



JBoss Enterprise Data Services 5.1
Hands on Lab

EDS ON SOA-P

Table of Contents

Table of Contents.....	2
Introduction.....	4
Overview.....	4
EDS Architecture.....	5
Included Files.....	6
System Expectations	6
What is Expected of You	7
Lab Number 1: Install and Configure SOA & EDS Platform.....	8
Get the SOA-P File.....	8
Unzip to Install, Part 1.....	8
Get the EDS File.....	8
Unzip to Install, Part 2.....	8
Finalize the Install with Ant.....	9
Enable the Admin Users.....	10
Install the JDBC Driver.....	11
Start the Server.....	11
Open the Admin Console.....	12
Lab Number 2: Installation of JBDS.....	16
Get the File.....	16
Running the Installer.....	16
Start JBDS.....	26
Select a Workspace.....	28
JBDS Start Page.....	29
Install the Patches.....	30
Lab Number 3: Create a Project and Import Data Sources.....	34
Open the Teiid Perspective.....	34
Create a Teiid Server Instance.....	36
Create a Teiid Project.....	38
Create Project Folders.....	39
Create a New Source Model.....	41
Import Data Source Metadata (Product).....	42
Preview the Product Data.....	50
Import Data Source Metadata (US & EU Customers).....	51
Lab Number 4: Create a Virtual Base Layer.....	53
Rationale.....	53
Create the US_Customers VBL.....	53
Create the EU_Customers and Product VBLs.....	57
Lab Number 5: Create An Enterprise Data Layer.....	58
Rationale.....	58
Import the Data Dictionary.....	59
Examine the Data Dictionary.....	61
Create the EU_Customer Enterprise Data Layer.....	62

Create the US_Customer and Products Enterprise Data Layer.....	67
Lab Number 6: Data Federation & Deployment.....	73
Data Federation.....	73
VDB Deployment.....	75
Create the VDB Metadata Model.....	75
Create Data Sources on the Teiid Server.....	78
Deploy the VDB.....	80
Dependency Diagrams.....	81
Lab Number 7: Accessing Data Services Via the ESB.....	82

Introduction

Overview

JBoss Enterprise Data Services Platform (EDS) is a powerful set of tools and runtime components that make it easy for your applications and business processes to integrate and use data from many data sources. JBoss EDS includes:

- Tools for creating data views that are accessible through standard protocols (the Teiid Designer plug-in for JBoss Developer Studio).
- A robust runtime environment that provides enterprise-class performance, data integrity, and security (the Teiid Server, which executes as a process within the SOA-P Server).
- The JBoss Enterprise SOA Platform with its next generation ESB, which makes it easier to publish data services and integrate them into applications and business processes.
- A repository for storing metadata (ModeShape)

By enabling the high-performance, secure, and simultaneous access of multiple sources of data (databases, files, applications, services) and presenting this data in integrated, semantically-reconciled views, JBoss EDS insulates applications from the underlying data source details. Data is accessed in real time without the need to copy or move it to a new location.

Data services can be accessed in three ways: via standard SQL through Java Database Connectivity (JDBC) or Open Database Connectivity (ODBC); as web services that can be automatically generated (with the WSDL/schemas generated for you) or they can be made compliant to a custom schema; and as Java objects through Hibernate. The inclusion of the full SOA Platform offering with EDS enhances the ability to address many additional use cases, adding asynchronous capabilities through the enterprise service bus (ESB). Maximizing the return on your information assets (ROA) is key for operational and competitive advantage. JBoss EDS helps you rapidly and effectively leverage your data assets.

Data Services offers a highly scalable and high performance solution to information integration. By allowing integrated and enriched data to be consumed relationally or as XML over multiple protocols, Data Services simplifies data access for developers and consuming applications.

EDS Architecture

Understanding the overall components that comprise the EDS architecture will help you to understand what is happening in the following labs. Here is a high-level architecture overview of EDS:

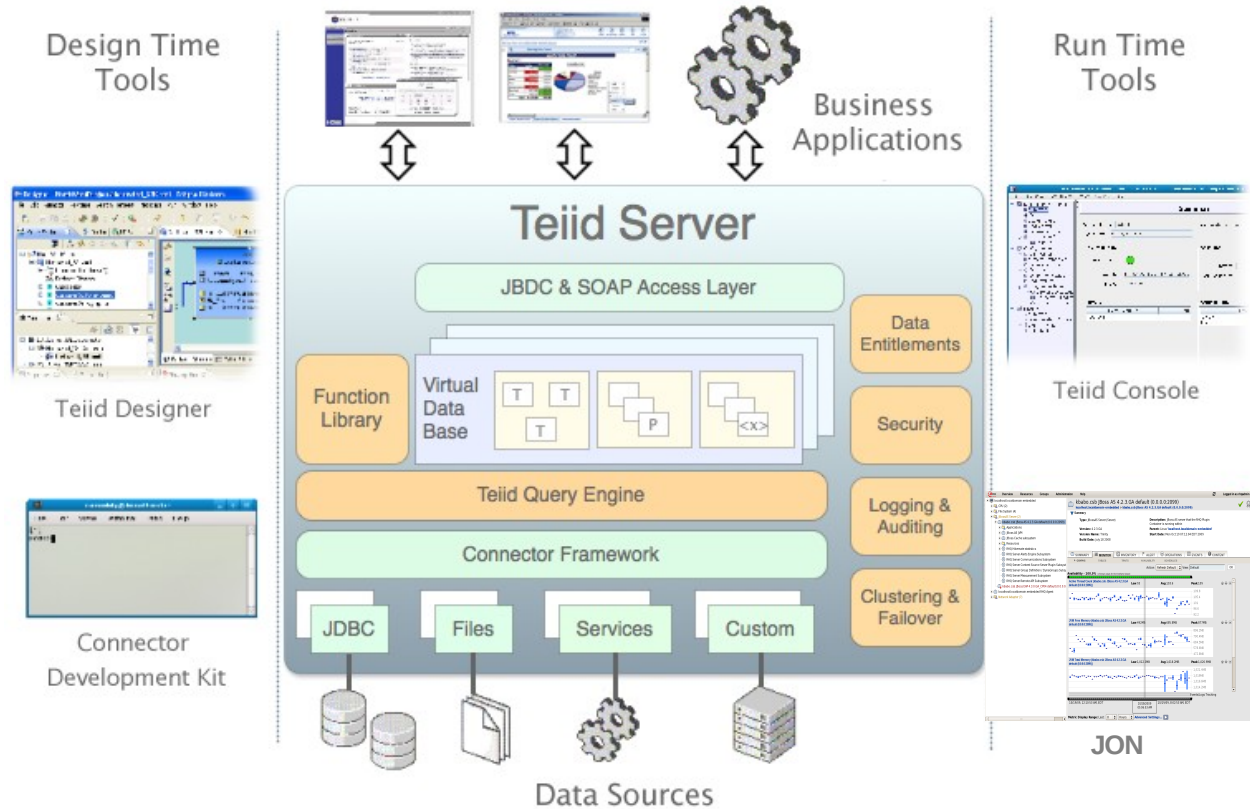


Illustration 1: EDS Architecture

JBoss Enterprise Data Services Platform is a data federation and virtualization engine (the Teiid Server) that allows you to query multiple data sources (RDBMSs, web services, files, etc) as though they were a single unified source. It is an optional, add-on extension to the JBoss SOA Platform. The Teiid Server Runtime executes (as illustrated above) as an additional process/capability within the SOA-P container. The Teiid Server fields data service requests through a number of different interface points (JDBC, ODBC, web services, Hibernate). It uses the metadata available from the deployed Virtual Database (VDB) definitions which contain the mapping of virtual tables and procedures to the actual entities in the underlying data sources. This metadata is combined with any additional criteria that is supplied with the data services request and a highly optimized query plan is generated by the Query Engine. Sub-requests are then sent out in parallel to all the data sources participating in the federation, to ensure the fastest possible response. Other components in the figure above are the resources provided by the SOA-P container (clustering, security, auditing, etc.), the connector framework through which the Query Engine communicates to data sources (using JCA), and the management, administration, and monitoring capabilities

that are provided by the JBoss Admin Console, but with far more features in the JBoss Operations Network (JON) product. JON is included with a managed EDS subscription.

As noted above, VDBs are the deployable artifacts that drive the runtime behavior of the system. The Teiid Designer Tool (plug-in) for JBoss Developer Studio 4.0 provides a highly intuitive, GUI-based mechanism to create VDBs. Teiid Designer lets you define physical data sources by importing the required metadata (schema) from disparate sources. Once the metadata is imported, Designer can assist in building additional view layers. The collection of physical and view models form the VDB. The bulk of this lab is devoted to showing how easy is it to create data service models (VDBs) with Designer.

Included Files

Several files are included with this workshop. There is a copy of SOA-P 5.1 (platform agnostic). There is a copy of the EDS 5.1 add-on (also platform agnostic). There are copies of JBoss Developer Studio for Windows, Mac, and Linux. For the EDS Preview feature to work correctly, you will also need to install the JBDS 4.0.1 patch; that .zip is also included. Details on how to install all of these files are provided in Labs 1 & 2.

System Expectations

It is expected that you have a Windows, Linux or Mac notebook and you are comfortable working and running Java programs on it. It is expected you will have the environment PATH set to include a JDK 6.0 to use for these labs. It is also a good idea to have JAVA_HOME set to your JDK that you plan on using. You will also need to have the 1.7 version of Apache Ant installed. Please make sure you do this before running any of the labs. Two examples of what these settings might look like is below:

```
PATH=${Some Path}/jdk1.6.0_17/bin:${Some Path}/ant/apache-ant-1.7.1/bin: ${More Path Info}
```

```
JAVA_HOME=${Some Path}jdk1.6.0_17
```

To verify that this is correct you will have to look at these values on your system. One simple way to check the JDK version that you have is to run:

```
java -version
```

to see which one is in your path, and it should be a JDK 6 version to run this lab. Also note that Ant has been installed for you and you will have to install a few other things as the lab progresses.

Please note that having an existing CLASSPATH environment variable set may cause odd issues with jar class loading, it is recommended to have this empty and not set. Please make sure to back up this value for when the lab is over. You are welcome to not do this, however weird things may happen when you are running through the labs.

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.

Lab Number 1: Install and Configure SOA & EDS Platform

Get the SOA-P File

In the `${USER_HOME}Downloads/Platforms` directory you will find the SOA-P installer, it platform agnostic and it should look something like this:

```
soa-5.1.0.GA.zip
```

Unzip to Install, Part 1

Create a Servers directory in the user home directory and make this unique based on your initials. Unzip the contents of the file above into that directory:

```
mkdir ${USER_HOME}/ServerXXX
cd ${USER_HOME}/ServerXXX
unzip ${USER_HOME}/Downloads/Platforms/soa-5.1.0.GA.zip
```

Your command should look something like this:



Illustration 2: Installing SOA-P

Get the EDS File

In the `${USER_HOME}Downloads/EDS` directory you will find the EDS installer, it is also platform agnostic and it should look something like this:

```
eds-5.1.0.GA.zip
```

Unzip to Install, Part 2

Change into the directory created by the unzip operation above (typically `ServerXXX/jboss-soa-p-5`). Unzip the contents of the file above into that directory:

```
cd ${USER_HOME}/ServerXXX/jboss-soa-p-5
unzip ${USER_HOME}/Downloads/EDS/eds-5.1.0.GA.zip
```

Your command should look something like this:


```

cmosher@cmosher:~/NotBackedUp/Demos/LabUser/ServerXXX/jboss-soa-p-5
File Edit View Search Terminal Help
[cmosher@cmosher ServerXXX]$ ls
jboss-soa-p-5/
[cmosher@cmosher ServerXXX]$ cd jboss-soa-p-5/
[cmosher@cmosher jboss-soa-p-5]$ unzip ../../Downloads/EDS/eds-5.1.0.GA.zip
Archive:  ../../Downloads/EDS/eds-5.1.0.GA.zip
  creating: eds/
  creating: eds/teiid/
  creating: eds/teiid/adminshell/
  inflating: eds/teiid/adminshell/adminshell.bat
  inflating: eds/teiid/adminshell/adminshell-console.sh
  creating: eds/teiid/adminshell/lib/
  inflating: eds/teiid/adminshell/lib/teiid-adminshell-7.1.1.GA.jar
  inflating: eds/teiid/adminshell/lib/jline-0.9.94.jar
  inflating: eds/teiid/adminshell/lib/teiid-client.jar
  inflating: eds/teiid/adminshell/lib/jansi-1.2.1.jar
  inflating: eds/teiid/adminshell/lib/commons-cli-1.2.jar
  inflating: eds/teiid/adminshell/lib/groovy-all-1.7.2.jar
  inflating: eds/teiid/adminshell/adminshell.sh
  inflating: eds/teiid/adminshell/connection.properties

```

Illustration 3: Installing EDS

Finalize the Install with Ant

Change into the `jboss-soa-p-5/eds` directory created by the unzip operation above. Execute “ant” with no arguments:

```

cd ${USER_HOME}/ServerXXX/jboss-soa-p-5/eds
ant

```

Take the defaults when queried by the script (profile = “default”, CXF install = ‘y’). The dialog will look like this:

```

cmosher@cmosher:~/NotBackedUp/Demos/LabUser/ServerXXX/jboss-soa-p-5/eds
File Edit View Search Terminal Help
[cmosher@cmosher jboss-soa-p-5]$ ls
eds/ jboss-as/ jbp-jpd/ Manifest/ mod_cluster/ picketlink/ resteasy/ seam/
[cmosher@cmosher jboss-soa-p-5]$ cd eds
[cmosher@cmosher eds]$ ant
Buildfile: build.xml

prompteds:
  [input] Enter profile to install EDS to: all, production, or default [default]

validate.profile:

promptcxf:
  [input] Teiid requires CXF to be installed for web services, do you wish to install (y or n) [y]

```

Illustration 4: Finalizing EDS Install with ANT

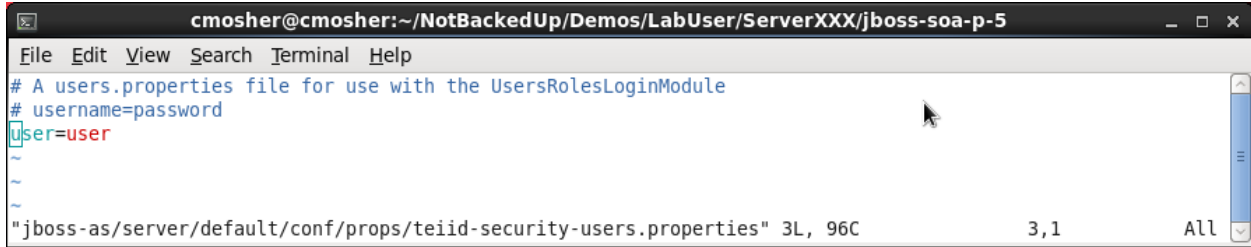


Illustration 8: Enable Teiid User (“user”)

Install JDBC Driver File

EDS does not ship with any JDBC drivers, so we will need to install a MySQL driver to work with the data sources that have been set up for this lab. The driver is in the Downloads/EDS directory and is called:

mysql-connector-java-5.1.13-bin.jar

cd into the default/lib directory and copy the driver into it:

```
cd ${USER_HOME}/ServerXXX/jboss-soa-p-5/jboss-as/server/default/lib
cp ${USER_HOME}/Downloads/EDS/mysql-connector-java-5.1.13-bin.jar .
```

It will look something like this:

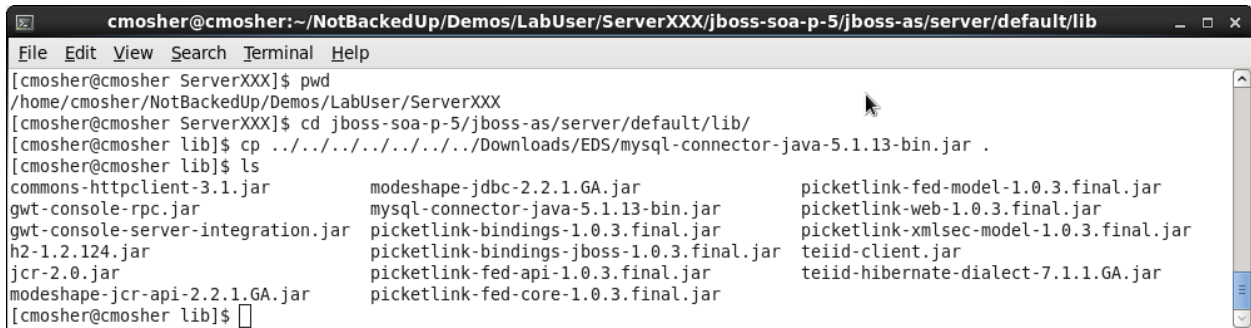
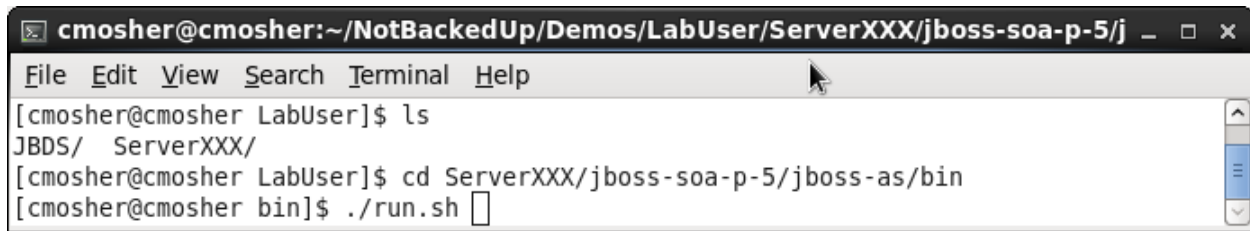


