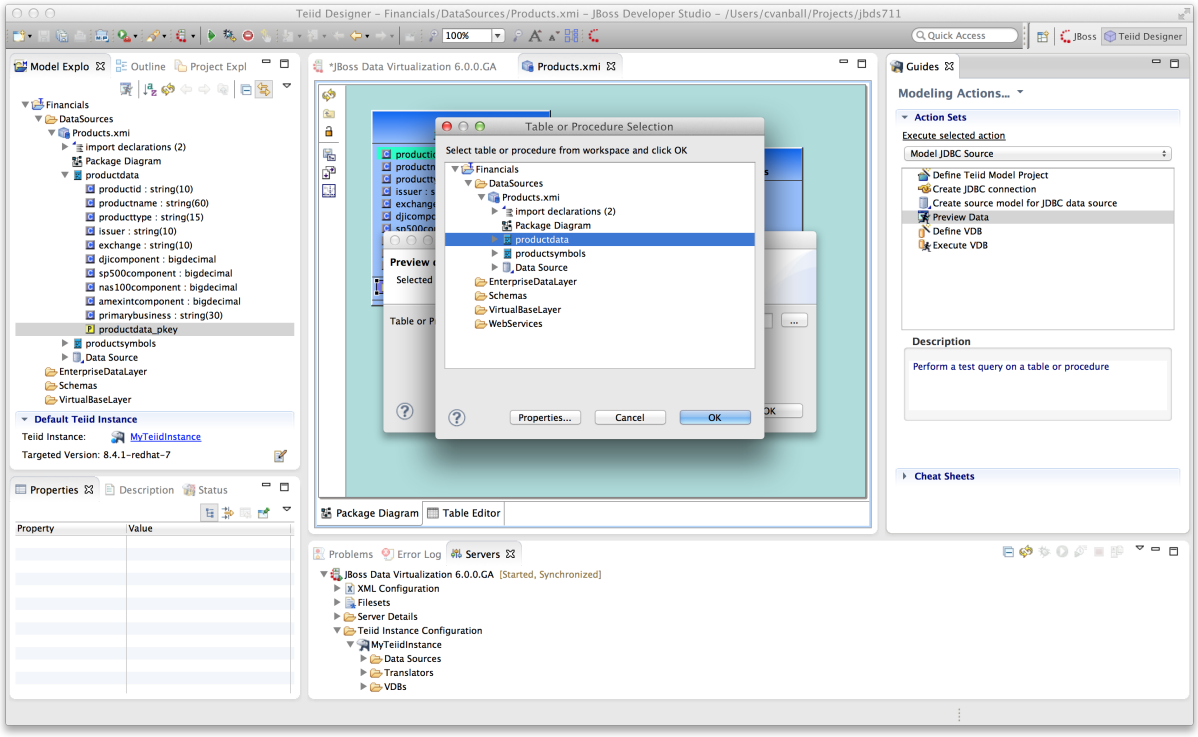

Figure 4.20.

This will bring us back to the Preview Data window where it should look like the one below.

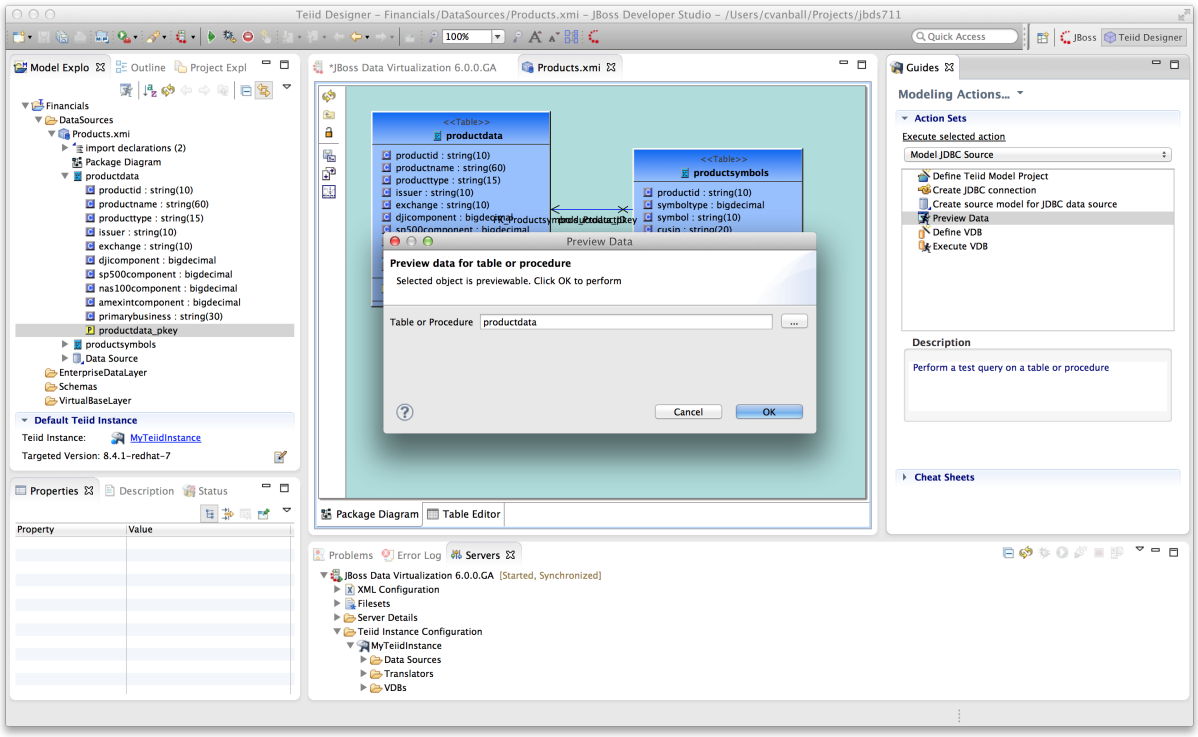

Figure 4.21.

Once you click OK, a pop-up window will indicate that there are some temporary artifacts being deployed to the Teiid Server in order to preview the data. Finally, there will be two additional views that will open along the bottom of JBDS. Specifically, the SQL Results and Teiid Execution Plan tab views. A successful execution will yield sample results as indicated in the illustration below.

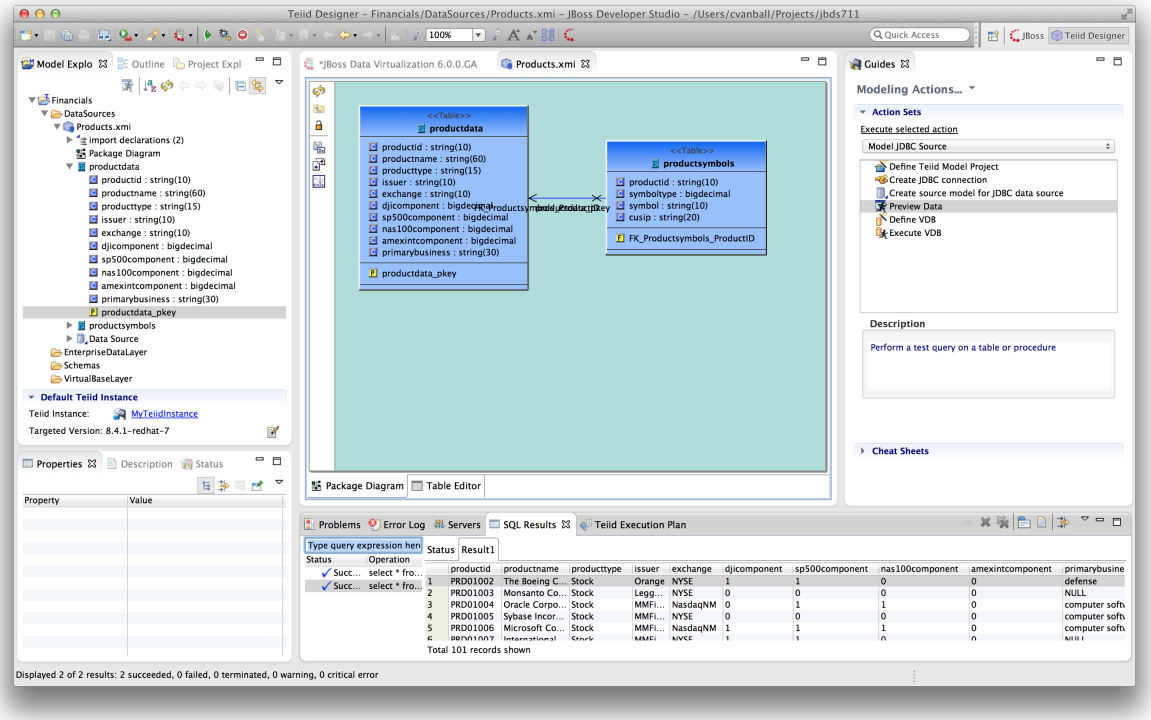

Figure 4.22.

4.6. Import Metadata from the US_Customers and EU_Customers Databases

We will now create source models that represent the US_Customers and EU_Customers schemas from our database. We will again import the metadata using the JDBC Database Import Wizard to create the model. Use the steps from the previous section to import the two schemas. Name the Models US_Customers and EU_Customers and only import the table metadata for the tables account, accountholdings, and customer. The database names for these two sources are uscustomers and eucustomers respectively. The username/password combination is the same as for the product database (postgres / postgres). You will need to create a new Connection Profile for each source but you can reuse the PostgreSQL JDBC driver that was previously referenced. Additionally, feel free to preview data for these two additional data sources using the steps that were outlined above. When you have

completed the imports, the Package Diagram and Model Explorer for US_Customers, for example, will look similar to the following illustration.

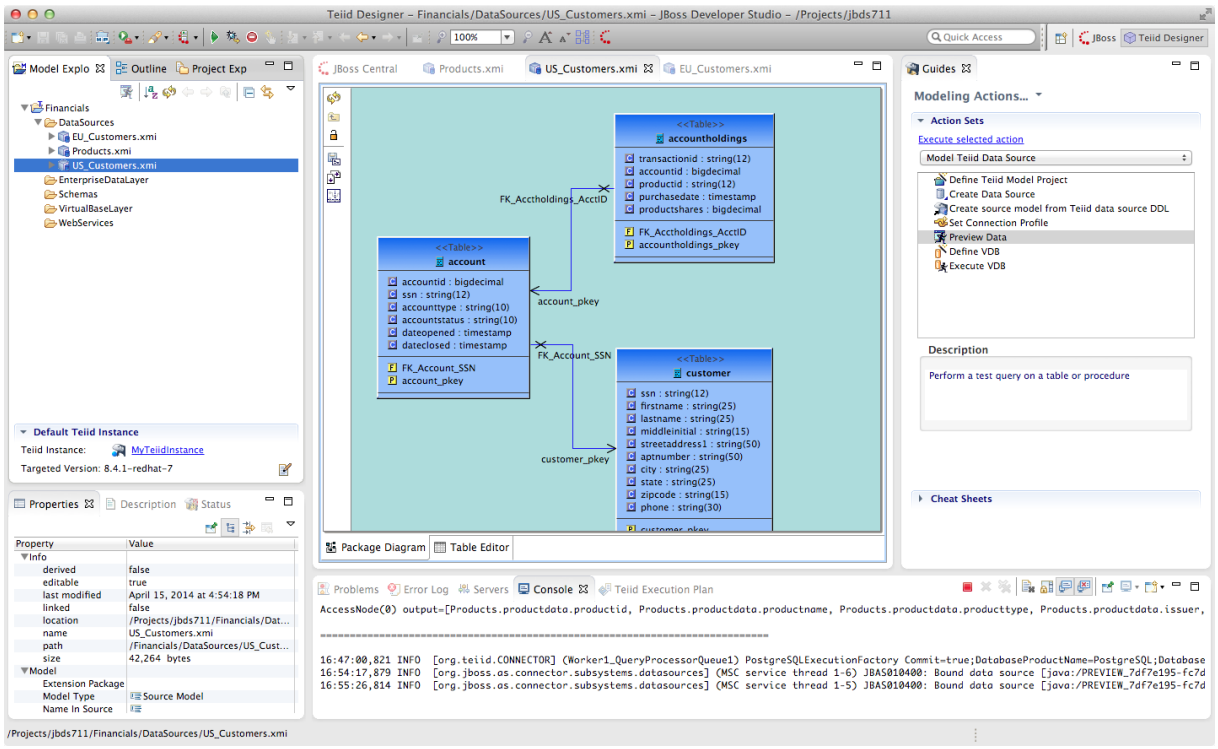

Figure 4.23.

4.7. Import Metadata from a flat file

So far we have been connecting to relational databases and their tables. You can connect to other types of data sources within the perspective. In this section we will connect to a flat file in a CSV (Comma Separated Values) format which contains market information as seen in the image below:

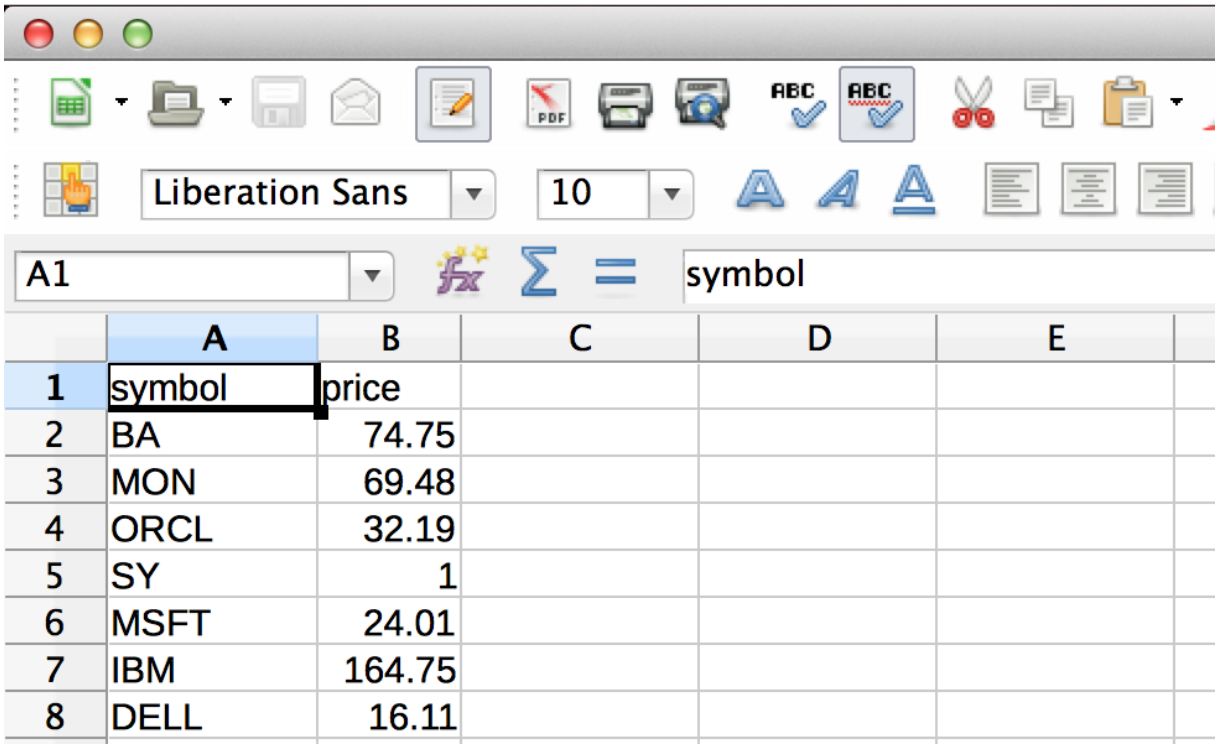

Figure 4.24.

As with the steps above you will need to right-click on the DataSources folder and select import. This time you will select the File Source (Flat) >> Source and View Model to import a data source.

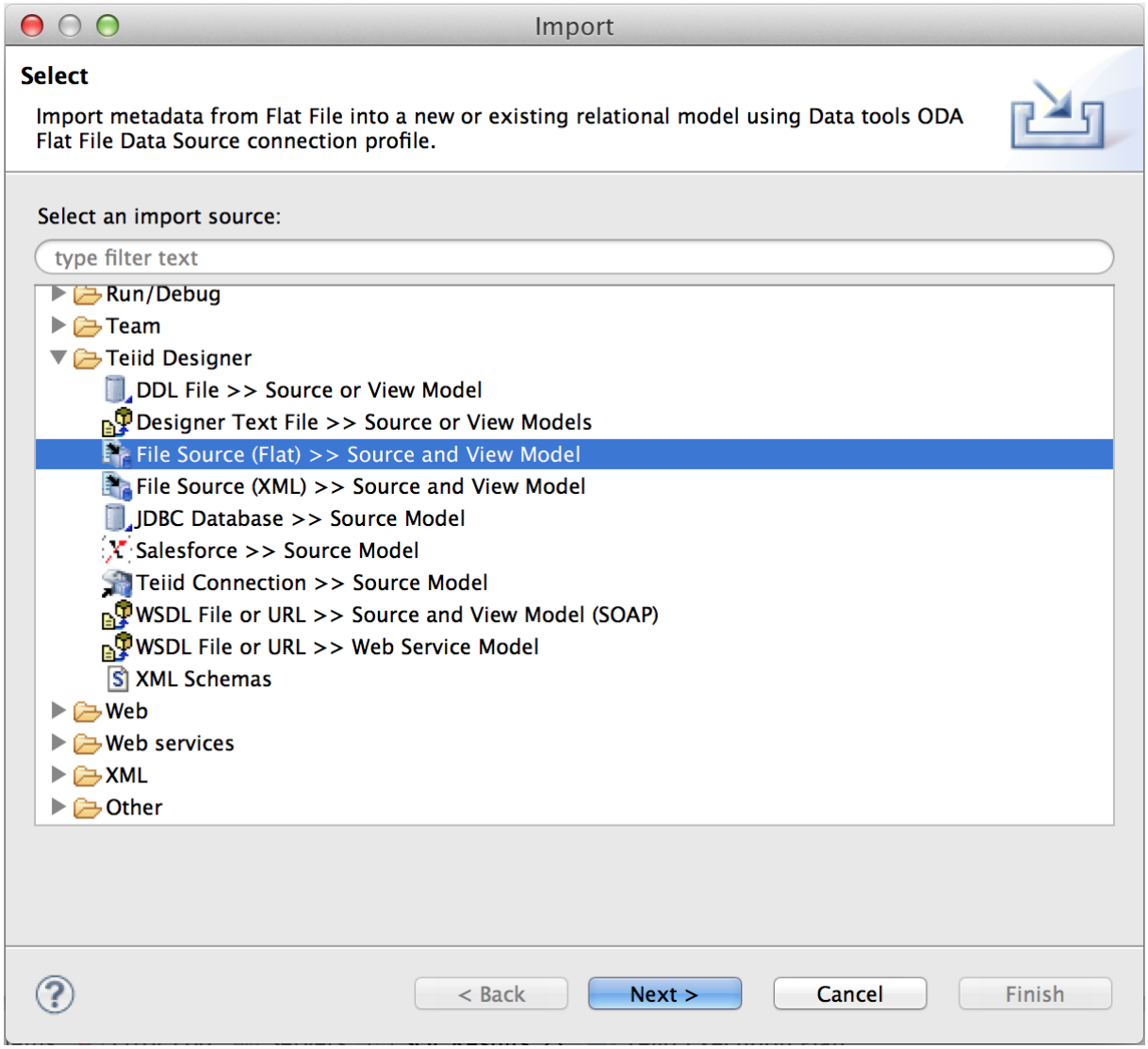

Figure 4.25.

Click the Next > button. The File Import File Options dialog box will appear.

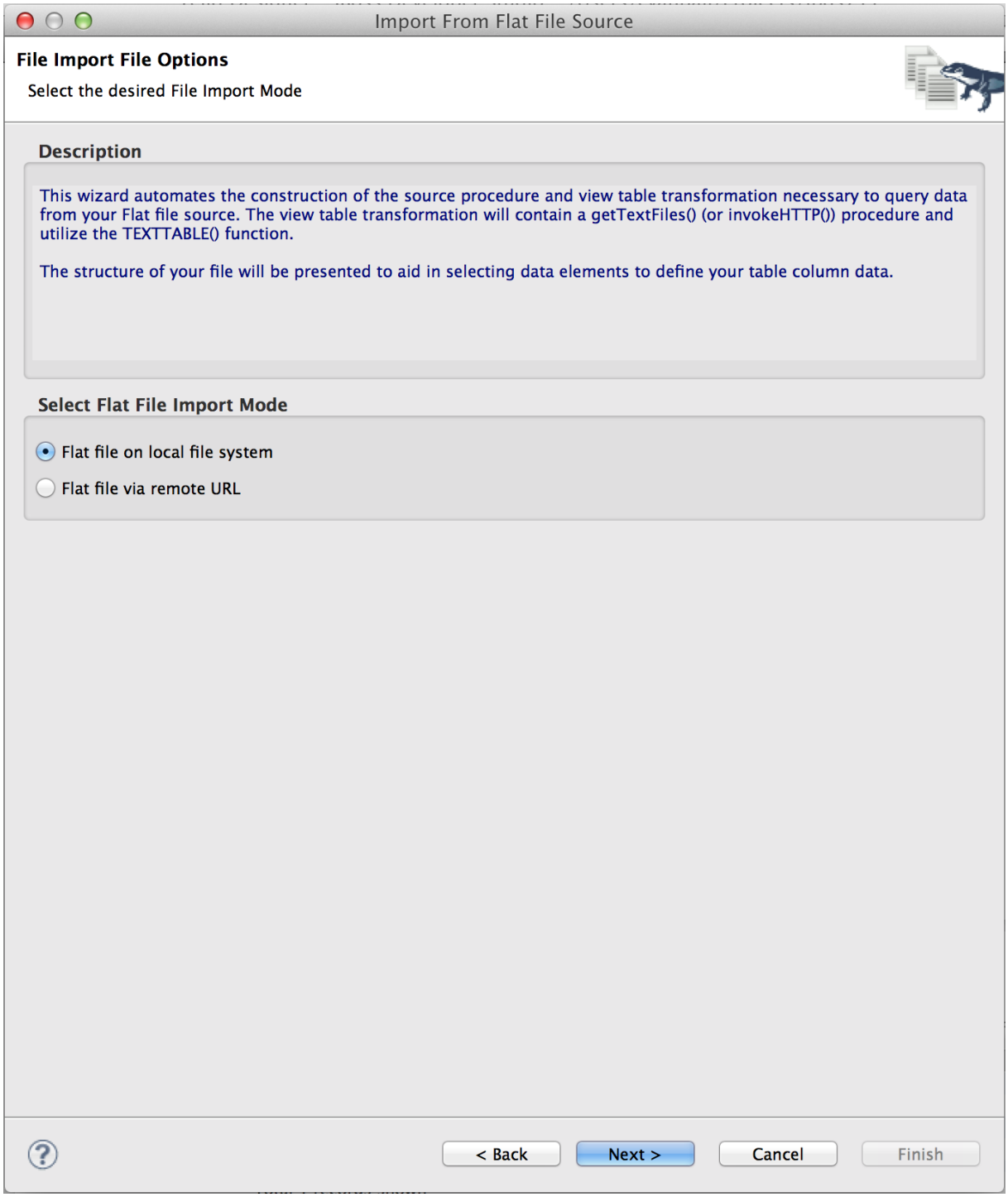

Figure 4.26.

Select option Flat file on local file system and click the Next > button.

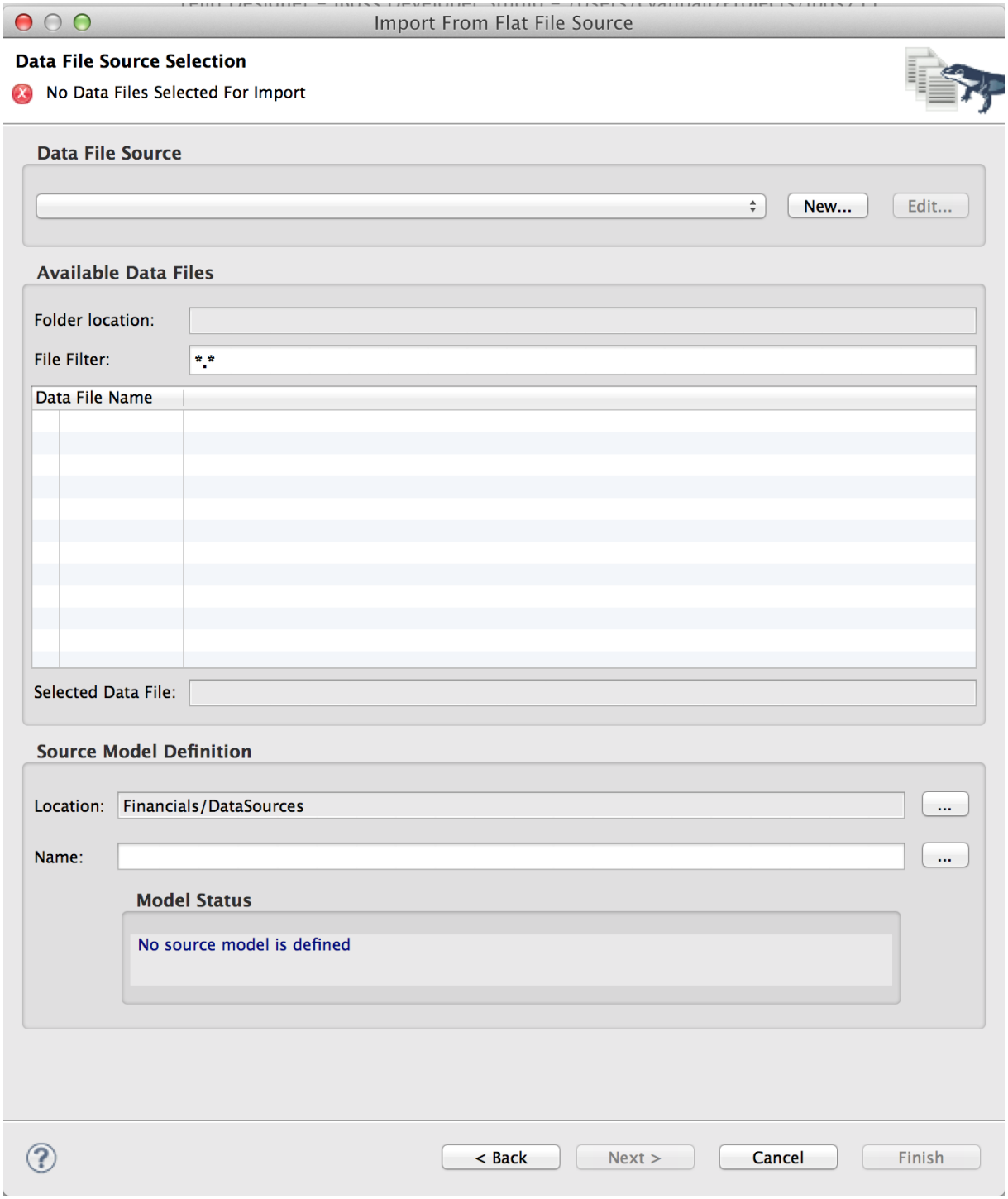

Figure 4.27.

The next screen that is displayed is the dialog box to select the file connection profile. Click on the New… button. The new connection profile dialog is displayed.

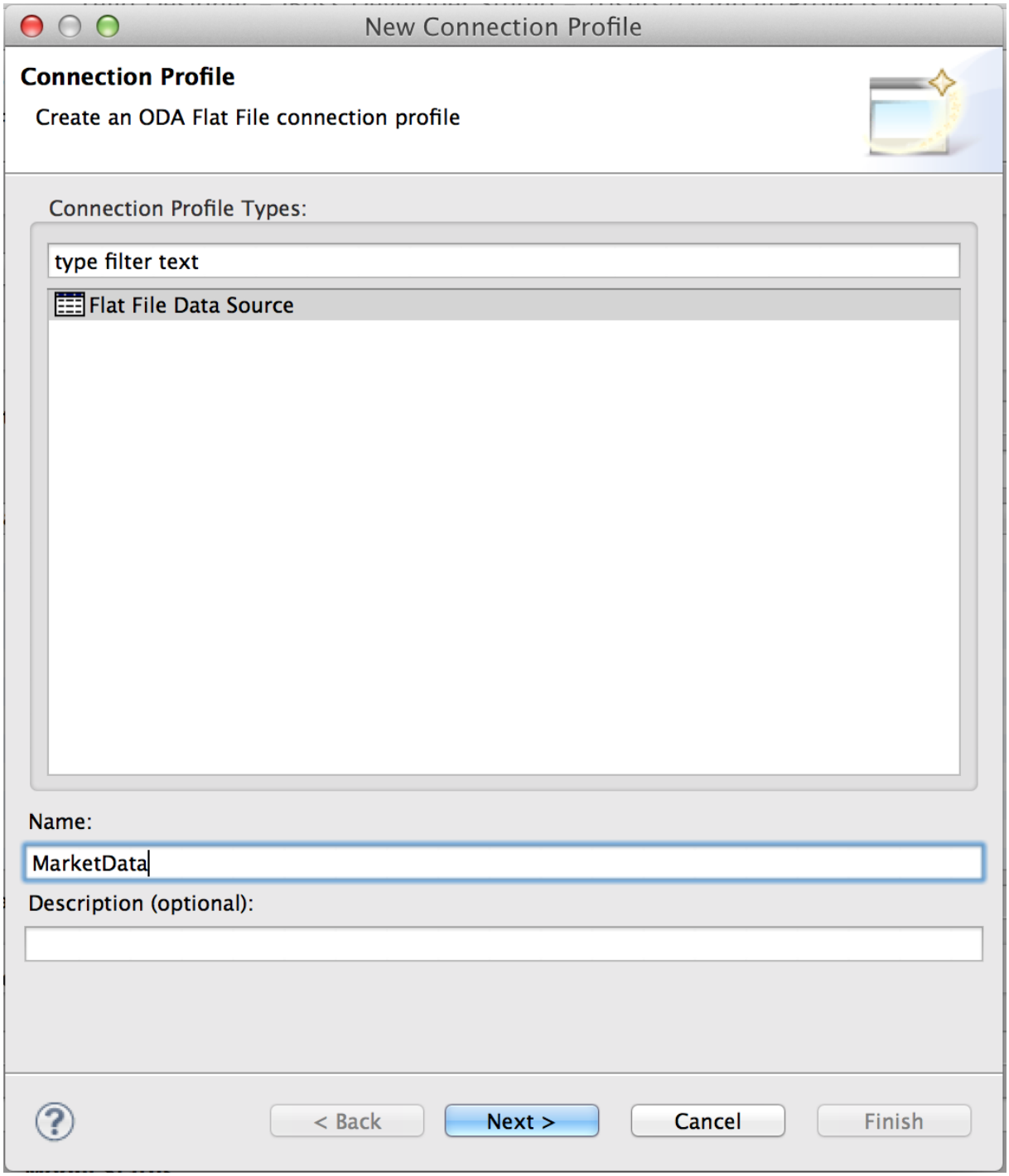

Figure 4.28.

Type in MarketData for the name and click the Next > button.

Use the browse (see image below) to locate the folder where the market data CSV file resides. The location should be DVWorkshop/dv_docker/demo. Ensure Use first line as column name indicator is checked. You can click the Test Connection button, it will just ensure that the program can get to the directory specified.

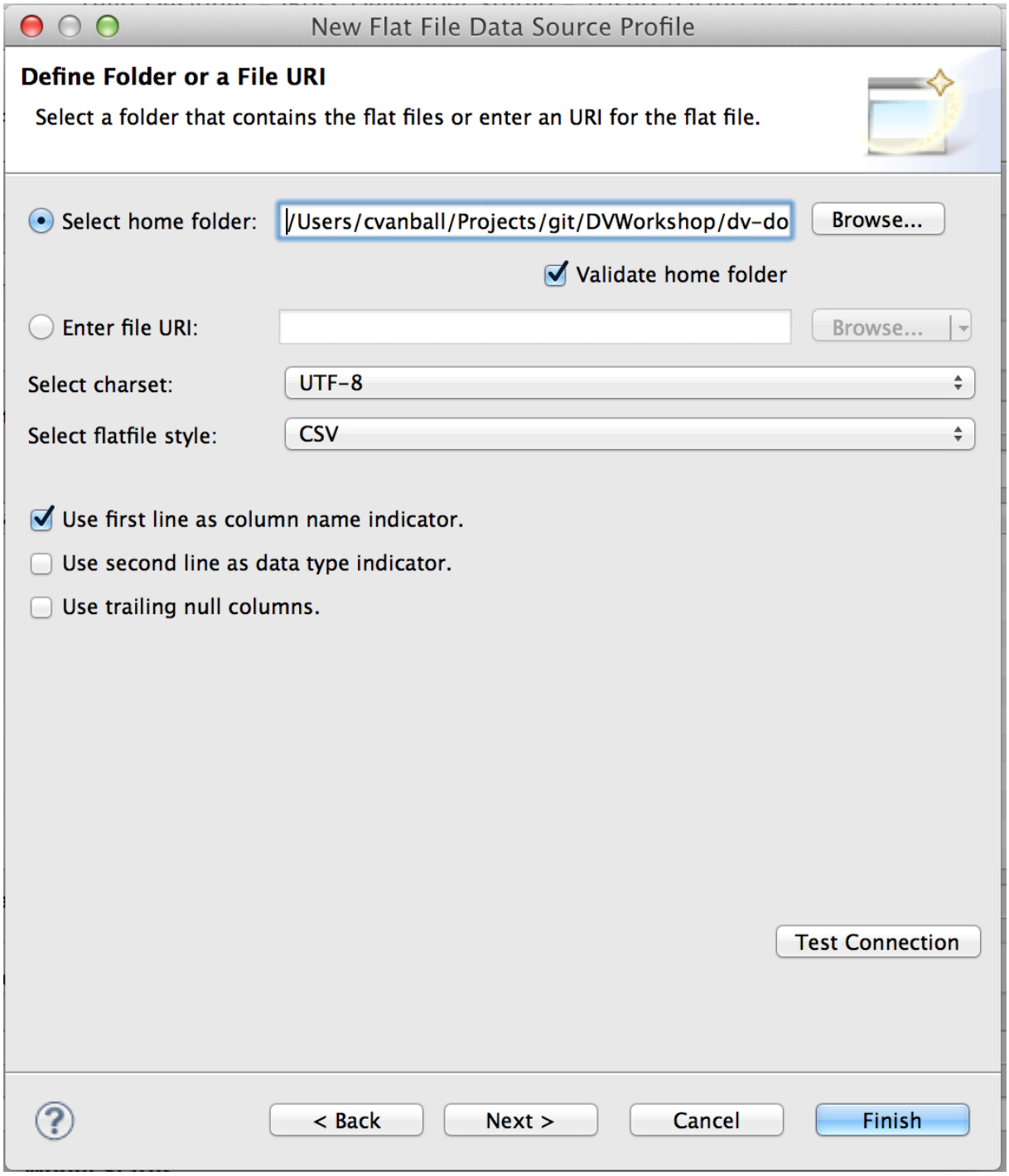

Figure 4.29.

Click the Next > button. The next dialog is the Summary dialog box that displays what has been selected for this connection so far.