Illustration 9: Copy JDBC Driver File into Default Server lib/ directory

Start the Server

The rest of the lab exercises will need to be able to connect to and work with a running Teiid Server instance. To start it all you need to do is start SOA-P with the default profile (the profile into which EDS was installed). CD into the bin directory of the server and execute ./run.sh with no options to use the default profile:

```
cd ${USER_HOME}/ServerXXX/jboss-soa-p-5/jboss-as/bin
./run.sh
```



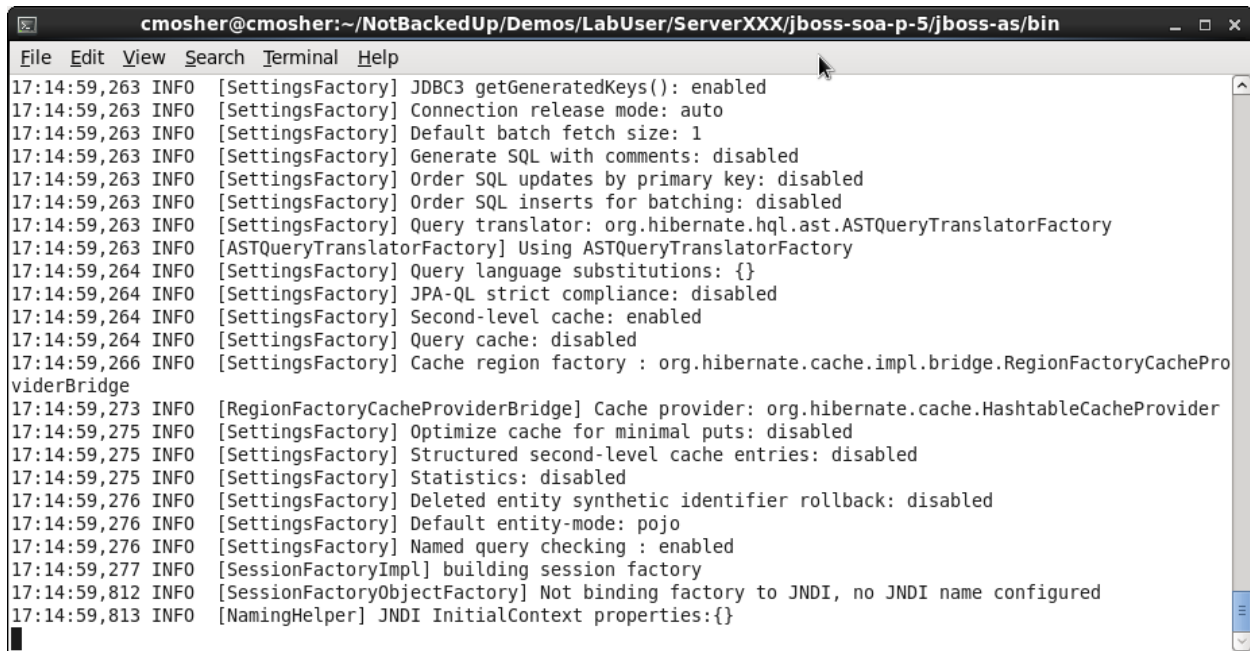
```

cmosher@cmosher:~/NotBackedUp/Demos/LabUser/ServerXXX/jboss-soa-p-5/j _ □ ×
File Edit View Search Terminal Help
[cmosher@cmosher LabUser]$ ls
JBDS/ ServerXXX/
[cmosher@cmosher LabUser]$ cd ServerXXX/jboss-soa-p-5/jboss-as/bin
[cmosher@cmosher bin]$ ./run.sh

```

Illustration 10: Start SOA-P/EDS

A successful Server start will finish with messages like the following:



```

cmosher@cmosher:~/NotBackedUp/Demos/LabUser/ServerXXX/jboss-soa-p-5/jboss-as/bin _ □ ×
File Edit View Search Terminal Help
17:14:59,263 INFO [SettingsFactory] JDBC3 getGeneratedKeys(): enabled
17:14:59,263 INFO [SettingsFactory] Connection release mode: auto
17:14:59,263 INFO [SettingsFactory] Default batch fetch size: 1
17:14:59,263 INFO [SettingsFactory] Generate SQL with comments: disabled
17:14:59,263 INFO [SettingsFactory] Order SQL updates by primary key: disabled
17:14:59,263 INFO [SettingsFactory] Order SQL inserts for batching: disabled
17:14:59,263 INFO [SettingsFactory] Query translator: org.hibernate.hql.ast.ASTQueryTranslatorFactory
17:14:59,263 INFO [ASTQueryTranslatorFactory] Using ASTQueryTranslatorFactory
17:14:59,264 INFO [SettingsFactory] Query language substitutions: {}
17:14:59,264 INFO [SettingsFactory] JPA-QL strict compliance: disabled
17:14:59,264 INFO [SettingsFactory] Second-level cache: enabled
17:14:59,264 INFO [SettingsFactory] Query cache: disabled
17:14:59,266 INFO [SettingsFactory] Cache region factory : org.hibernate.cache.impl.bridge.RegionFactoryCacheProviderBridge
17:14:59,273 INFO [RegionFactoryCacheProviderBridge] Cache provider: org.hibernate.cache.HashtableCacheProvider
17:14:59,275 INFO [SettingsFactory] Optimize cache for minimal puts: disabled
17:14:59,275 INFO [SettingsFactory] Structured second-level cache entries: disabled
17:14:59,275 INFO [SettingsFactory] Statistics: disabled
17:14:59,276 INFO [SettingsFactory] Deleted entity synthetic identifier rollback: disabled
17:14:59,276 INFO [SettingsFactory] Default entity-mode: pojo
17:14:59,276 INFO [SettingsFactory] Named query checking : enabled
17:14:59,277 INFO [SessionFactoryImpl] building session factory
17:14:59,812 INFO [SessionFactoryObjectFactory] Not binding factory to JNDI, no JNDI name configured
17:14:59,813 INFO [NamingHelper] JNDI InitialContext properties:{}

```

Illustration 11: Successful start of SOA-P/EDS

Open the Admin Console

We can now open the Admin Console to examine the properties of the running Server instance. Open a web browser and enter `http://localhost:8080/admin-console` as the URL to get to the login screen. Username is "admin" and password is "admin" (as specified in the `soa-users.properties` file above):

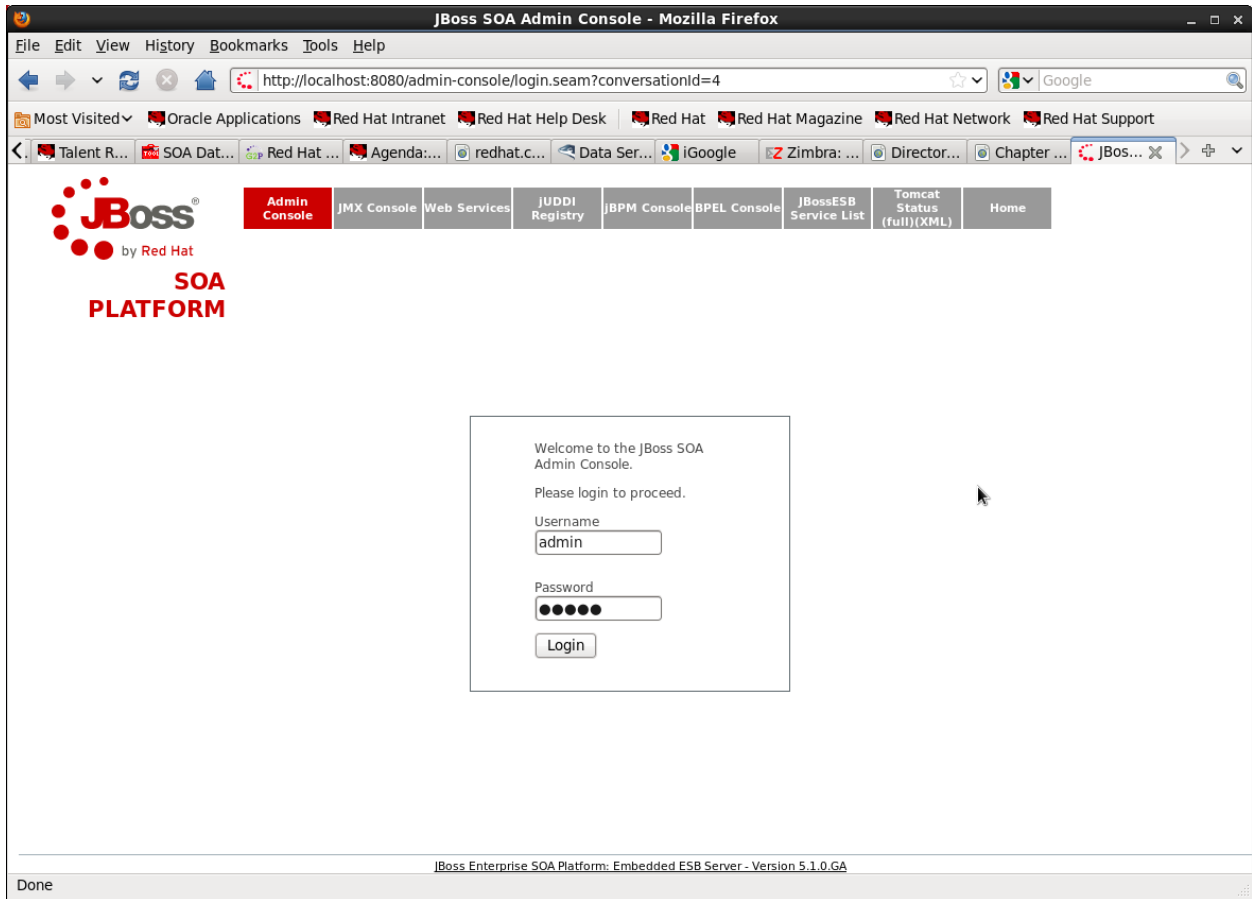


Illustration 12: Admin Console login screen

The Console will open with a view to the top-level server:

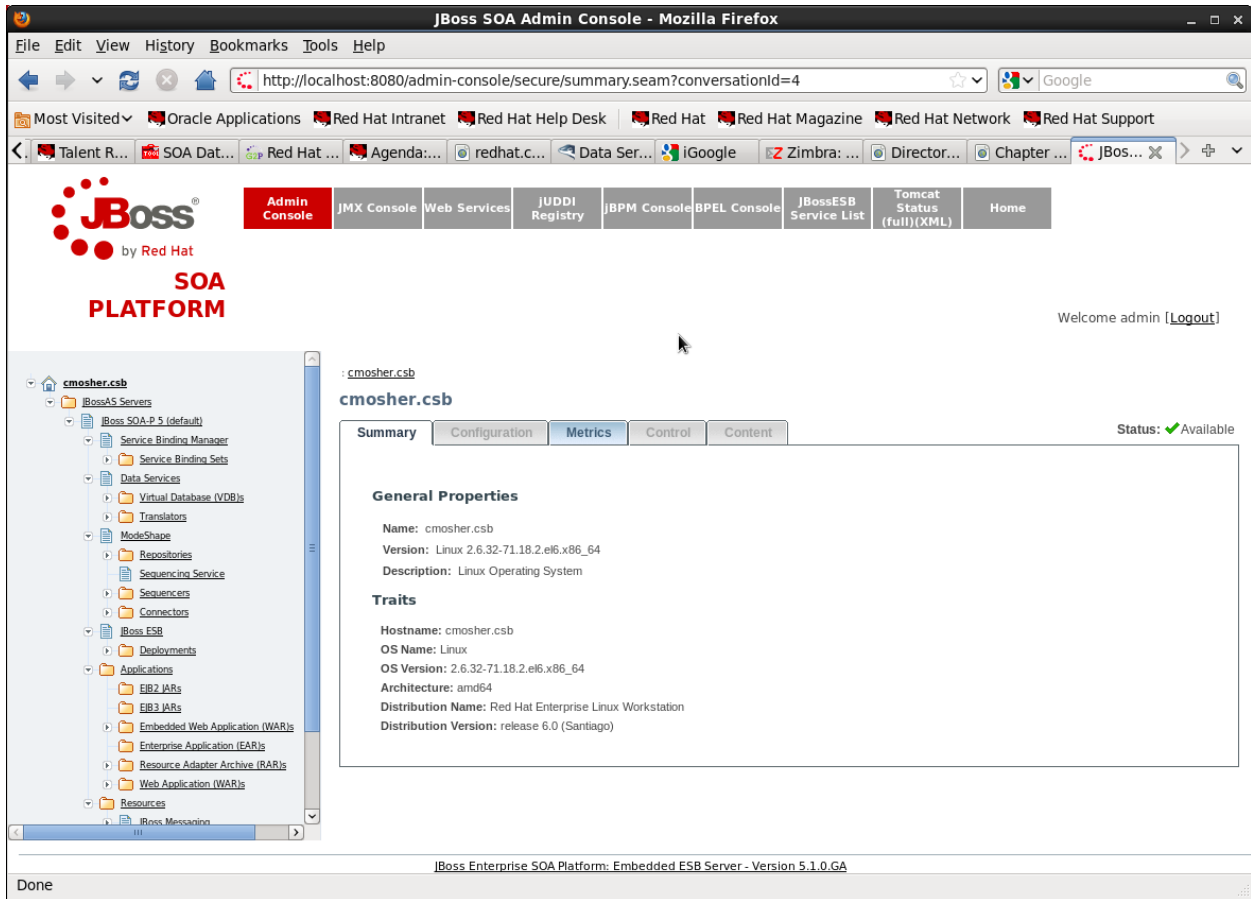


Illustration 13: Admin Console Top View

Select the Data Services tab in the tree-view on the left, and then select the Configurations tab. Scroll around in the Configurations tab to get an idea of the kinds of properties that can be configured:

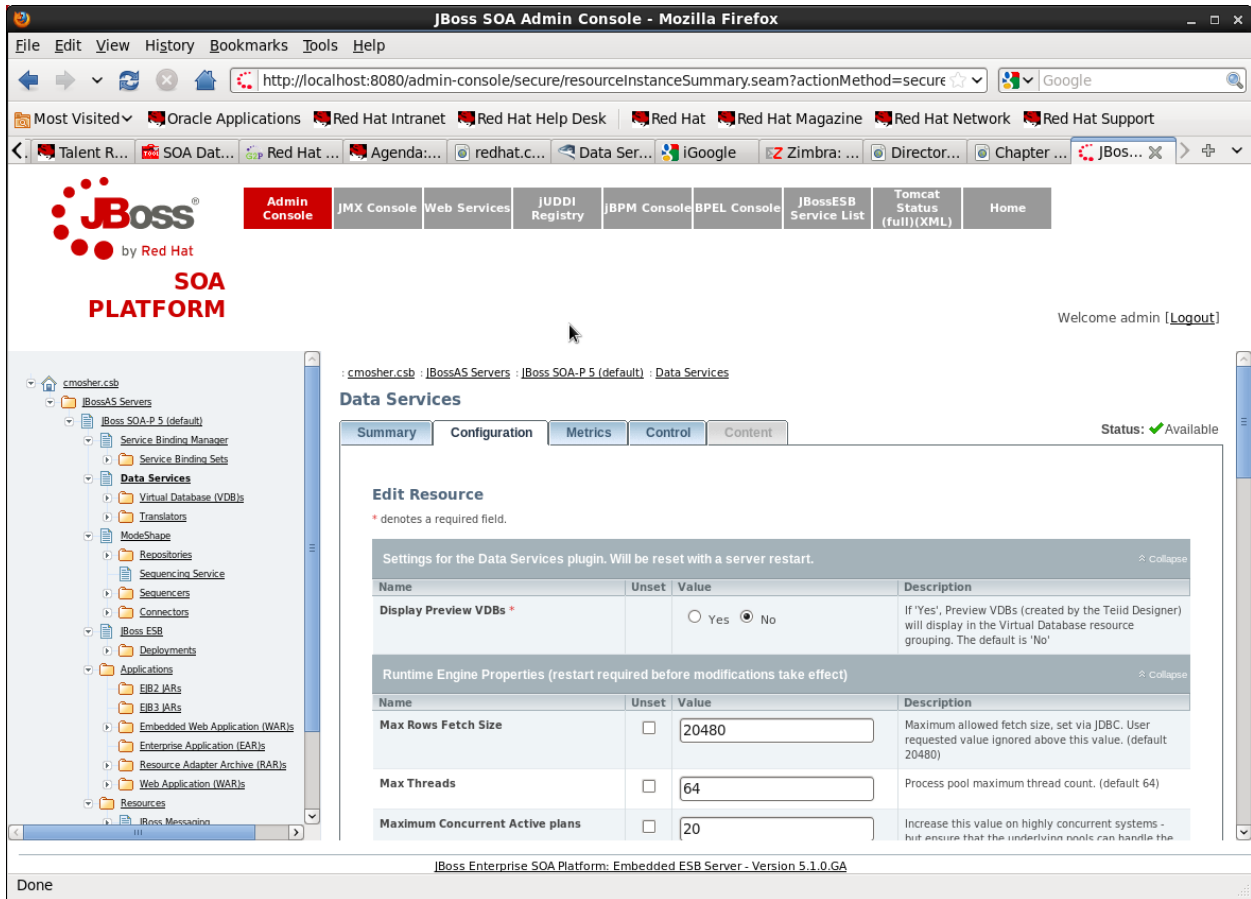


Illustration 14: Data Services Admin Console View, Configuration Tab

Congratulations, you have now completed the first lab. EDS has been installed and configured into a SOA-P instance, an instance has been started and is awaiting requests.