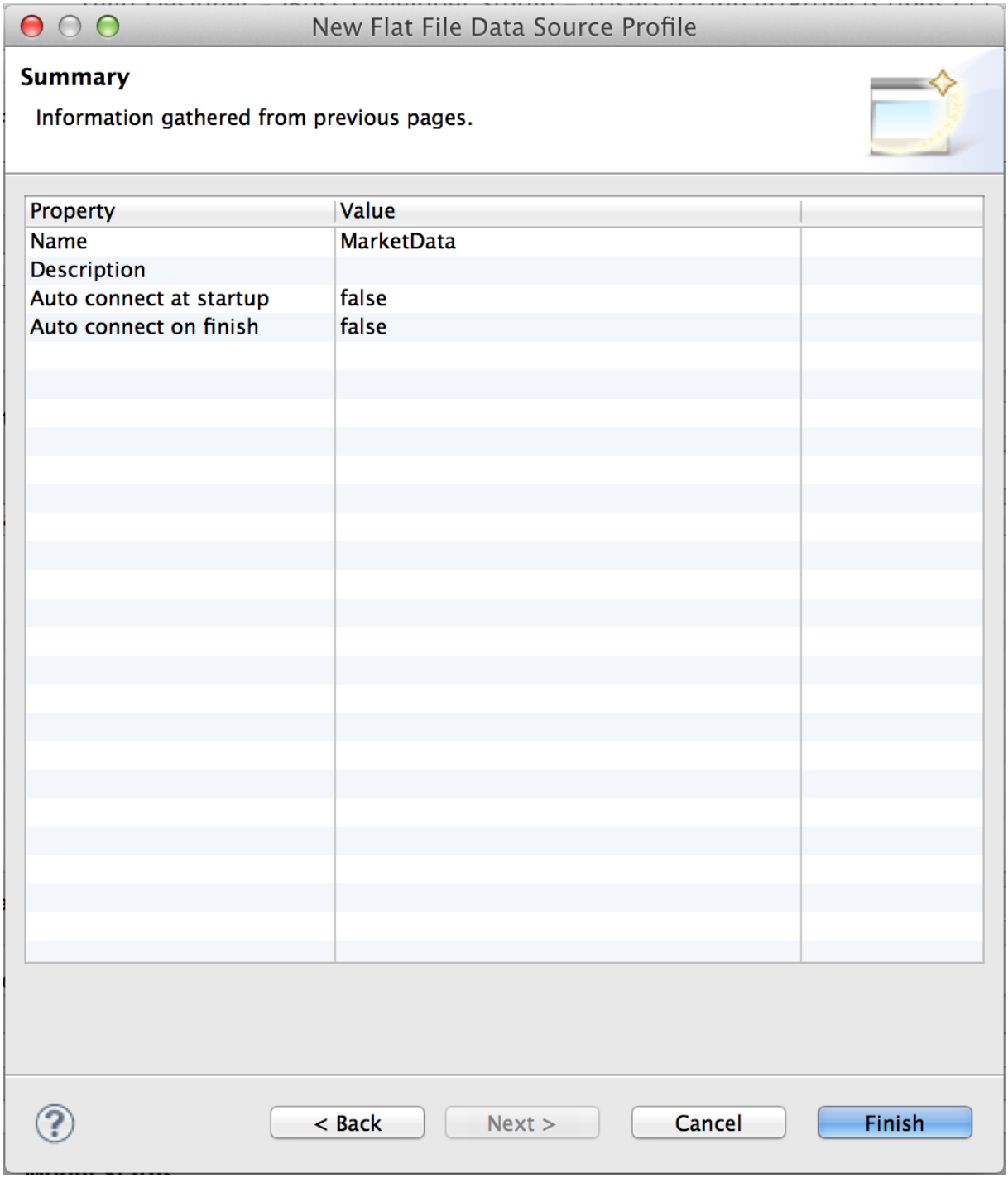

Figure 4.30.

Click the Finish button to continue. All of the CSV files are listed from the data source folder selected in the previous steps. We want to make sure that the check box is checked beside the marketdata.csv file, the file that you need to connect to. In the Source Model Definition, enter the model name, MarketData in the Name: field. (See image below)

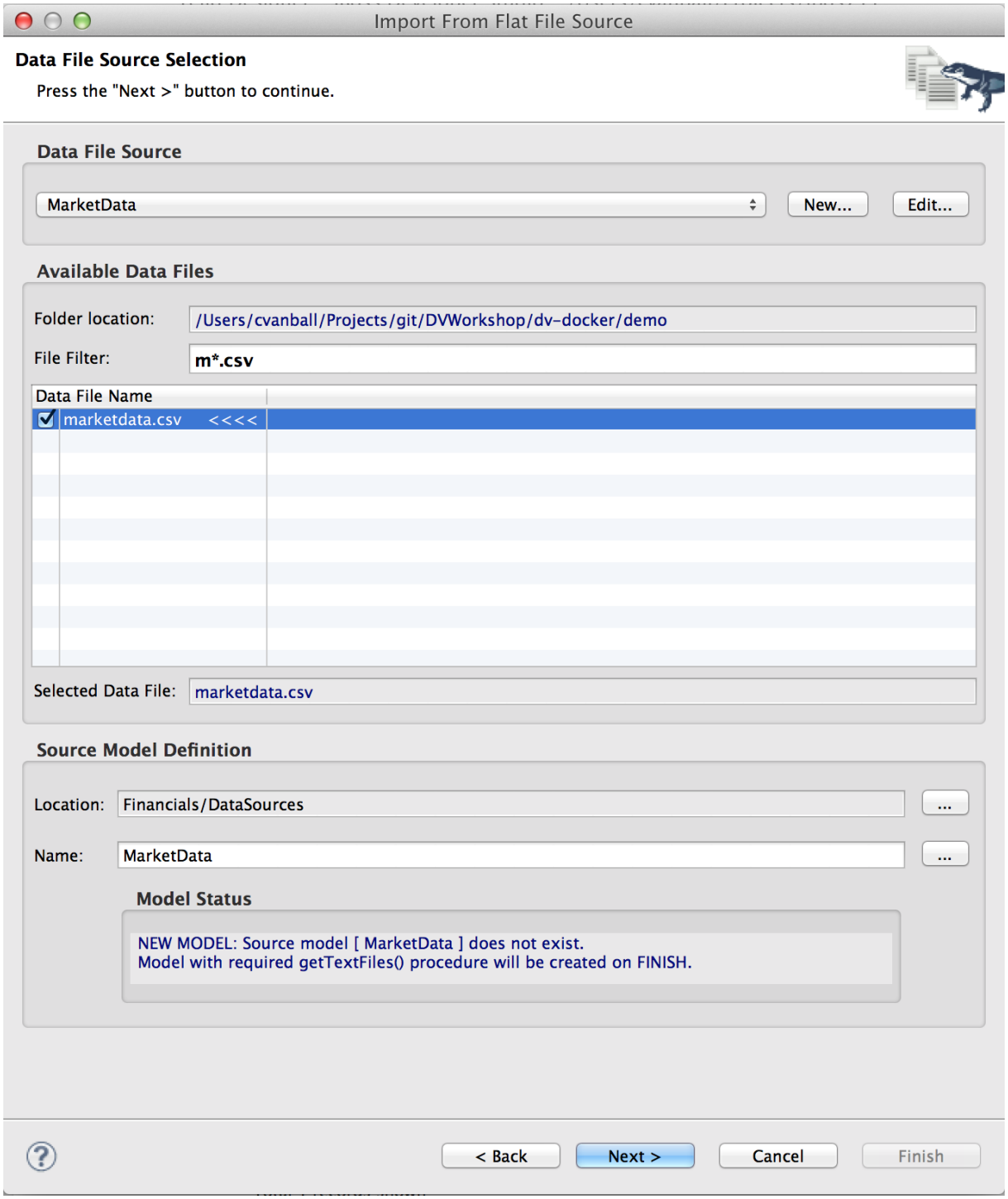

Figure 4.31.

Click on the Next > button to continue.

The next dialog box that opens allows you to select how the CSV file is formatted. In this case, the file is Character delimited (Delimited with a comma). See the image below for the settings that you will select.

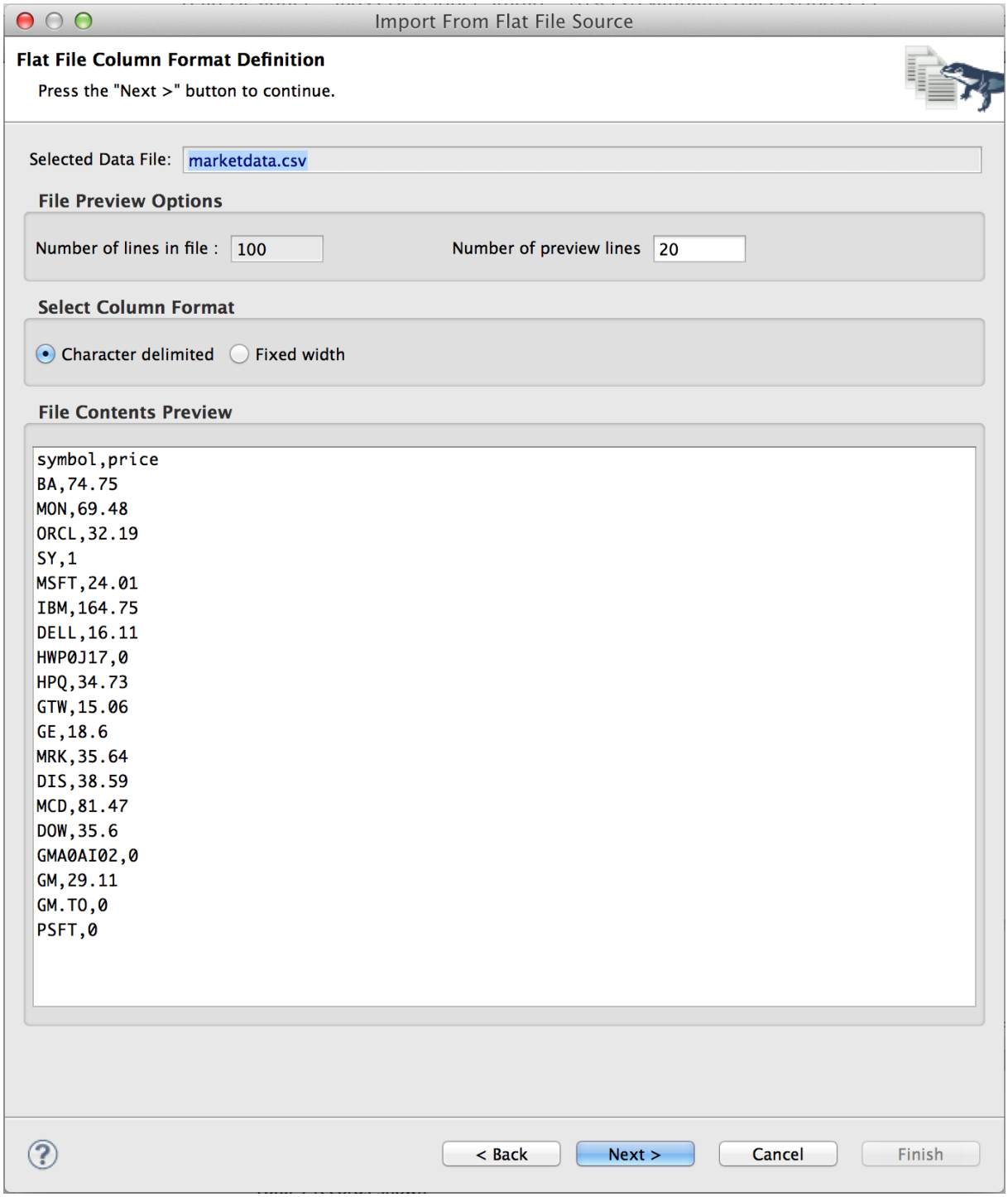

Figure 4.32.

Click on the Next > button to continue. Next specify parameters for how the CSV file to be imported. You will change the Datatype of the price column to bigdecimal.

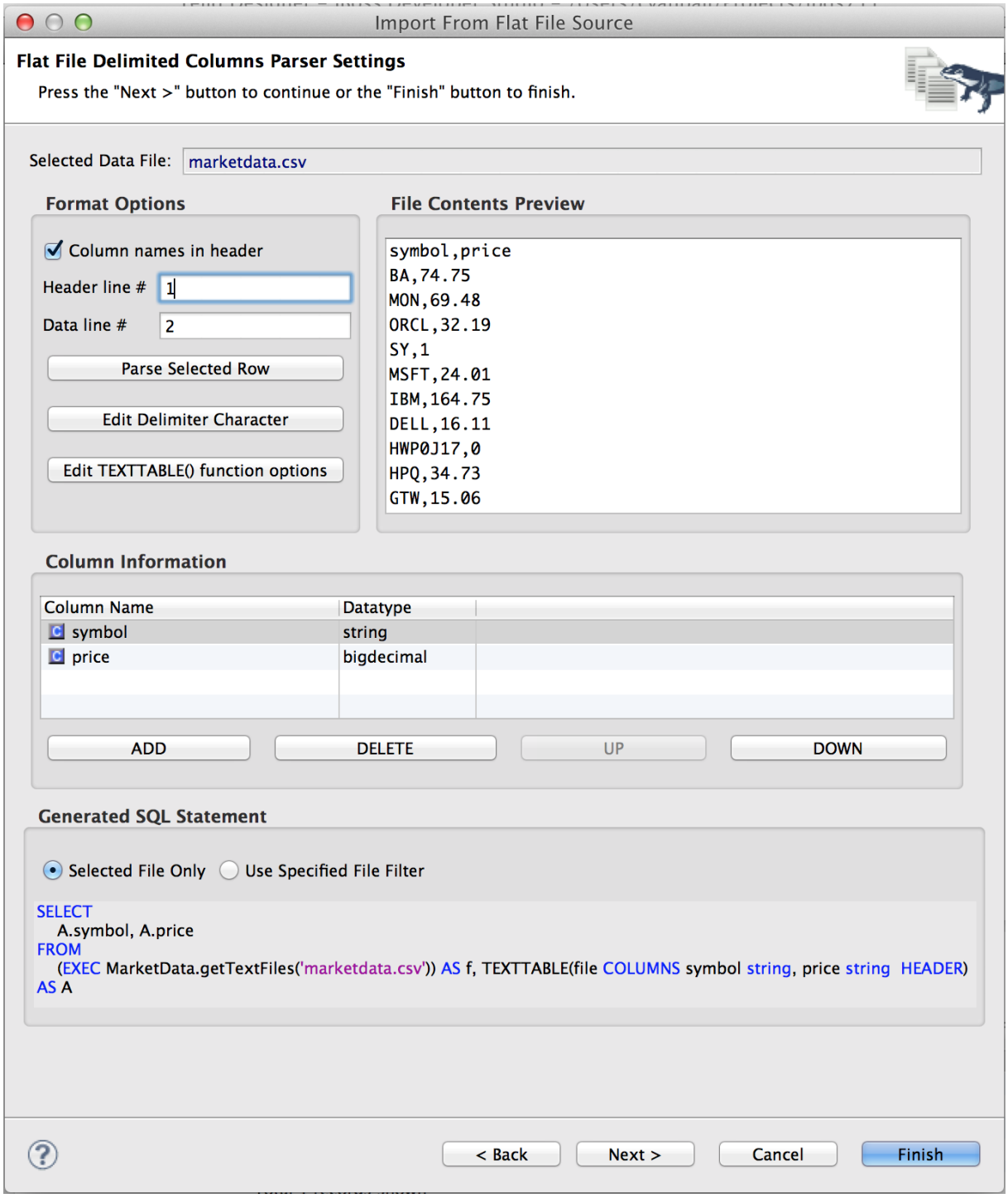

Figure 4.33.

Click the Next > button when complete.

The last step is to specify the View Model Definition. (See image below) Enter in the data from the image below.

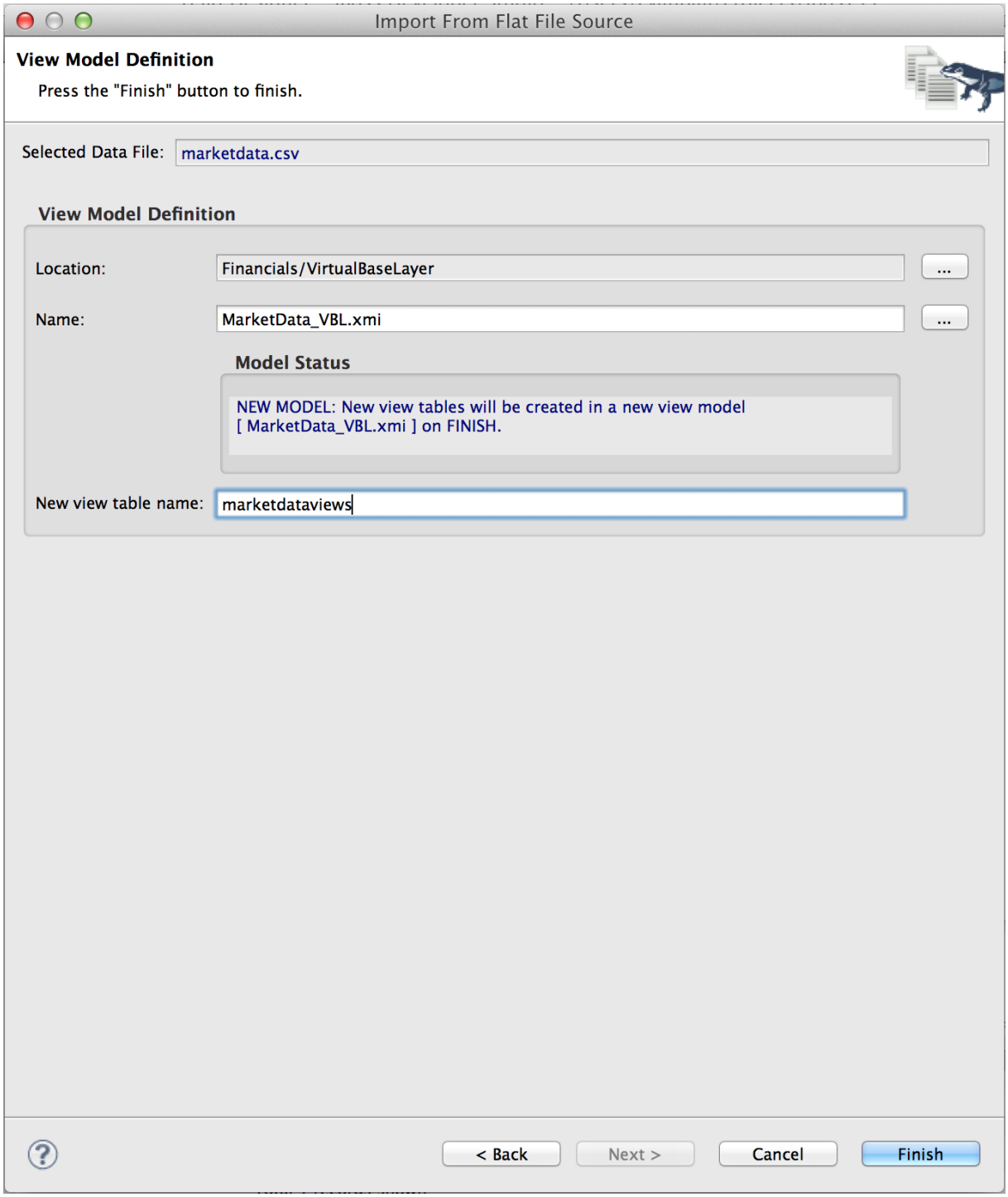

Figure 4.34.

Click on the Finish button to import the model.

Congratulations, you have now completed this lab.

Chapter 5. Create a Virtual Base Layer

Beyond the obvious advantages of integrating disparate data sources using an intuitive, easy-to-use technology such as JBoss Data Virtualization, another significant advantage of creating a data services abstraction layer is to provide a level of isolation from the physical sources themselves. By creating a layered set of data service models, from fine-grained to more granular data services, the developer can build a stable data integration layer than can easily adapt to changes in the source systems. This is especially advantageous when the consumers of the data are separate from/have no control of the providers of the data. A recommended best practice to begin building this "future-proof" abstraction layer is to create a Virtual Base Layer (VBL), a one-to-one mapping of each physical source that isolates any future changes that may arise in the data source(s) to a specific transformation in the model. These VBL components can then be used as building blocks for higher level services; all of the transformations built on top of them will not need to be changed should the need arise to accommodate a change in the source(s).

5.1. Create a virtual base layer: US_Customers_VBL

To create a VBL for each of the source metadata models that you have imported, right-click on the VirtualBaseLayer folder that you created earlier and select New → Teiid Metadata Model.

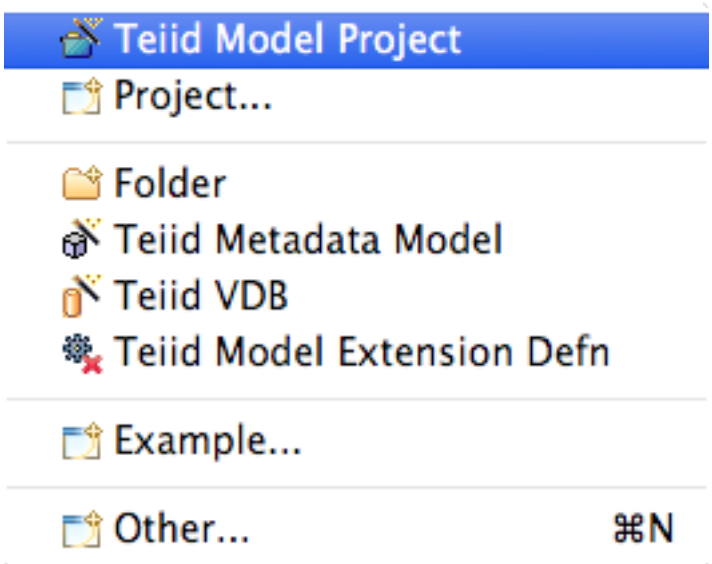

Figure 5.1.

Enter US_Customers_VBL as the Model Name, Relational as the Model Class, and View Model as the Model Type. Select “Transform from an existing model” in the Select a model builder panel and click Next.

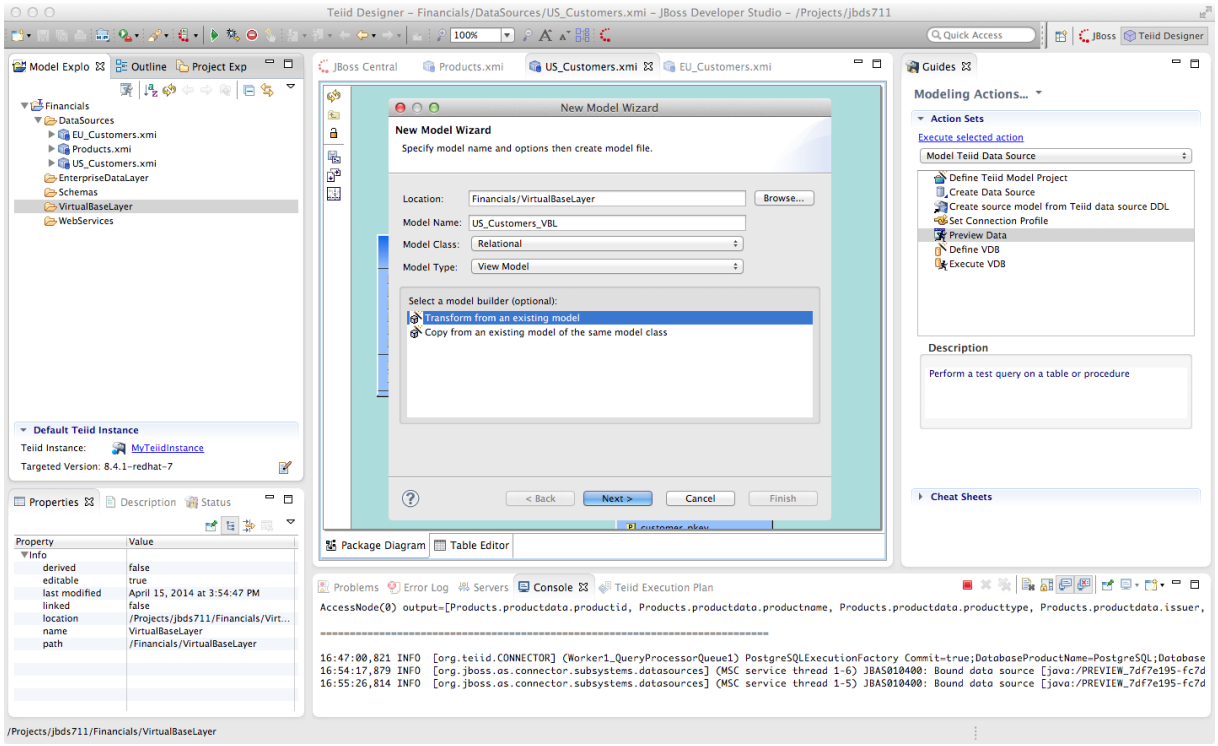

Figure 5.2.

The “Transform from an Existing Model” wizard will appear.

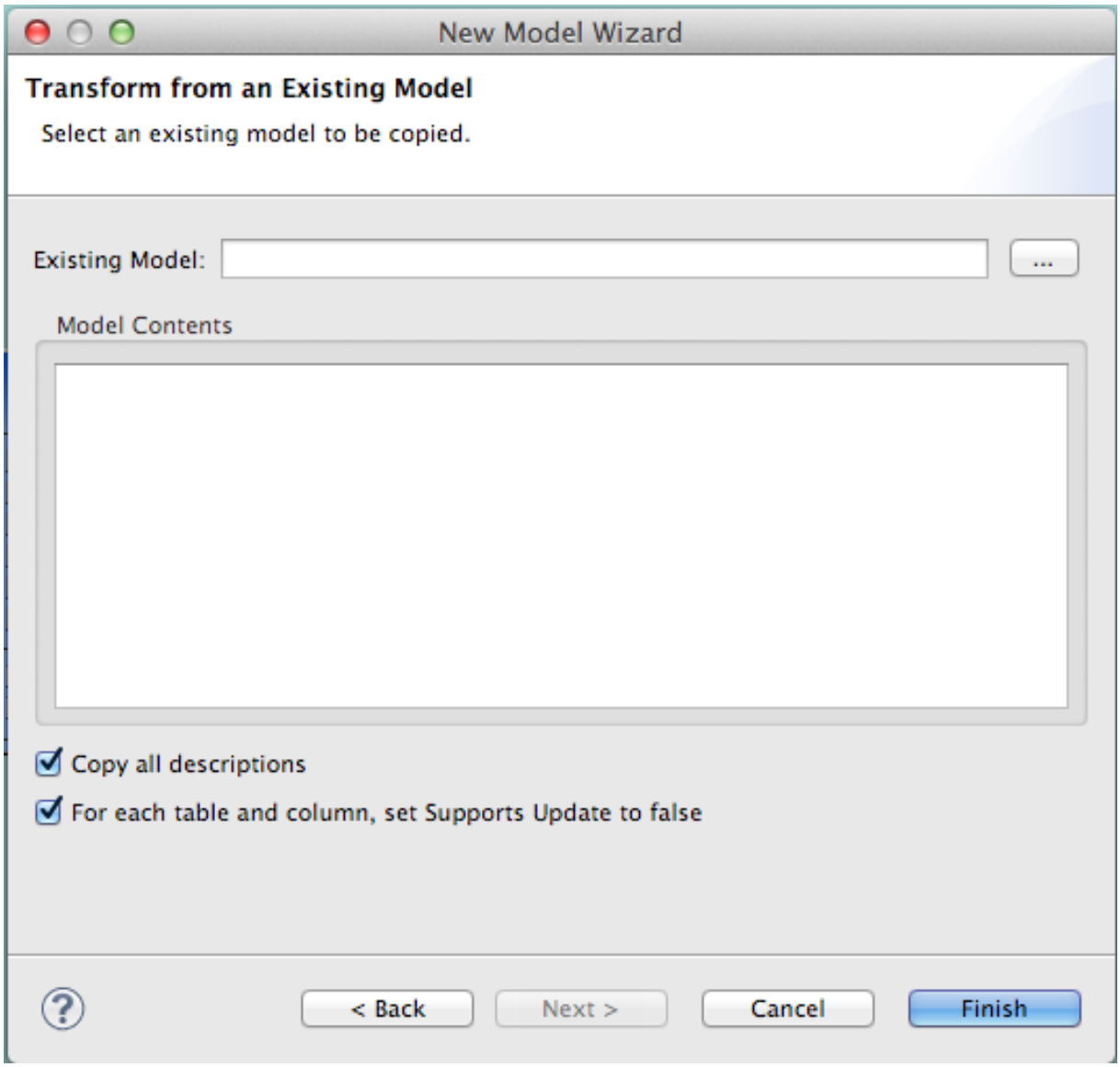

Figure 5.3.

Click the btn[…] to the right of Existing Model and open Financials → DataSources → US_Customers. Click OK. You will be returned to the New Model Wizard. We are going to copy all model elements so simply click Finish on the screen as indicated below.

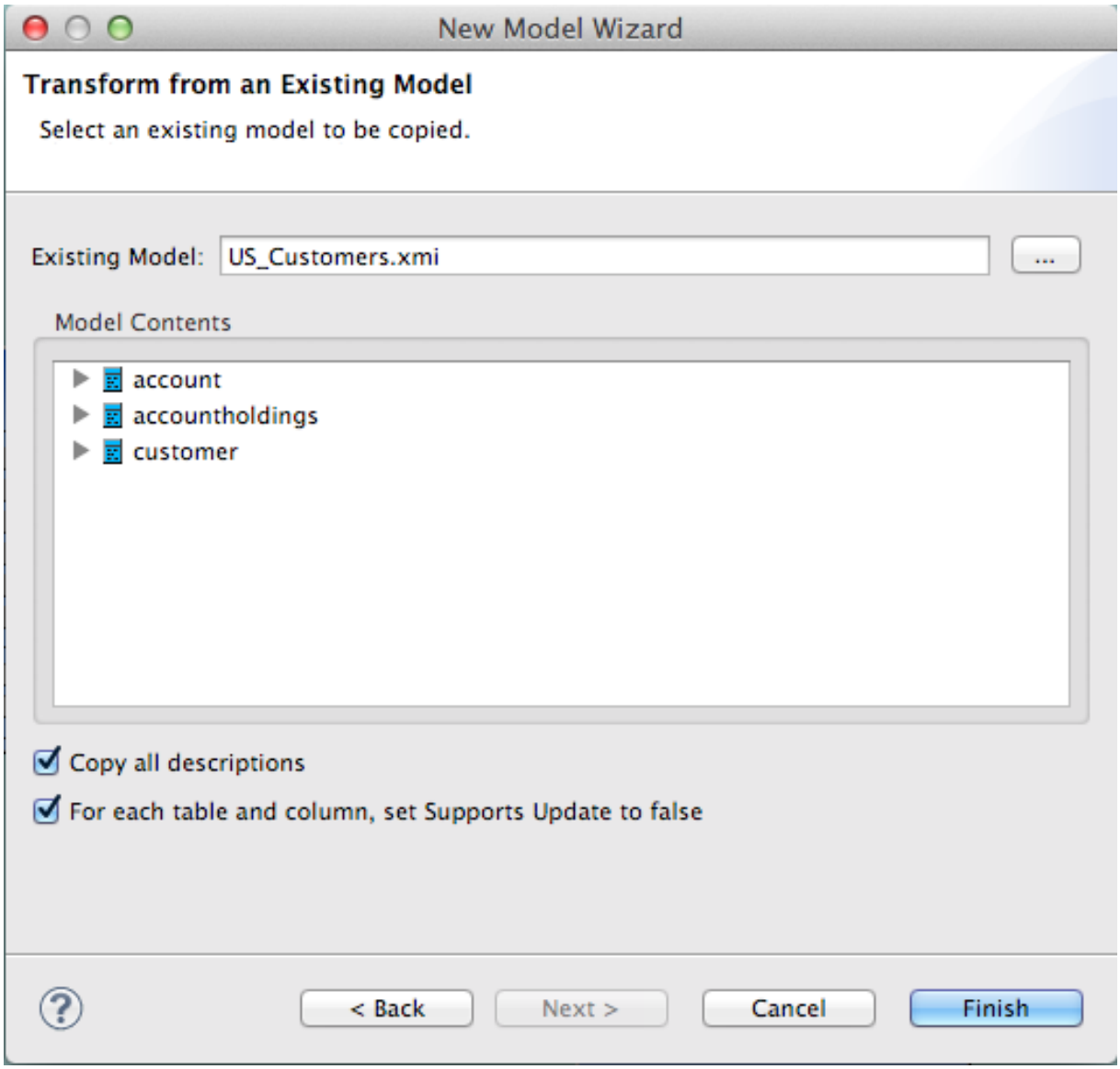

Figure 5.4.

The Package Diagram and Model Explorer for the US_Customers_VBL virtual model will now open in the workspace. Observe that the virtual models are rendered in yellow whereas physical models are rendered in blue. Save the changes to the project (do this periodically as you progress) and note that you can also do previews of the virtual model.

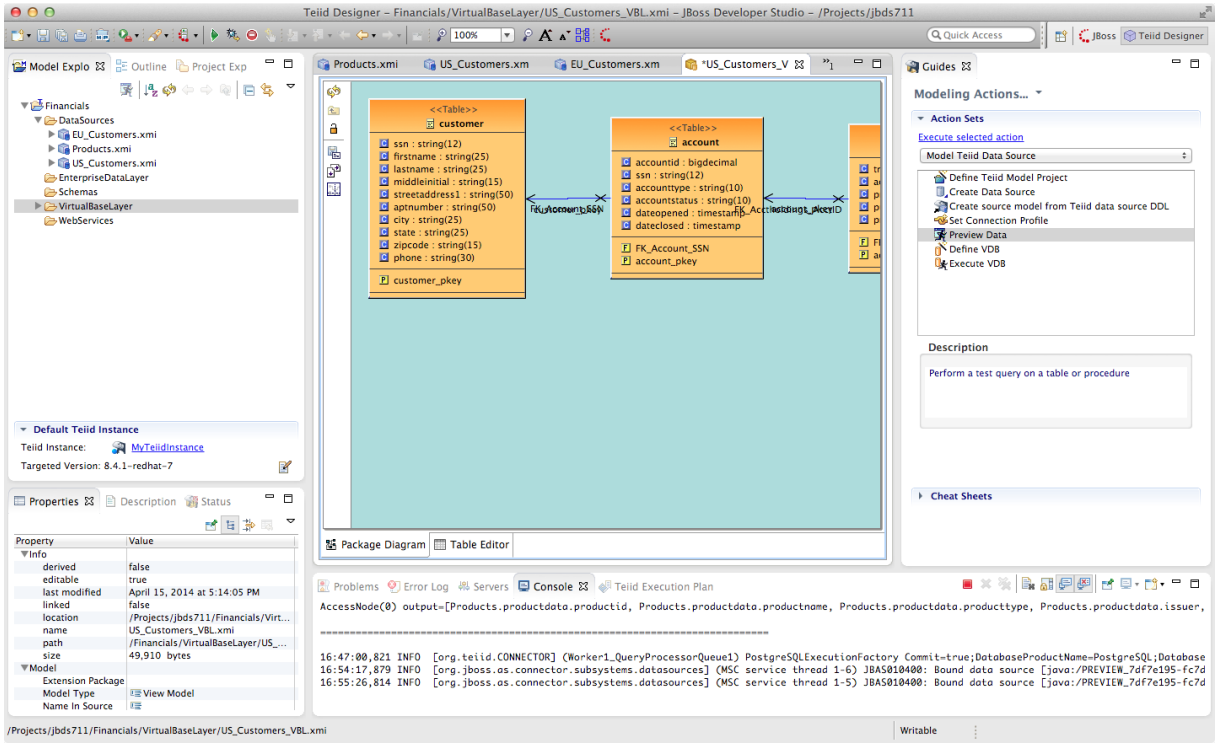

Figure 5.5.