Lab Number 2: Installation of JBDS

Get the File

In the `${USER_HOME}Downloads/JBDS` directory you will find the JBDS installer, which for Linux will look something like this:

```
jbdevstudio-product-eap-linux-gtk-x86_64-4.0.0.v201102161941R-H180-GA.jar
```

Running the Installer

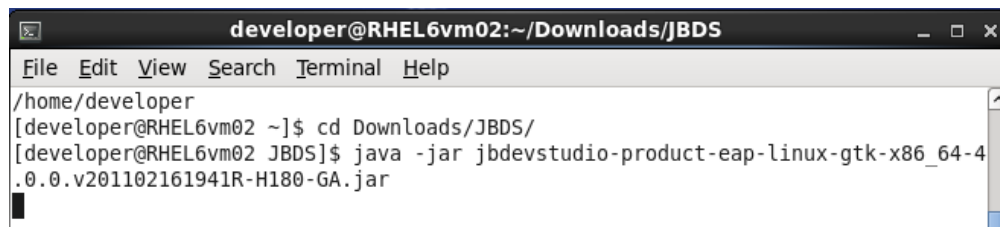
The next step is to run the installer. The command to do that is very simple:

```
cd ${User_Home}/Downloads/JBDS
```

type in `java -jar` and `j` and the tab key to bring up the correct file

```
java -jar ${The_File_Available_For_Linux_Above}
```

A command prompt will be used for this and on Linux it looks like this

A screenshot of a terminal window titled "developer@RHEL6vm02:~/Downloads/JBDS". The terminal shows the following commands and output:

```
File Edit View Search Terminal Help
/home/developer
[developer@RHEL6vm02 ~]$ cd Downloads/JBDS/
[developer@RHEL6vm02 JBDS]$ java -jar jbdevstudio-product-eap-linux-gtk-x86_64-4
.0.0.v201102161941R-H180-GA.jar
```

Illustration 15: Running JBDS Installer

Running this command will then get you to the next step in the process: the visual installer. It will look something like this:



Illustration 16: JBDS Installer Welcome Screen

Select the Next Button and this will get you to this screen:



Illustration 17: Accept License

Please select the "I accept..." radio button to enable the next button. Click the next Button.

The next screen prompts you on where to install JBDS. Make the name unique and don't just select the default. Make sure you write down or remember this path. Please make sure it does not have spaces in it, as older versions had problems running with spaces in the installed path and it should be fixed, but you never know. Also if you already have an existing copy of JBoss Developer Studio installed please raise your hand. Also please put in something unique in the name (like your initials) so that someone else after you will be able to install this lab without any problems.



Illustration 18: Select the Installation Path

Make any changes needed to the installation path and then please select Next. You will be prompted to make this directory if it does not exist. If you do not see a prompt like this, select previous and make the installation path unique.

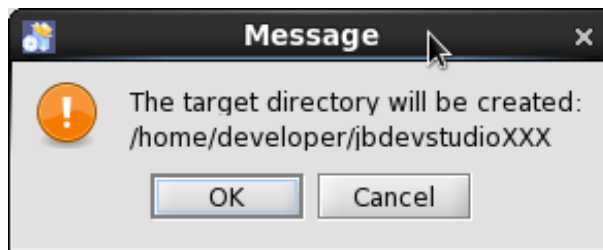


Illustration 19: Confirm Installation Directory

Press OK to confirm that the directory can be created.

Next you will be prompted to use the JDK that was used from your path/java_home properties. No changes are needed with this setting.



Illustration 20: Select Default JVM

Please select Next

The next box will prompt you to install and make available the embedded EAP instance. Please leave this section alone and click Next. Later, we will show you how to add a SOA-P instance that you already installed. You also could add another JBoss EAP/AS instance, but for the purposes of this lab this is not required and not recommended.



Illustration 21: Select existing platforms

You could choose to click the add button and find the SOA-P instance, however we will show you how to do that in the next lab. You could also click find, and hunt down the the installed instance that way.

Select Next

The next screen just shows you what will be installed:

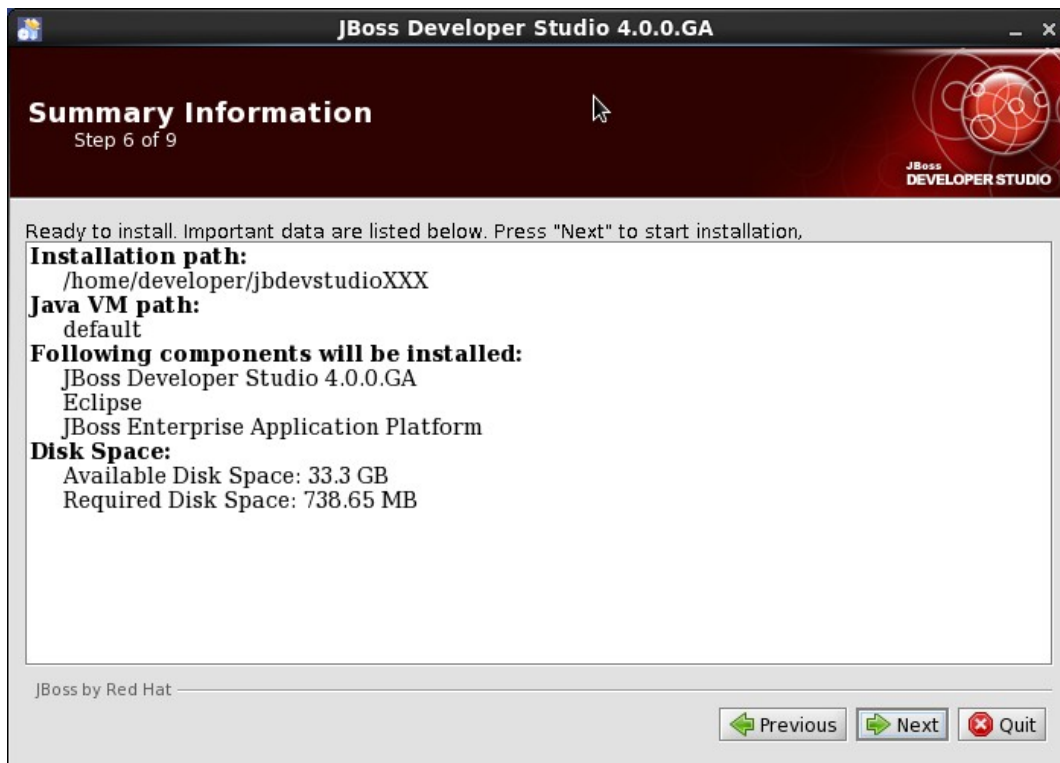


Illustration 22: Confirm installation settings

Click Next

The installer will run for a few minutes, expanding and copying all the necessary files.

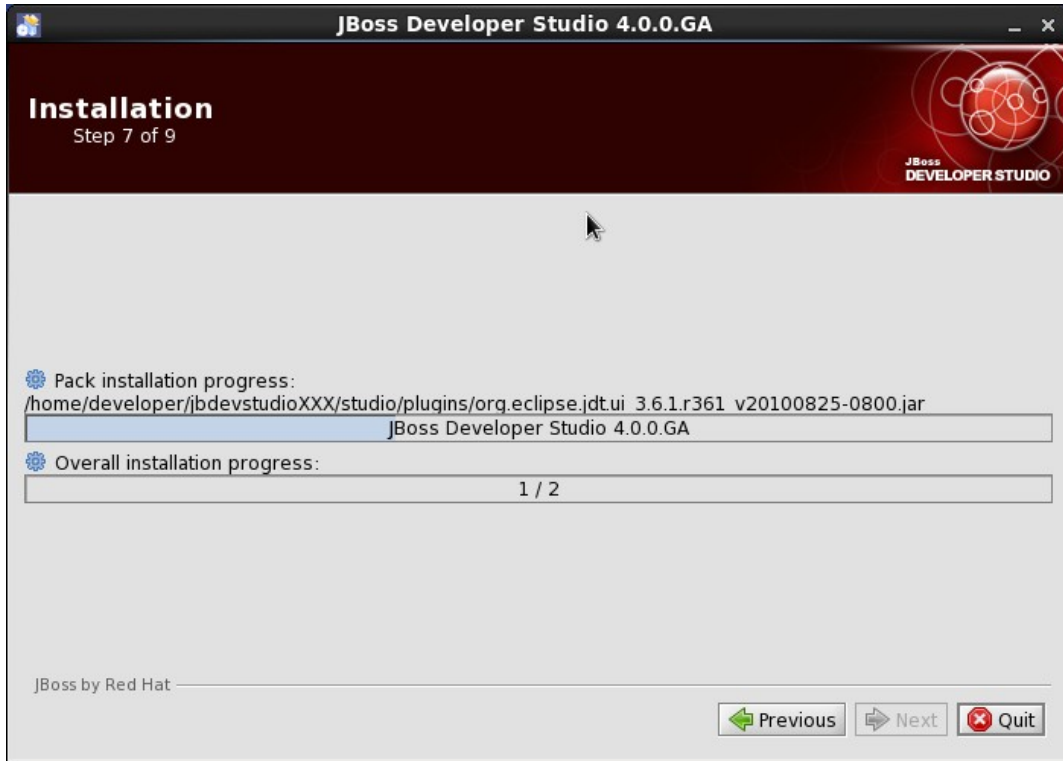


Illustration 23: Running installation

Wait for Next to be available to you once the installer is completed.

The next step will ask where to place short cuts and the like. The defaults are fine for this lab.



Illustration 24: Shortcut locations

Select Next.

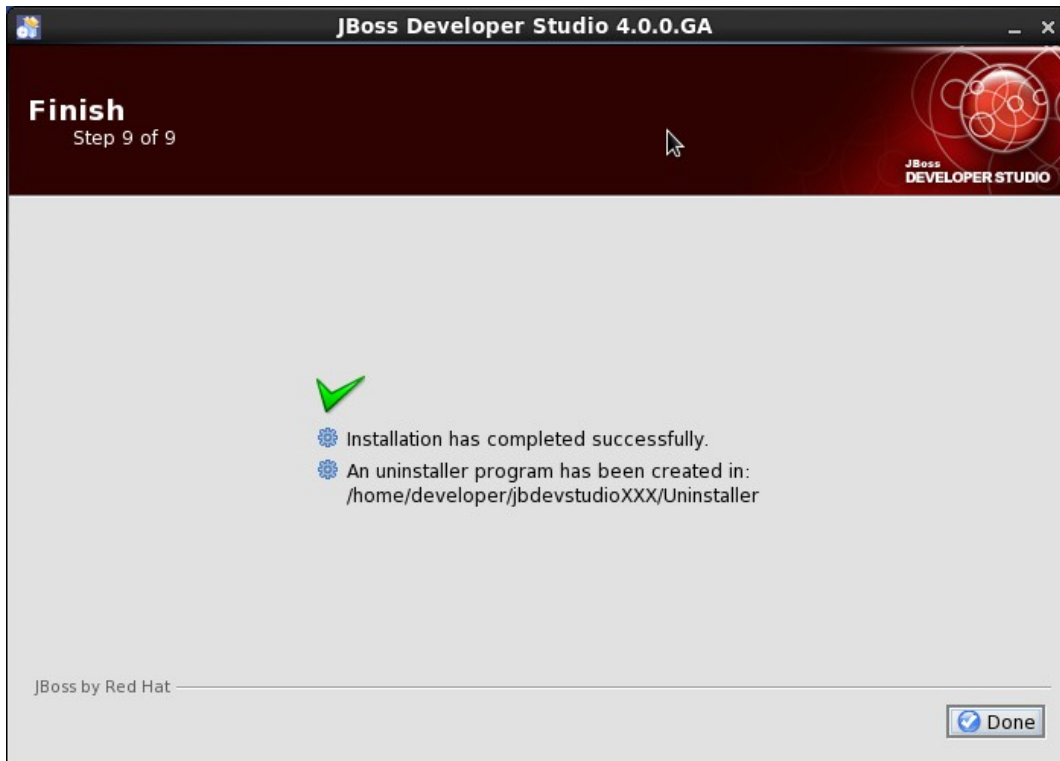


Illustration 25: Installation complete

Please Select Done.

The final step is to install the Teiid update patch via the file located at Downloads/JBDS/jbosstools-teiid-designer-7.1.1-Update-2011-04-21_13-58-39-H167.zip

To do this we will need to start up JBDS for the first time.

Start JBDS

To start JBDS you will have to find the JBDS instance we just installed. There should be shortcuts in your programs menu or maybe even on your desktop depending on what you selected above when installing. On Linux it will be installed in Applications -> Programming -> JBoss Developer Studio 4.0.0.GA as seen below. You might need to restart your machine to see this:

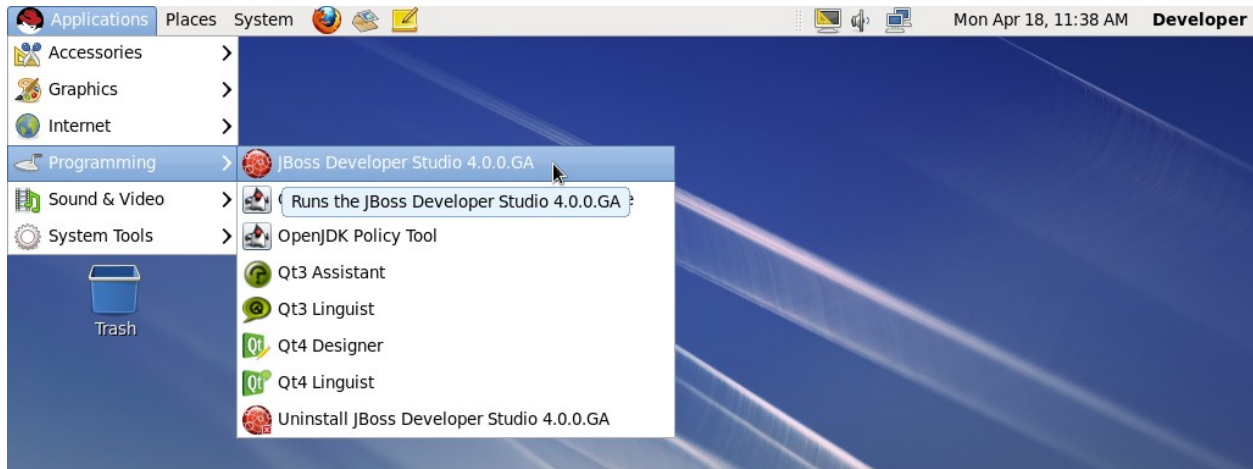


Illustration 26: Launch JBDS

If you did not install the shortcuts, navigate to the directory you installed JBDS and execute the link to the binary directly:

```
cd /home/developer/jbdevstudioXXX (for example)
./jbdevstudio
```

Select a Work Space

Please when selecting a workspace select a new directory or one that does not exist yet:

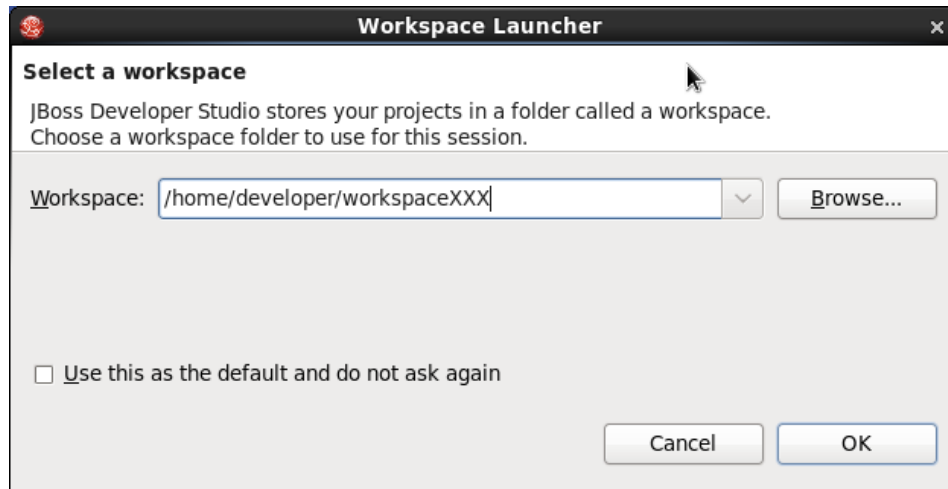


Illustration 27: Select workspace

Select OK and wait for JBDS to fully open

JBDS Start Page

You will then be brought to the main landing page. This page comes up every time you create and open a new Work Space.

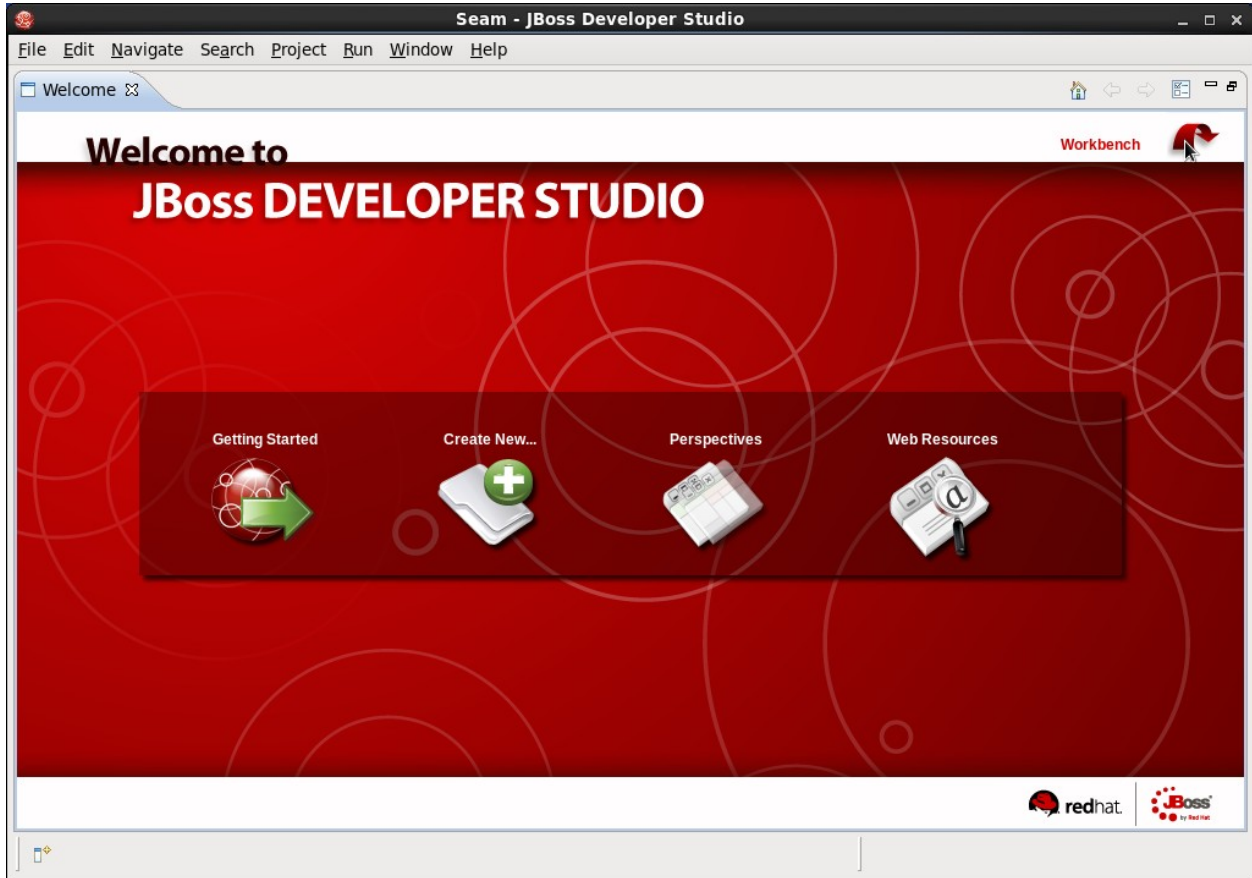


Illustration 28: JBDS initial screen

Select the Workbench Icon above which will bring up JBDS as shown below:

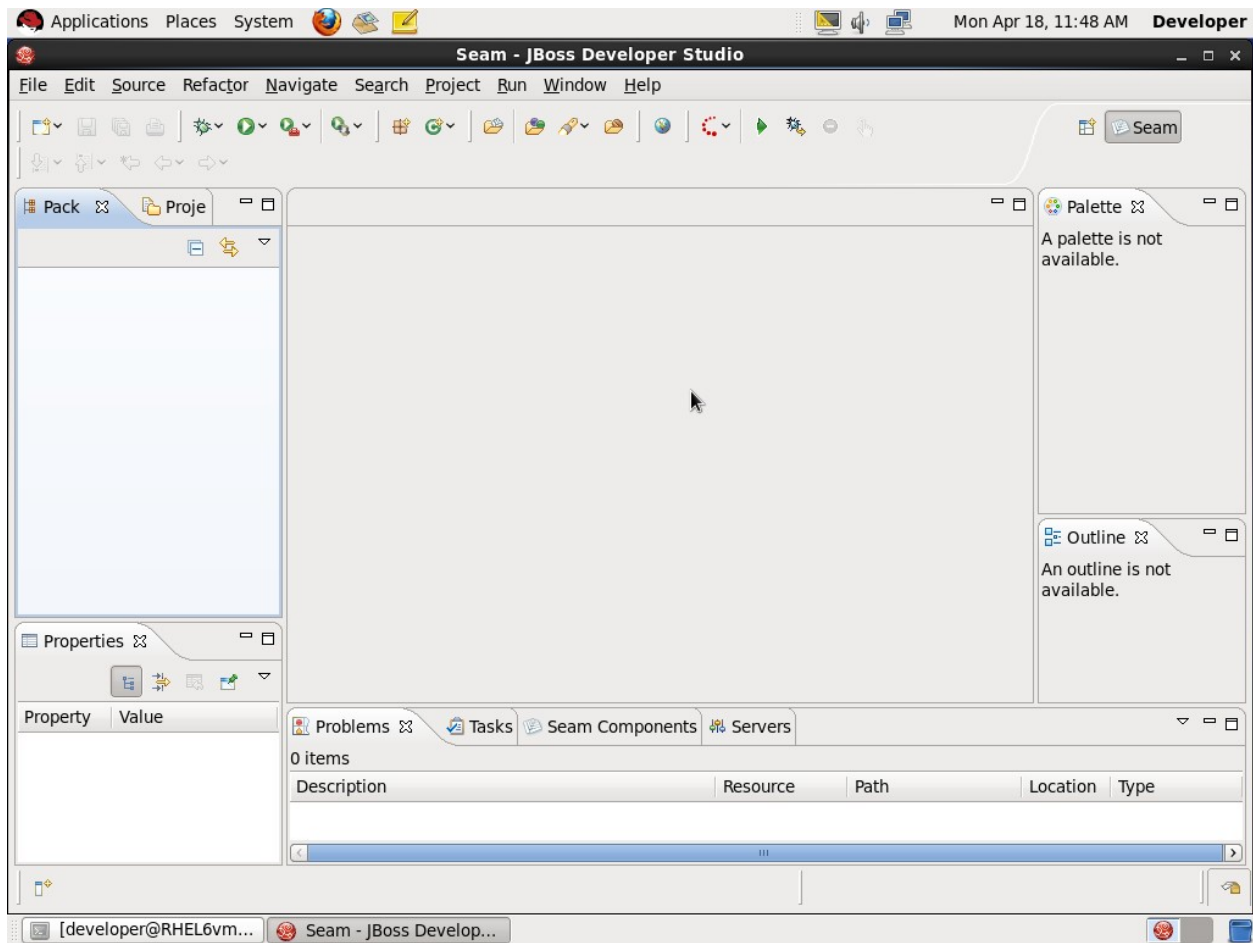


Illustration 29: JBDS initial perspective

Install the Patches

Select Help -> Install New Software... to open the wizard and provide the path to the patch file in Downloads/JBDS as shown below. Supply the full filename of the patch file in the “Work with” box and press return. Then select all of the modules (the top-level check-box) to install:

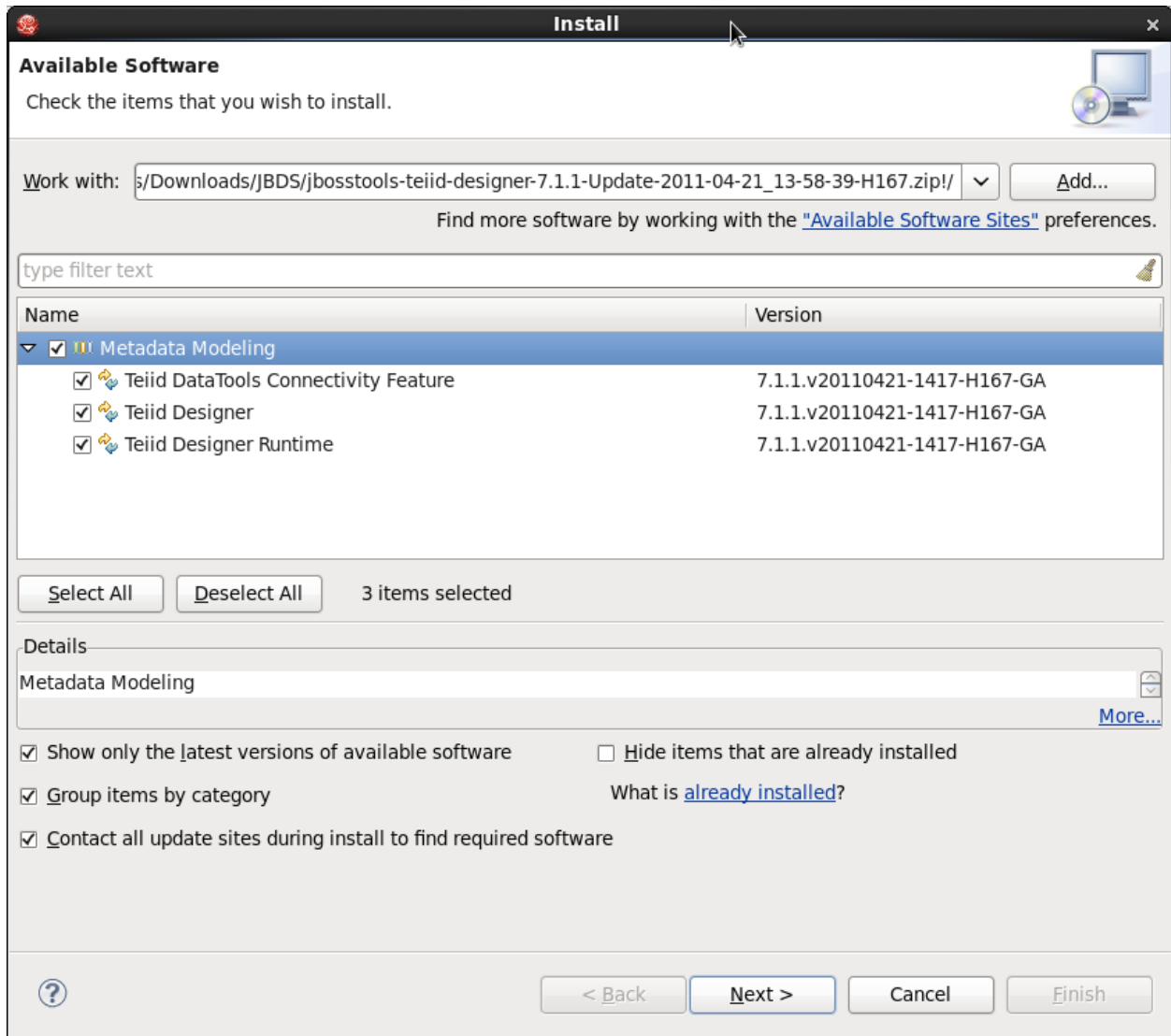


Illustration 30: Install New Software Wizard

Choose Next.

The tool will modify the update as follows:

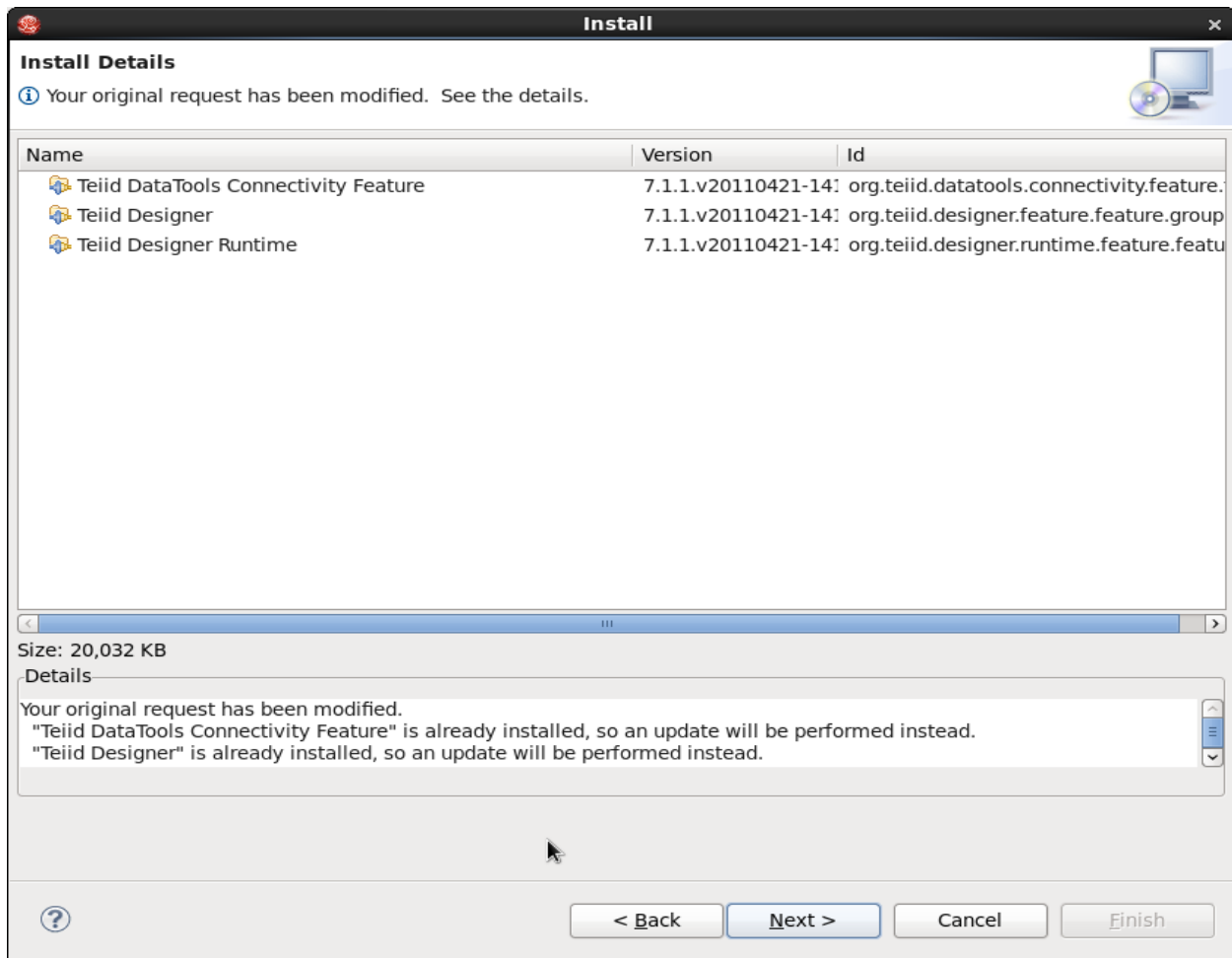


Illustration 31: Install Details

Select Next. You will be prompted to Accept the License terms. Choose Accept and select Finish:

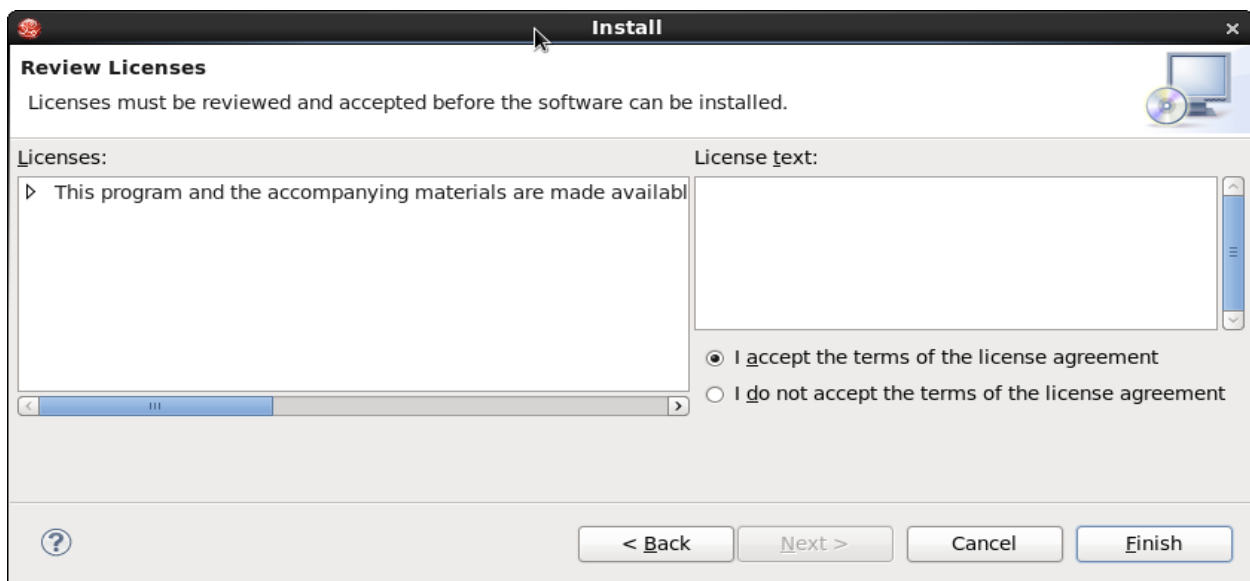


Illustration 32: Accept License Terms

The update will take a minute or two. During the update you will get a security warning, as JBDS cannot verify the patch file provided (it is unsigned). Select OK to continue the update:



Illustration 33: Security Warning Due to Unsigned Content

You will then be prompted to restart JBDS. Select **Restart Now**:



Illustration 34: Restart after Updates

You have successfully installed and patched JBDS. The next lab will show you how to import data source metadata and begin to work with physical and virtual models.

Lab Number 3: Create a Project and Import Data Sources

Open the Teiid Perspective

To begin this exercise, launch JBDS (if it is not already open from the previous lab), and open the “Teiid Designer” perspective. This is because the Seam perspective is opened by default.