Double-click on the customer table in the US_Customer_VBL model. This will open the Transformation Editor.

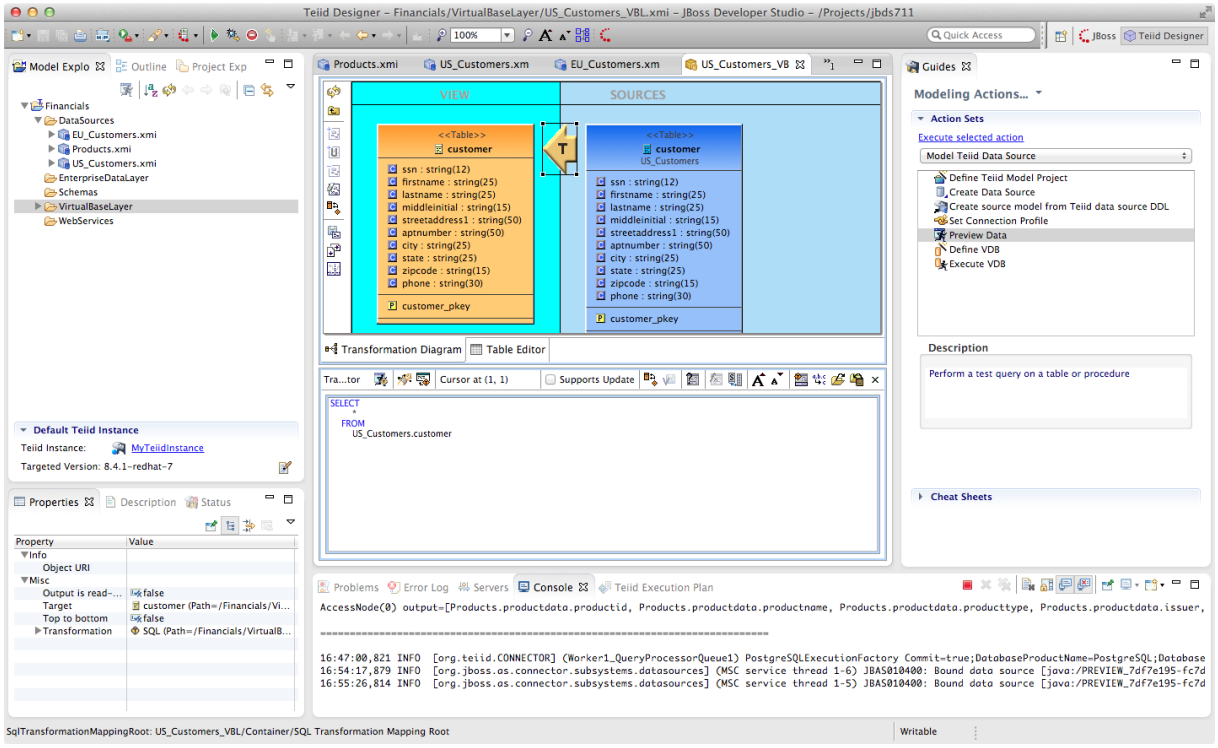

Figure 5.6.

The transformation has already been created for you, and as noted above it is a simple one-to-one mapping of the underlying physical source. By building up the data services in layers like this, it allows the designer to keep the transformational logic for each view fairly simple, and complex data transformations are achieved by uses several layers of such views. Now, in a traditional relational database, such a design would have a fairly heavy performance penalty at runtime to deal with all of these layers of views, which is why in a traditional database you'd see a use case like this defined as a single view defined with a very lengthy and complex SQL statement. The reason this is not an issue with JBoss Data Virtualization is because the Query Engine in the JBoss Data Virtualization Server compresses all of these layers at run time down to a single (potentially highly complex) query that is then optimized to run (in parallel if possible) against the backend data sources. Thus there is no penalty to using layered views with JBoss Data Virtualization.

The "transformation language" in the Transformation Editor is ANSI-standard SQL. We don't make you learn a new language to define transformations, you just use normal SQL syntax, which is something that we find most data architects are already fairly comfortable with. We will be spending much more time with the Transformation Editor further on in the lab.

5.2. Create the EU_Customers_VBL and Products_VBL Models

Create VBLs for the other two physical models you have imported (Products_VBL and EU_Customers_VBL). Be sure to create them within the VirtualBaseLayer folder. Right-clicking on the folder to get to the New → Teiid Metadata Model wizard. Save your changes. When you are finished, your project should look like the illustration below.

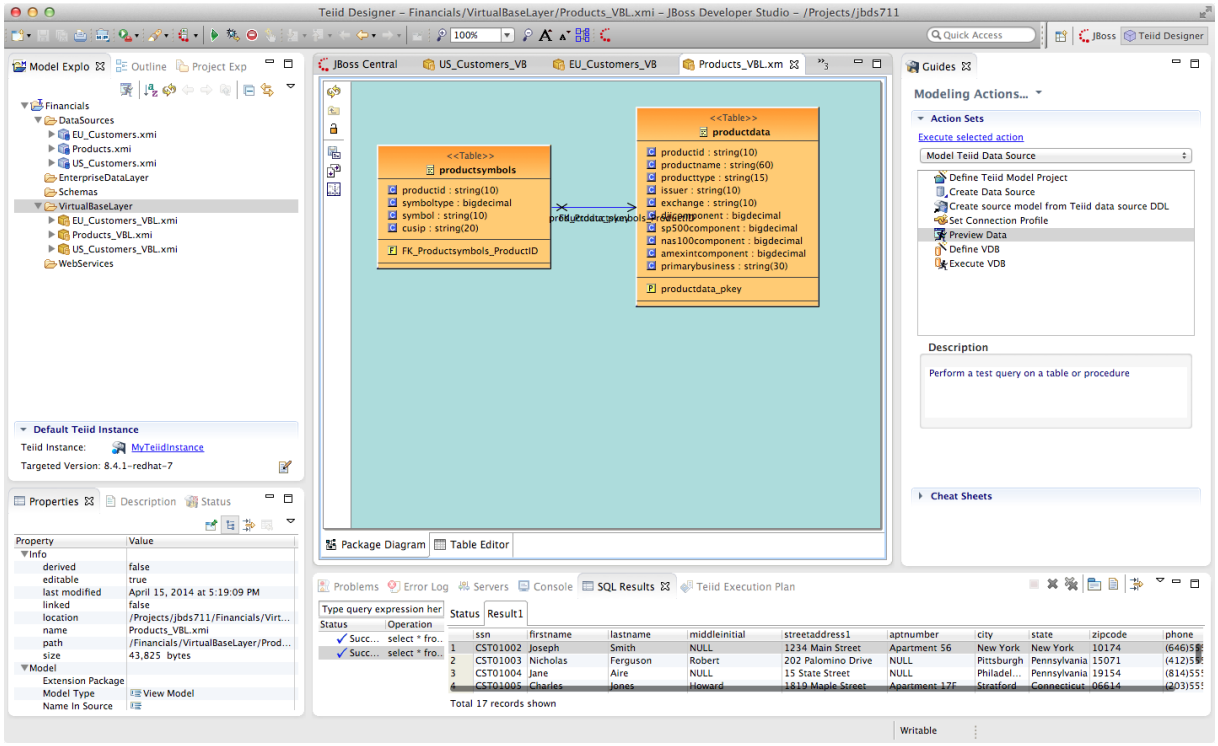

Figure 5.7.

Congratulations, you have now completed this lab.

Chapter 6. Create an Enterprise Data Layer

The next layer to build are the models in the EnterpriseDataLayer folder. What these models do is resolve the slight semantic differences between the EU_Customers and US_Customers database tables – for example, the US version includes a field for middle initial only in the customer table where the EU version of the customer table has a corresponding field for middle name. Now, one technique we use here to make resolving the semantic differences a little bit easier here is to convert all of the data types for these tables from the standard SQL datatypes to one of a set of custom datatypes we've defined using JBoss Data Virtualization's data dictionary capabilities. This way we can ensure that we get all data for a particular field into the same datatype. It also gives us a guide to resolving any differences in field names between the two. Note that using a domain-specific (or enterprise-wide standard) data dictionary is only a recommendation. There is no hard requirement to define or use a data dictionary but instead do this semantic mediation without one (by manually comparing datatypes and field lengths between the two data sources). Our goal with JBoss Data Virtualization is to not impose any roadblocks for those users that want to get their use cases addressed as rapidly as possible – so we don't force any requirement on users to define a data dictionary or a taxonomy or anything like that.

6.1. Import the Data Dictionary Schema

An XML schema containing the definitions of a number of datatypes applicable to the Financial domain used by this lab is located in DVWorkshop/dv_docker/demo. The name of the file is DataDictionary.xsd. We want to import the schema into the Schemas folder that was created in an earlier lab. Perform the following steps. Right-click on the Schema folder in the Model Explorer and choose Import → Teiid Designer → XML Schemas. Then click the Next > button.

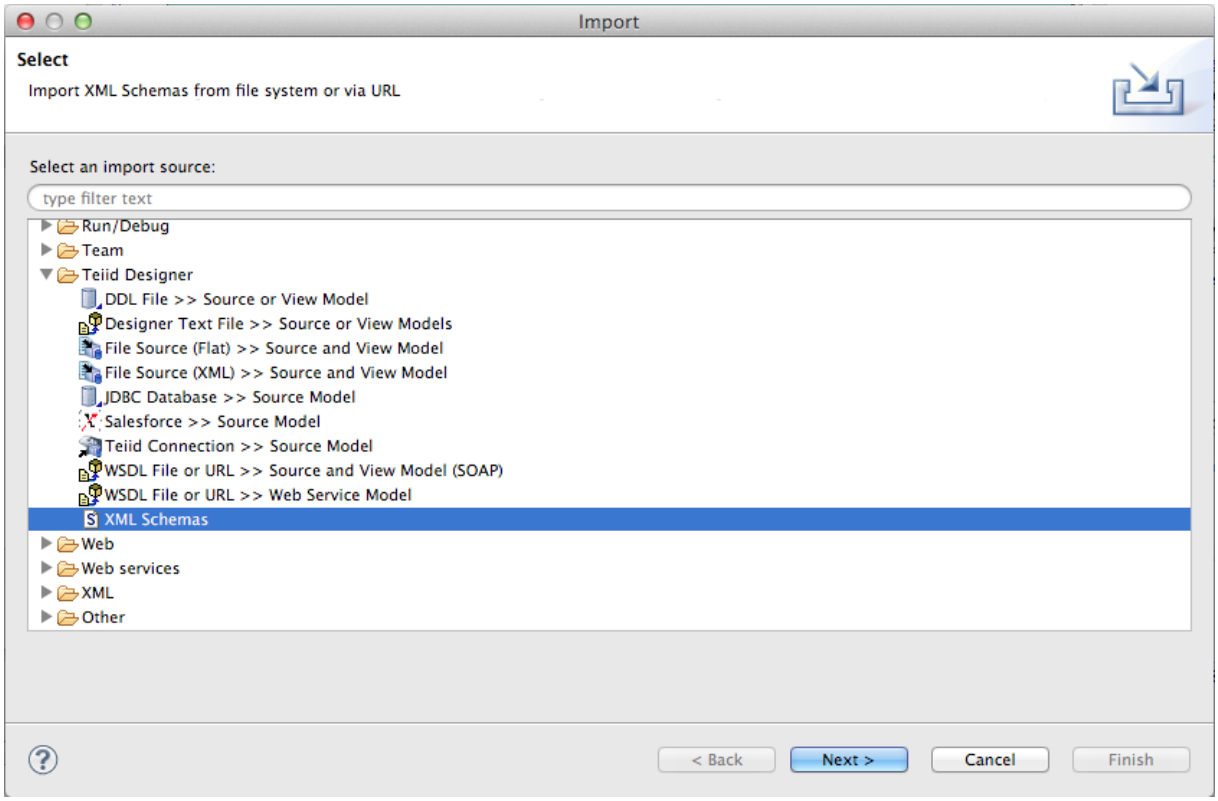

Figure 6.1.

Choose to import the schema from the file system and click Next >.

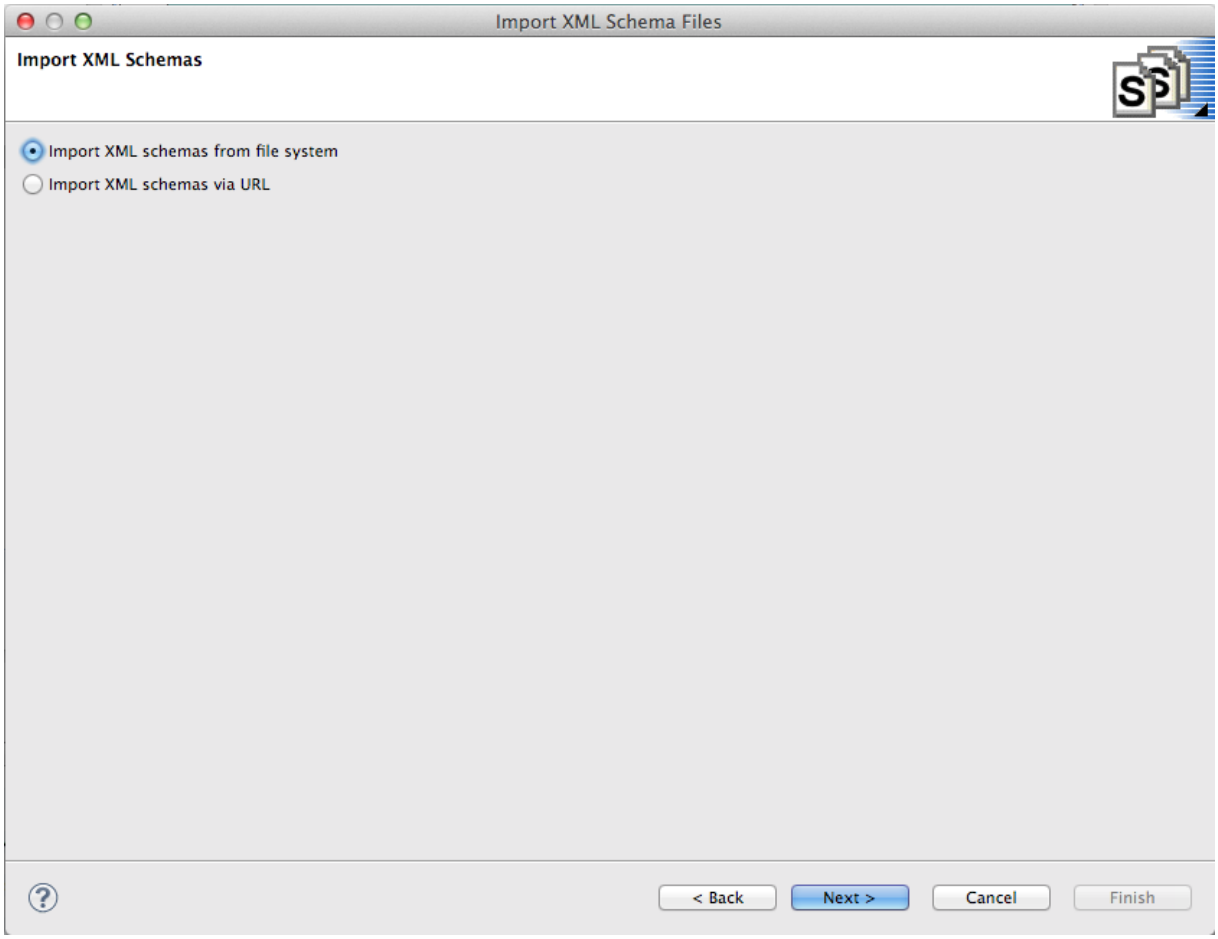

Figure 6.2.

Browse to the DVWorkshop/dv_docker/demo directory. Once the directory is opened in the “From Directory” field, you will be able to select (checkbox) the DataDictionary.xsd file. Then click Finish.

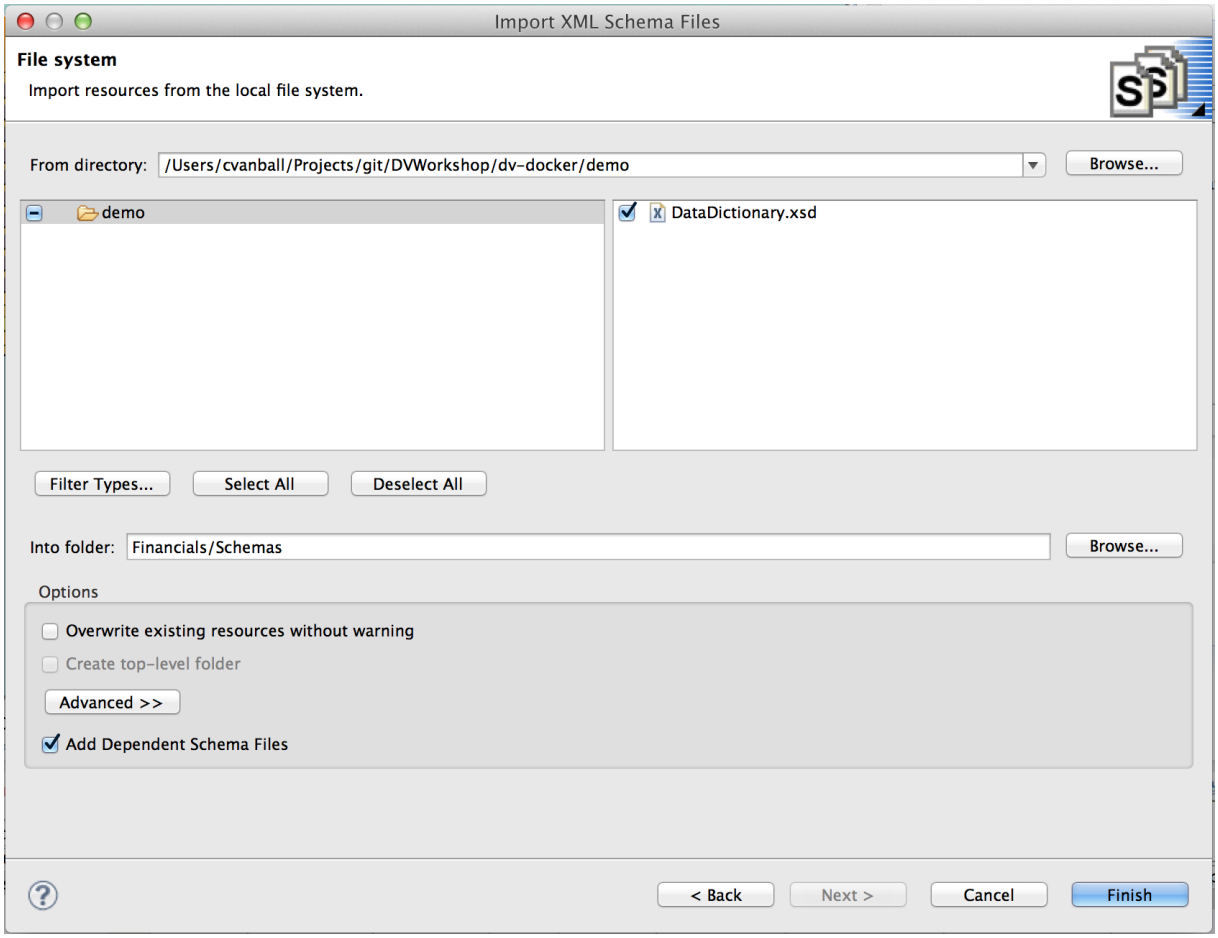

Figure 6.3.

You may get a warning that the Financials/Schemas already exists. Choose yes to overwrite.

6.2. Examine the Data Dictionary

Double-click on the DataDictionary.xsd file you just imported to open the XML Schema viewer. This is a very basic viewer that allows you to see the design model, the source (note the tabs at the bottom), and choose any of the data types to view it's properties (in the Properties tab to the left). The schema can be edited/created here, but most customers use XML-specific tooling to create these files.

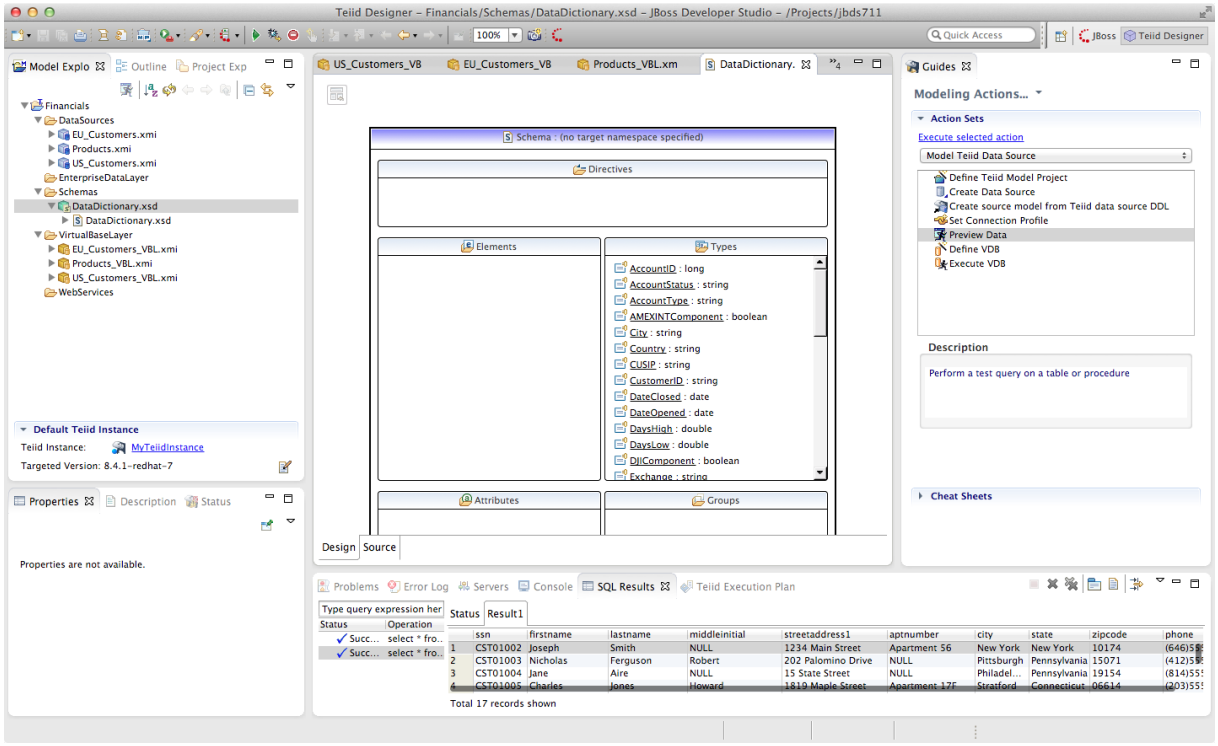

Figure 6.4.

6.3. Create the EU_Customers_DDC Enterprise Data Layer

Create a new virtual metadata model called EU_Customers_DDC (for Domain Data Dictionary) under the EnterpriseDataLayer folder. Right-click on the folder EnterpriseDataLayer and select New → Teiid Metadata Model to bring up the New Model Wizard and make the following selections as indicated in the illustration below.

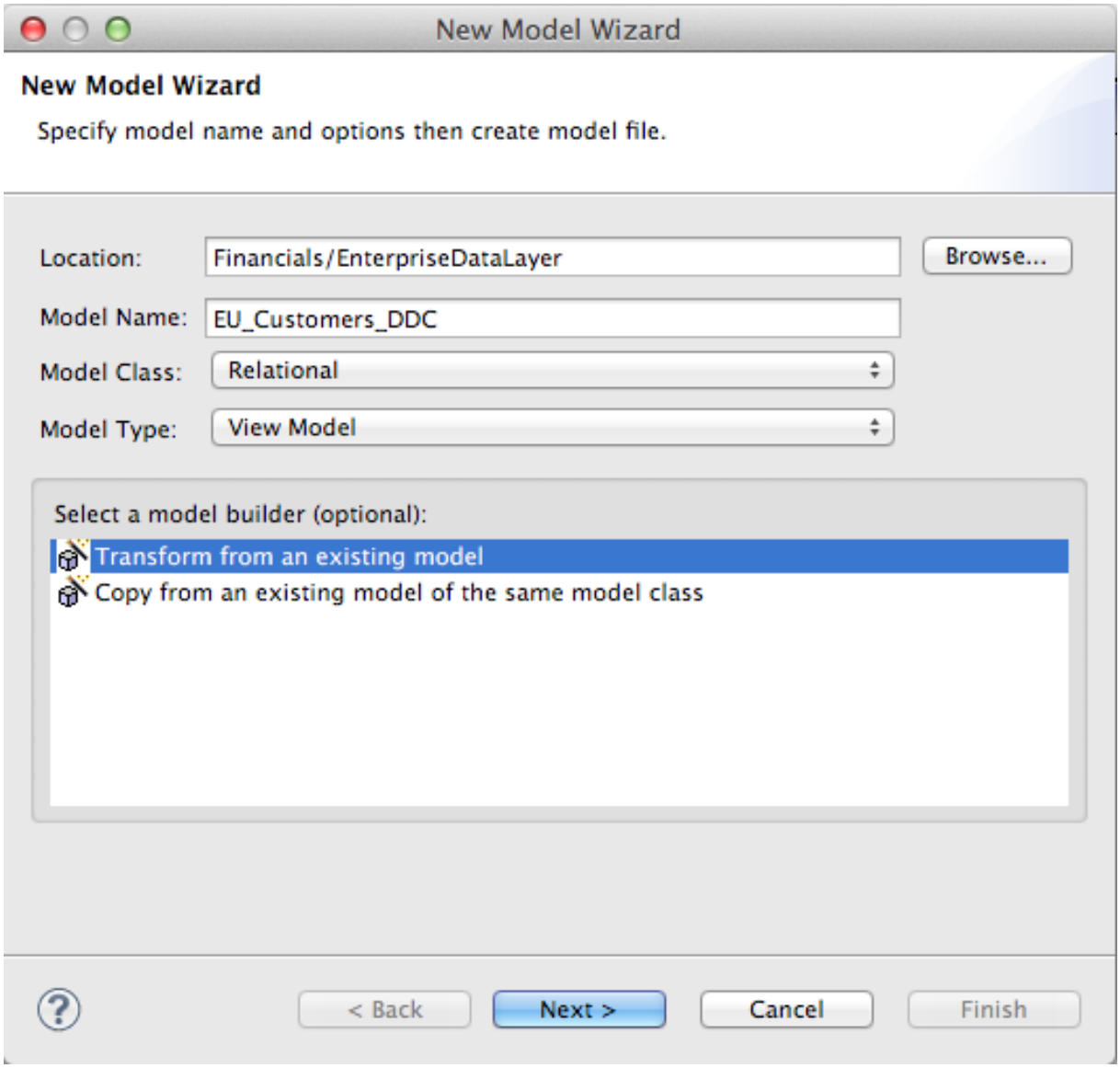

Figure 6.5.

Click Next > to go the next step of the New Model Wizard.

This time, use the EU_Customers_VBL as the model as the basis for the DDC model and click OK and Finish.

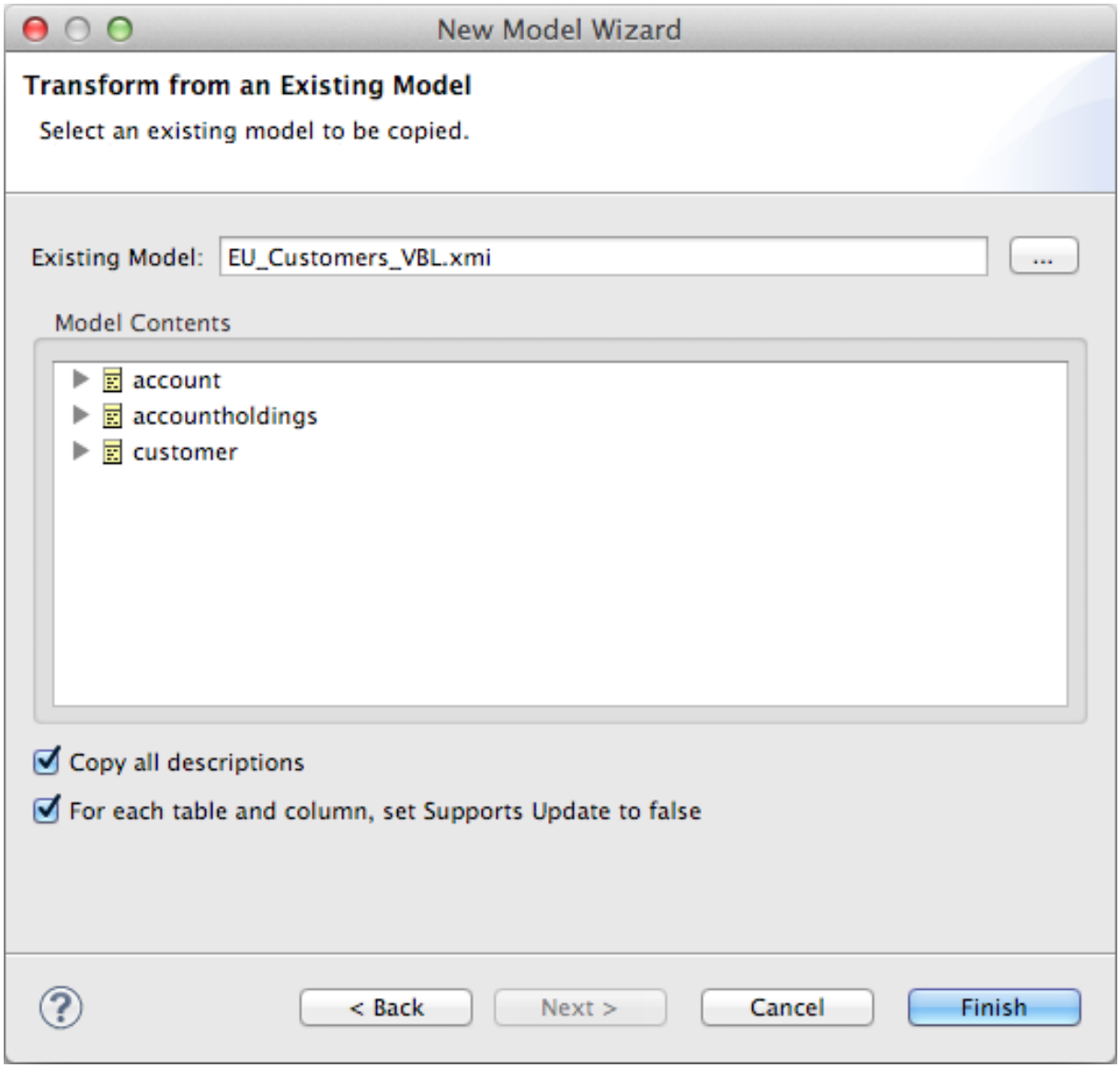

Figure 6.6.

We are going to use the database schema of the EU_Customers database as the standard upon which to base our enterprise model. Therefore we won't be changing any of the table names or column names, but we will be modifying the transformation to incorporate our Data Dictionary. When we get to the transformations for the US_Customers and EU_Customers DDC models we will see that more changes will be necessary to make them semantically align with our enterprise model. Open the transformation Editor for the Customer table of EU_Customers by double-clicking on it. Select CustomerID and note that there is a Datatype field in the Properties Panel on the lower left.

Click on the value field to the right of datatype in the Properties Panel in the lower left-hand side of JBDS.

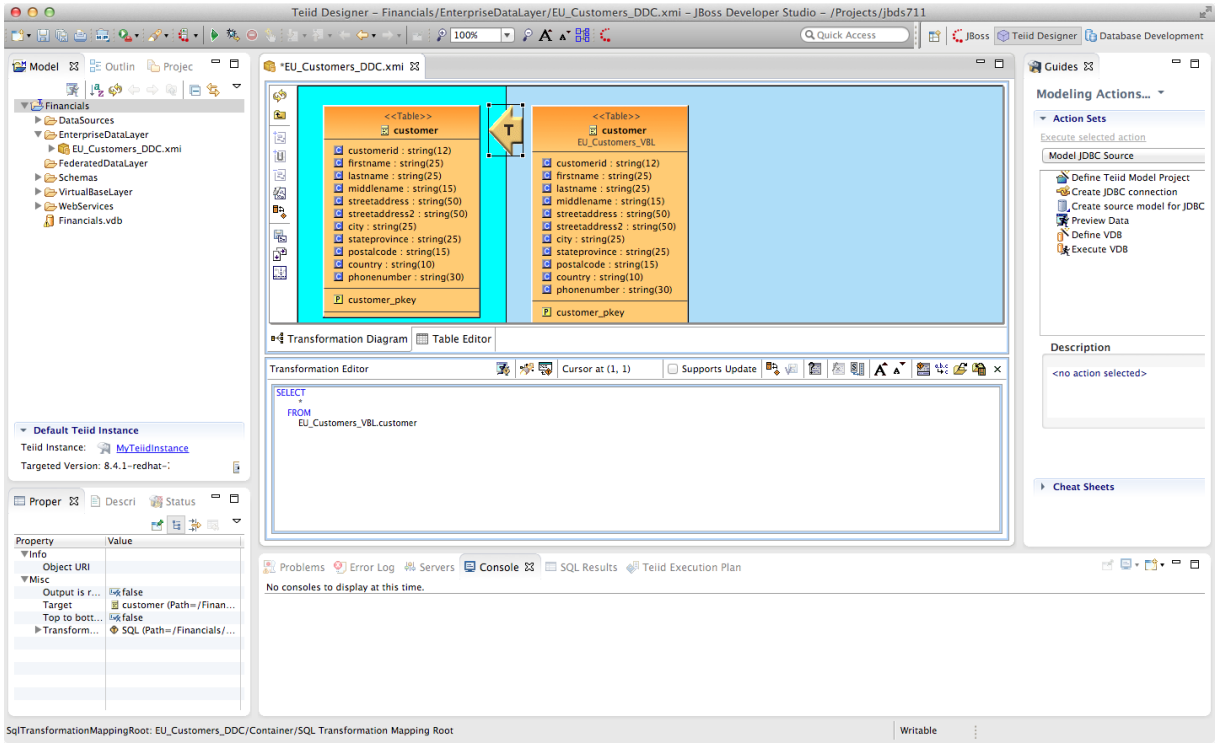

Figure 6.7.

When clicking in this field in the Properties Panel, the Select a Datatype Wizard is presented. The User-Defined Types were populated when we imported the DataDictionary.xsd schema. Browse through the datatypes, check and un-check the boxes to see which types are built-in and which are user-defined.

In the “Select a Datatype” window choose CustomerID user-defined type for the CustomerID column and click OK. You can begin typing the name of the datatype in the top window to filter the selection types displayed. Go through this exercise with every column in the Customer table choosing the appropriate datatype for each. Be sure to save your work. At the completion of the exercise, your Customer table should look like the one below.