To Open “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:

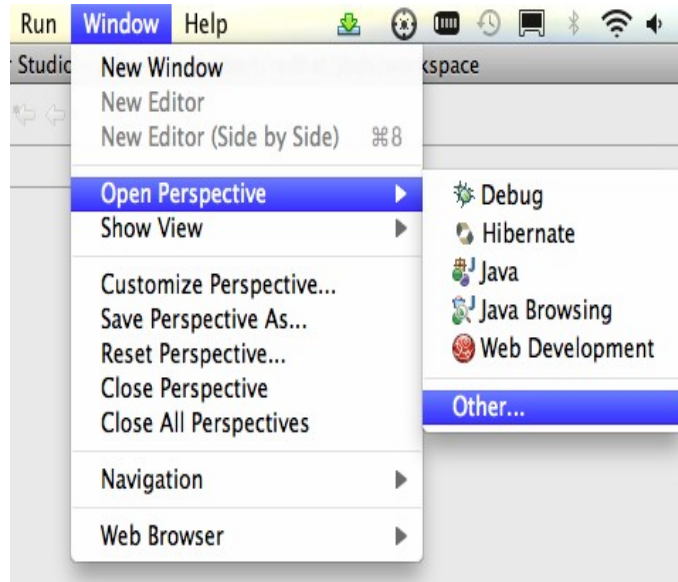


Illustration 35: Select “Other” Perspectives

Select the Teiid Designer from the perspective list as shown:

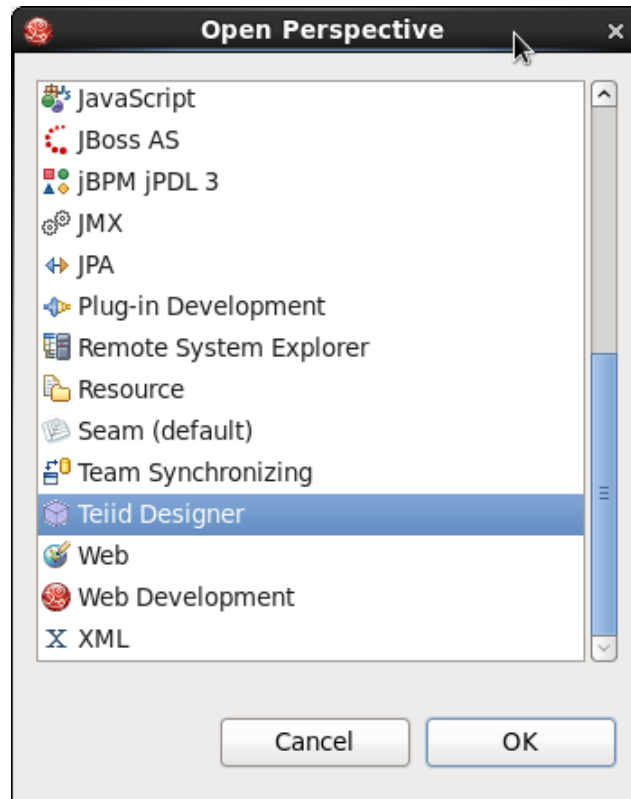


Illustration 36: Select JBoss AS perspective

Choose Teiid Designer and press OK. This will bring you to a screen that looks like this:

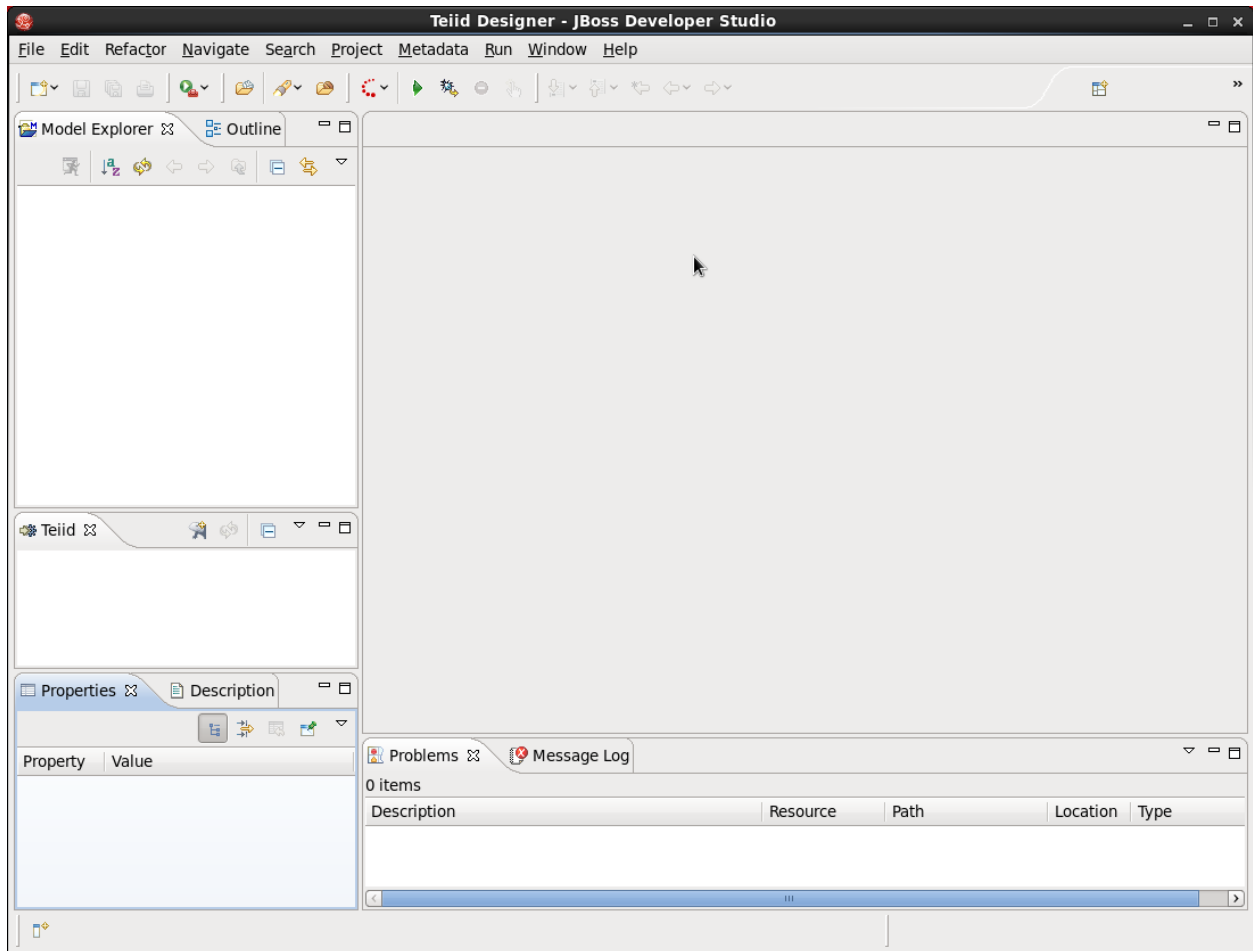


Illustration 37: Teiid Designer Perspective

Create a Connection to the Teiid Server

Connecting to a running Server instance is necessary to execute previews of the data services that we will create. Right-click in the Teiid panel on the left side and select New Teiid Instance. This will bring up the wizard below. Use all of the provided defaults, and fill in admin/admin for the Teiid Admin connection username/password, and user/user for the Teiid JDBC connection. Check “Save” and “SSL” boxes for the Admin connection, and “Save” (not SSL) for the JDBC connection. Click on Test to test the server connection, the select Finish. The wizard should look like this:

New Teiid Instance

Enter Teiid Instance Information
Define the Teiid Instance connection information.

Teiid Admin Connection Info

Host: localhost

Port number: 31443

User name: admin

Password: ***** Save

URL: mms://localhost:31443 SSL

Set as default Teiid instance on 'Finish'

Teiid JDBC Connection Info

Port number: 31000

User name: user

Password: **** Save

URL: jdbc:teiid:<vdbname>@mm://localhost:31000 SSL

Illustration 38: Connect to Teiid Server Instance

The workspace will now look like this:

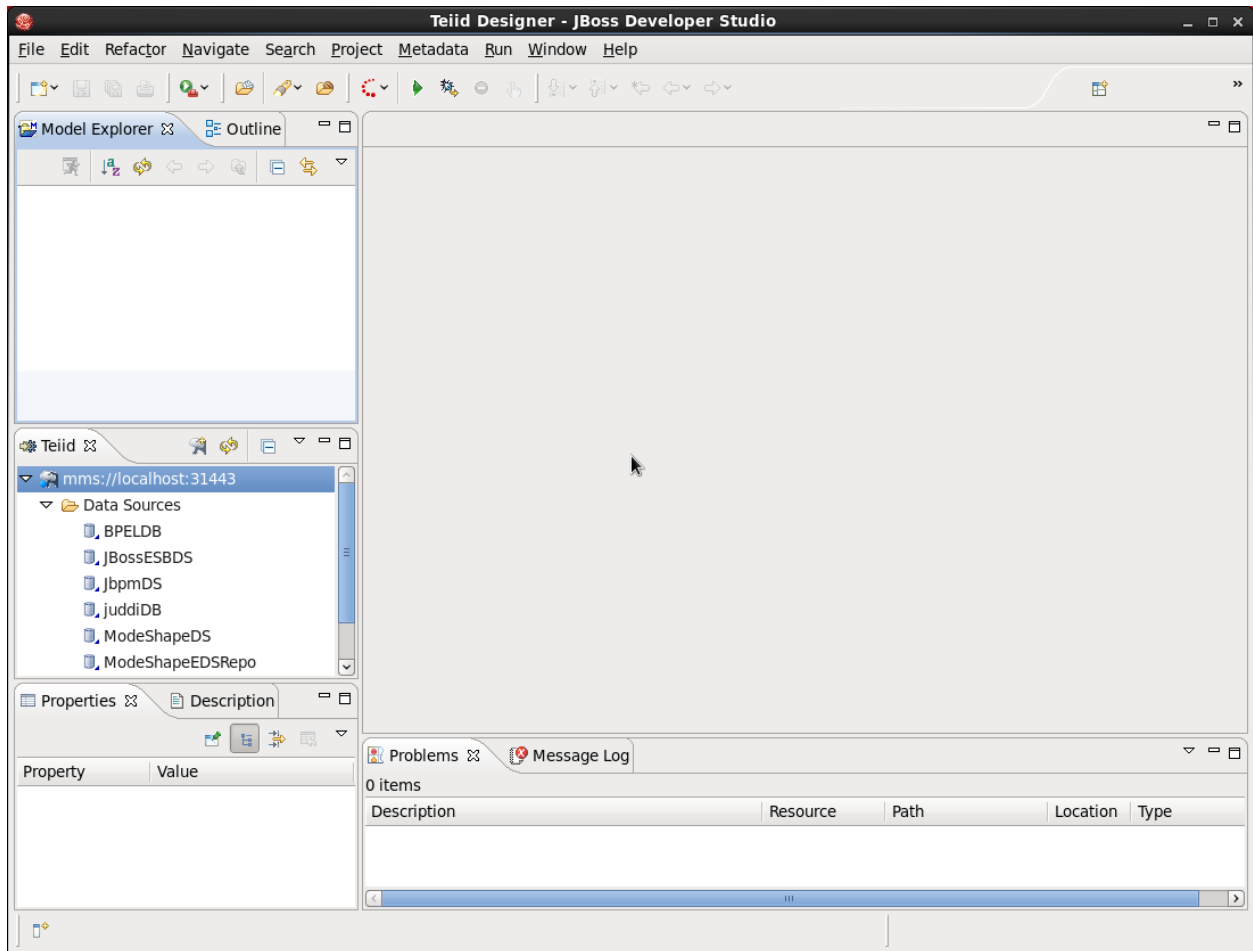
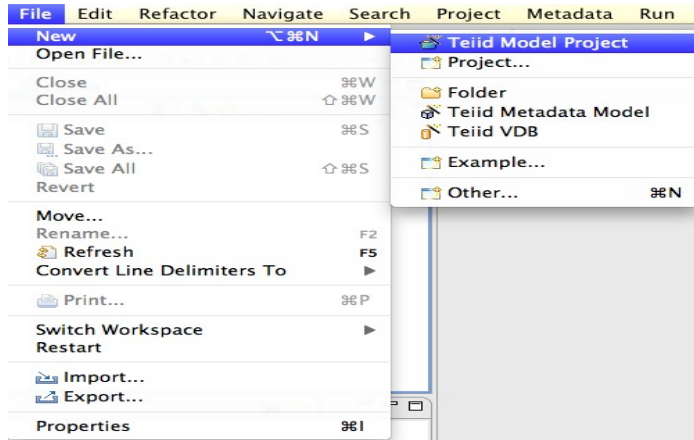


Illustration 39: Teiid Designer Perspective with Active Teiid Server Instance

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 exercises 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:



a) From the menu bar, select File -> New -> Teiid Model Project. The New Model Project Wizard will be displayed.

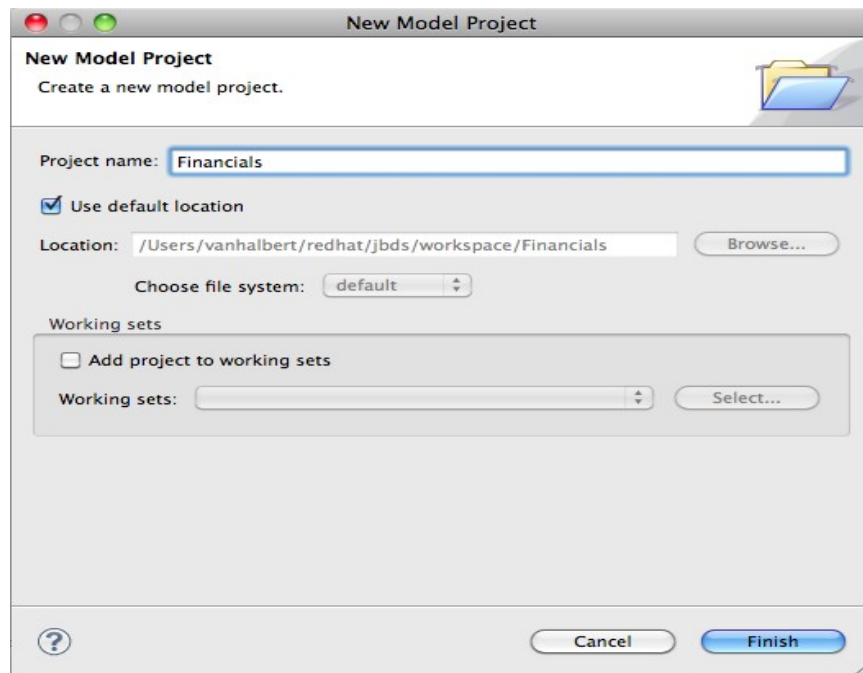


Illustration 40: Create a New Teiid Model Project

b) In the "Project name" field, enter "Financials":

c) Click "Finish", you should now see the Financials project folder in the Model Explorer

Creating Folders

You can use folders to organize models and other files within your projects. This allows the ability to group like items within a single folder or a hierarchical fashion. For the purpose of these exercises we will group our models into folders based on their functional role.

The first folder that we will create will contain our source models. Source models are essentially our link to the physical data. We will group our source models into a folder named DataSources.

- a) Within the Model Explorer, right-click on the Financials project folder and select New -> Folder

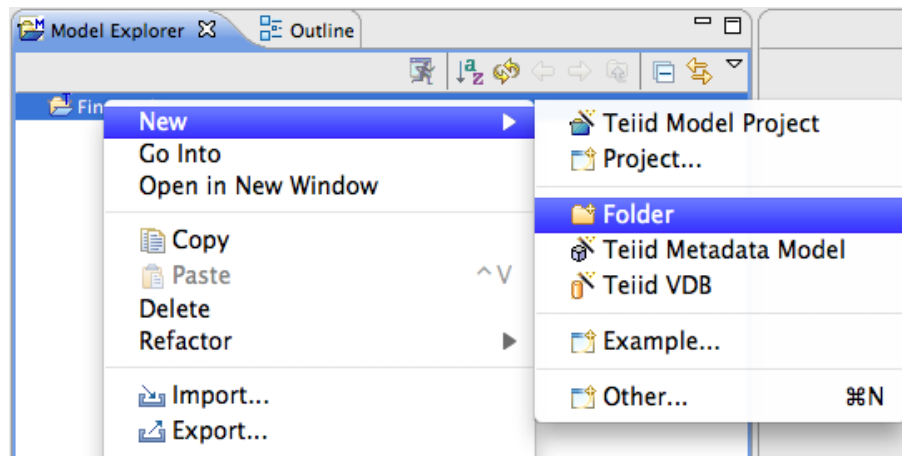


Illustration 41: Create a Folder

- b) Select "Financials" and verify that "Financials" is displayed in the field under "Enter or select the parent folder"
- c) In the "Folder name" field, enter "DataSources"
- d) Click "Finish"

These steps are seen in the wizard below:

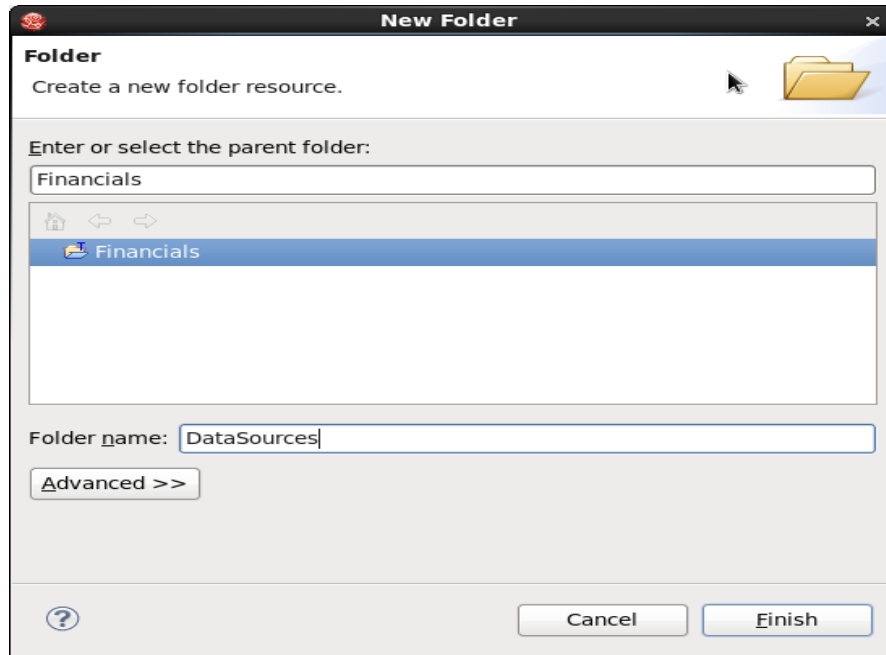
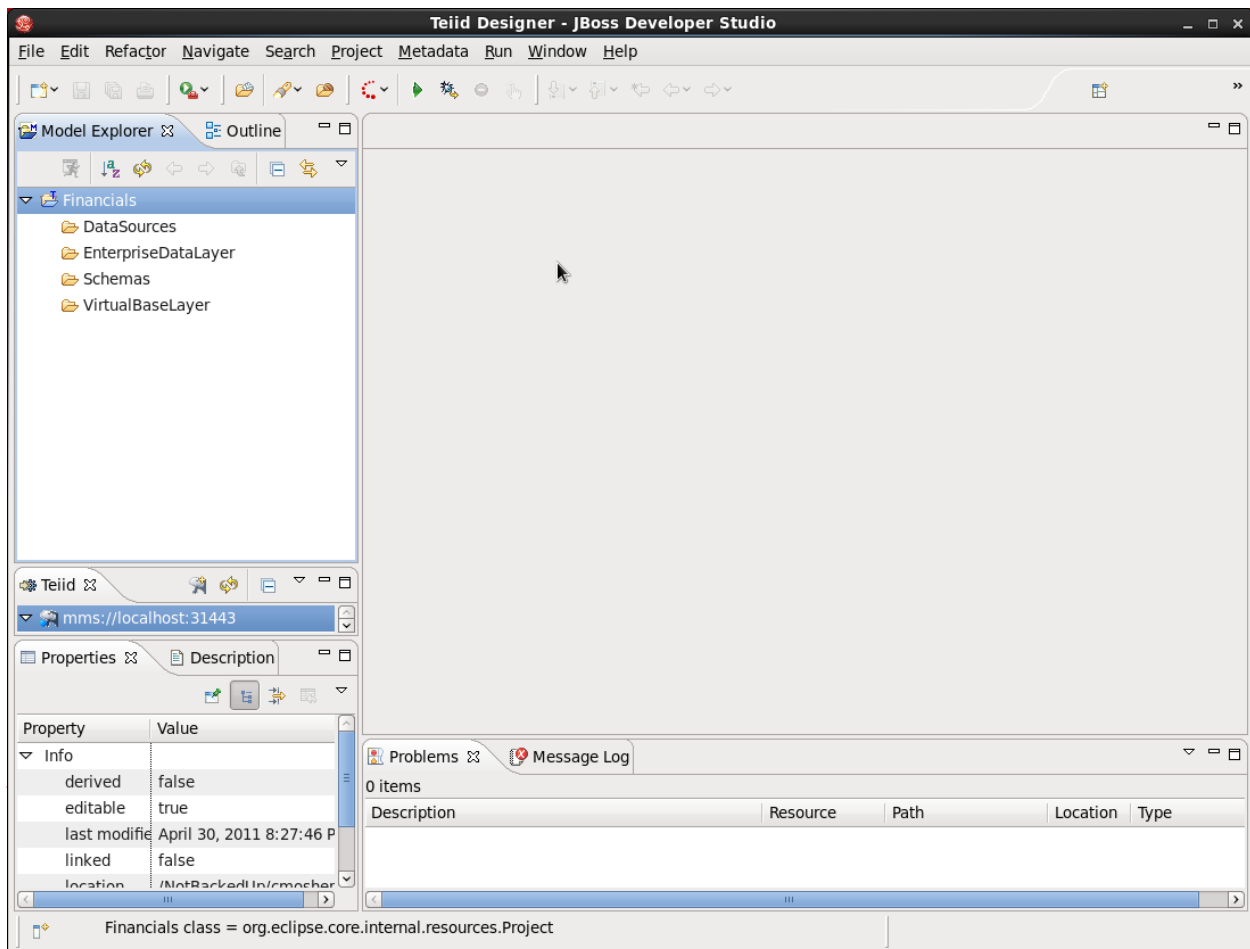


Illustration 42: Name the New Folder

e) Go through the same steps above to create three more folders under Financials called "Schemas", "VirtualBaseLayer", and "EnterpriseDataLayer". The result will look like this:



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 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".

Importing Metadata from the Product Database

For the first part of this exercise, we will import metadata from a JDBC Database that contains the Products schema. This will create this source model by reading the metadata from our Products schema and selecting what features and items we want included in our source model.

- 1) Right-click on the "DataSources" folder and select "Import ...". In Import wizard selection dialog, select the arrow next to "Teiid Designer" to expand the import options. Now select "Metadata from JDBC Database" and click "Next" :

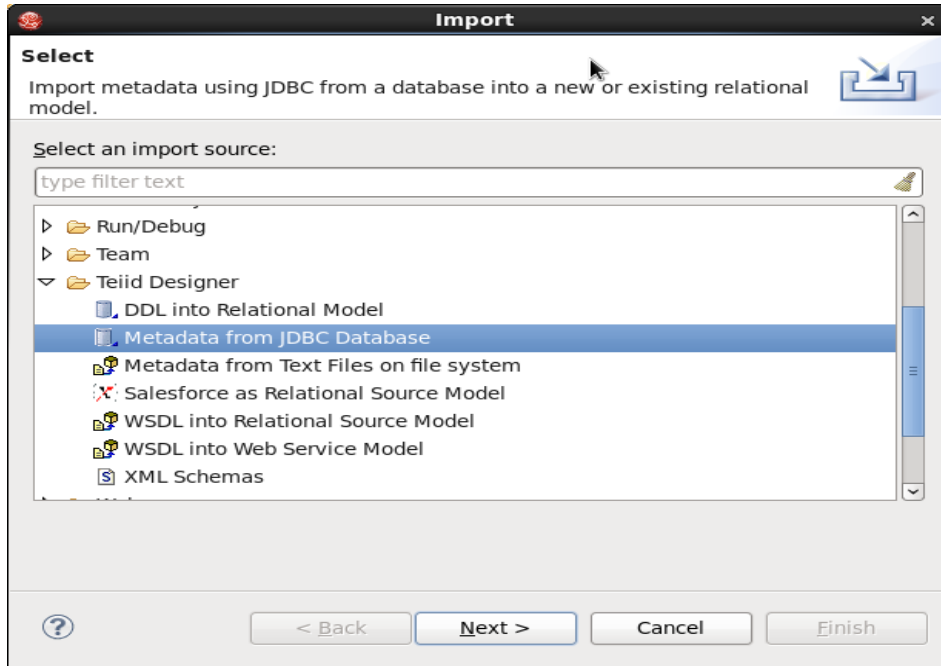


Illustration 44: Select Metadata from JDBC Database

2) In the “Import Database via JDBC” wizard, you will need select a Connection Profile. If a connection profile doesn't exist for the database that contains the Product schema, then select the “New” button to create it:

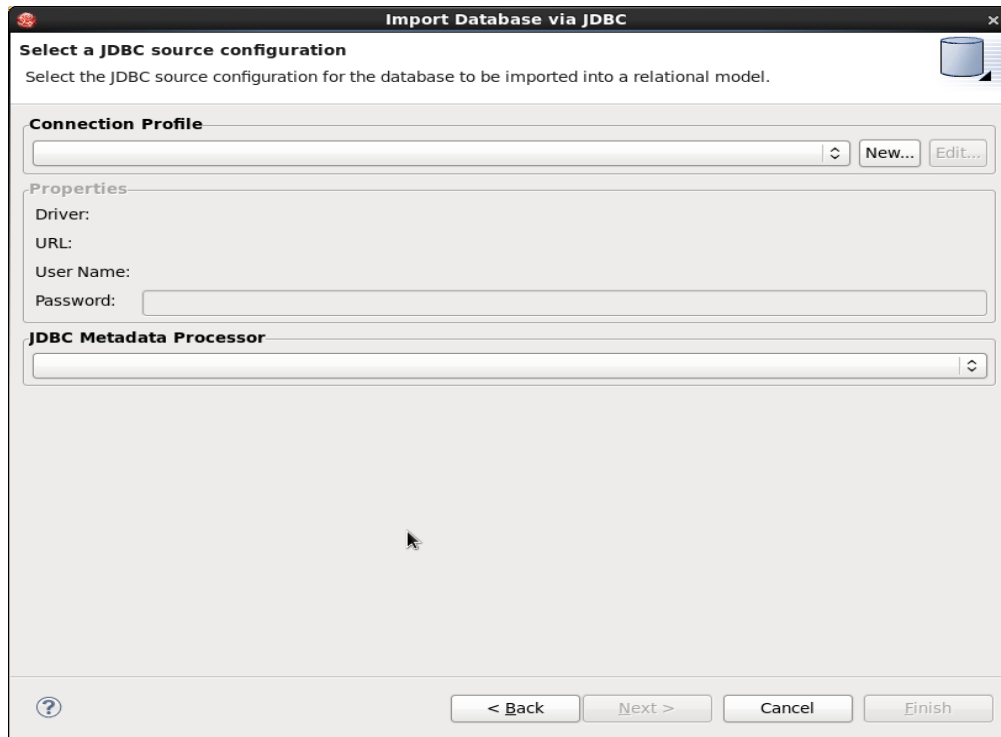


Illustration 45: Select New Connection Profile.

3) The Connection Profile Wizard will come up. Scroll through the list to see the supported databases, then choose "MySQL" for the Connection Profile Type, and enter "Products" for the Name. Press Next:

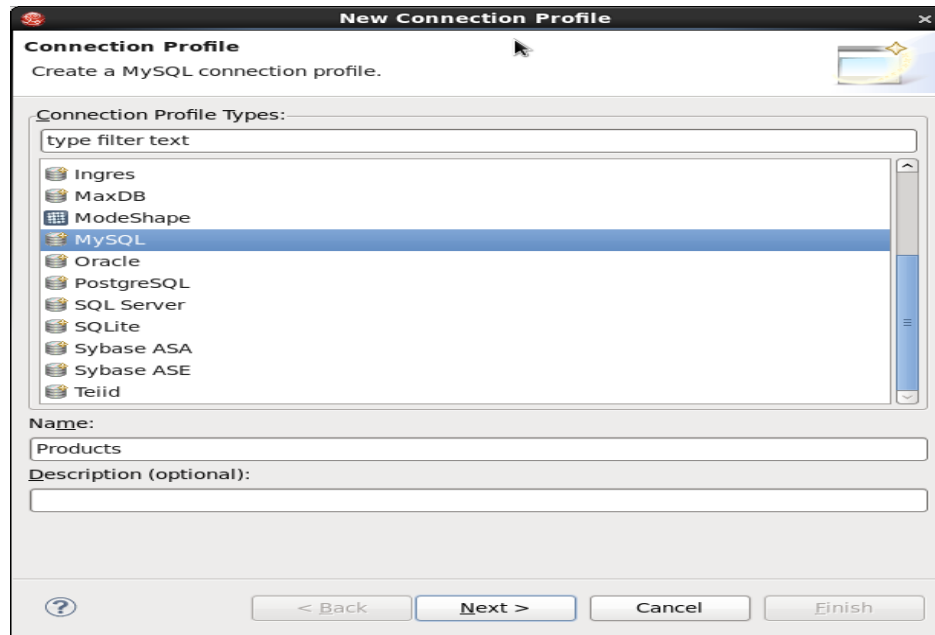


Illustration 46: Select MySQL & Name the Connection

4) 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 won't be, the first time you run through these steps), then select the cross-hair icon, "New Driver Definition", which is to the right of the Drivers:

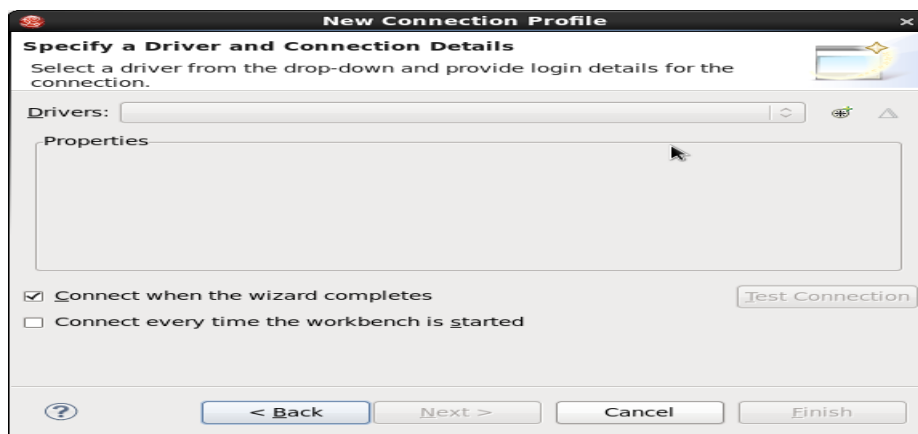


Illustration 47: New Connection Profile, Specify Driver

5) In the “New Driver Definition” dialog, select the MySQL JDBC 5.1 Driver. It will indicate that the driver jar is not found:

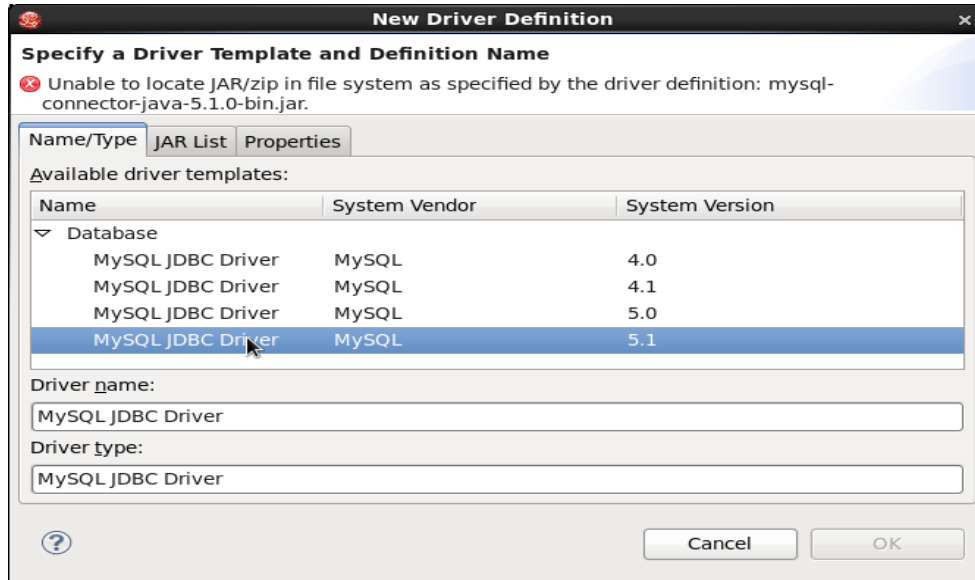


Illustration 48: JDBC .jar Cannot be Located

6) Click on the "JAR List" tab in the New Driver Definition wizard, and select the “Add JAR/Zip” option to select the jdbc driver file to use to access the MySQL .jar that you installed in the EDS/SOA-P profile. Navigate to \${USER_HOME}/ServerXXX/jboss-soa-p-5/jboss-as/server/default/lib to select the mysql-connector-java-5.1.13-bin.jar. Press OK:

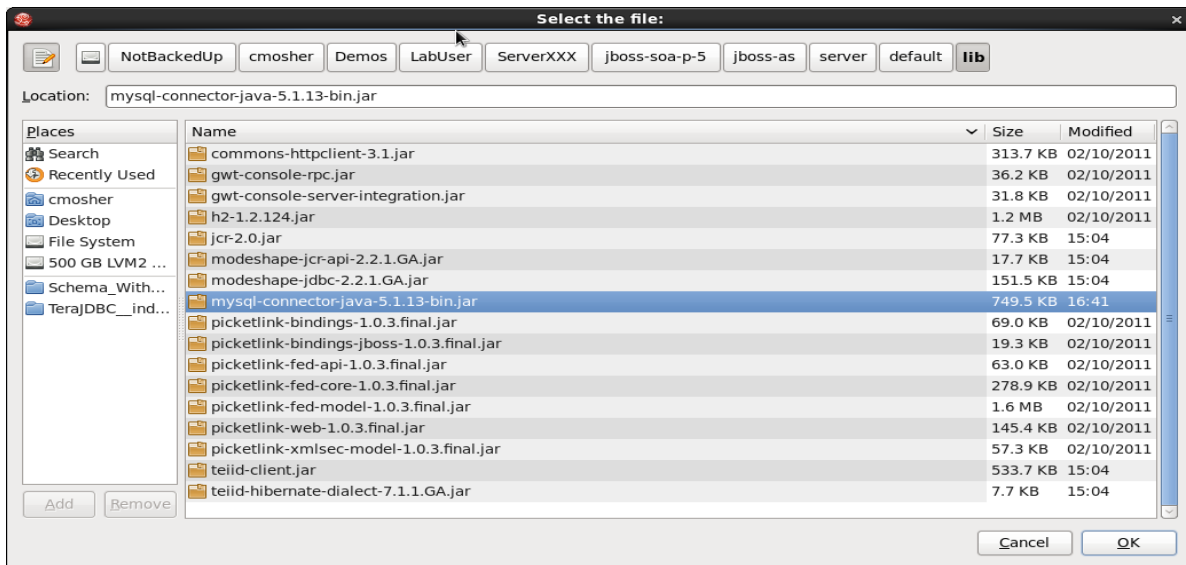


Illustration 49: Select the MySQL Driver .jar from the Server's default/lib directory