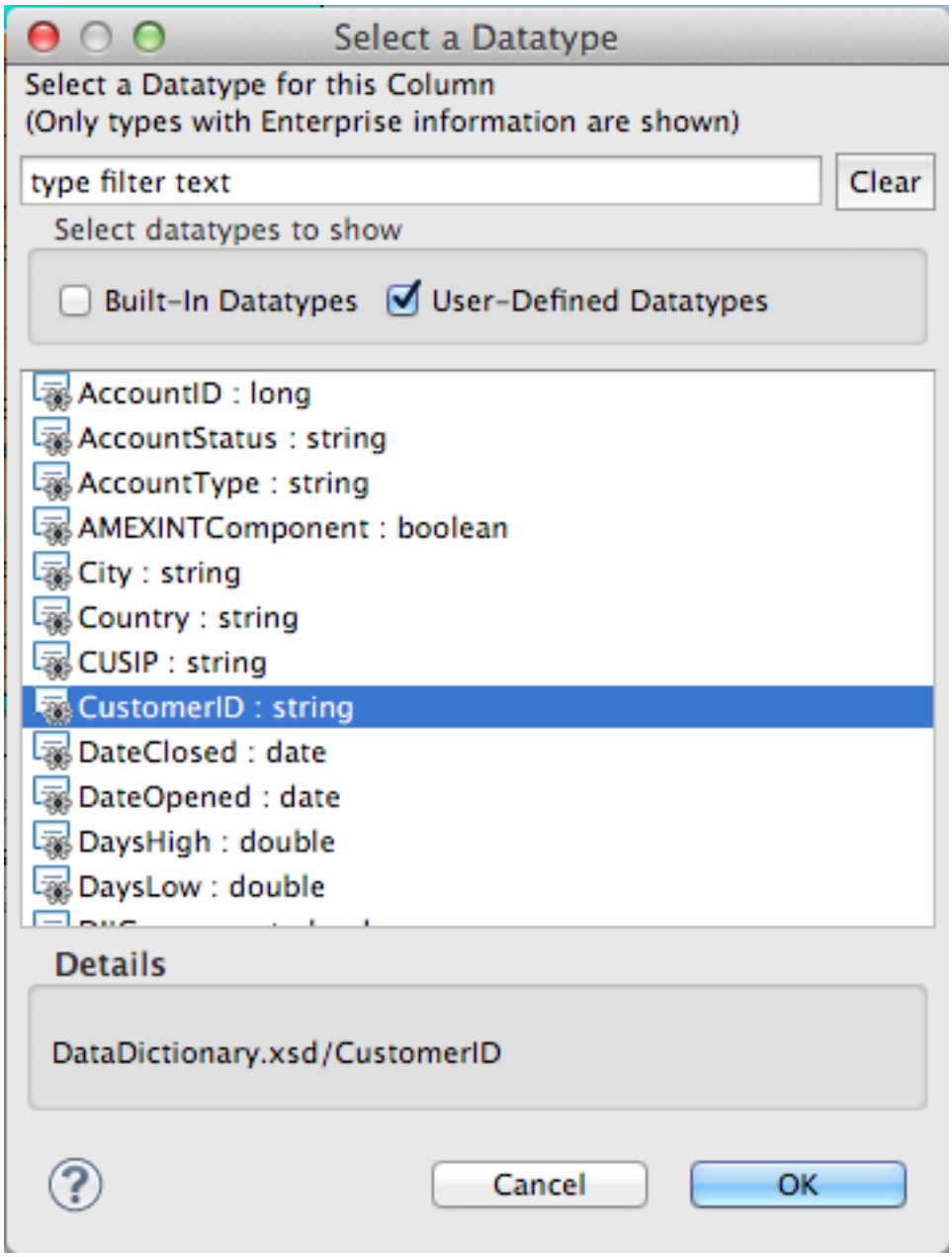
Select a Datatype
Select a Datatype for this Column
(Only types with Enterprise information are shown)
type filter text
Clear
Select datatypes to show
Built-In Datatypes
User-Defined Datatypes
AccountID : long
AccountStatus : string
AccountType : string
AMEXINTComponent : boolean
City : string
Country : string
CUSIP : string
CustomerID : string
DateClosed : date
DateOpened : date
DaysHigh : double
DaysLow : double
Details
DataDictionary.xsd/CustomerID
Cancel
OK

Figure 6.8.

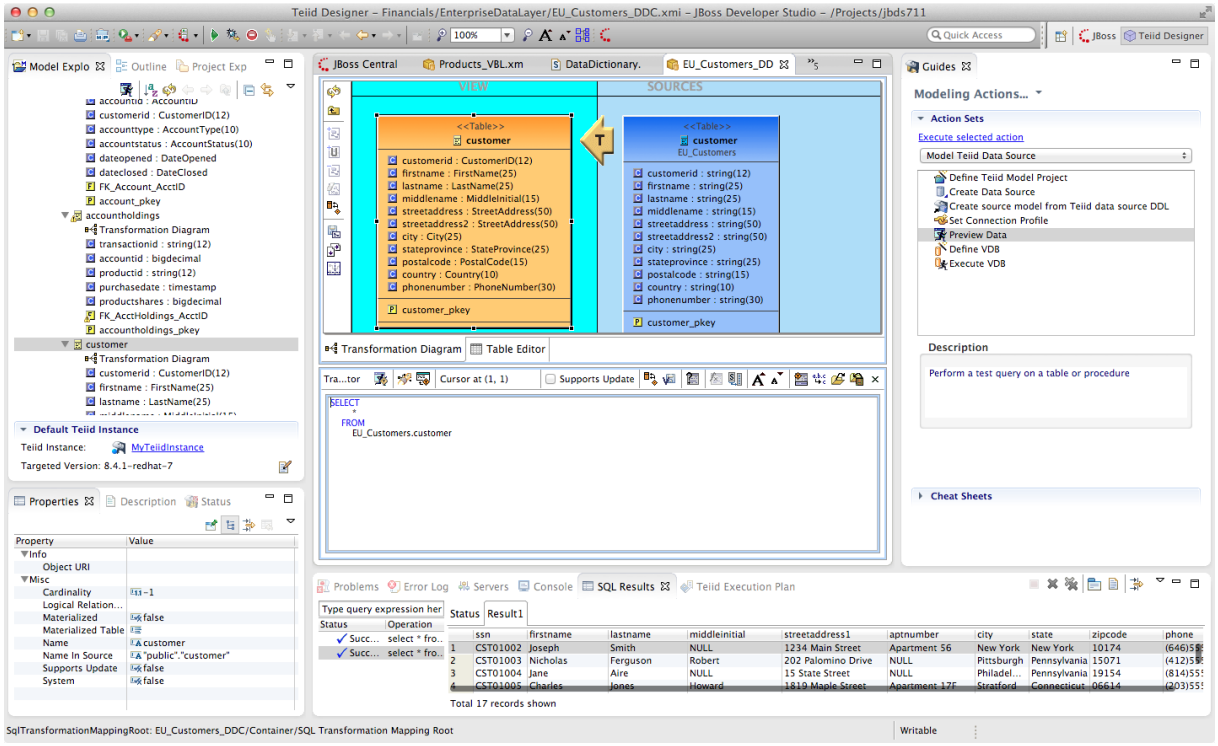

Figure 6.9.

Perform the same process for the other tables (Account, AccountHoldings) in the EU_Customer_DDC model. A way to "go back" to the parent model is to right-click on the teal background surrounding the View model and select "Show Parent Diagram". Alternatively, you can select the table you want to work with by double-clicking on it in the Model Explorer tree-view. Note: however when you save your changes the following warnings appear in the Transformation Editor: The SELECT transformation is valid, but NOT fully reconciled: The transformation output types do not match the target attribute types. This is because some of the enterprise datatypes we have chosen do not match the underlying format of the source data (AccountID, for example we are defining as a long, but it is a bigdecimal in the source). To address these type mismatches we will open the Reconciler. This is done by clicking on the small icon just to the right of "Supports Update" in the Transformation Editor.

Clicking on the Reconciler for AccountHolding brings up the following view.

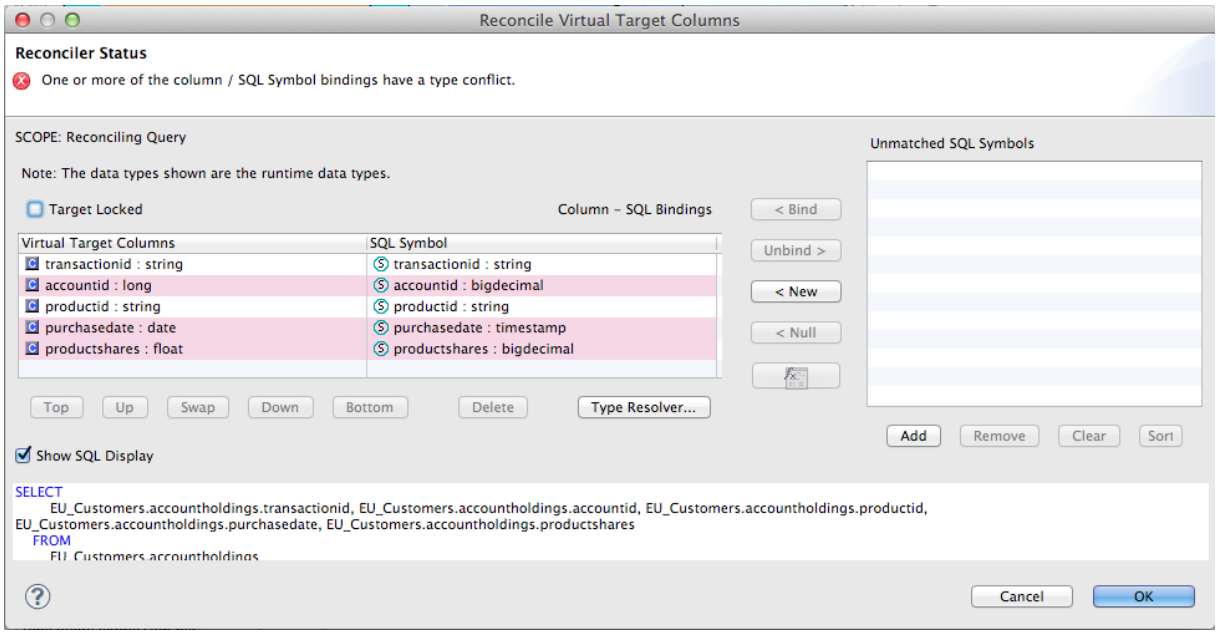

Figure 6.10.

The highlighted fields indicate some problems. We can address all of these in one go by clicking on the Type Resolver… button. That will bring up the following wizard.

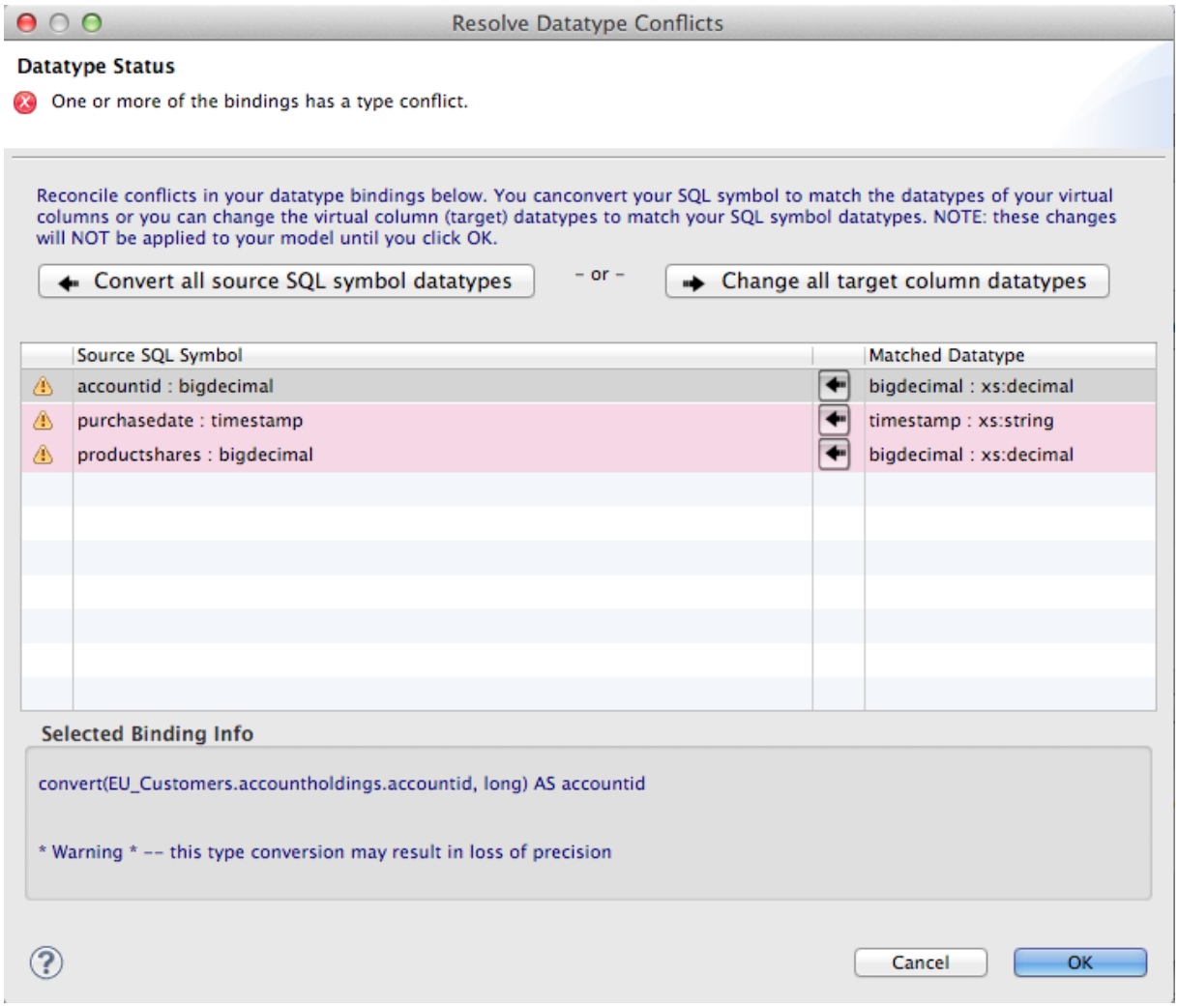

Figure 6.11.

Select each mapping at the top and see that the proposed transformation at the bottom changes. This is under the SQL Symbol panel; we want to convert to the datatype that is assigned to the enterprise datatype. (The Virtual Target Attribute section of the wizard above lets you modify that datatype; we don't want to do that here.) Note that there are three different type mappings being handled here: bigdecimal to long, timestamp to date, and bigdecimal to float. We are not worried about precision in this case, so we can simply press the Convert all button (the second one, in the SQL Symbol section), followed by clicking OK. This will return you to the Reconciler. Note that the SQL has been re-written in the transformation to handle all the type conversions. Click OK on the Reconciler to finish the process.

Now go back and perform the same steps with the account table. If we were worried about precision, we could take any number of steps to refine/modify the transformation including coding it by hand in the Transformation Editor, using one of the large set of

out-of-the-box functions provided with the product, or by creating our own User-Defined function.

When you are finished with the Account table, it will look like the following.

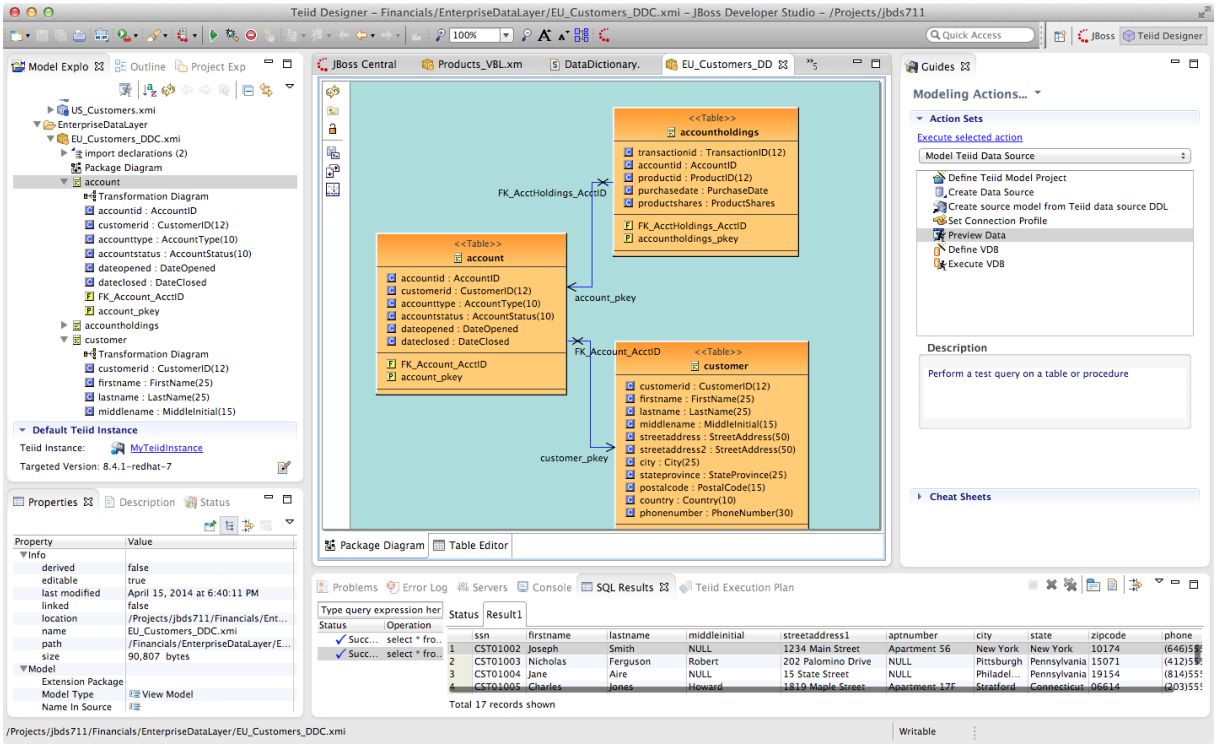

Figure 6.12.

6.4. Create the US_Customers_DDC Enterprise Data Layer

Now that we have finished with building the first enterprise data service layer in our model, we can take a short-cut to creating the same type of model for the US_Customers model. Essentially we are going to use the EU_Customers_DDC model as a template for creating the US_Customers_DDC model, and then replace the sources of the transformations for each of the tables with the correct ones. Here is how to do it: Right-click on the EnterpriseDataLayer folder and select New → Teiid Metadata Model. Fill in the wizard with the following fields (below) and click Next.

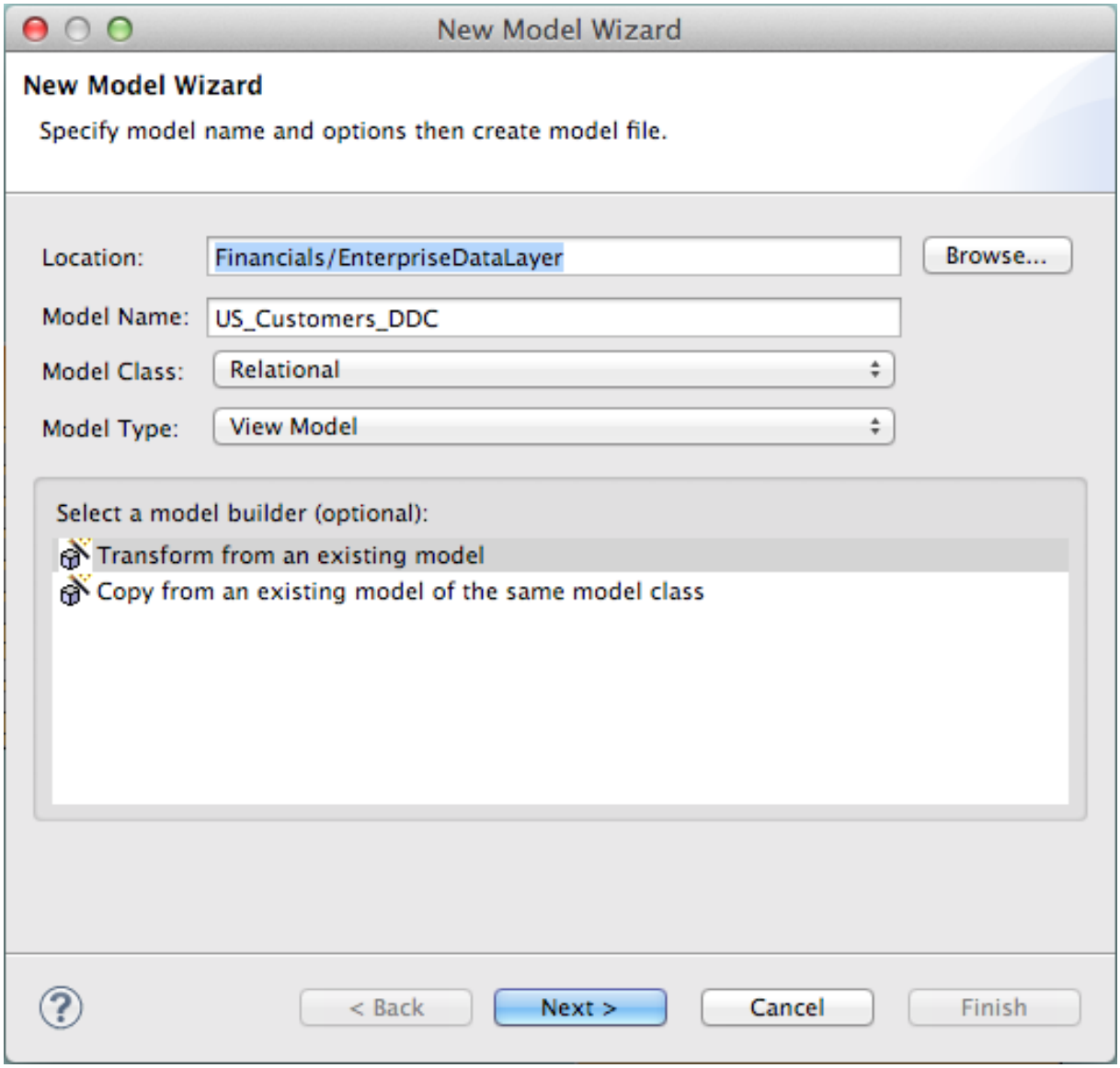

Figure 6.13.

In the “New Model Wizard” window choose the EU_Customers_DDC model in the EnterpriseDataLayer folder and click OK followed by Next > and Finish. Your selection should be as indicated below.

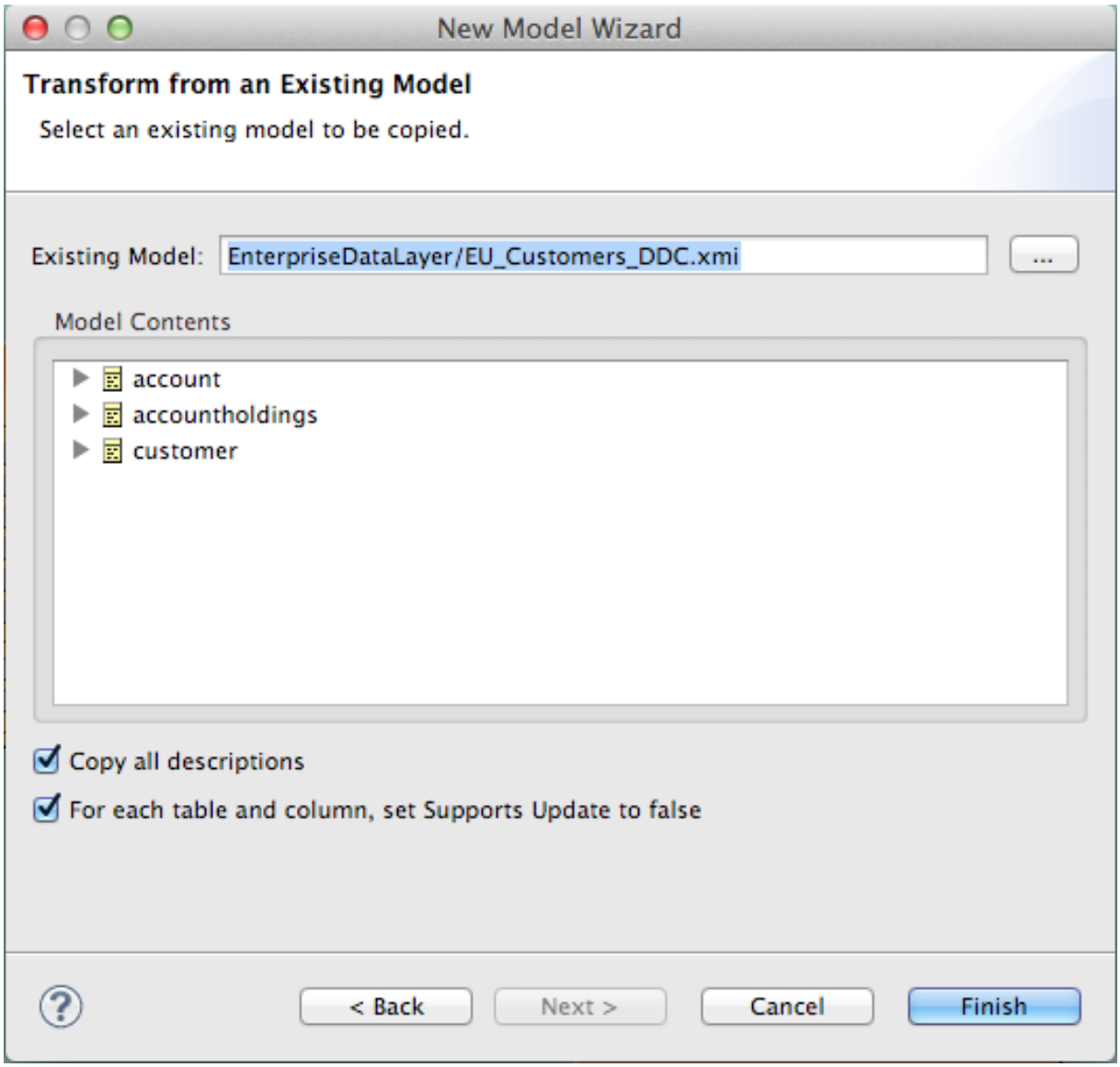

Figure 6.14.

Open the Transformation Editor on US_Customers_DDC.customer. Note that the Source of the transformation is the EU_Customers_DDC.eucustomer table. We want to replace that with the US_Customers_VBL table. Right-click on the Source table and select Remove Transformation Source(s).

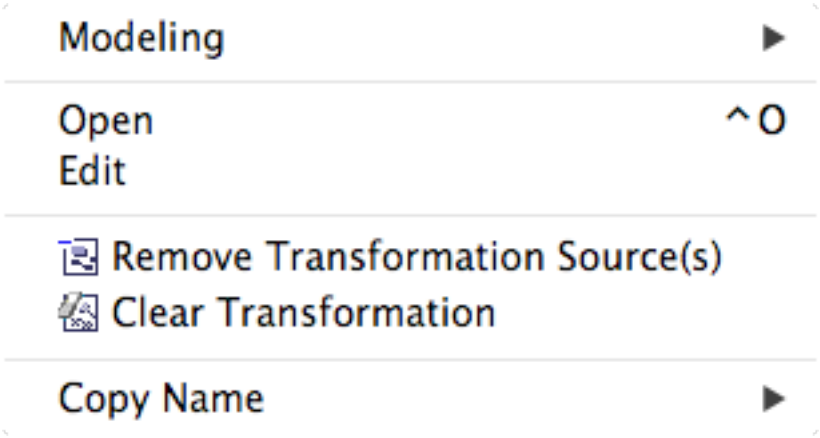

Figure 6.15.

The following pop-up window will be presented.

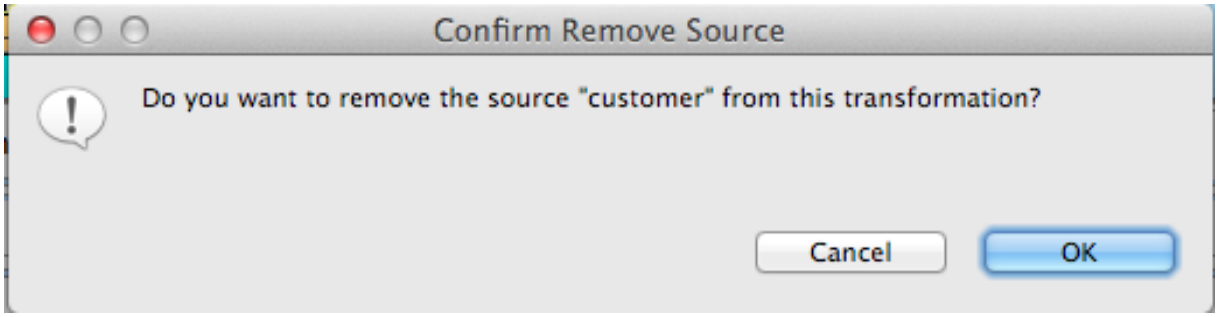

Figure 6.16.

Click OK. The following illustration indicates what your view in Teiid Designer should now resemble.

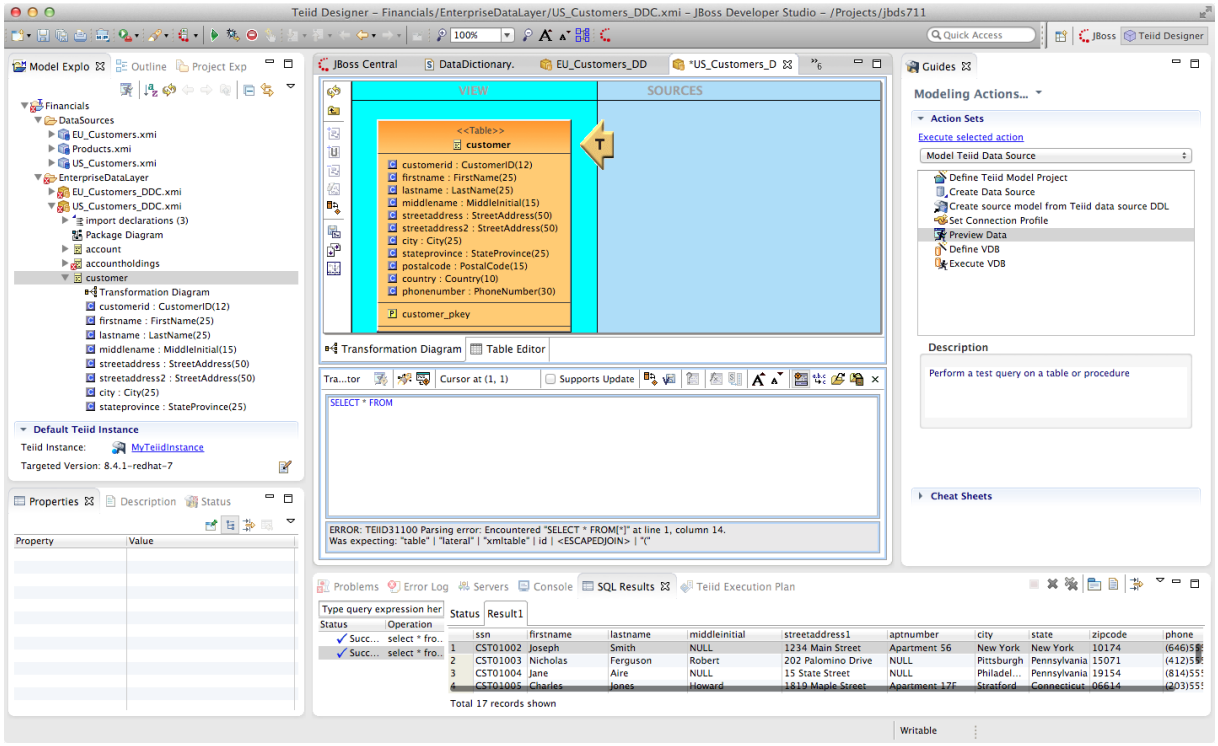

Figure 6.17.

Select the VirtualBaseLayer → US_Customers_VBL → customer table. This is highlighted in the illustration below.

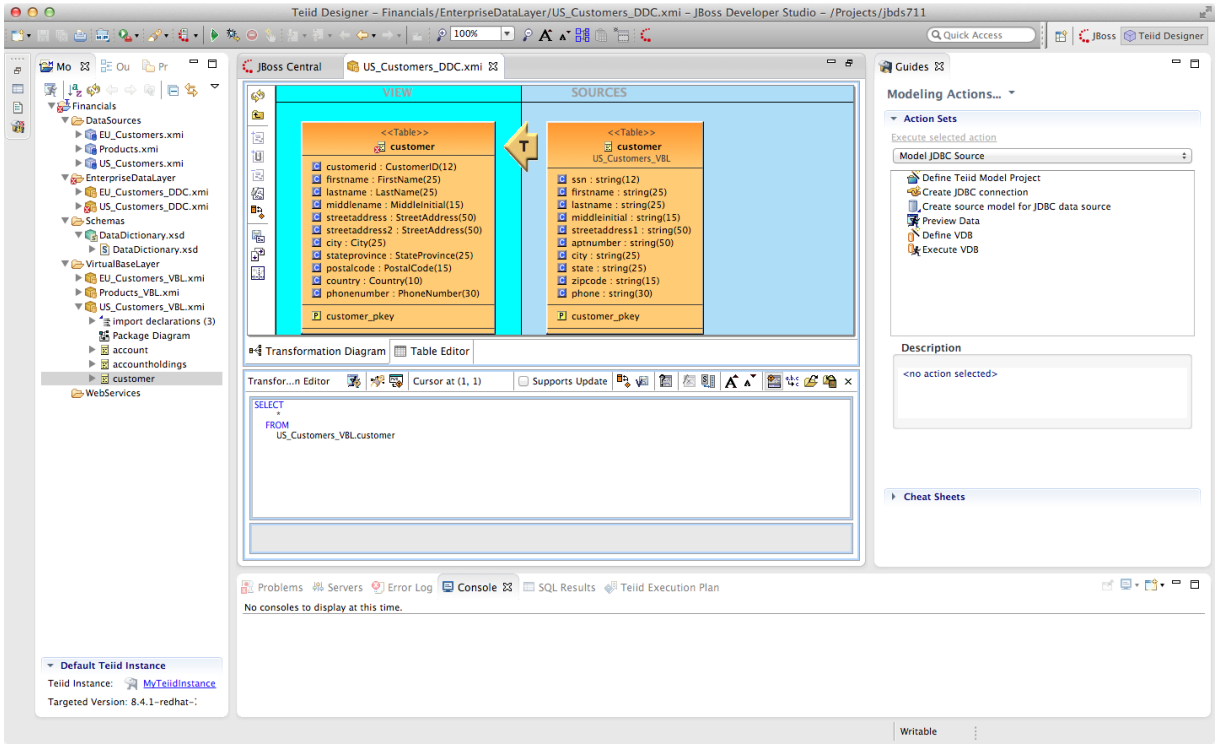

Figure 6.18.

There will be an error on our current model, US_Customers_DDC. At this point, we need to add a source model. Simply drag the highlighted customer table indicated in the previous illustration to the sources column on the right-hand side. This will add this table from our VirtualBaseLayer to our US_Customers_DDC model.

As is indicated on the model, the transformation is valid but is not fully reconciled. Click on the reconciler to bring up the wizard. As you can see from the reconciler, we have a bit of work to do.