7) After pressing OK (above) the warning that the .jar could not be found will go away. Press OK on the dialog below to return to the previous wizard:

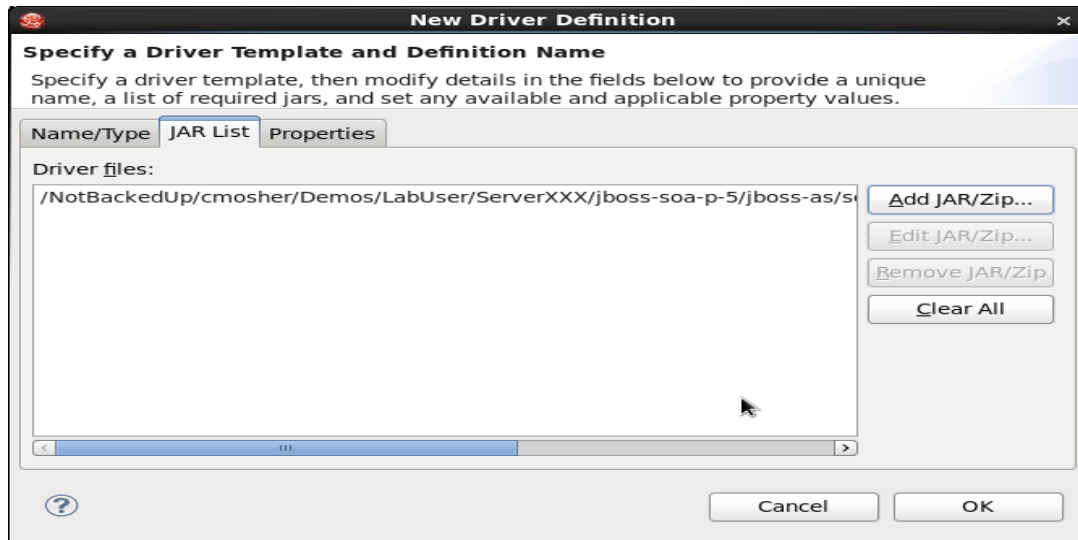


Illustration 50: Select the MySQL Driver .jar from the Server's default/lib directory

8) Fill in the database name (Products), URL (IP address will be provided during the lab), username (root), and password (eds4me) and press "Test Connection" to test the database connection properties. Check the "Save Password" and "Connect when the wizard completes" checkboxes and press Finish:

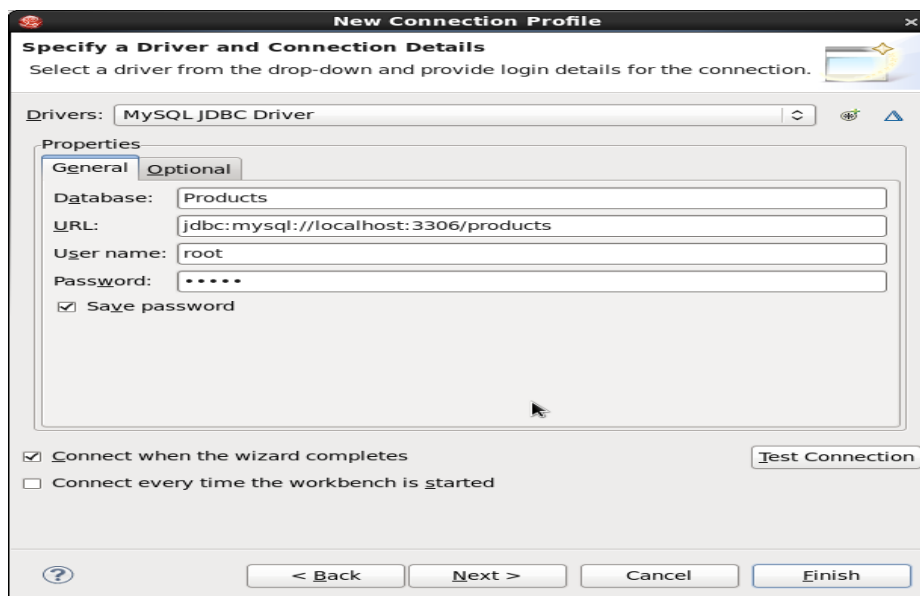


Illustration 51: Fill in the Products database connection properties

9) Clicking "Finish" above will return to the "Import Database via JDBC" dialog, which started the creation of the connection profile in Step 4). Click "Next" to select the metadata types that will be

included when imported. If you click “Finish”, then the defaults will be used and everything will be imported. For this tutorial, we will click “Next”:

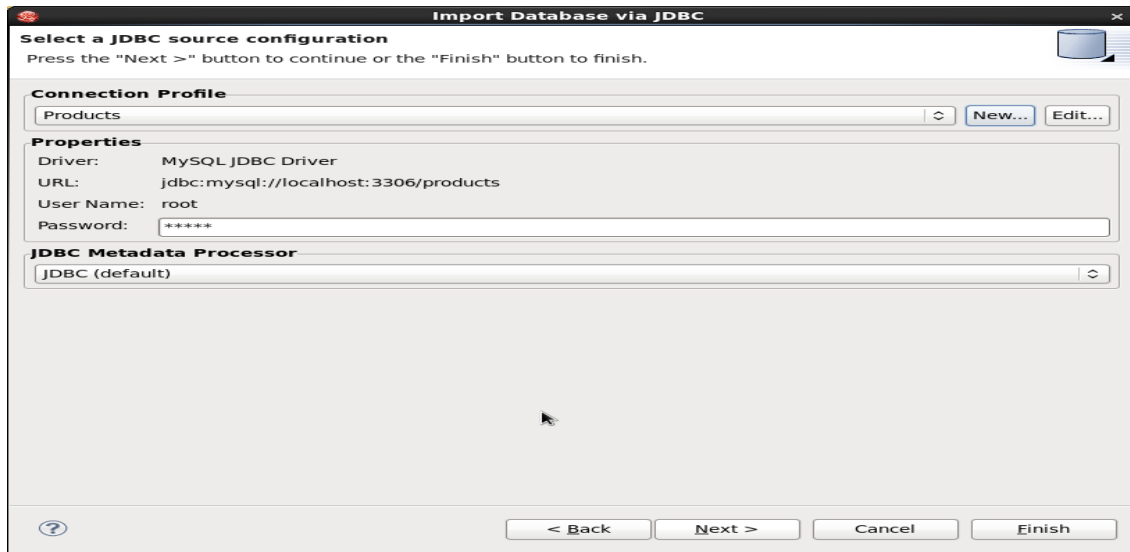


Illustration 52: Fill in the Products database connection properties

10) Select the metadata to be imported. For this tutorial, the diagram indicates what is being selected. Click “Next” to select which tables to be imported. If “Finish” is clicked, then all tables will be included. For this lab, click "Next":

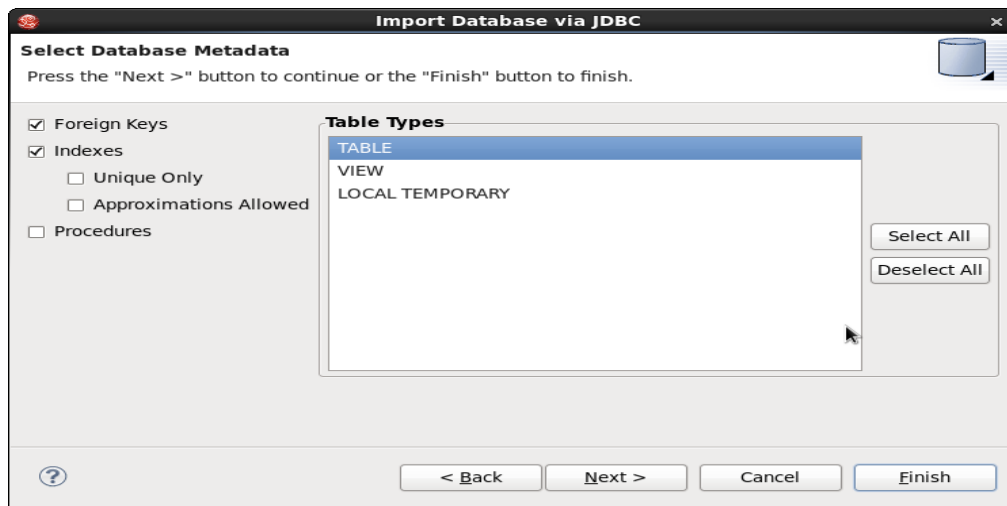


Illustration 53: Select JDBC Metadata to Import

11) Select all of the tables in the products database and click "Next" to see the Model Import Options:

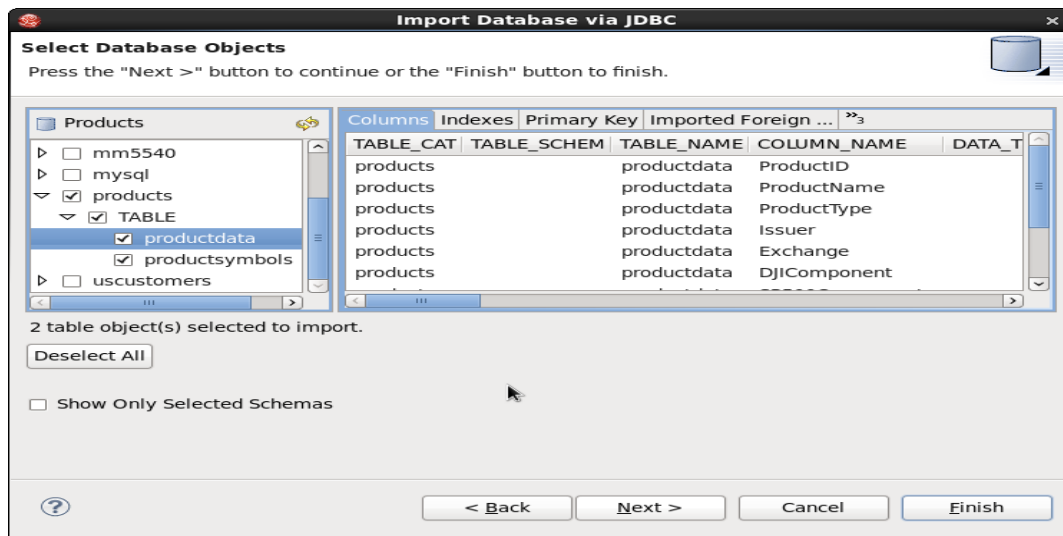


Illustration 54: Select All Tables in "products"

12) This is the final dialog before the actual importing of the metadata to create the model "Products.xmi". For this exercise, we want to keep the model structure as simple as possible. Therefore, remove the check marks next to any items (i.e., Database, Schema) in the "Include In Model" section and click "Finish":

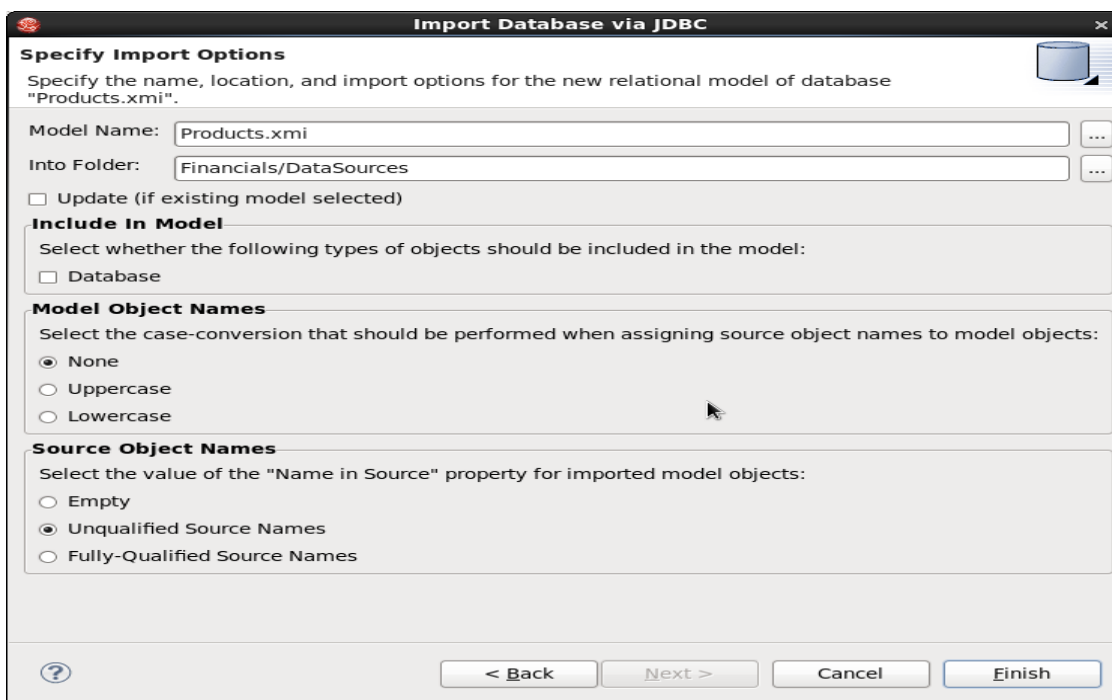


Illustration 55: Last Screen of JDBC Import Wizard. Uncheck "Database" and Finish.

You will now see the Products.xmi source model was opened and was opened and its Package Diagram can be seen in the model view area. Click on PRIMARY (the primary key of the **productdata** table at the bottom) and note that the Key (ProductID) in **productdata** and the Foreign Key in the **productsymbols** table are highlighted. This is because Designer knows via the metadata that all these elements are related:

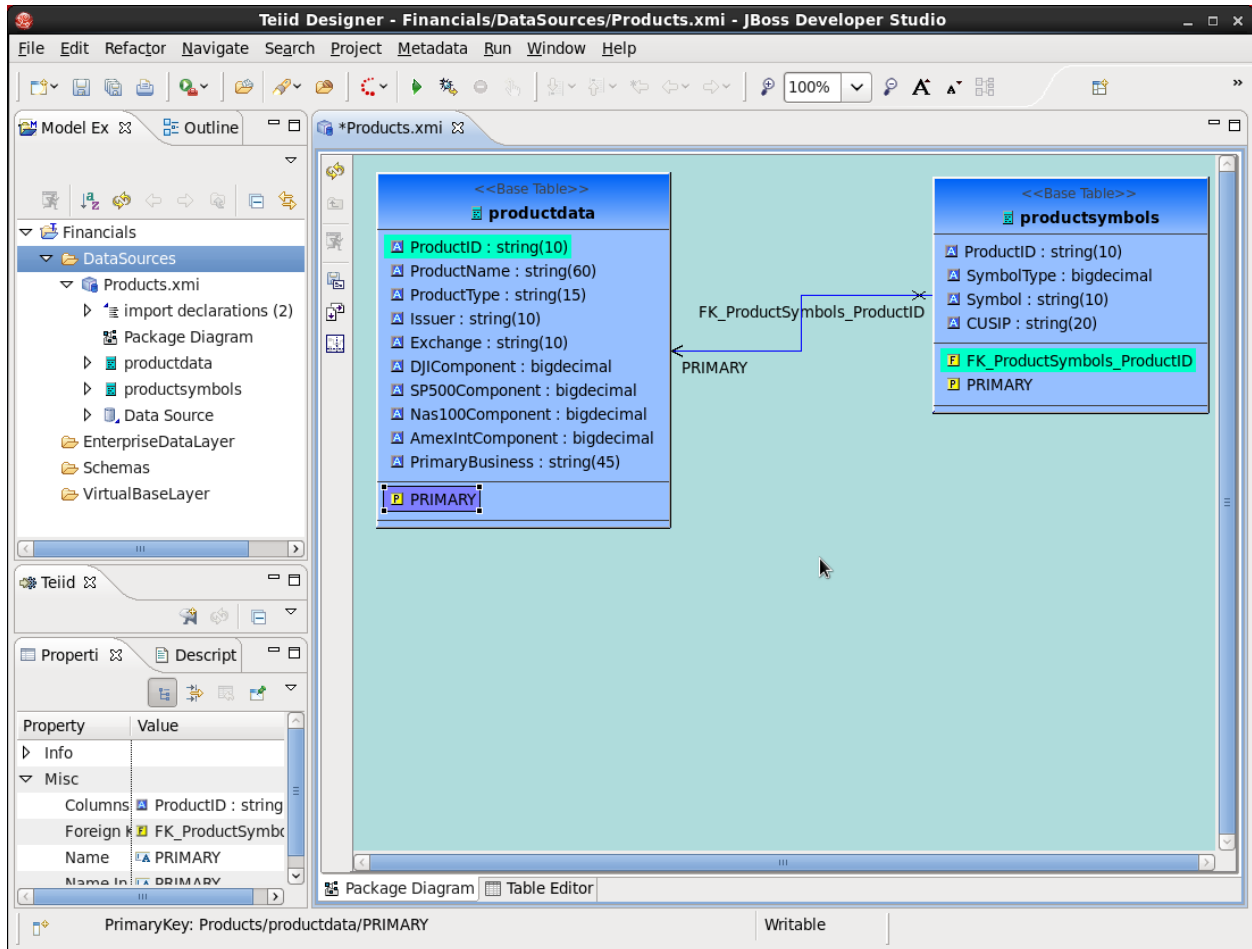


Illustration 56: Physical Metadata Model of Products Database.