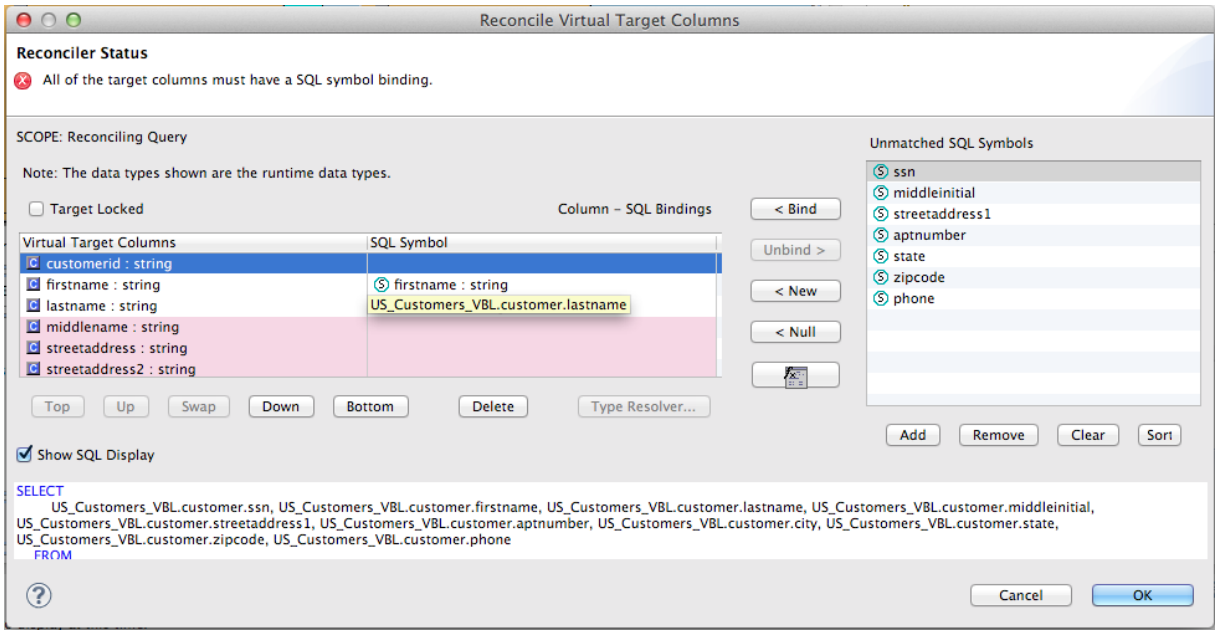

Figure 6.19.

There are two things that we need to do in order to fix this transformation.

1. Assign (Bind) variables that do not automatically match. By selecting the source on the left and the target on the right, we can then bind each of the following:
    - ssn to customerid
    - middleinitial to middlename
    - streetaddress1 to streetaddress
    - aptnumber to streetaddress2
    - state to stateprovince
    - zipcode to postalcode
    - phone to phonenumber (be sure to assign this on the left too and not map it to country!)

When we are finished, we have one more step.

1. Create a (simple) function to assign a value to Country as it does not exist in the source. To do this we will open up the Expression Builder by clicking on the "f(x)" button. This is right under the "< Null" button in the middle area of the wizard. Since all that is needed is a simple (static) assignment, the Expression Builder comes up with the following screen.

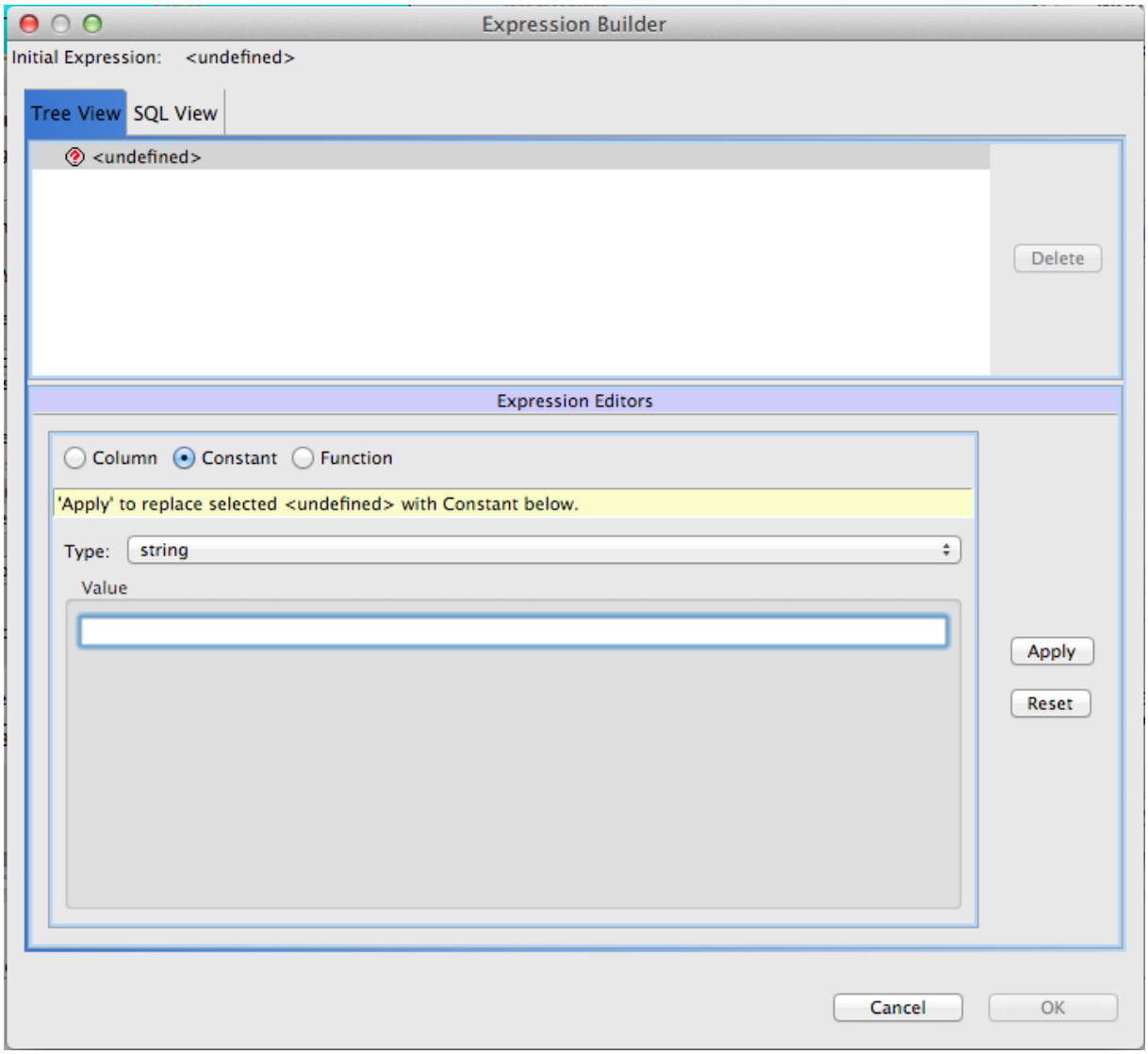

Figure 6.20.

All we need to do is type "USA" into the Value field, click Apply, then OK. However, while you are in the Expression Builder, you may want to select the Function radio button to check out the many out-of-the-box functions and operations that ship with JBoss Data Virtualization. When you are finished, be sure to set it back to Constant and complete the instructions as outlined above.

When the Expression Builder exits back into the Reconciler, you will notice that the function ('USA' AS Country) has been properly assigned. Click OK in the Reconciler and save your changes. Now perform the same process with the other two tables (Account and AccountHoldings). Delete the EU_Customers_DDC source and drag & drop the appropriate US_Customers_VBL source and perform any necessary reconciliations. Again, you can go through the required steps to Preview data that was outlined in an earlier lab. The Data Dictionary has also defined enterprise types for Product data. Create a Products_DDC model in the EnterpriseDataLayer folder, source it from the Products_VBL model, and correct the datatypes in the DDC model. Finally, reconcile any datatype conversion issues.

Congratulations, you have now completed this lab.

Chapter 7. Create a Data Federation Layer

Now that we have created our Virtual Base Layer (to isolate us from changes in the sources), defined an enterprise data dictionary to semantically reconcile the various terminologies in our disparate data sources, and created an Enterprise Data Layer that captures those mappings, we are ready to integrate some different data sources. And, since we have the ground work already, it could not be simpler.

7.1. Create a Data Federation Layer

Right-click on the Financials Project and go New → Folder to create a new folder called “FederatedDataLayer”. Create a new Teiid Metadata Model in this folder called “All_Customers". Create it by transforming it from the EU_Customers_DDC model. Your new model wizard should look like the illustration below.

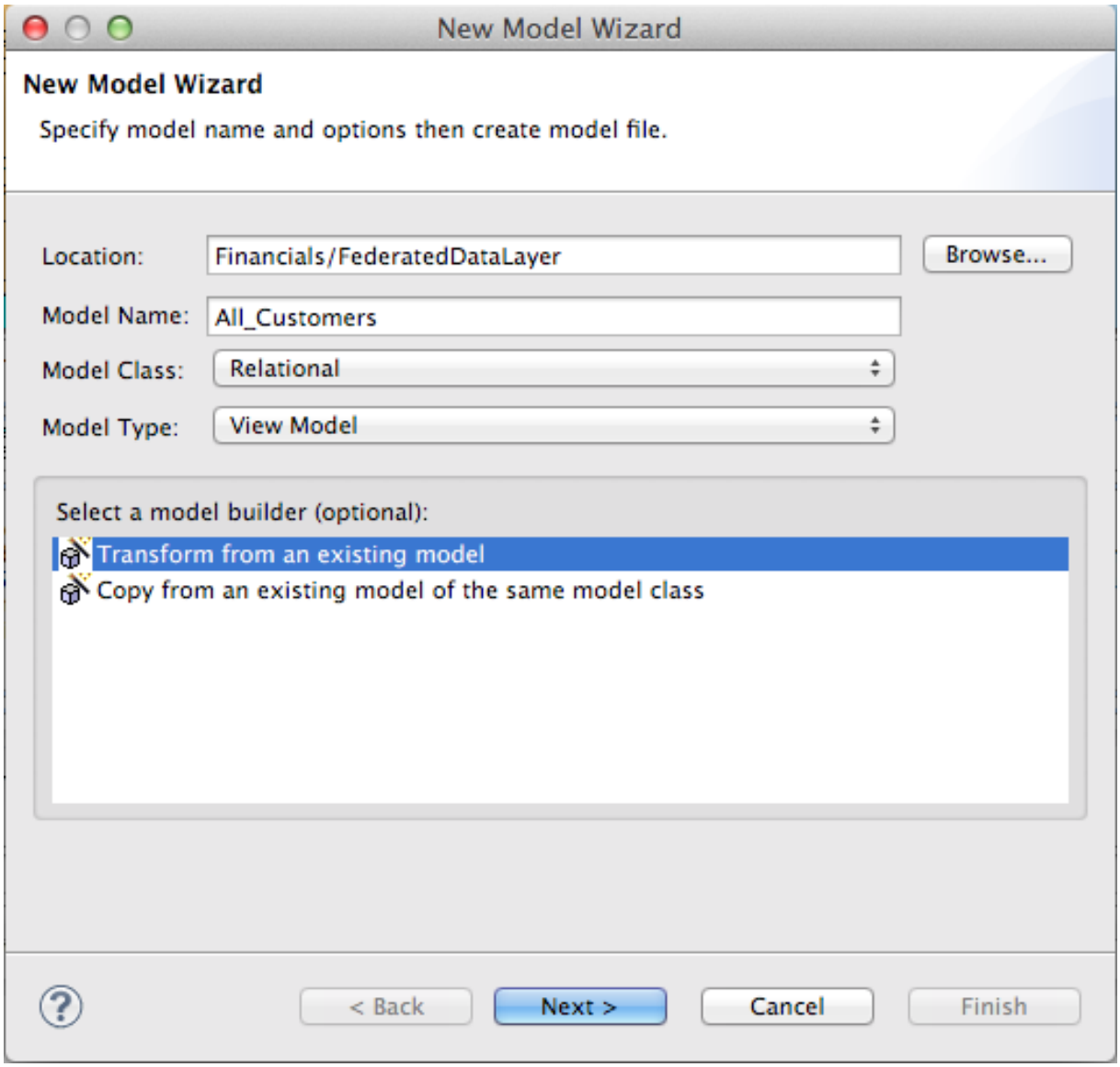

Figure 7.1.

After clicking the Next button, be sure to select the EU_Customers_DDC.xmi model as indicated below.

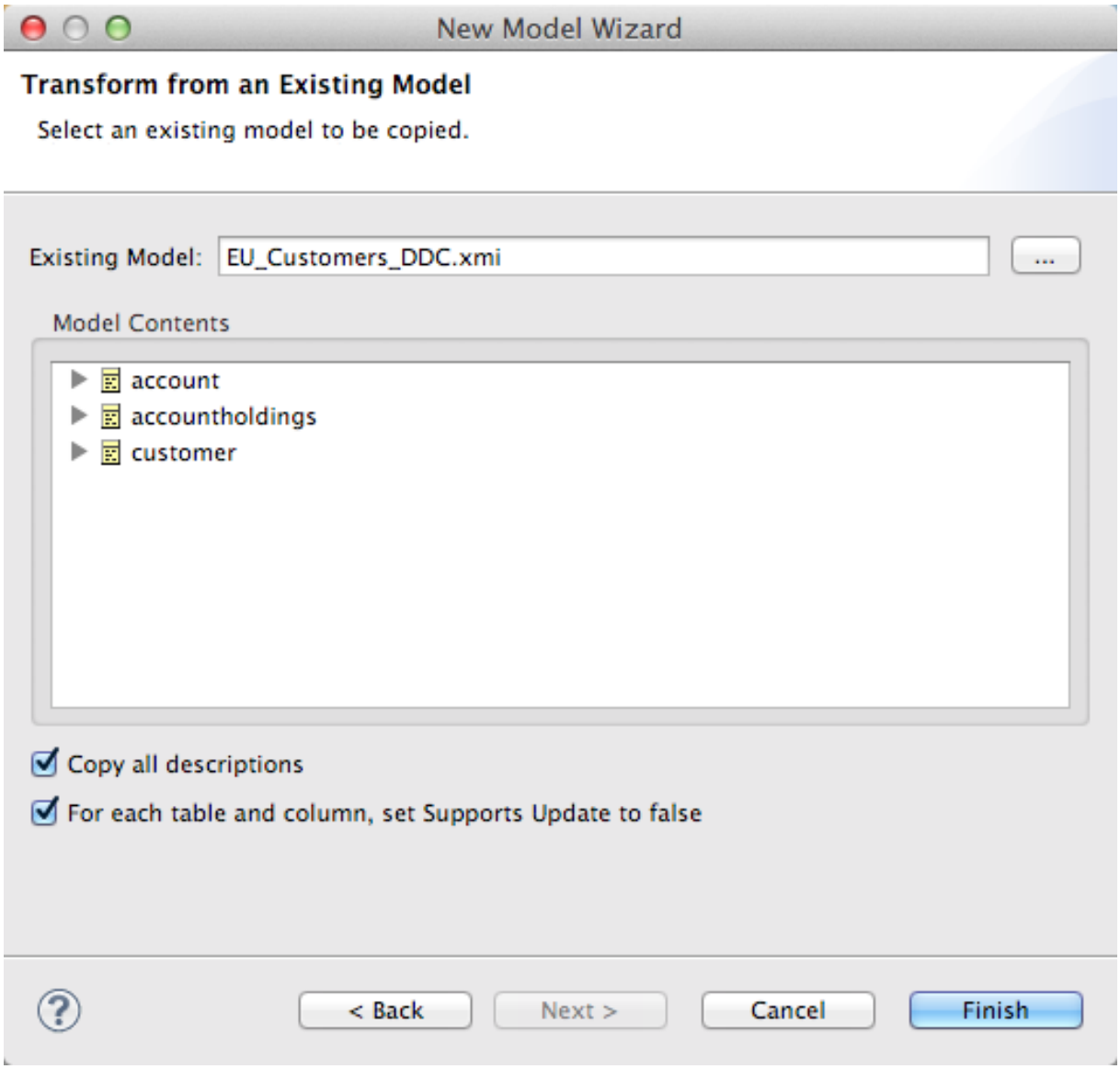

Figure 7.2.

Now, open up the Transformation Editor for All_Customers.customer. In the Model Explorer on the left, open up the Enterprise Data Layer → US_Customers_DDC and highlight the customer table as indicated in the illustration below.

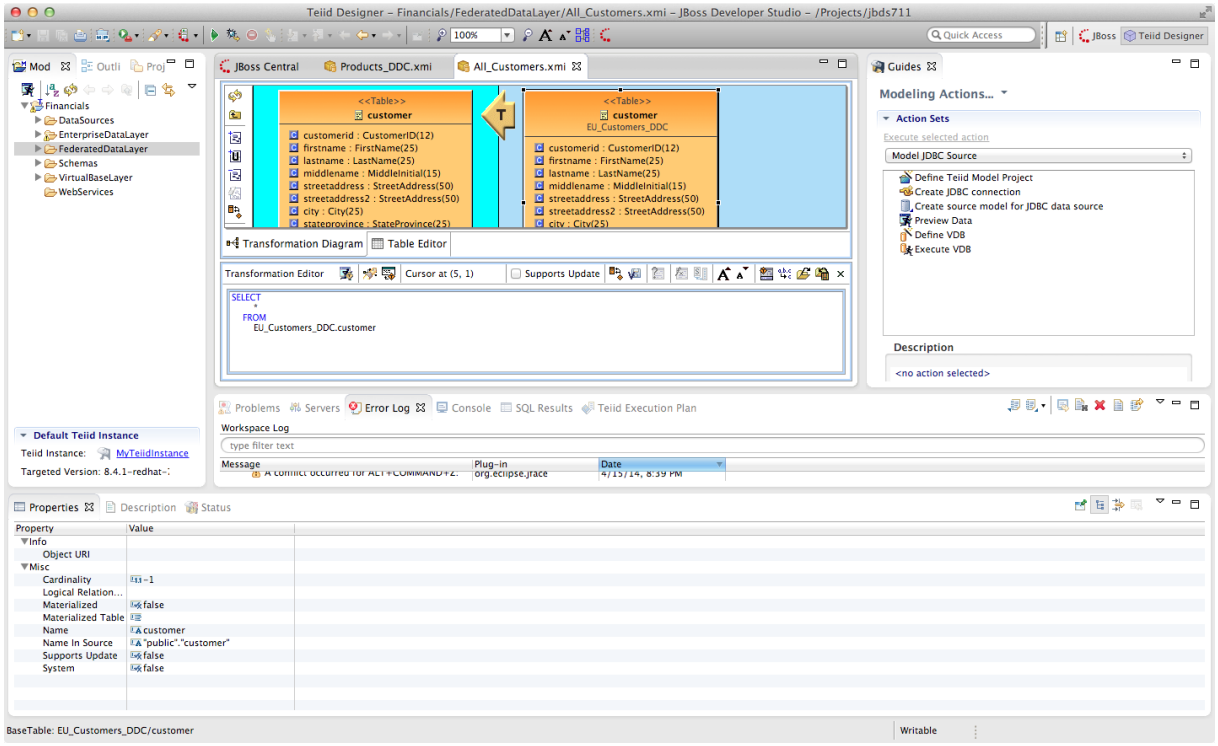

Figure 7.3.

Several new icons have been activated in the middle tool bar. Click on the fourth one down on the middle tool bar. If you hover your mouse over the label it reads "Add Union Transformation Source(s)". It is possible that you may have to click this button twice for the action to occur. This will add the customer table from US_Customers_DDC to the All_Customers model. Additionally, the "T" icon will now have a small "U" to the lower-right indicating that there is a union that is responsible for this transformation. Now, all that is required is to save the model. That is it! You now have a union of two tables from two separate databases. In order to view the result, be sure the customer table is highlighted in the Model Explorer under the FederatedDataLayer. This is indicated in the illustration below.

With the All_Customers.customer table highlighted in the Model Explorer, the "Running Man" icon is now enabled. This icon is located on the toolbar directly below the Model Explorer tab. Click on this icon to preview the data of the combined eu_customers and us_customers database tables. If for some reason this does not work, select the Preview Data Action on the right-hand side of JBDS that was utilized in earlier labs to preview the data. Now that the exercise for federating the customer table is complete, give it a try for the other two tables, account and accountholdings.Clearly, this is a simple example where we assume that the entries in the US Customers database and

the EU Customers database do not overlap. But, JBoss Data Virtualization is capable of very sophisticated modeling and transformations.

Congratulations, you have now completed this lab.

Chapter 8. Create a Web Service

In this lab, you will be utilizing the Teiid Designer and JBDS to create a Web Service. It should be noted that this lab is not a guide on Web Services or how or when to use them. Rather, this is an instructional approach for some of the advanced capabilities of Teiid Designer. To begin, right-click on the All_Customers.xmi model in the FederatedDataLayer folder and select option Modeling → Create Web Service. This is indicated in the illustration below.

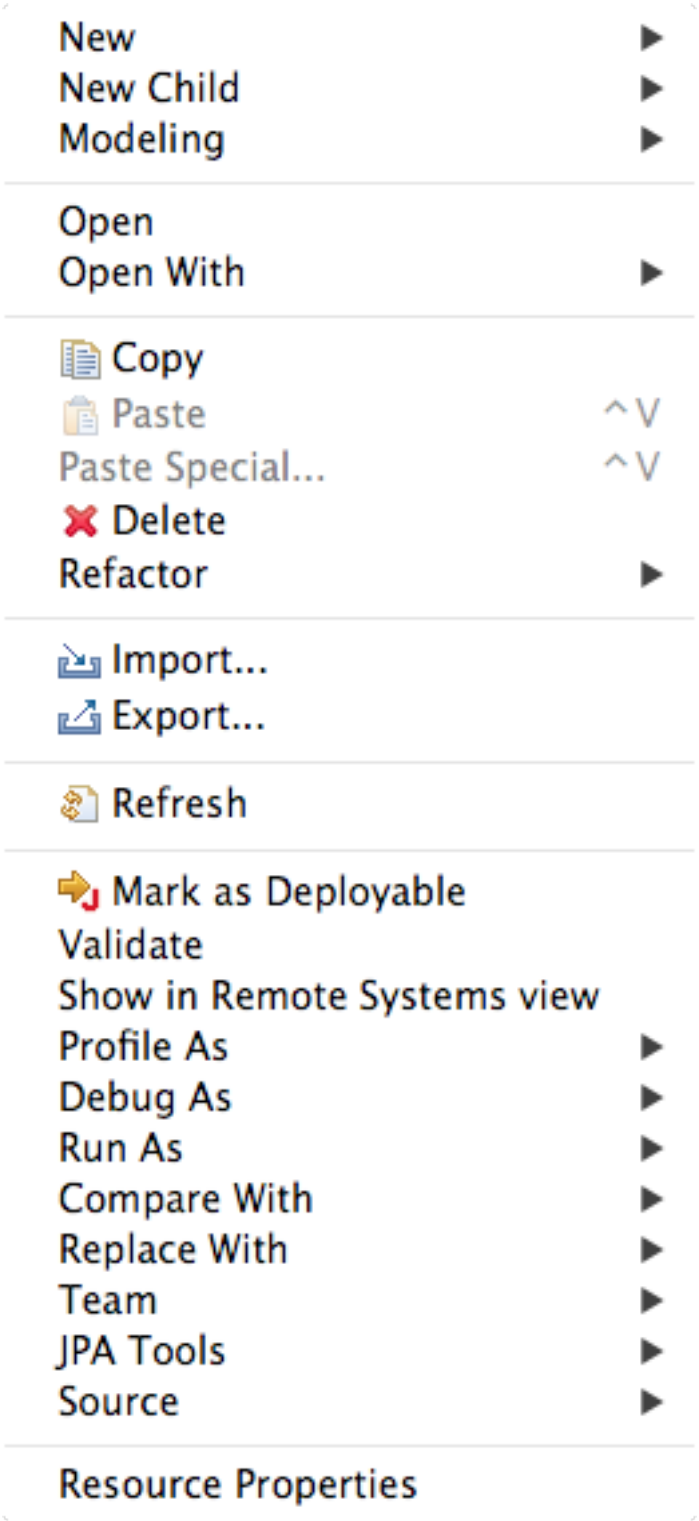
New
New Child
Modeling
Open
Open With
Copy
Paste ^V
Paste Special... ^V
Delete
Refactor
Import...
Export...
Refresh
Mark as Deployable
Validate
Show in Remote Systems view
Profile As
Debug As
Run As
Compare With
Replace With
Team
JPA Tools
Source
Resource Properties

Figure 8.1.

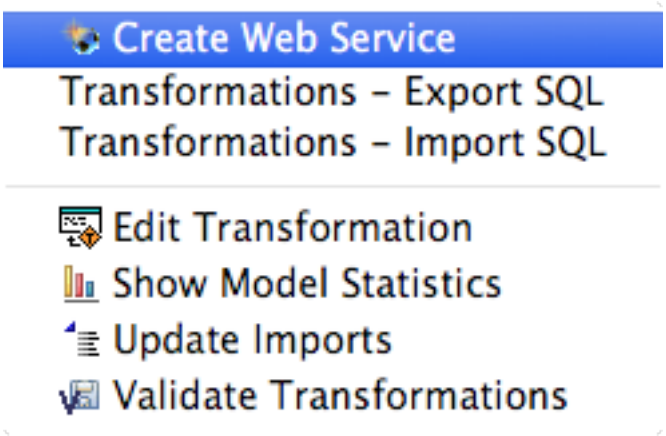

Figure 8.2.

This will bring up the following wizard.

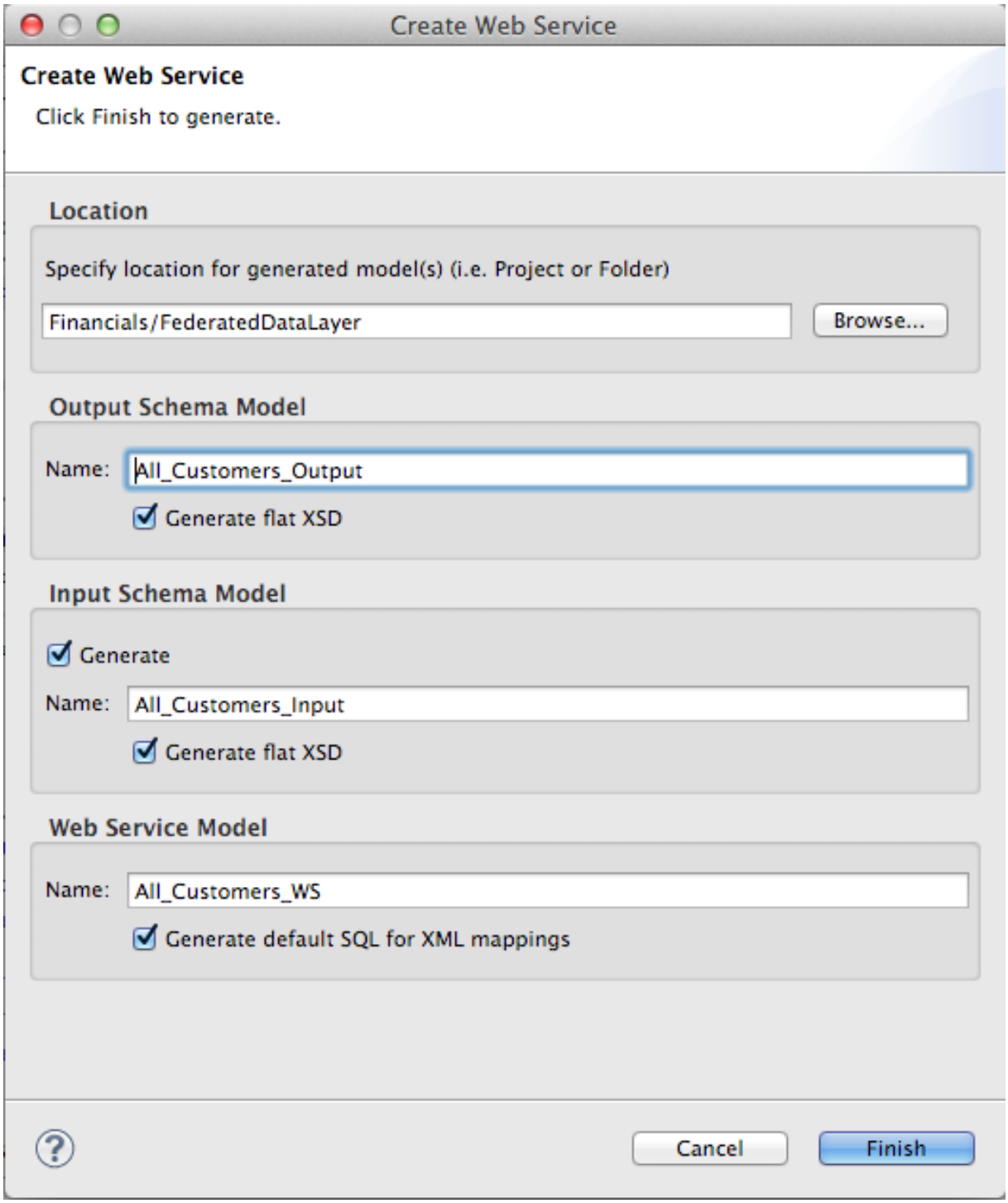

Figure 8.3.

In order to keep things organized, be sure to select “Browse” and select the WebServices folder we created in an earlier lab. This will drop all web service artifacts within this folder. Then, click Finish. If the generation was successful, the following result will occur.

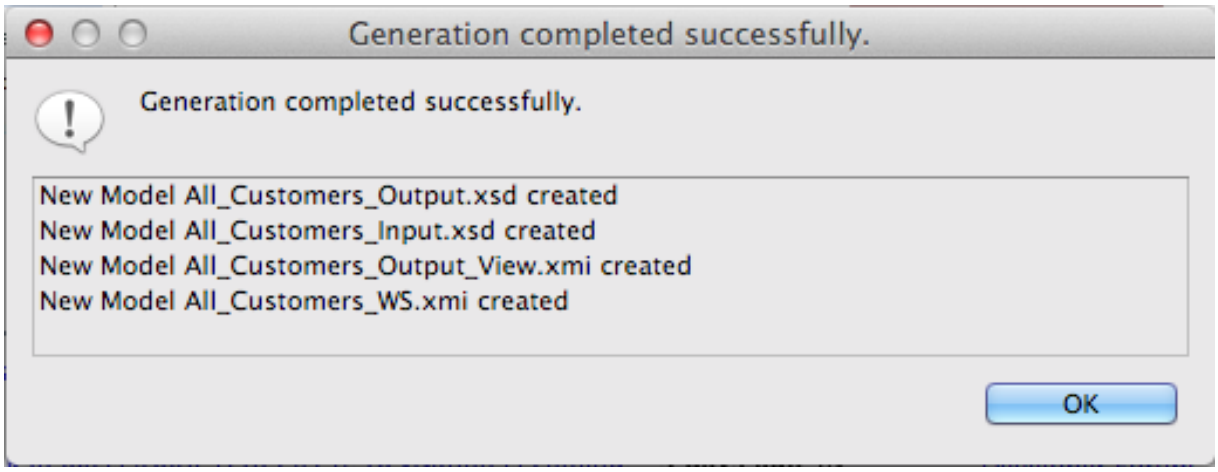

Figure 8.4.

The generated Web Service will open after clicking OK. Your Teiid Designer view should resemble that below.

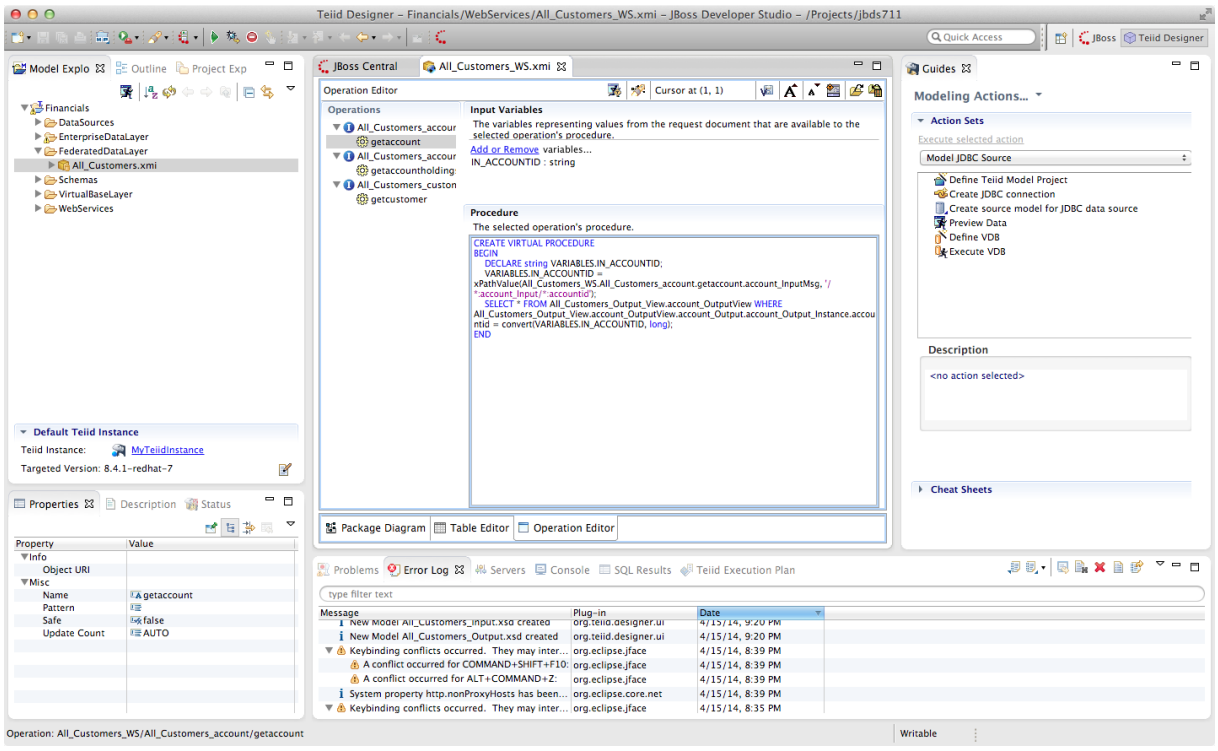

Figure 8.5.

If not selected, be sure to click on the “getcustomer” operation as indicated below. With the getcustomer operation highlighted, click the “Running man icon with a diamond T”. This icon is located across the top of the All_Customers_WS.xmi view and to the right of the Operation Editor. Click this icon to bring up the following window.

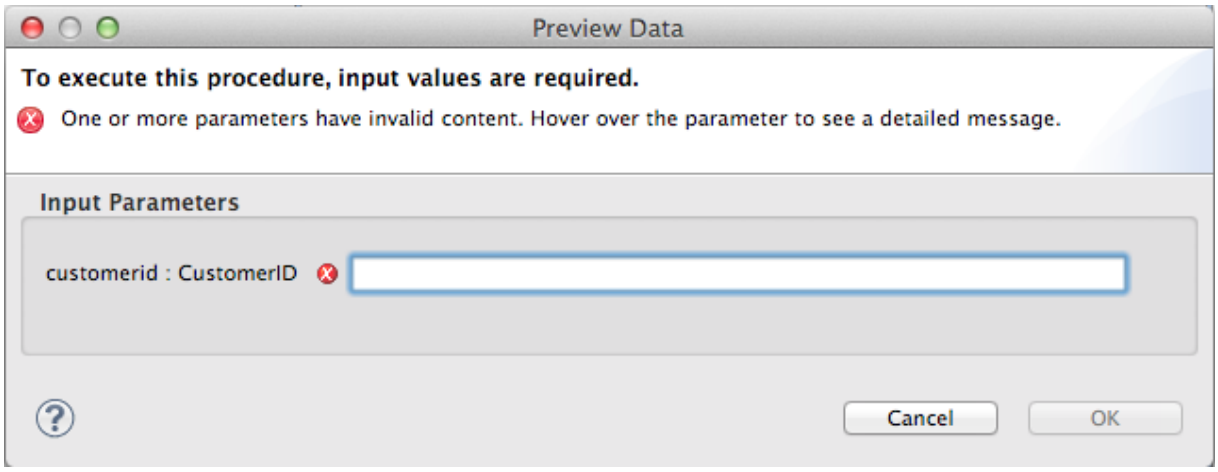

Figure 8.6.

When initially opened, the customerid : CustomerID field will be blank. Enter in the value "CST01033". Then, click OK.

Upon inspection of the Result1 tab reveals the XML output of the Web Service invocation as indicated below.

```
<?xml version="1.0" encoding="UTF-8"?>
<account_Output:customer_Output   xmlns:account_Output="http://
www.metamatrix.com/account_Output"  xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
 <customer_Output_Instance>
  <customerid>CST01033</customerid>
  <firstname>Alexander</firstname>
  <lastname>Franken</lastname>
  <middlename>Horst</middlename>
  <streetaddress>Berliner Platz 85</streetaddress>
  <streetaddress2 xsi:nil="true"/>
  <city>Munich</city>
  <stateprovince xsi:nil="true"/>
  <postalcode>80806</postalcode>
  <country>Germany</country>
  <phonenumber>0890877435</phonenumber>
 </customer_Output_Instance>
</account_Output:customer_Output>
```

Congratulations, you have now completed this lab.

Chapter 9. Virtual Database Deployment

We have been doing all of our querying directly through the Preview interface of Teiid Designer, but in order to make our data services available to external clients, we will need to package them up into a Virtual Database (VDB), the deployable artifact that drives the run-time behavior of the server. It is exactly analogous to a WAR or an EAR file; once created it can simply be dropped into the deploy directory of a running server (with JBoss Data Virtualization installed) and the Teiid Server process will hot-deploy the data services modeled within it. The process is very simple.

9.1. Create the VDB Metadata Model

Right-click on the Financials project (top layer folder in the Model Explorer) and select New → Teiid VDB.

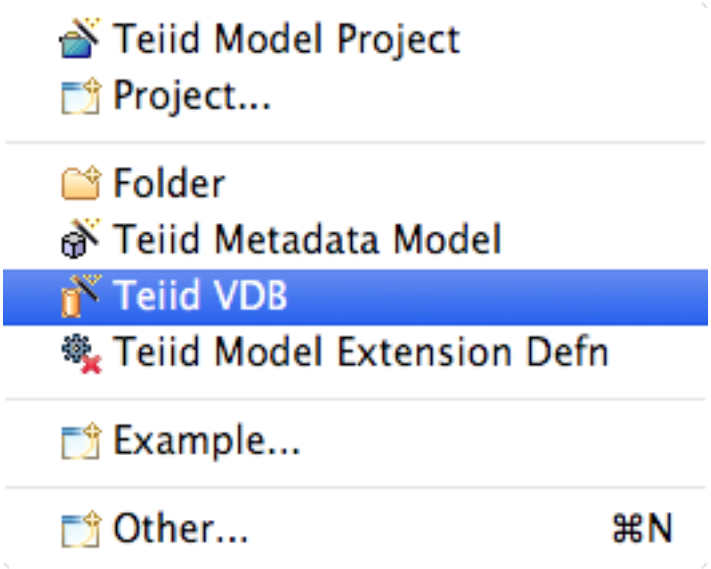

Figure 9.1.

This will open the New VDB wizard. The New VDB wizard is part of the illustration below.

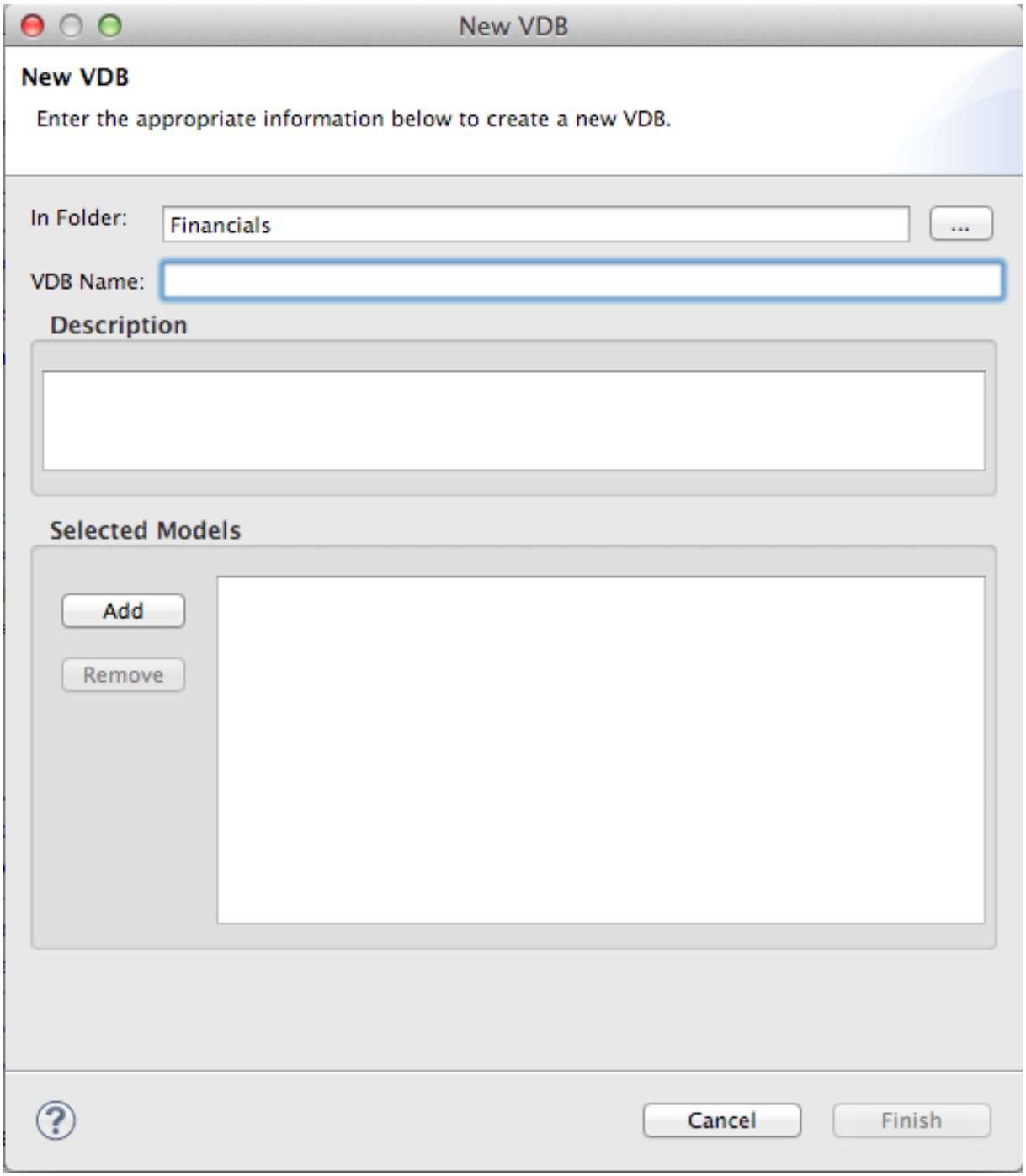

Figure 9.2.

Enter a VDB Name, “Financials”. Next, click the Add button in order to select the model(s) to add to the VDB. Next, select the Add button so we can select a model(s) to add to the VDB.Models wizard as indicated in the illustration below. This will bring up the Select Models window.

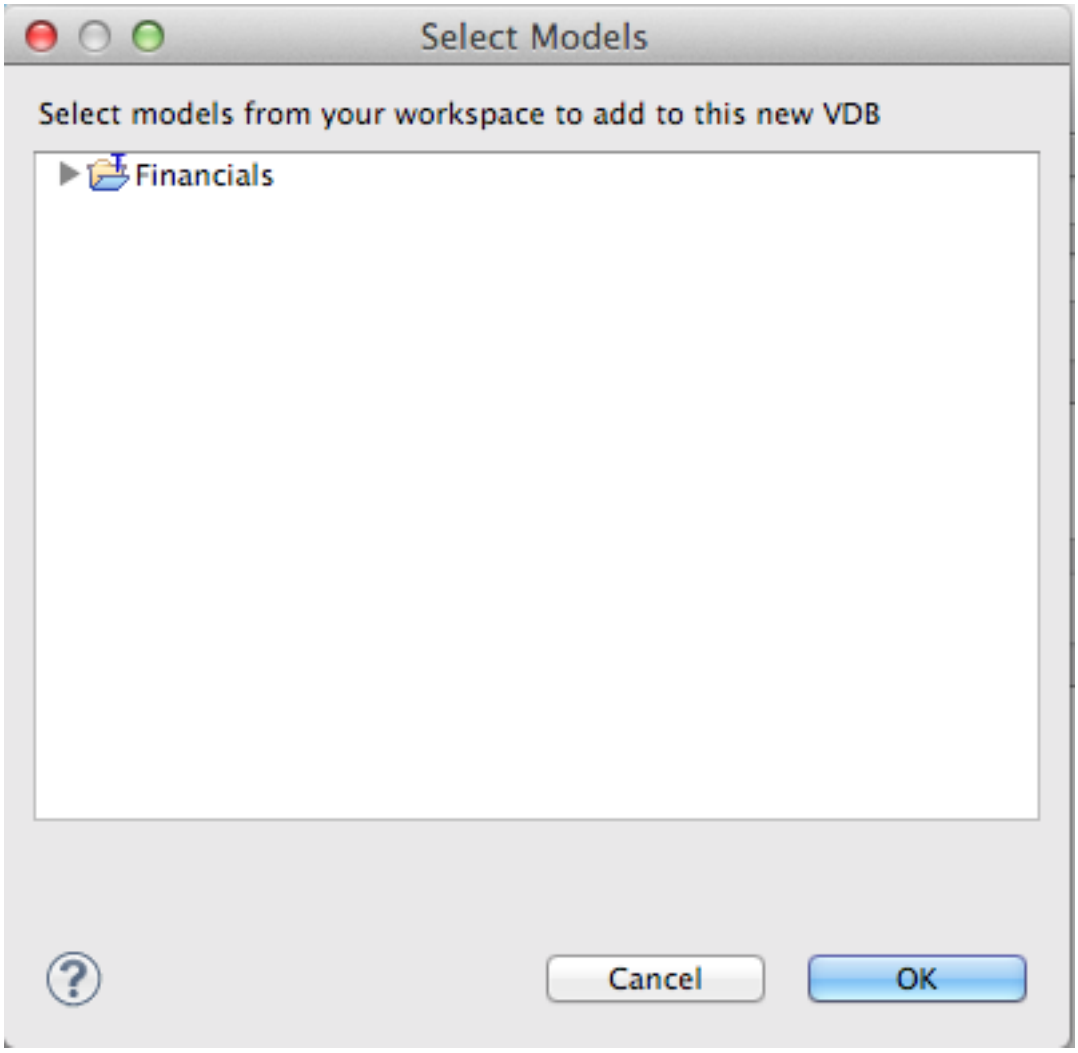

Figure 9.3.

For this lab, we will select the Financials → FederatedDataLayer → All_Customers.xmi model and click the OK button. Your New VDB wizard should look like the illustration below. At this point, click the Finish button.

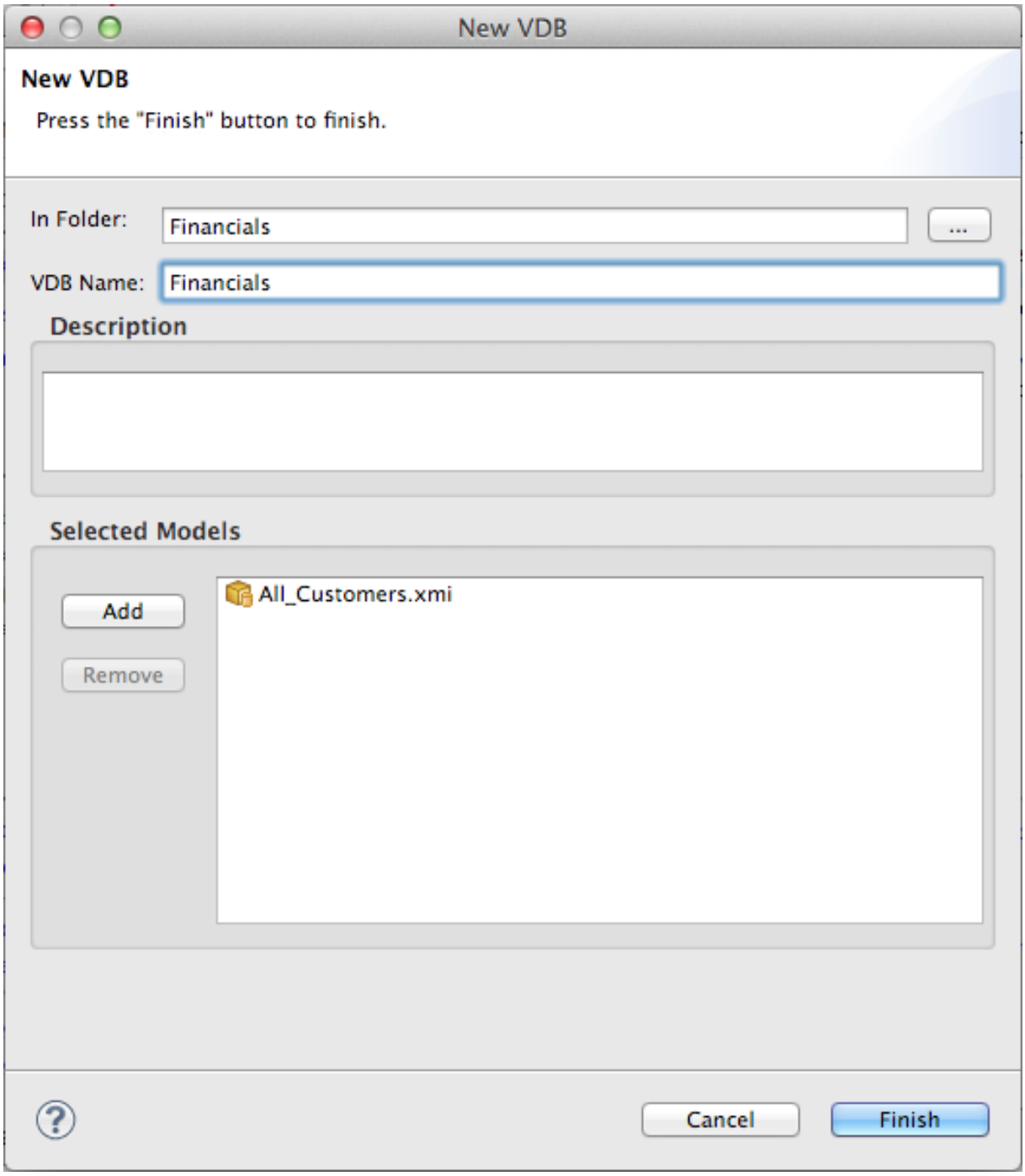

Figure 9.4.

This will now open the VDB Viewer in Teiid Designer. Your view should look like the illustration below.

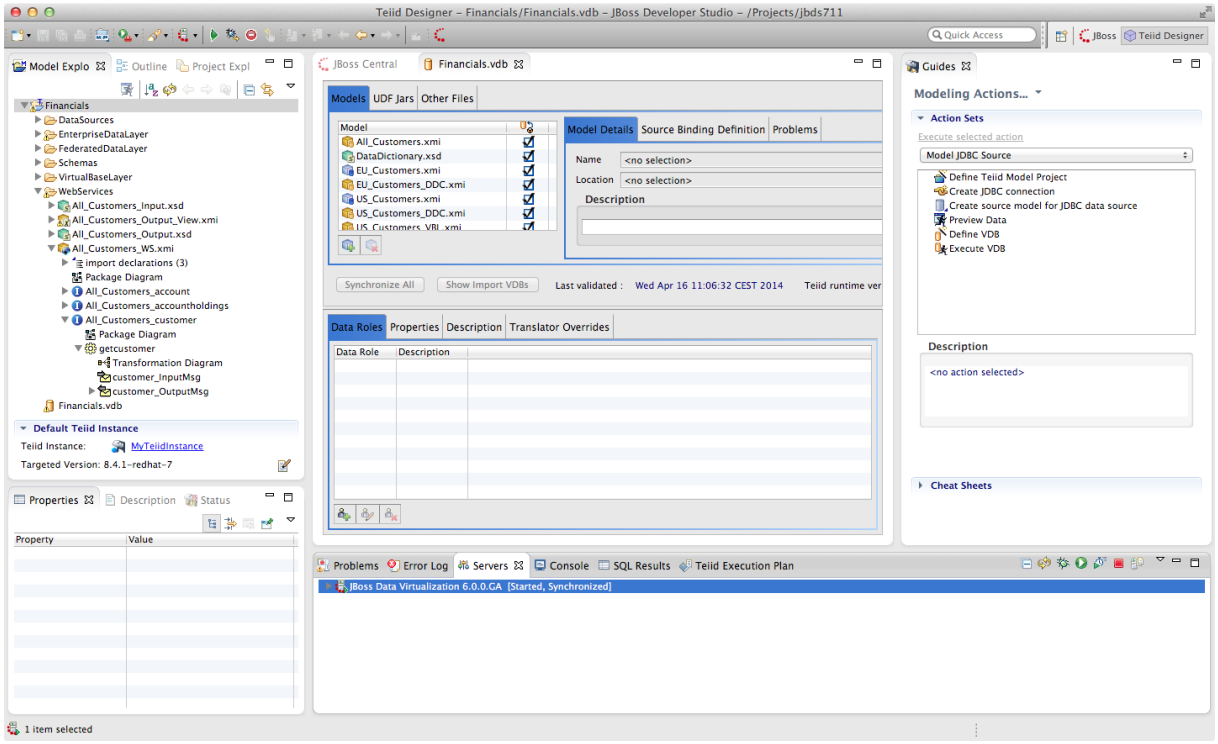

Figure 9.5.

All models, virtual and physical (as well as the DataDictionary schema we used) have been brought into the VDB. This is because the Query Engine will need all of the models and their associated metadata/transformations, in order to drive the run-time behavior. However, that does not mean that the data services developer is forced to expose all of these more granular data services if they should choose not to. Indeed it is a best practice to at least completely hide the source data systems, to prevent users of the virtual layer from going directly to the sources. In this way JBoss Data Virtualization can add additional layers of security and authorization/ authentication to protect sensitive data. To illustrate this, uncheck the boxes in the second column (annotated with the magnifying glass icon) on the physical models. This will make them invisibile/ unavailable to any client connecting to this VDB. The models are still there (and must be, for the rest of the federation to work), but they cannot be accessed other than through the higher-level data services that have been defined.

Here is what the view should look like with the physical source visibility turned off. Specifically, the EU_Customers.xmi and the US_Customers.xmi have their visibility turned off.

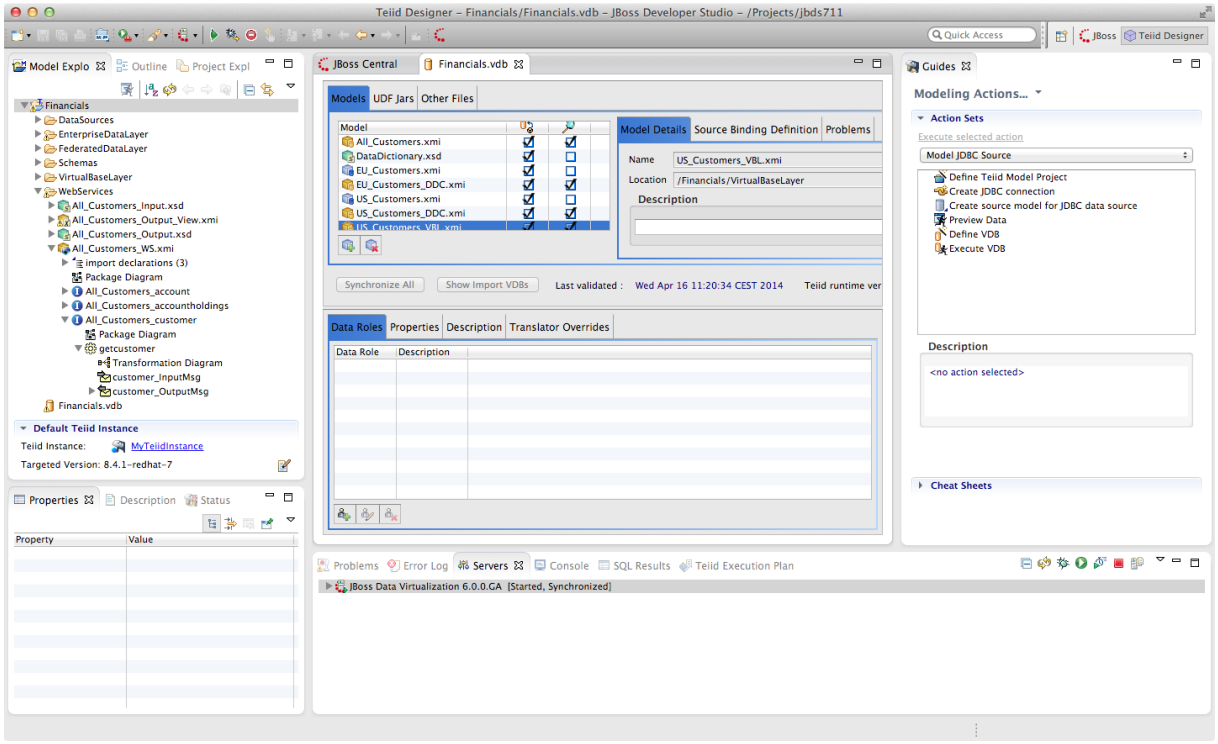

Figure 9.6.

Be sure to "Save All" to save all of the changes to the VDB.

9.2. Create data sources on the JBoss Data Virtualization server

There is one thing we need to do before we deploy the VDB to the server. We need to install the data sources into the JBoss Data Virtualization server's deploy directory. This can be done in a number of ways. If we have the DataSource-ds.xml file we can simply drop it into the deploy directory. Note however that the JNDI name that the VDB expects must match. The JNDI name the VDB expects is in a column named JNDI Name. For this example, the JNDI names are EU_Customers and US_Customers. This column is visible in the illustration below.

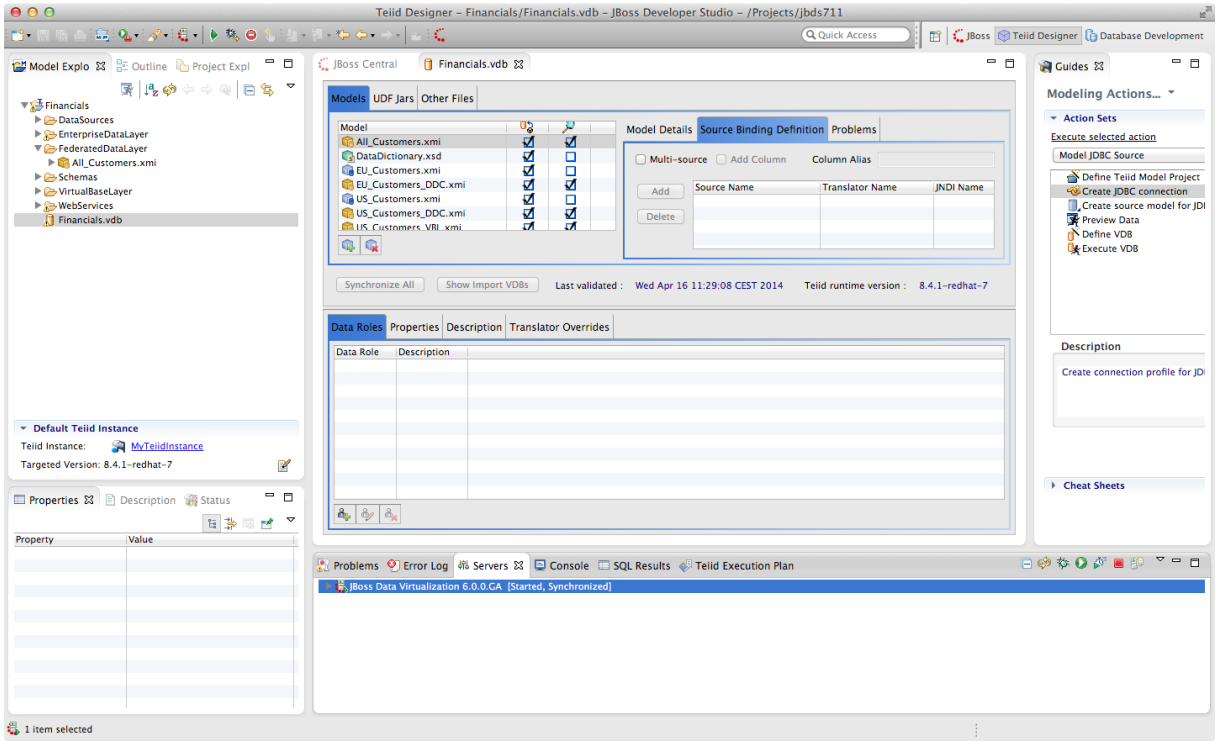

Figure 9.7.

We can also use Teiid Designer to pass the data source information that we captured in our model on to the Server. Not doing this automatically is an intentional security feature. To begin creating the required data sources, we need to ensure that the Teiid Panel is open. The panel is located in the Teiid tab along the bottom portion of JBDS. In fact, this is a "view" into the Teiid Instance that we created a connection to in an earlier lab. In the Teiid tab, there will be a top-level folder called MyTeiidInstance. Again, this name will be the name that you used in an earlier lab when we were connecting to this instance. Below the top-level folder, there are two additional folders, Data Sources and VDBs. The view of this tab can be seen in the illustration below.

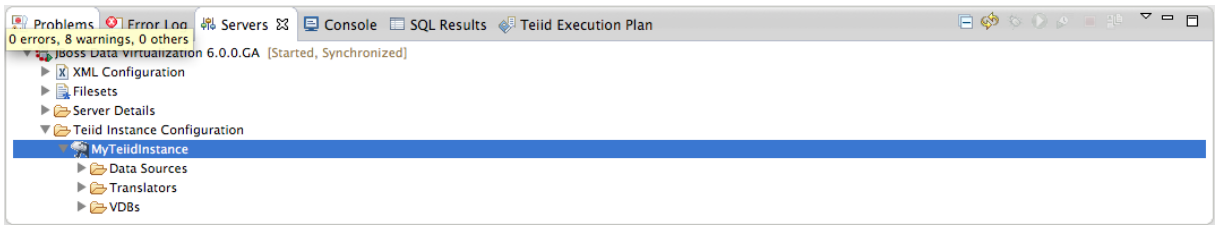

Figure 9.8.

Once complete, you should note that these data sources are already available, along with some of the internal sources used by JBoss Data Virtualization. Your Data Sources view should look like that below.

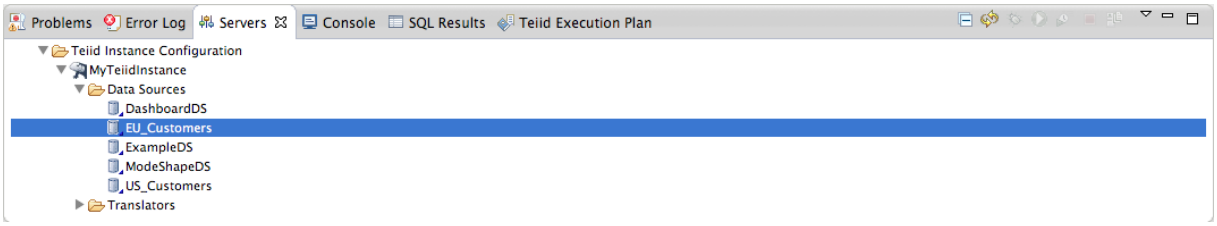

Figure 9.9.

To add new Data Source right-click on Data Sources. This will bring up a menu option to create a new data source. This will open the Create Data Source Wizard. At this point, we are ready to deploy the VDB to the server. This could not be easier. Simply right-click on Financials VDB model and select Modeling → Deploy. This will deploy the VDB to the running Teiid Server instance. Now, open up a browser and point it to http://localhost:8080. This will open the console for administrative services for the JBoss Data Virtualization server. Click on "Administration Console". This is displayed in the illustration below.

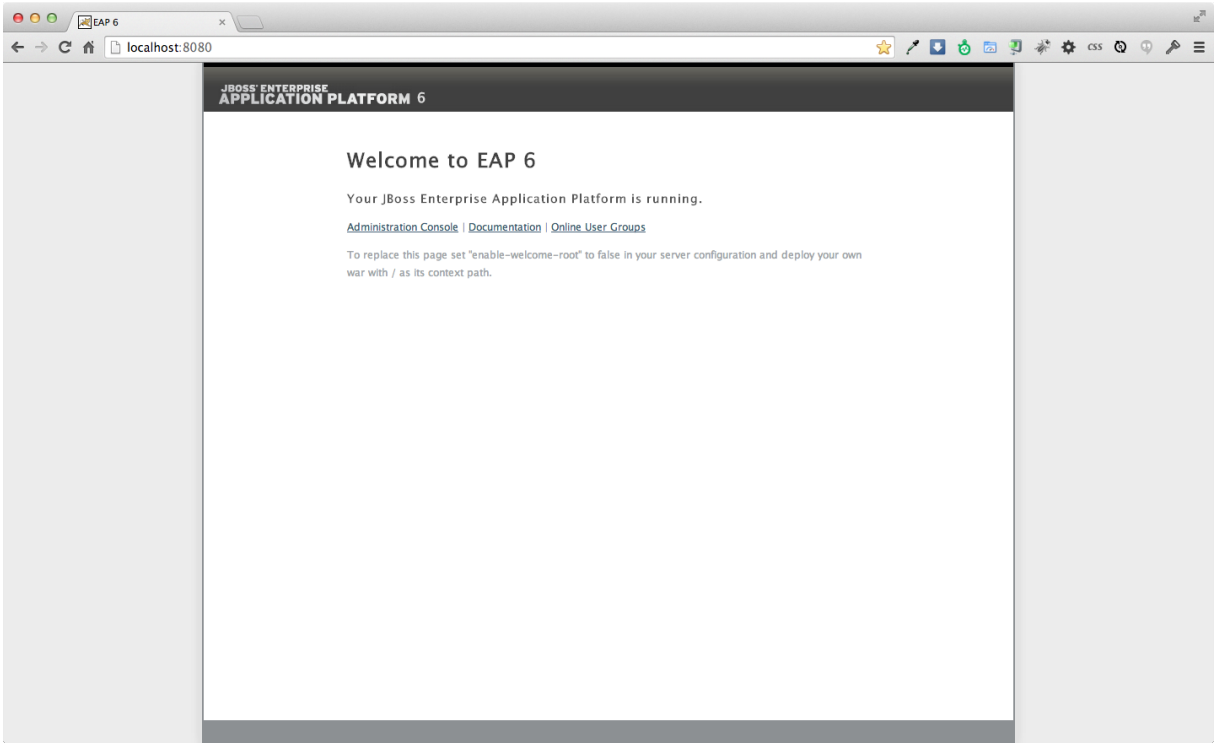

Figure 9.10.

Once the console is open, verify that the VDB has been deployed. Along the left-hand pane there is a Virtual Databases menu option. By clicking this option, we can see that the Financials VDB has been successfully deployed. By following the illustration below, click tab Models, you should be able to see the deployed data services.

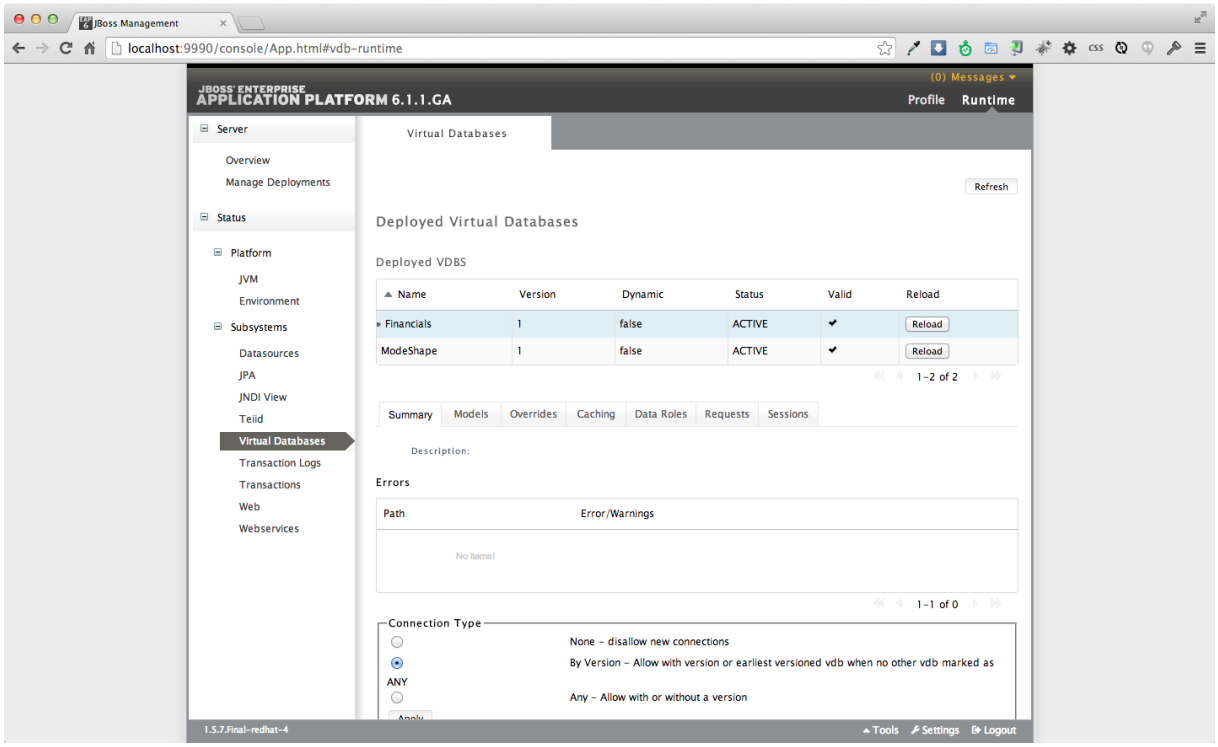

Figure 9.11.

Upon closer inspection, you will notice that there is a "1" that has been appended to the VDB. This is simply the version number of the VDB. This enables you to deploy multiple versions of a VDB and have them accessible by your applications.

9.3. Dependency Diagrams

There is a very useful feature for modeling in the Teiid Designer. Right-click on the All_Customers.customer table and select Modeling → Show Dependency Diagram.

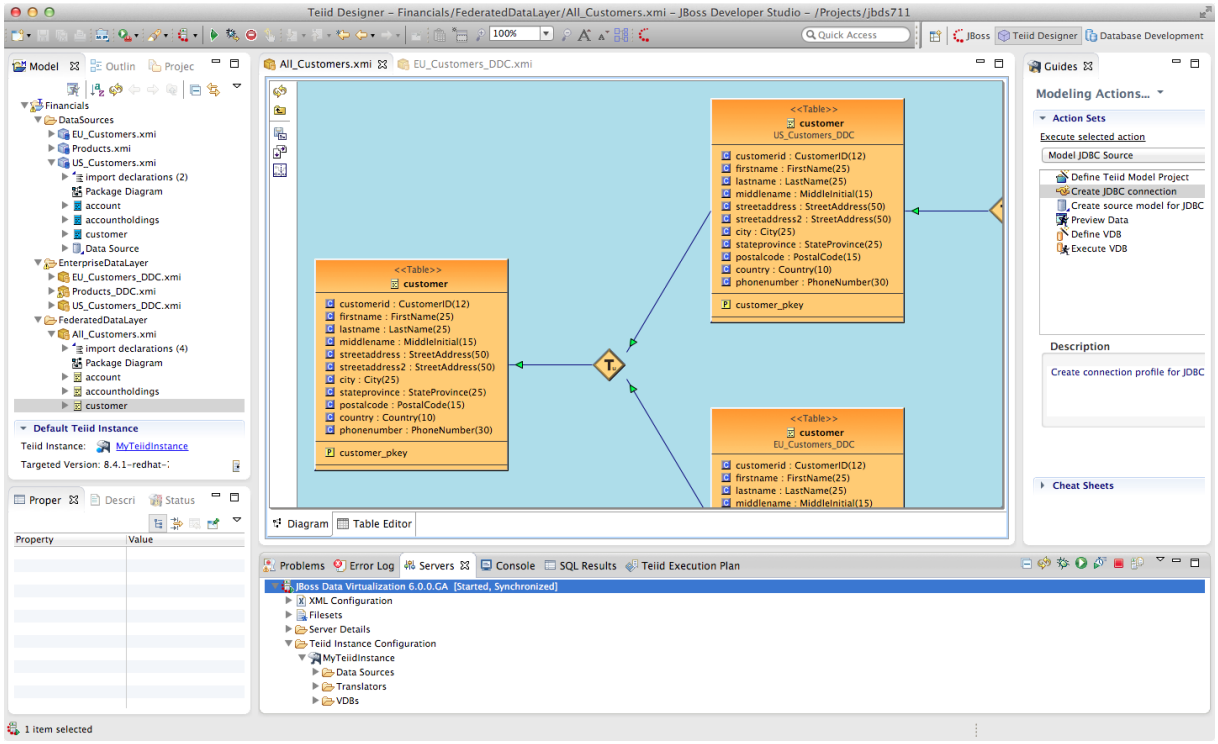

Figure 9.12.

Then, choose the Outline view. This is the tab that is right next to the Model Explorer tab in the left-hand pane of JBDS. If the Outline View is not visible, simply enable it by going to Window → Show View → Outline. Finally, select the Show Diagram Overview button. This is the button located right under the Outline tab. You should see something like the following illustration.

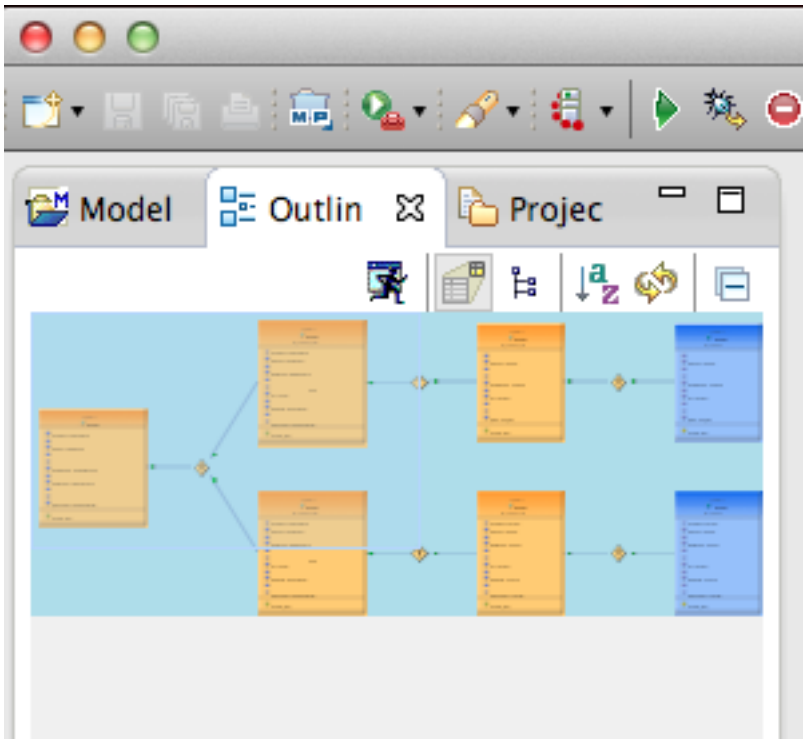

Figure 9.13.

You can scroll the "viewfinder" on the left in the Outline view to navigate the model on the right. You can also hover your mouse over the transformations (the diamond-shaped "T" nodes) on the right to examine the content of the transformations.

This capability is helpful for navigating among large, complicated models, and also shows the provenance/lineage of any selected data service. As was noted at the beginning, there is tremendous power and reusablility that results from the ability to create data service layers without a performance penalty.

Congratulations, you have now completed this lab.

Chapter 10. XML and Service Sources

In this lab you will be using the JDBS tool to create a Web Service. This lab is not a guide on Web Services, on how or when to use or create Web Services, but an instructional approach for some of the capabilities of the JBDS tool to create a Web Service. In this lab you will create the CountryInfoService.xmi model and the CountryInfoServiceView.xmi model. These will be created from the WSDL:

http://www.oorsprong.org/websamples.countryinfo/CountryInfoService.wso?WSDL

10.1. Create relational model from WSDL

Using the WSDL importer results in the creation of query-able relational procedures that represent your desired request and response web service structure defined through your WSDL's schema definition. Create two new folders called WSDL and User-specified Procedures in your Model Explorer view. Right-click on the Financials project and select New → Folder. Call the new folder WSDL.

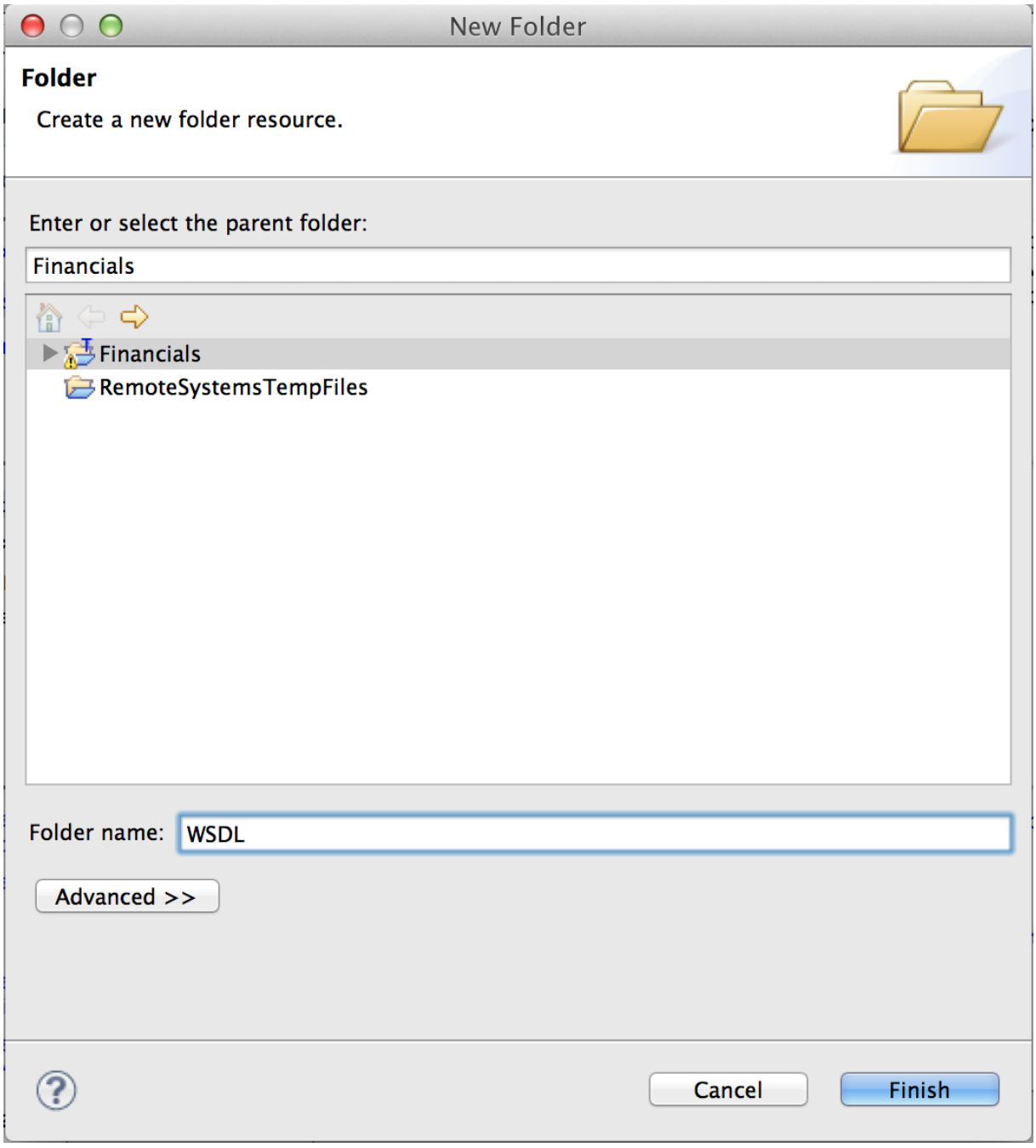

Figure 10.1.

Do the same for folder User-specified Procedures.

Right-click on the WSDL folder and select Import… from the right-hand menu.

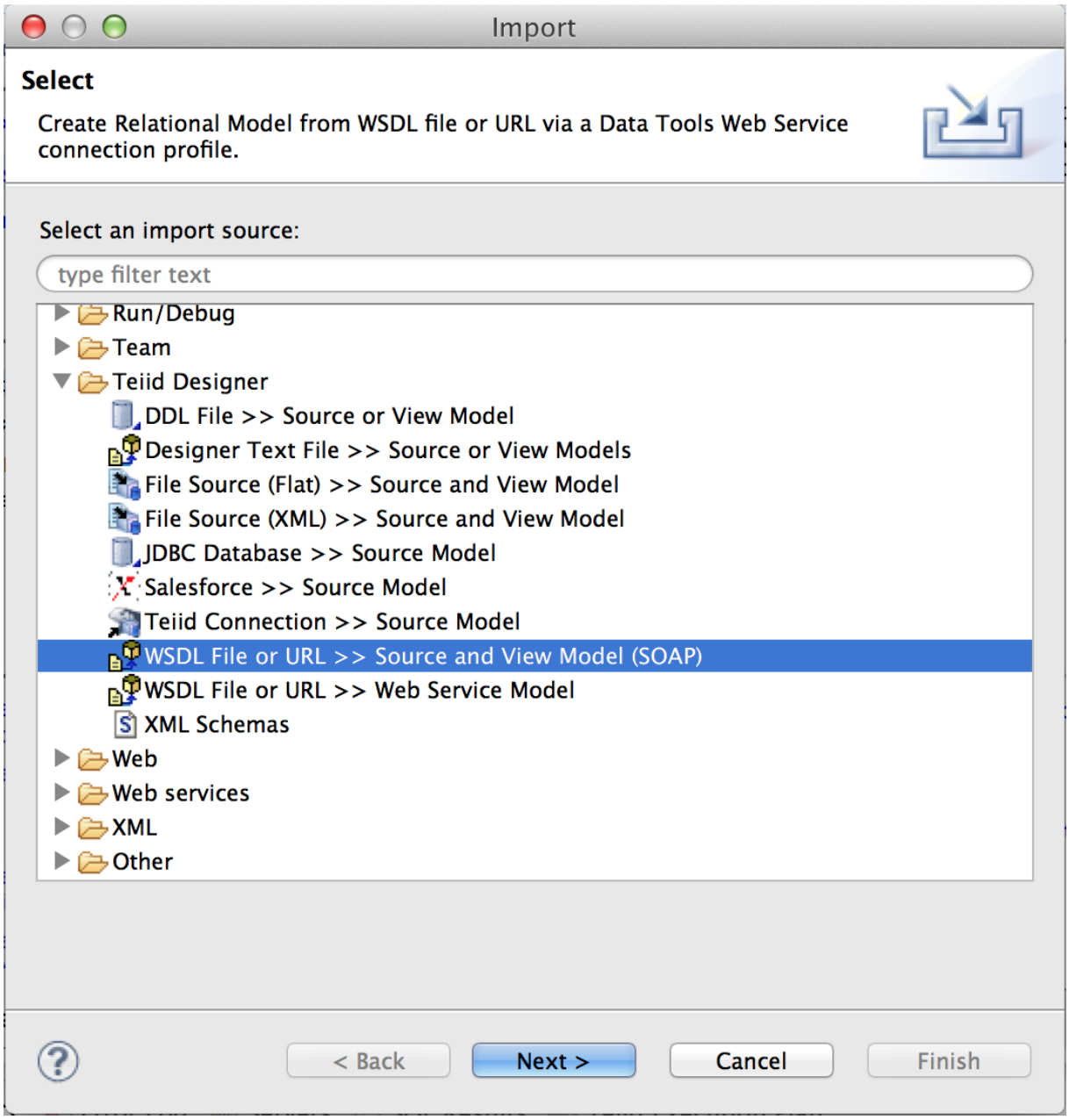

Figure 10.2.

The Import dialog box is opened. Select WSDL File or URL >> Source and View Model (SOAP) located under the Teiid Designer folder from the import source list that is displayed in the Import dialog box. Click the Next > button to continue. The Create Relational Model from Web Service Dialog box opens.

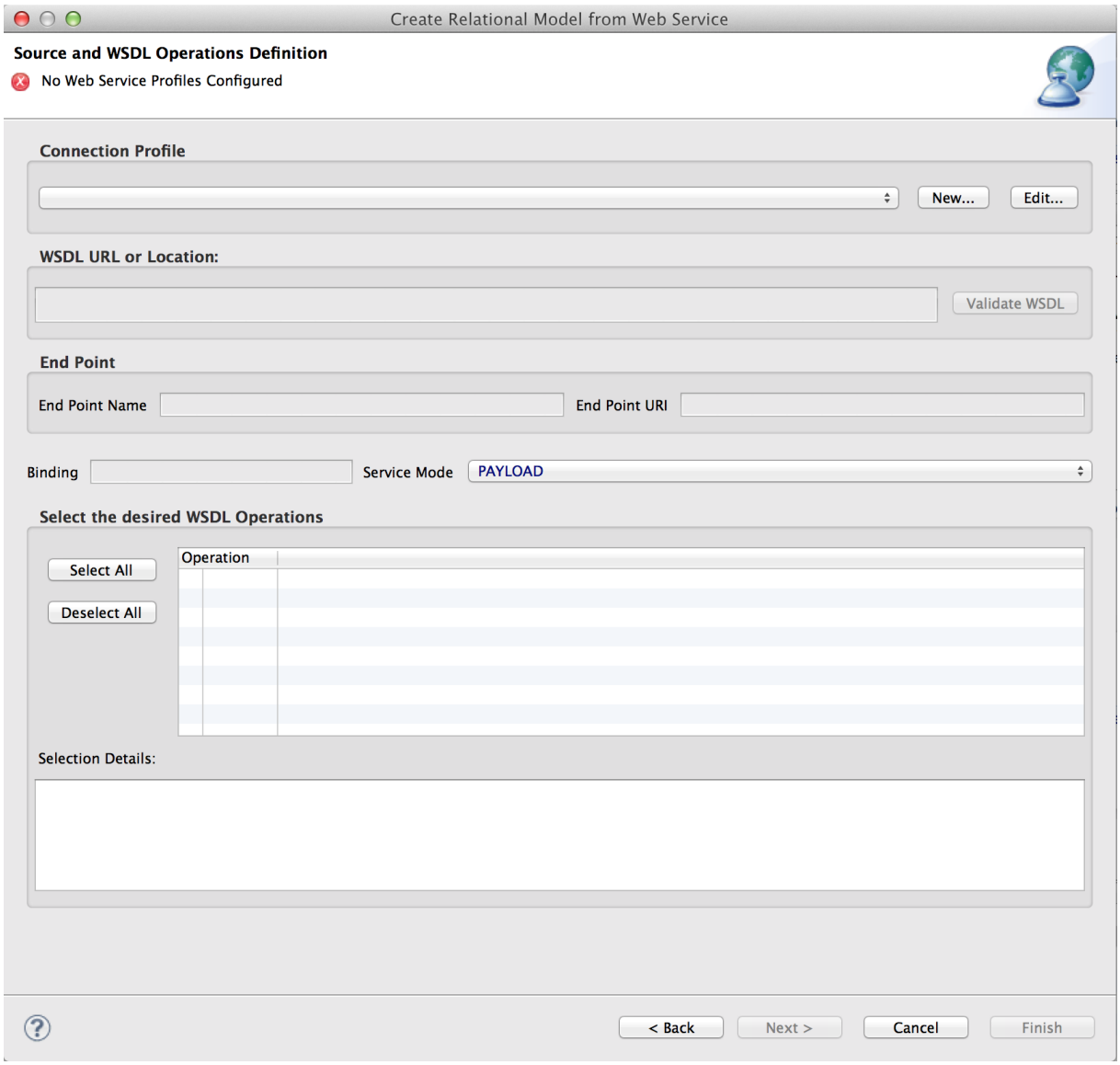

Figure 10.3.

Click on the New… button to create a new Connection Profile. The New Connection Profile dialog opens.

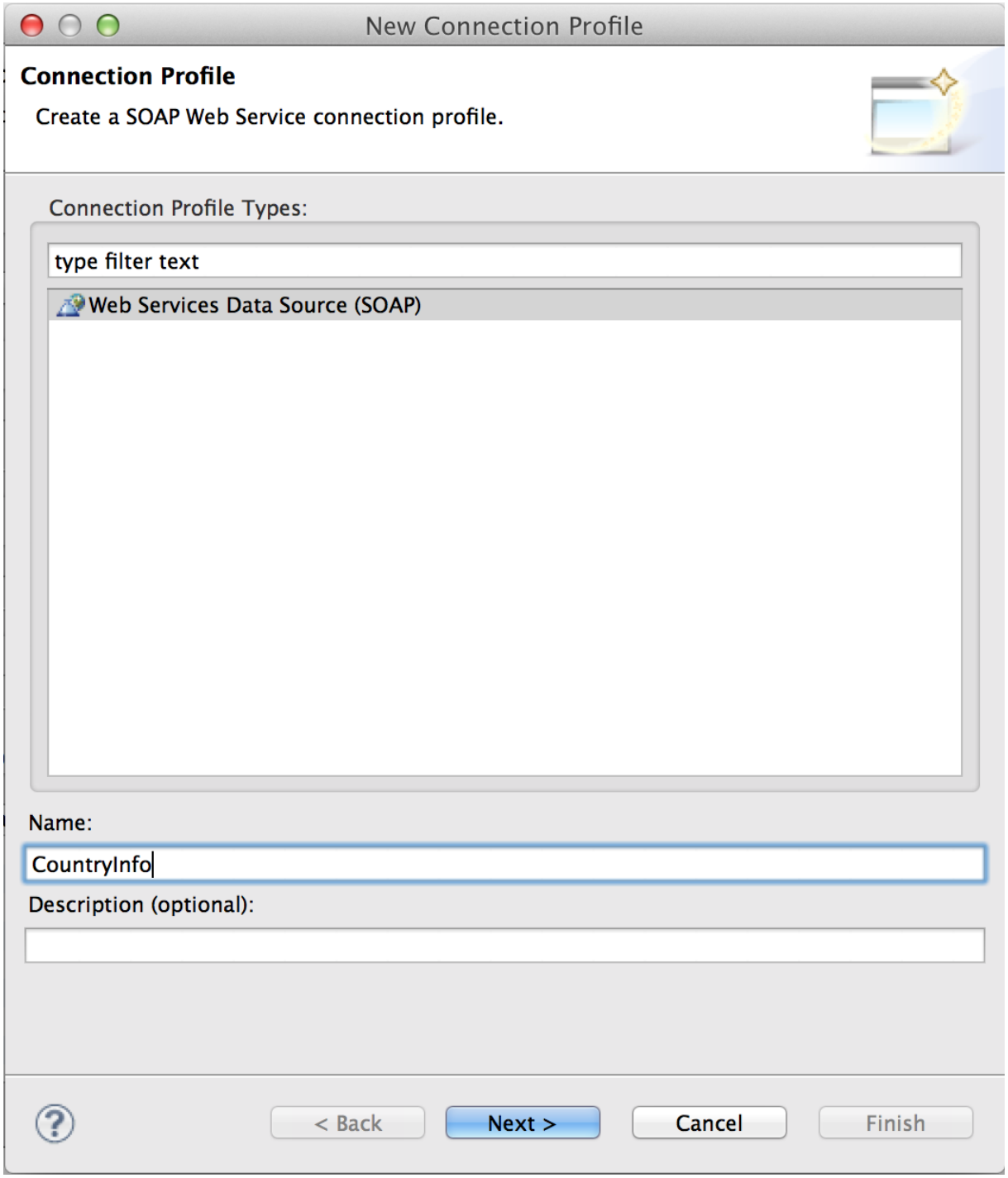

Figure 10.4.

Type CountryInfo in the Name field and click the Next > button. The New Connection Profile dialog box opens.

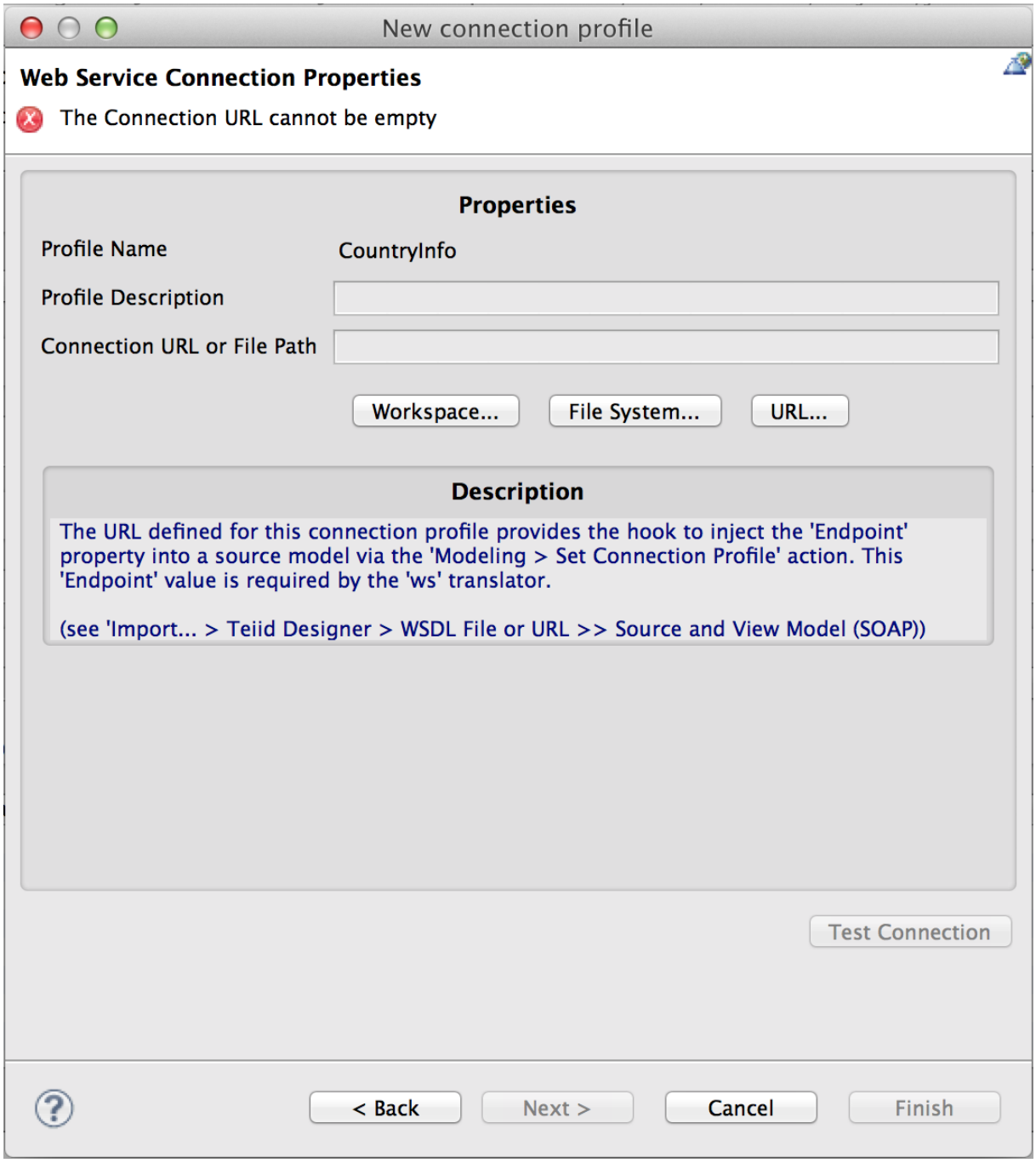

Figure 10.5.

The WSDL that you will connect to is a URL. Click on the URL… button to enter a URL. The WSDL URL dialog box opens.

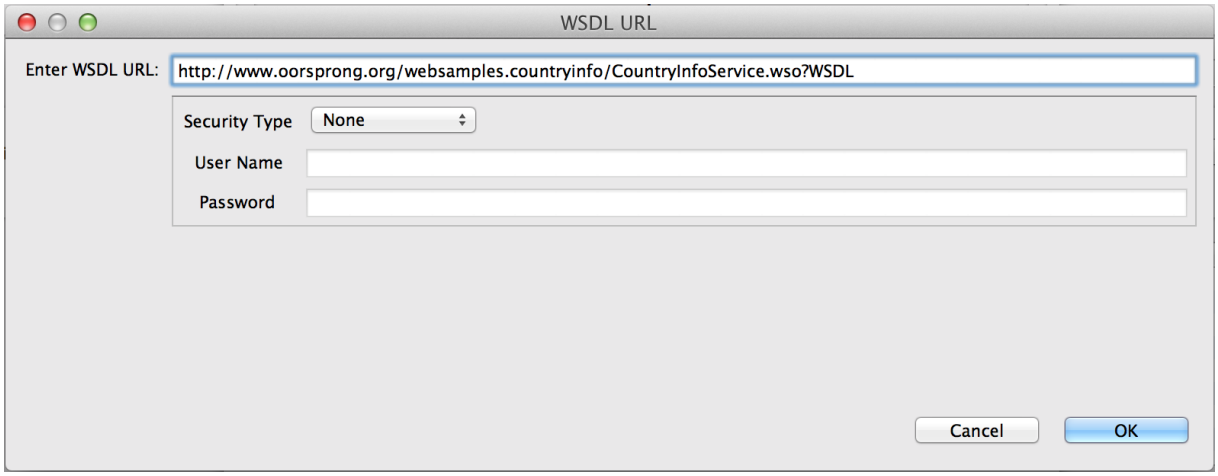

Figure 10.6.

In the WSDL URL field enter in the following URL:

http://www.oorsprong.org/websamples.countryinfo/CountryInfoService.wso?WSDL

The WSDL has no security so leave the Security Type to default of None. Click the OK button. When you click the OK button the software will validate the URL.

You will need an internet connection for this to validate correctly.

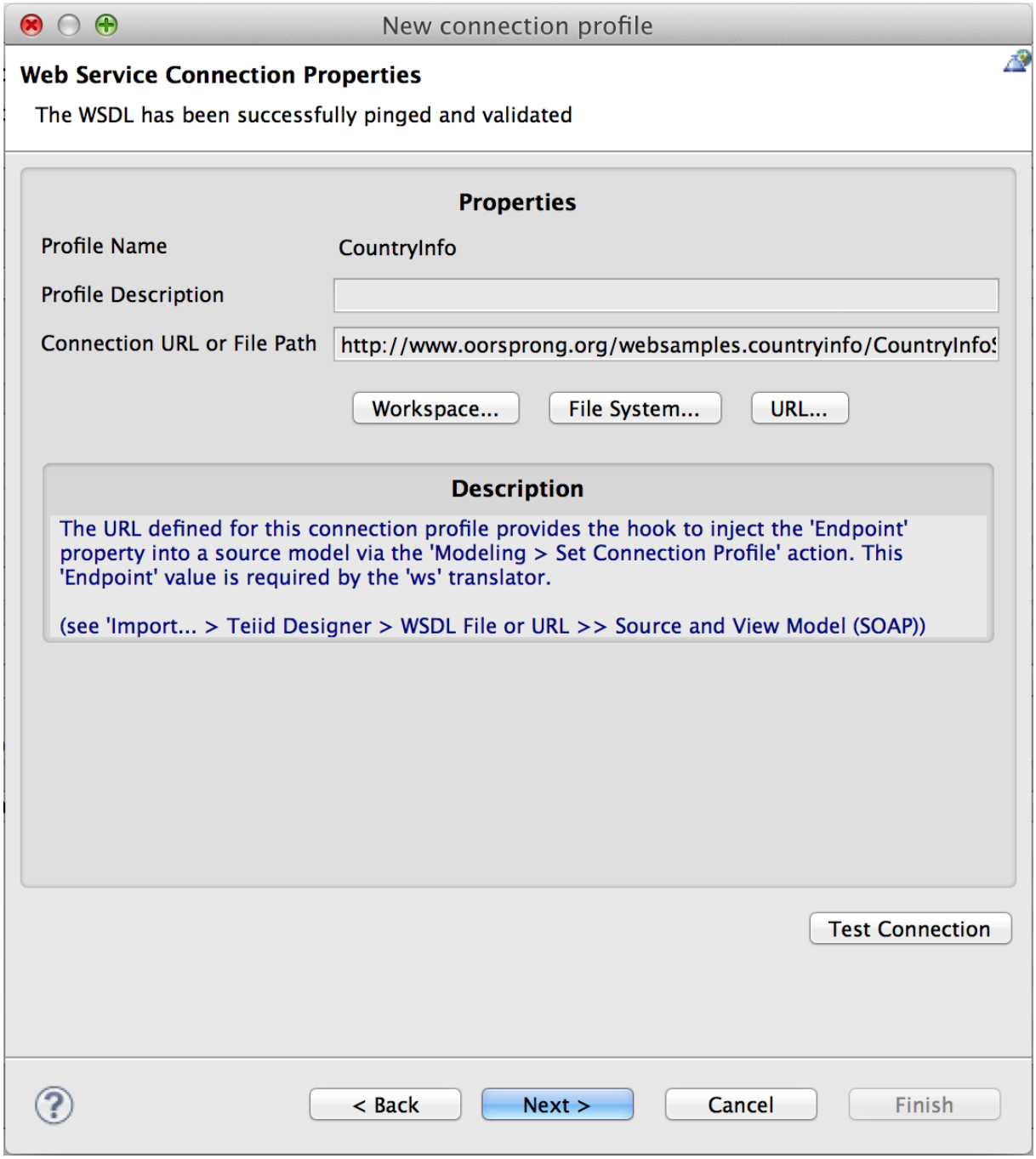

Figure 10.7.

Click on the Test Connection button to test the URL. Click on the Next > button.

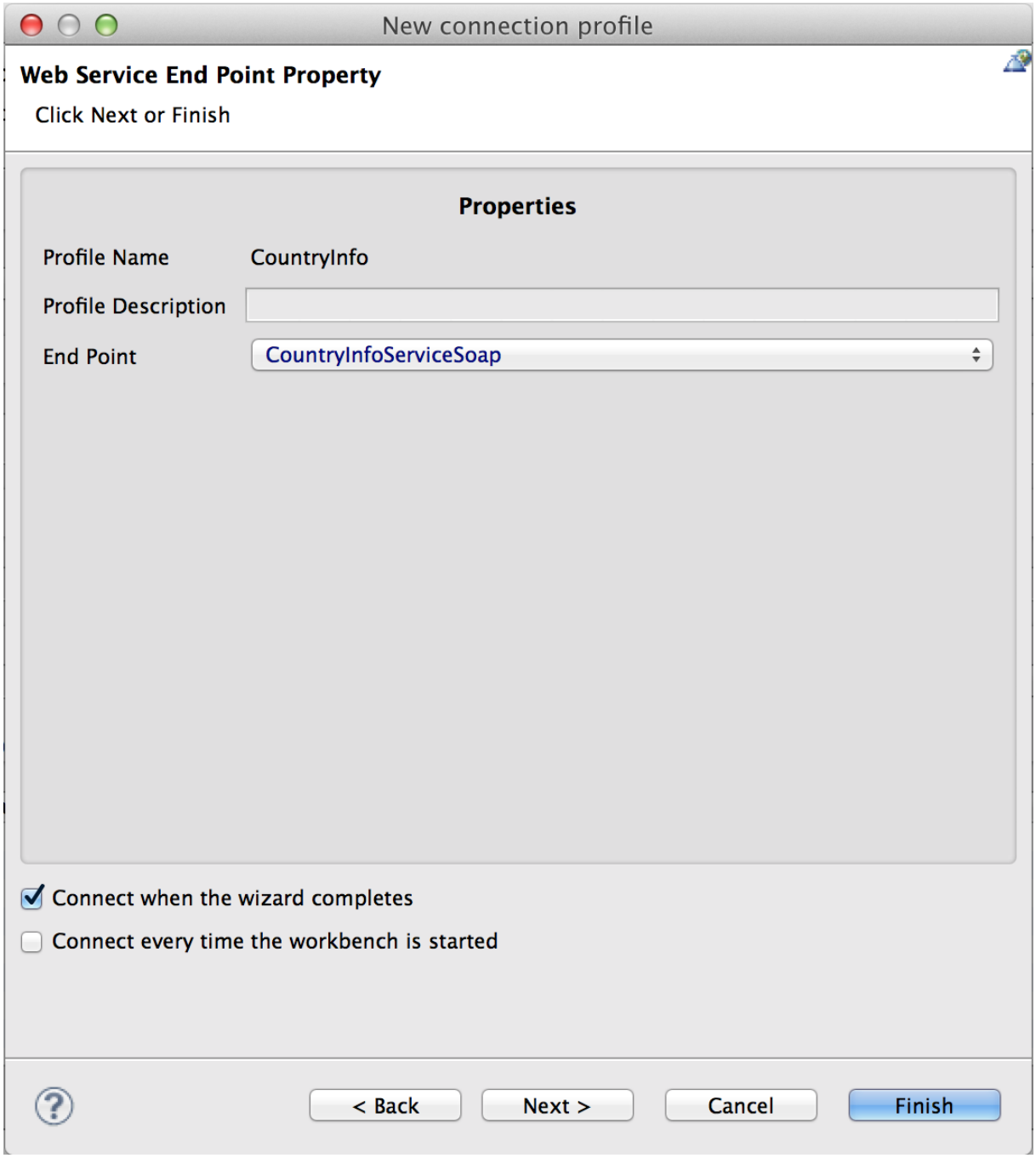

Figure 10.8.

Click on the Next > button.

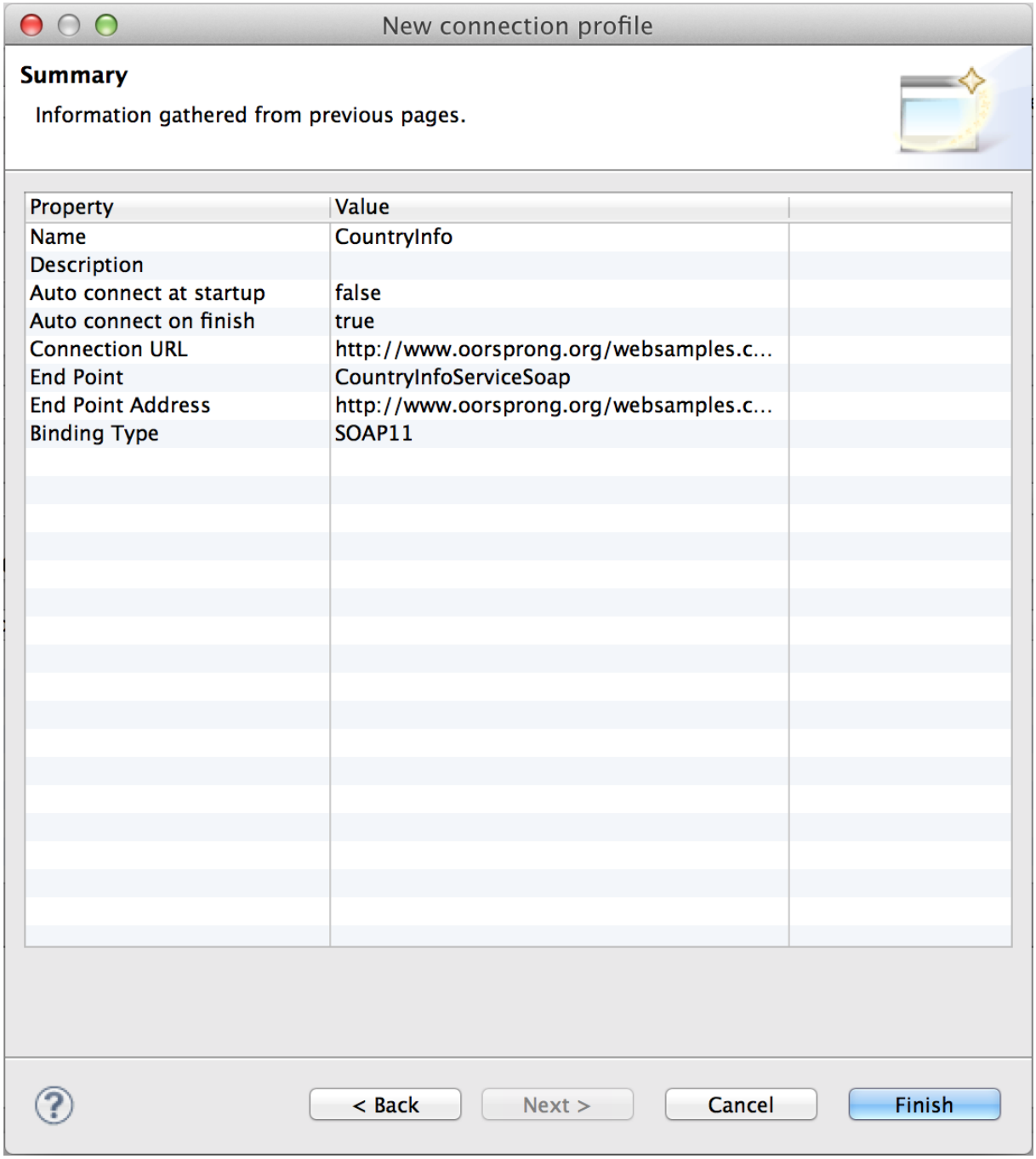

Figure 10.9.

The Summary window will appear. Click Finish to go the Source and WSDL Operations Definition.

The connection to the WSDL will return the WSDL Operations that are available. This first page involves defining your WSDL source connection profile. Select port (if multiple available), service mode (PAYLOAD vs MESSAGE) and check the operations you wish to use to generate your queryable relational procedures. Click on the Deselect All button

to deselect all of the WSDL Operations. Click on the Operations: CountryName and the FullCountryInfo, to select.

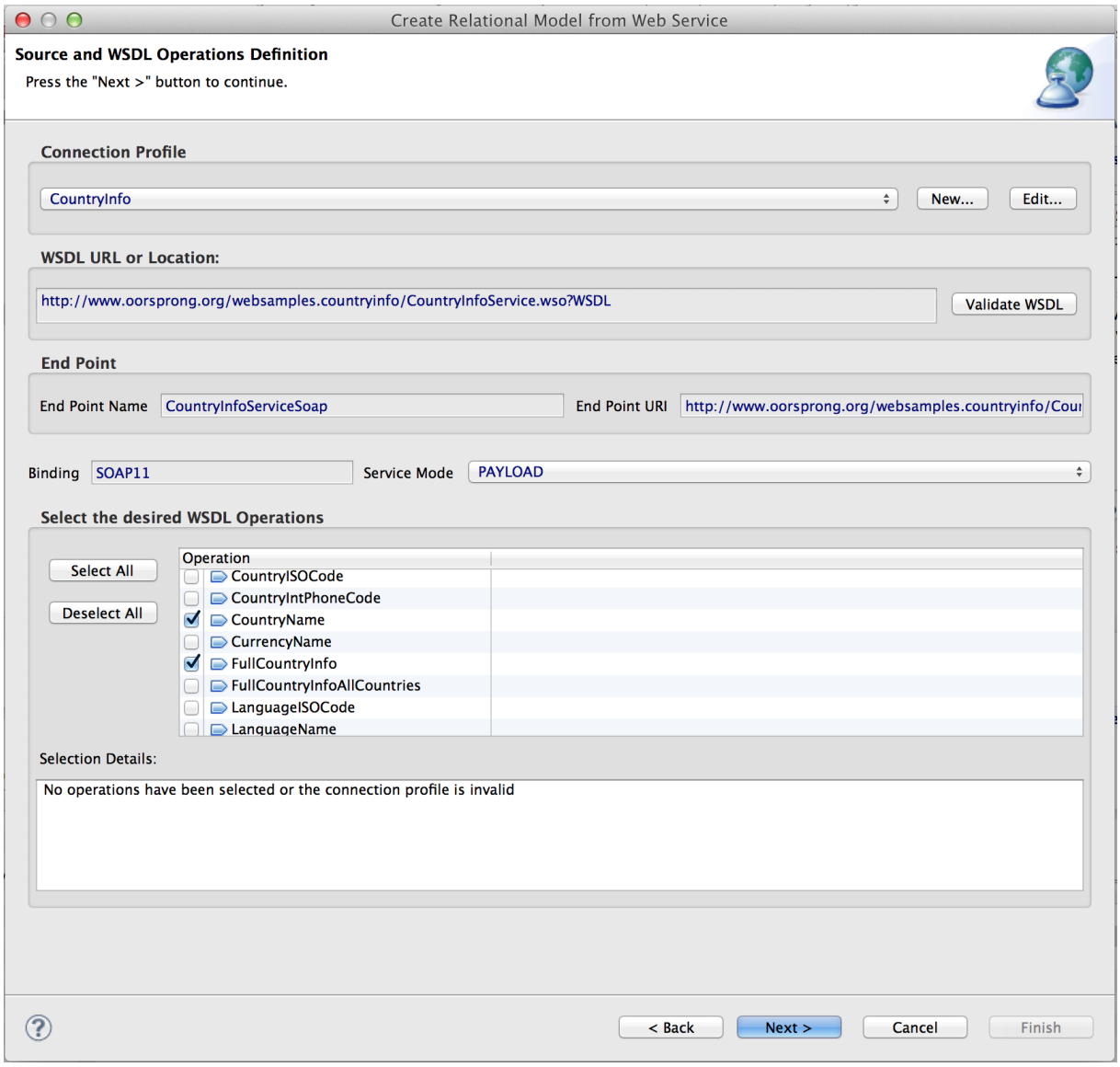

Figure 10.10.

Click on the Next > button.

The Create Relational Model from Web Service dialog is displayed. This page allows you to define the source and target models where the generated procedures will be defined in. Notice that the Name fields for the Source and View model are populated. Based on initial workspace selection, the location for these models may be preset but changeable and the source and view model names will be generated based on the service name defined in your WSDL. Select the browse button … to select locations and existing models in your workspace.

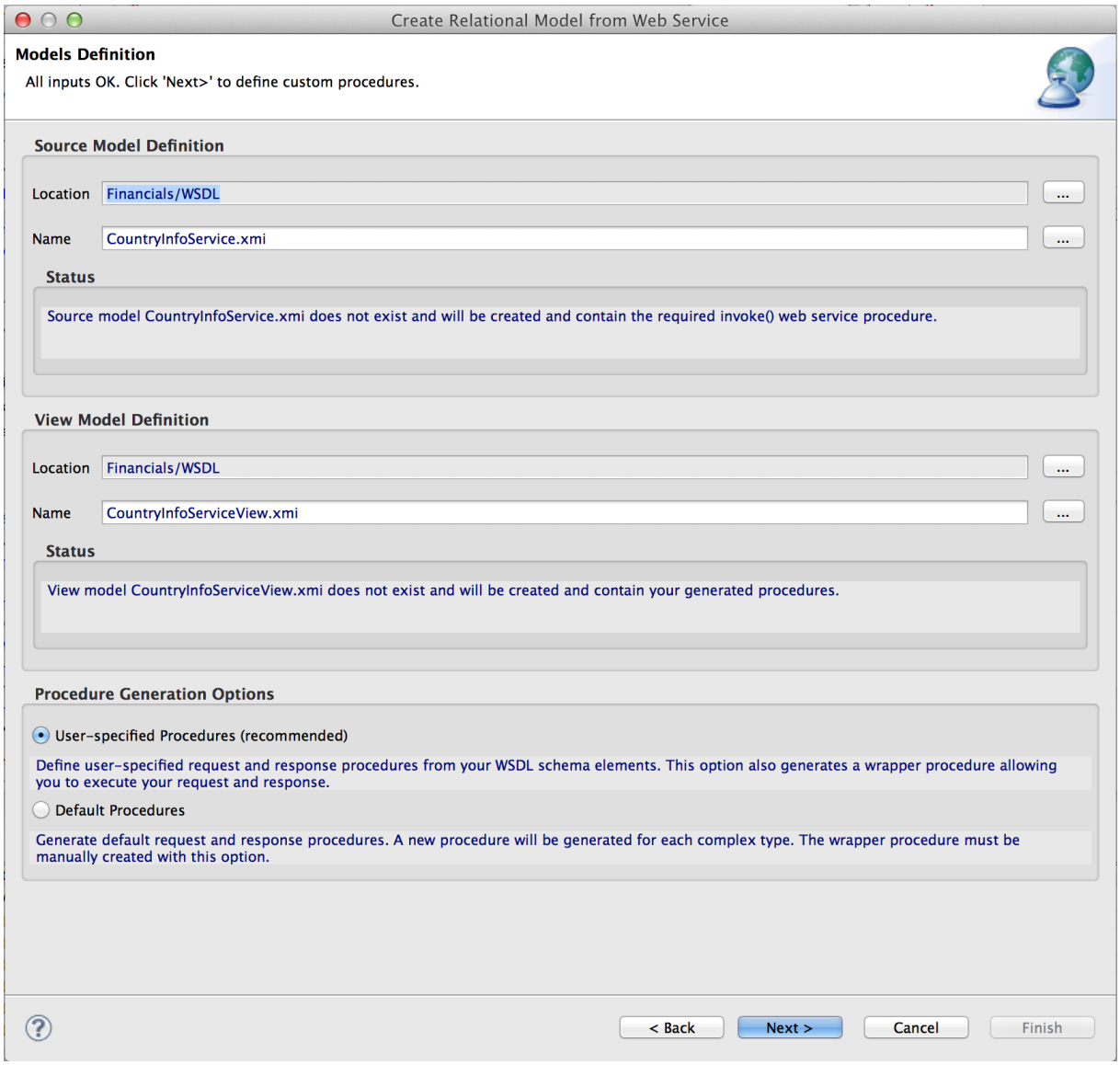

Figure 10.11.

Choose User-specified Procedures and click the Next > button.

In the next few steps add in the Element Info for both the CountryInfo and FullCountryInfo operations. Use the Operations combo selector at the top to switch between the operations. In the Request tab, select and double-click the schema elements you wish to set as input parameters for your request. These will be added to the Element Info panel and the resulting generated SQL statement will be updated to reflect the new element. If the selected service mode for this procedure is set to MESSAGE, the HEADER tab will be enabled and allow you to define the SOAP header variables utilizing the same schema tree. Select the Operation from the Operations pull-down (CountryInfo and FullCountryInfo). Add in the Schema Contents element by selecting the element in the left-hand side and clicking the Add button on the right-hand

side. Repeat these actions for both the Request and Response tabs. See image below for further clarification.

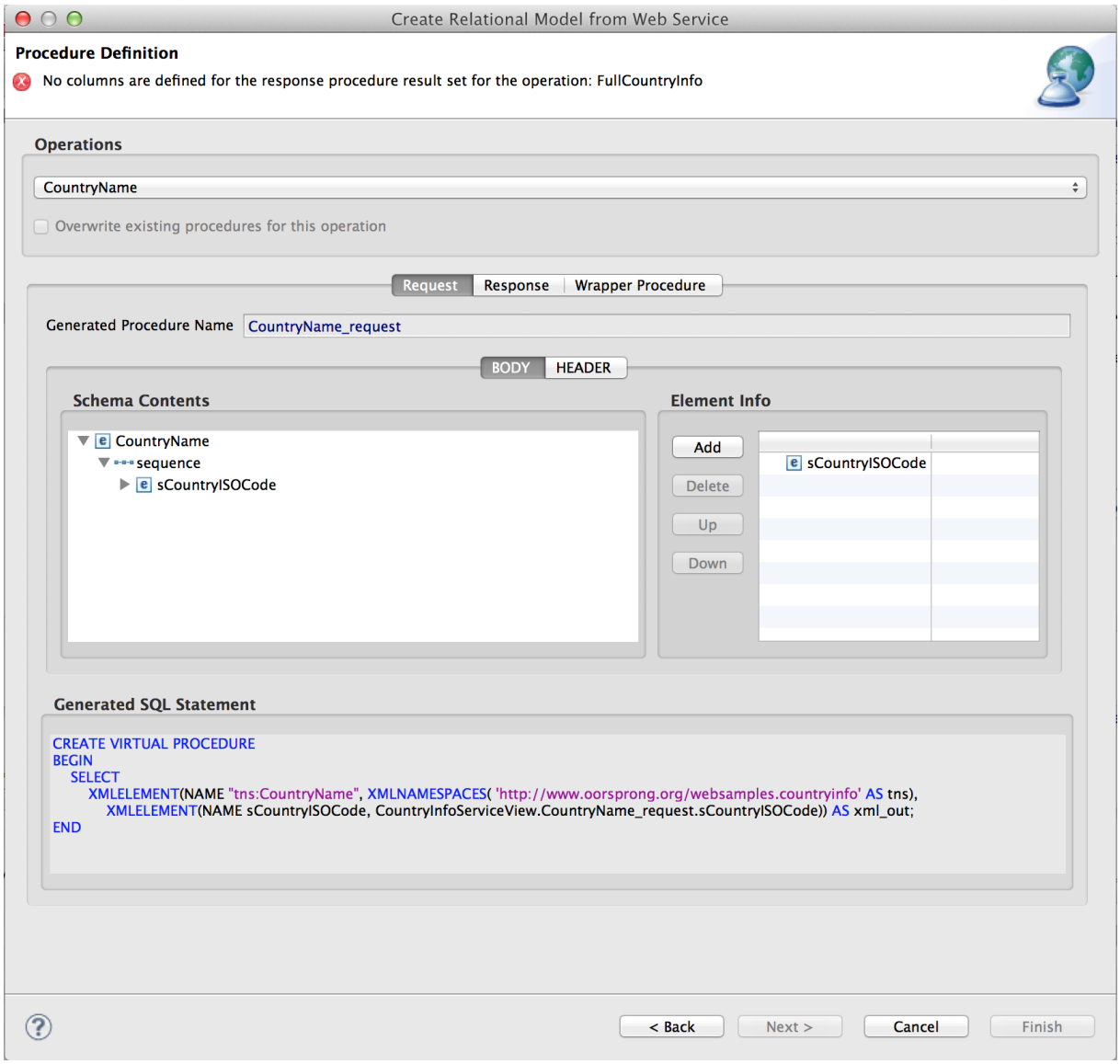

Figure 10.12.

This wizard generates both request and response procedures that are used in the query-able wrapped procedure. If you wish to generate Designer's legacy create and extract procedures choose the Legacy Procedures option in the lower section and click Finish.

Click on the Finish button. A status screen is displayed while the models are being created. The models will appear in the Model Explorer and will be open in the right-hand pane. Click the Save All button to save your models.

Double-click on the CountryName procedure to open the transformation view for the procedure.

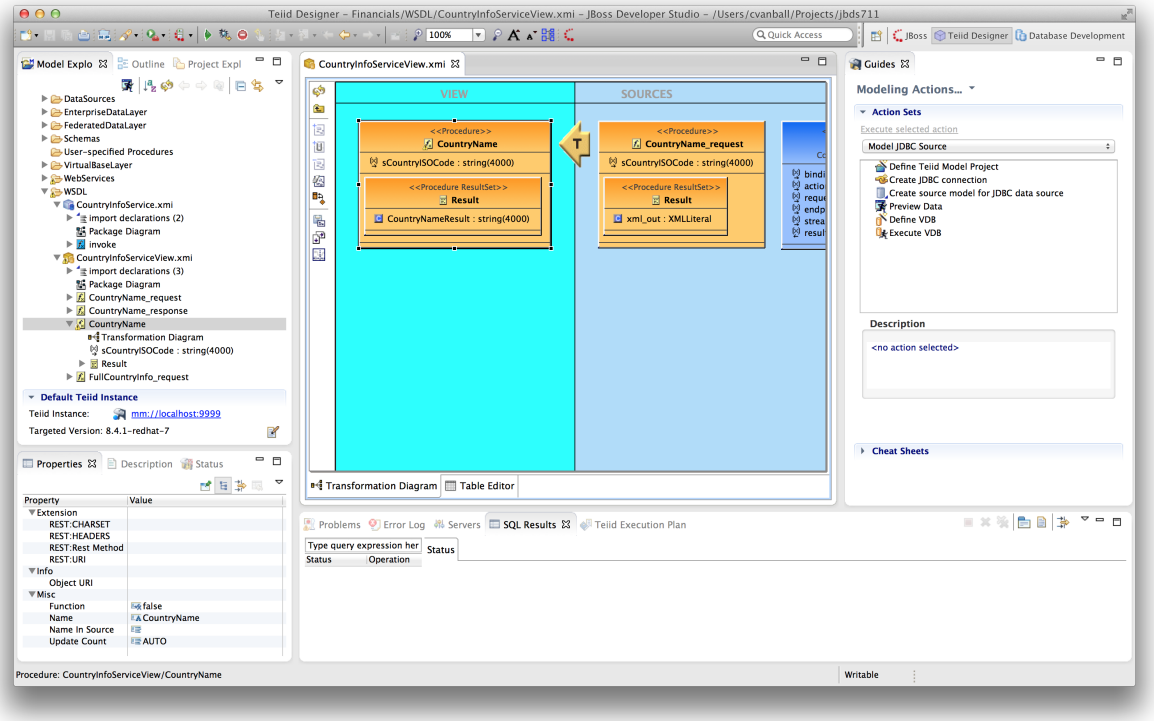

Figure 10.13.

Click on the Preview Data button to preview the data.

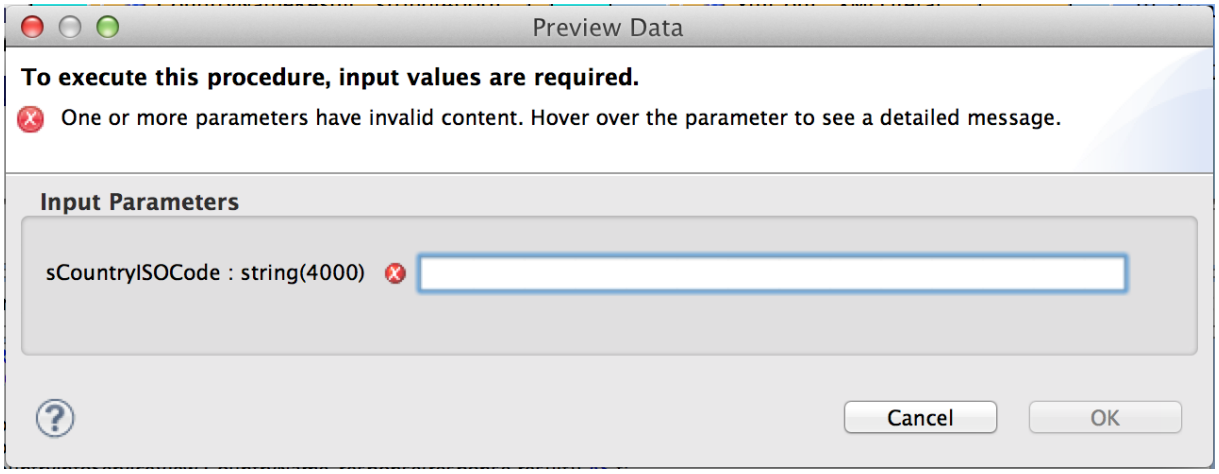

Figure 10.14.

Enter in a country code, such as AU for Australia, to retrieve the data. Click the OK button. View the SQL Status and Result1 tab to see the results returned from the procedure.

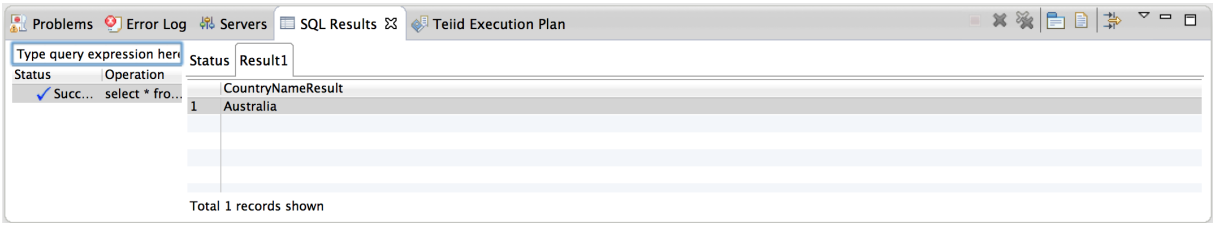

Figure 10.15.

Try returning other ISO country codes such as CA, BM, BR, etc. The ISO country codes can be found at the following URL:

http://userpage.chemie.fu-berlin.de/diverse/doc/ISO_3166.html

Save and close the models.

Congratulations, you have now completed this lab.

Chapter 11. OData

The Open Data Protocol (OData) is a Web protocol for querying and updating data that provides a way to unlock your data and free it from silos that exist in applications today. OData does this by applying and building upon Web technologies such as HTTP, Atom Publishing Protocol (AtomPub) and JSON to provide access to information from a variety of applications, services, and stores. OData is used to expose and access information from a variety of sources including, but not limited to, relational databases, file systems, content management systems and traditional Web sites. OData is consistent with the way the Web works, it makes a deep commitment to URIs for resource identification and commits to an HTTP-based, uniform interface for interacting with those resources (just like the Web). This allows OData to enable a new level of data integration and interoperability across a broad range of clients, servers, services, and tools. For more information on OData see http://www.odata.org.

11.1. How to access the data

When you have successfully deployed the Financials VDB into the JBoss Data Virtualization server, the OData protocol support is implicitly provided by the JBoss Data Virtualization server without any further configuration. Now, open up a browser and point it to the following URL: http://localhost:8080/odata/Financials.1/All_Customers.customer. If you are requested to type in a username/password enter user/user. This is similar to making a JDBC/ODBC connection and issuing a SQL call as SELECT * FROM All_Customers.customer; The returned results from OData query can be in Atom/AtomPub XML format or JSON format. By default AtomPub based XML result is returned as shown below.

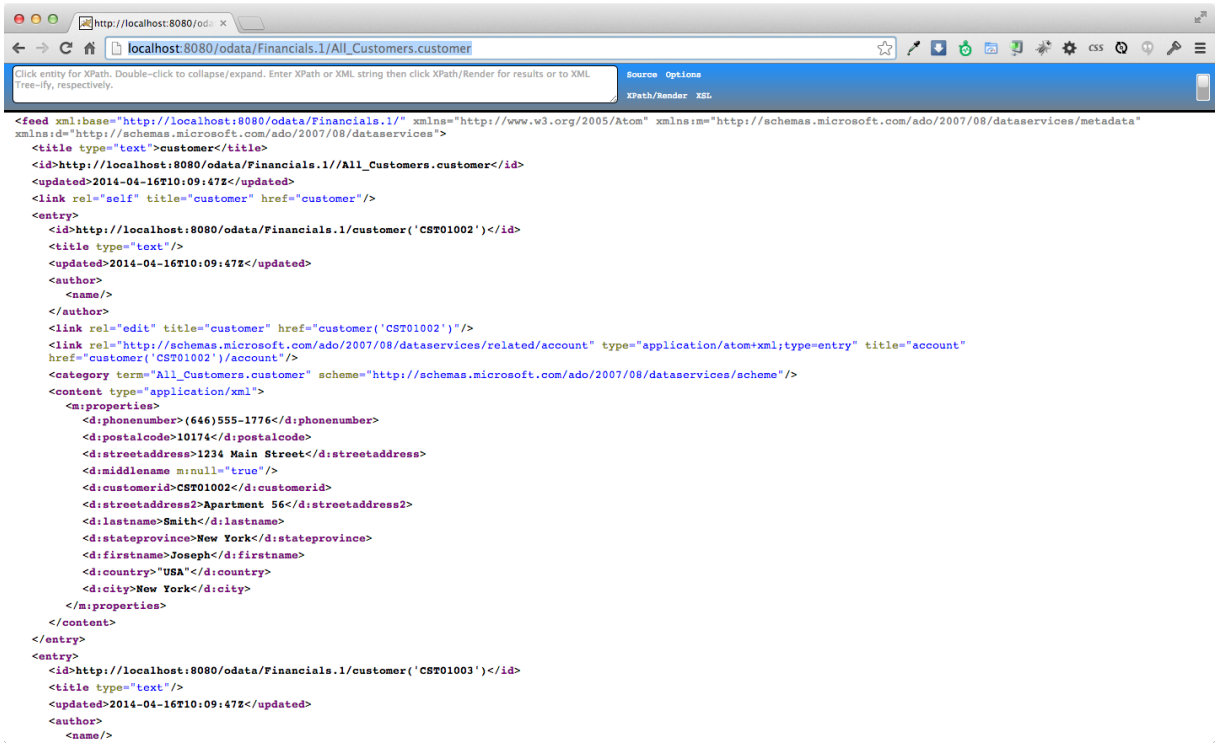

Figure 11.1.

To return the results in JSON format use the following URL as shown below: http://localhost:8080/odata/Financials.1/All_Customers.customer?$format=json

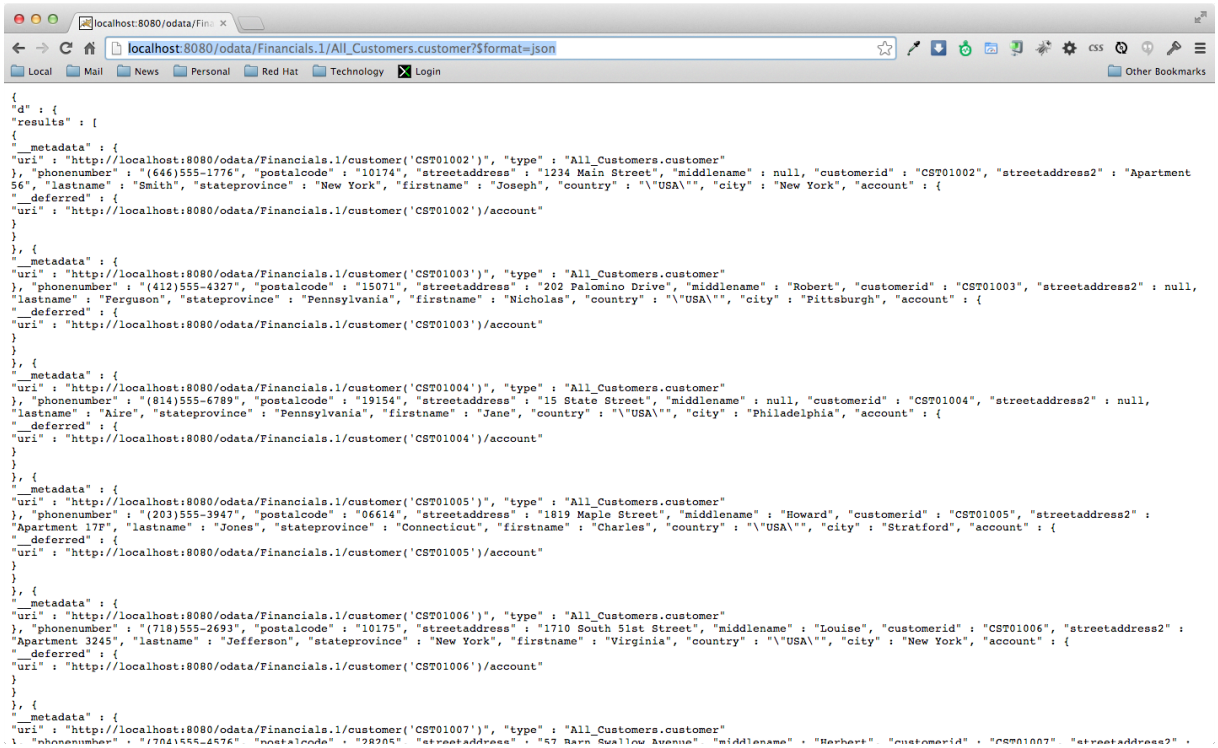

Figure 11.2.

11.2. How to query the data

To search for customer with customerid='CST01033' which is the SQL equivalent of SELECT * FROM All_Customers.customer where customerid='CST01033'; use following URL and it should return the results shown below: http://localhost:8080/odata/Financials.1/customer('CST01033')

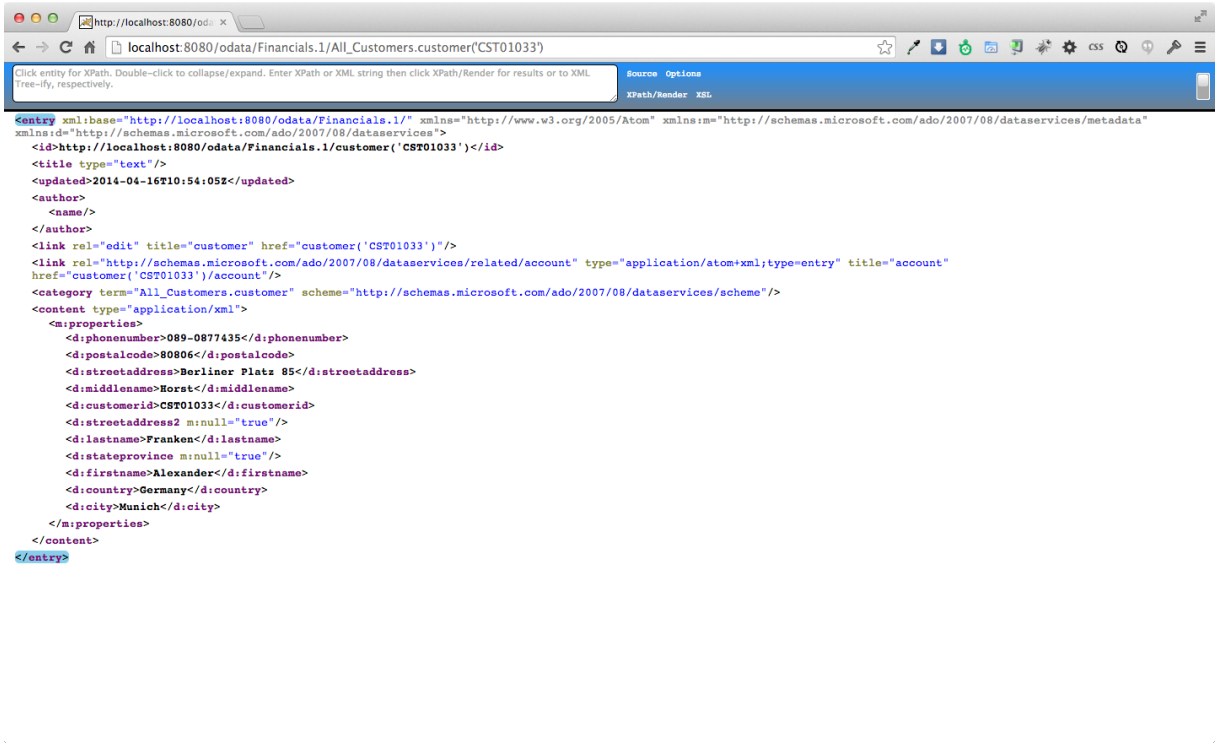

Figure 11.3.

Another way to query the data is to use the OData filter system query option using $filter in the URL. The $filter system query option allows clients to filter the set of resources that are addressed by a request URL. $filter specifies conditions that MUST be met by a resource for it to be returned in the set of matching resources. To search customers from the USA try following URL: http://localhost:8080/odata/Financials.1/All_Customers.customer?$filter=country eq *USA* and this should return the results as shown below.

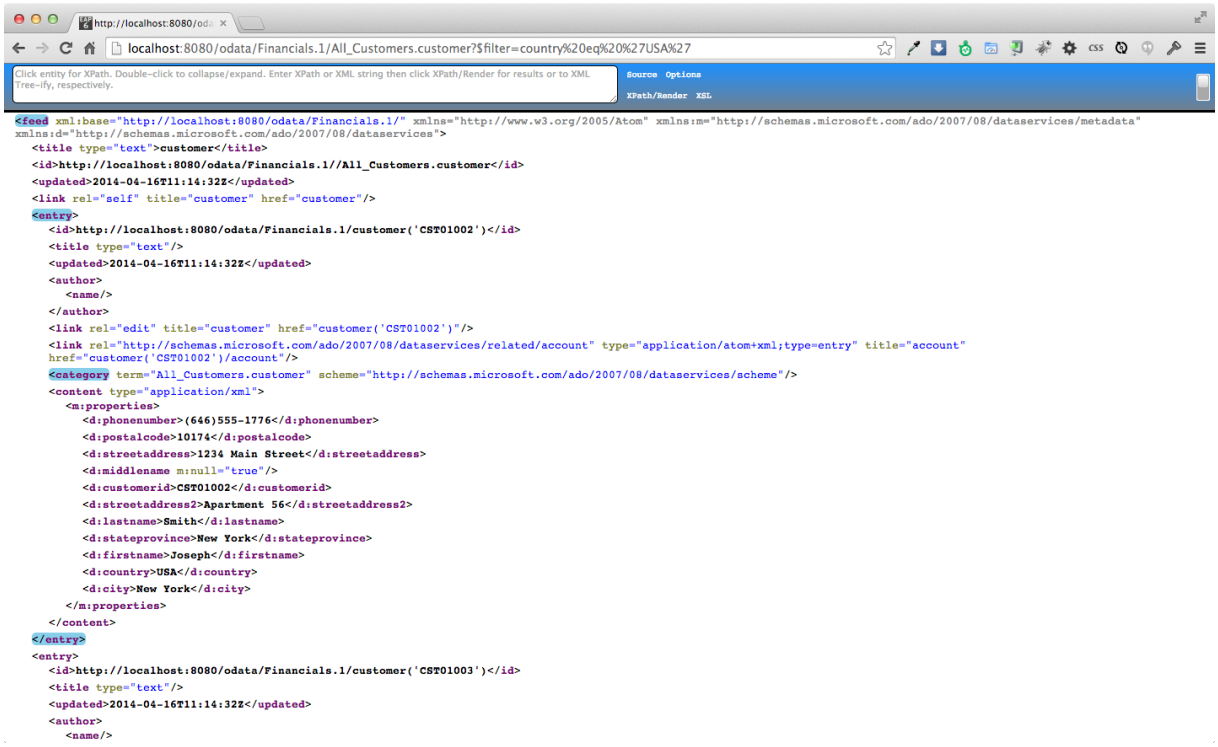

Figure 11.4.

Congratulations, you have now completed this lab.