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 a simple click of a button. Click on the **productdata** table and note that a small icon to the left of a running man is enabled. Click the button. (NOTE: There is currently an outstanding bug that prevents this from working on Macs.)

When this is first done the following dialogue will pop up:

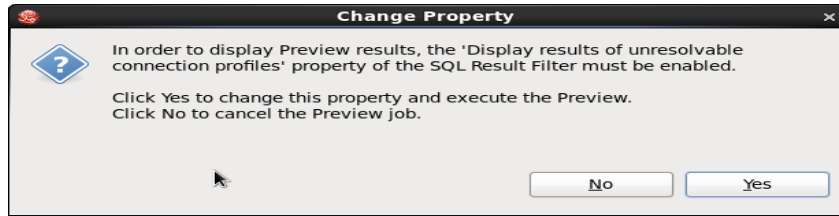


Illustration 57: Enable SQL Result Filter for Preview Capability

Click "Yes" and the query will complete, showing a sampling (the number of rows can be adjusted via preferences) of the data in the database. You will need to select the Result1 sub-tab under the SQL Results tab to see the row data:

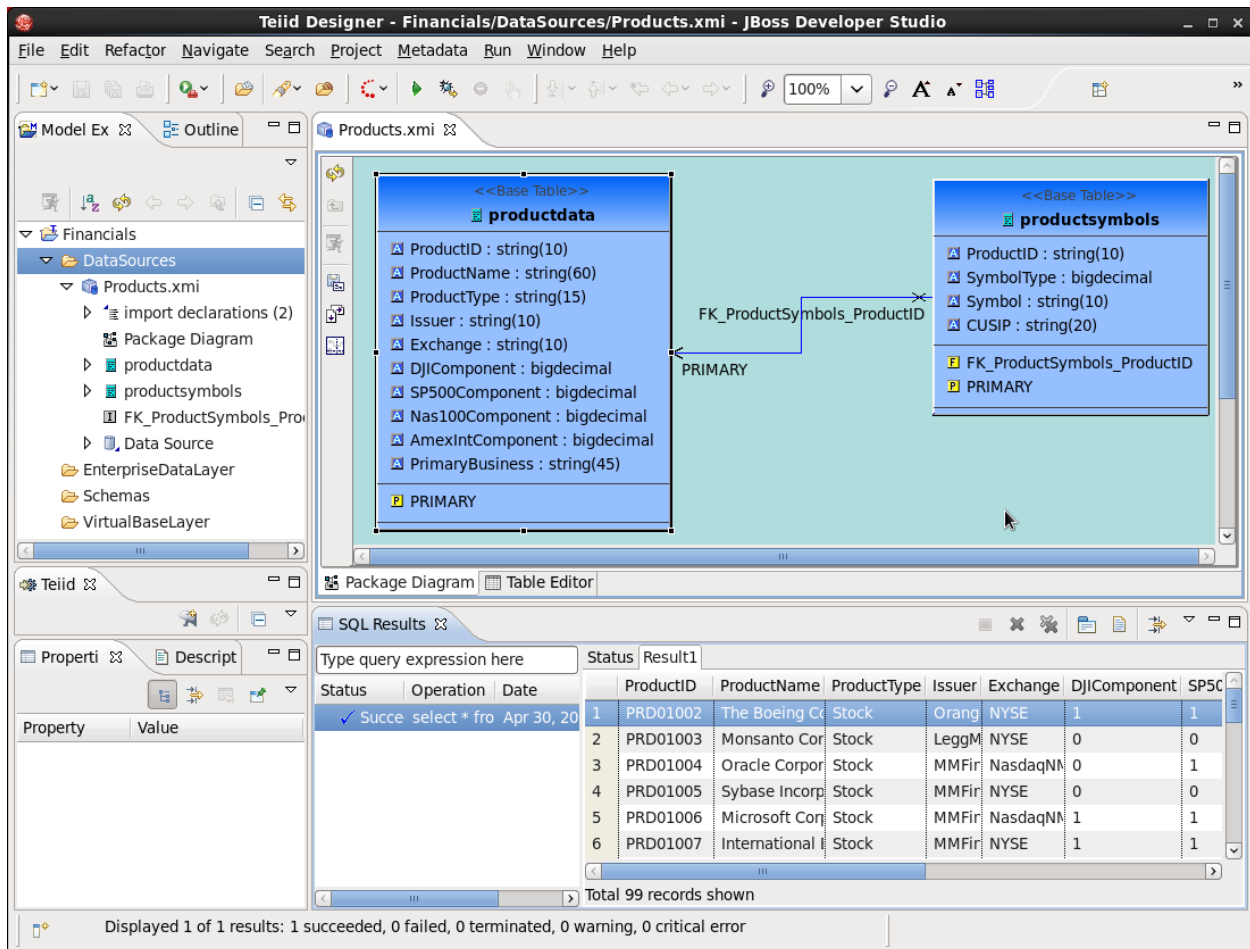


Illustration 58: Preview of ProductData.

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 these 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 usernames/passwords are the same as for the product database (root / eds4me). You will create a new Connection Profile for each source, but you can reuse the MySQL driver. Feel free to preview the data sources after you finish the imports.

When you have completed the imports, the Package Diagram and Model Explorer for US_Customers (for example) will look similar to the following figure:

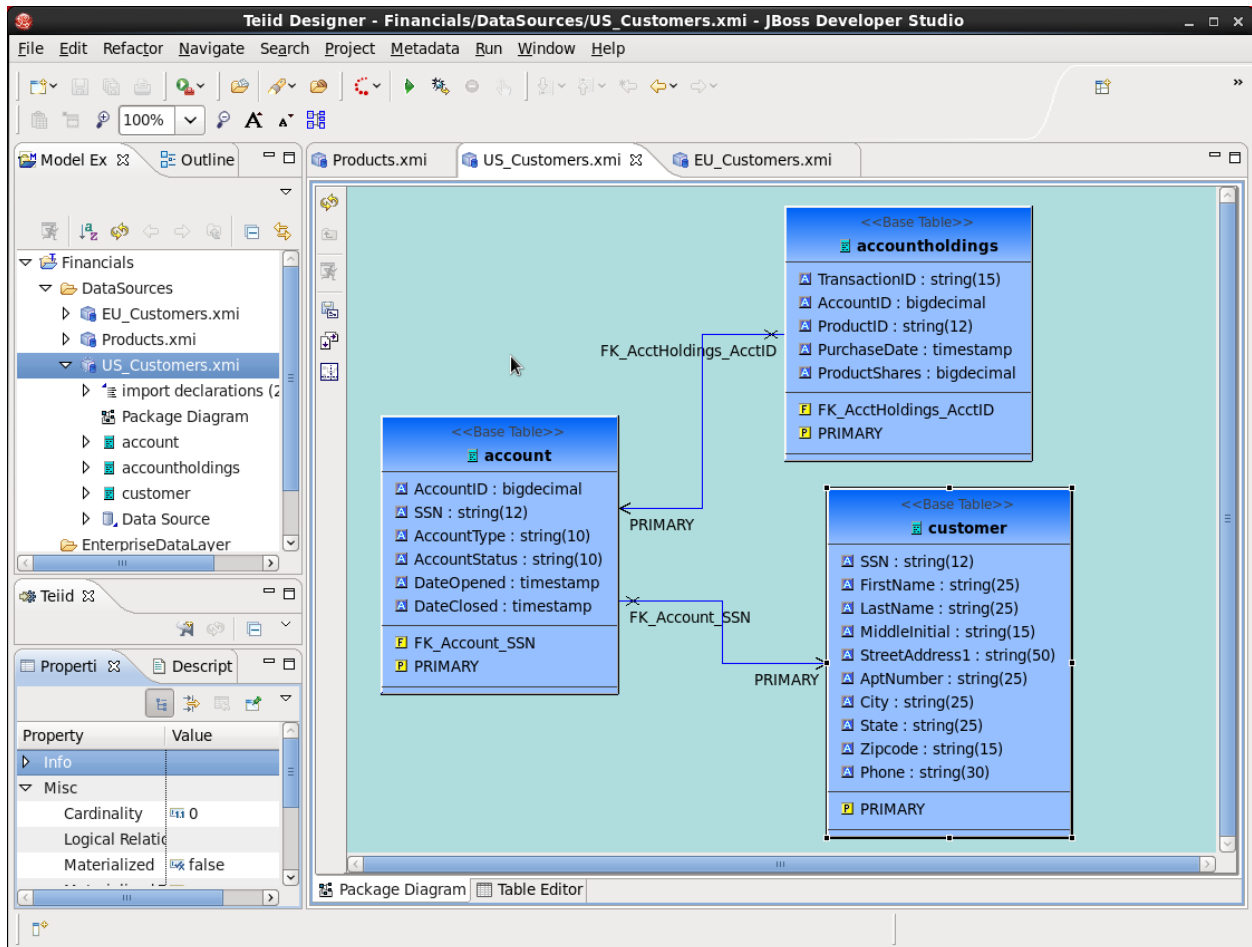


Illustration 59: US_Customer Physical Metadata Model

Congratulations, you have completed Lab 3.

JBoss by Red Hat EDS

Lab Number 4: Create a Virtual Base Layer

Rationale

Beyond the obvious advantages of integrating disparate data sources using an intuitive, easy-to-use technology such as EDS, 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 that 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).

Create a 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. 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:

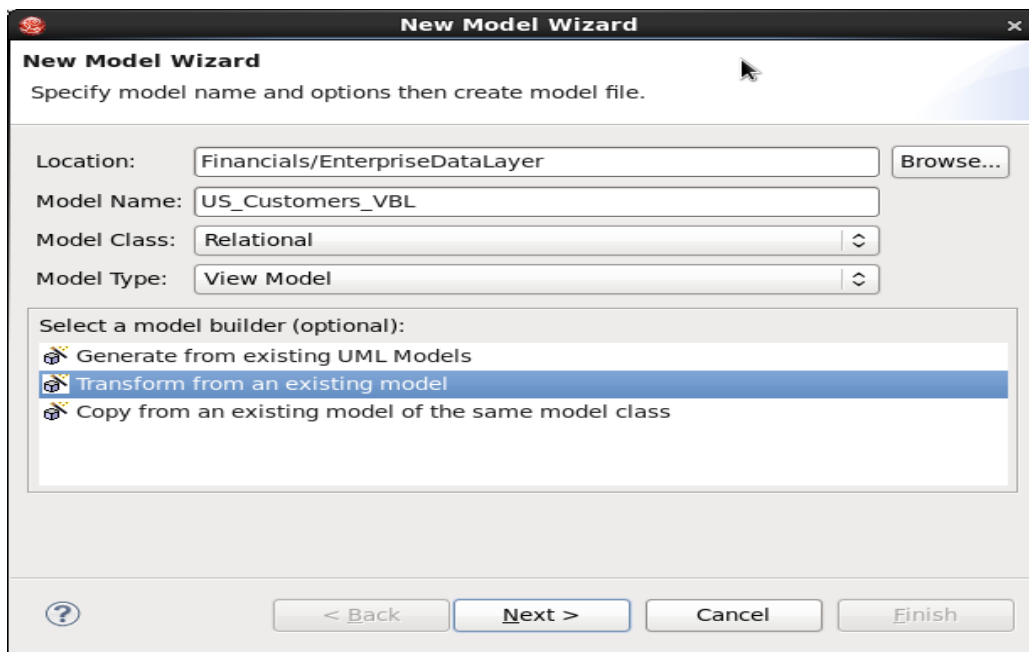


Illustration 60: Creating a Virtual Base Layer Model of US_Customers

The "Transform From An Existing Model" wizard will appear:

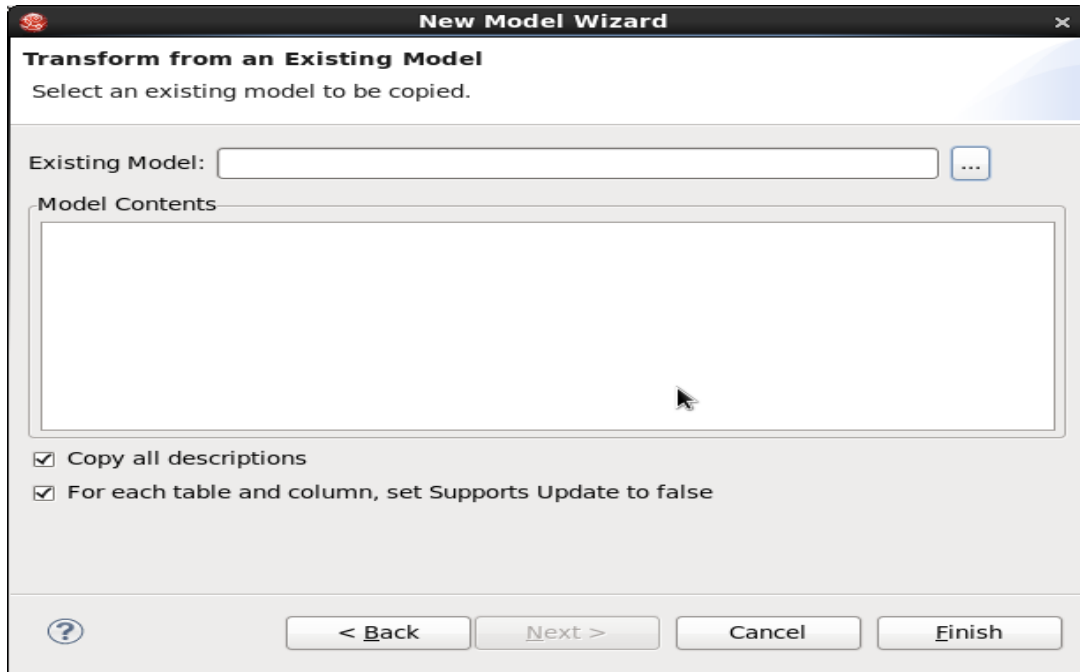


Illustration 61: Transform from Existing Model Wizard

Select the Chooser icon to the right of Existing Model, and open Financials -> DataSources ->US_Customers:

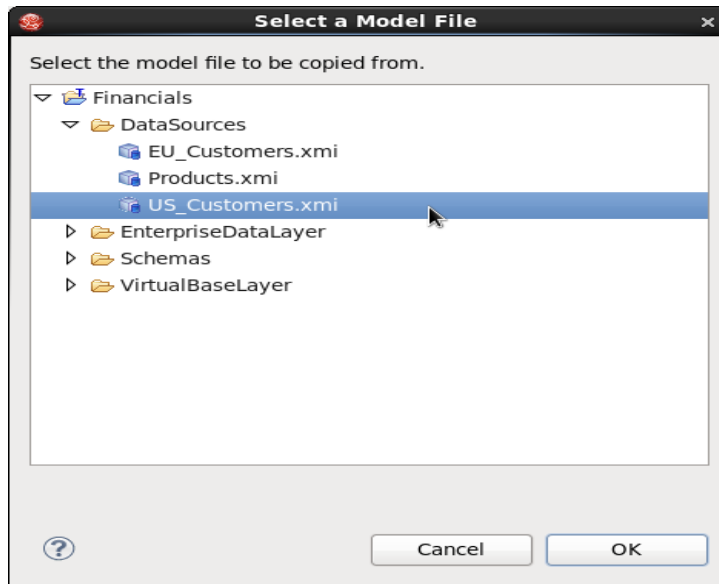


Illustration 62: Transform from Existing Model Wizard

Select OK. You will be returned to the New Model Wizard. We are going to copy all model elements, so just click Finish:

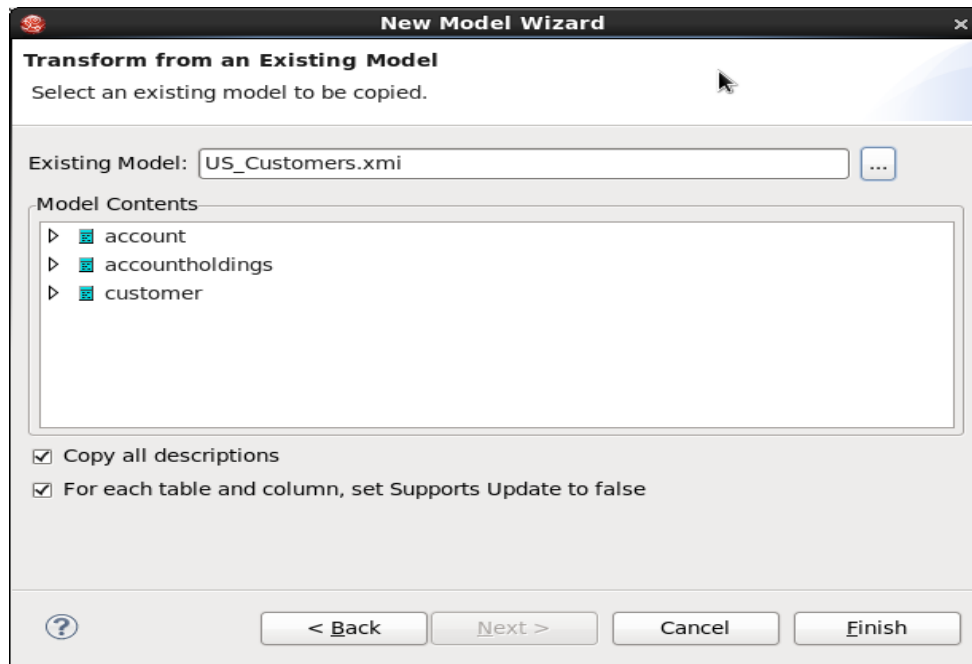
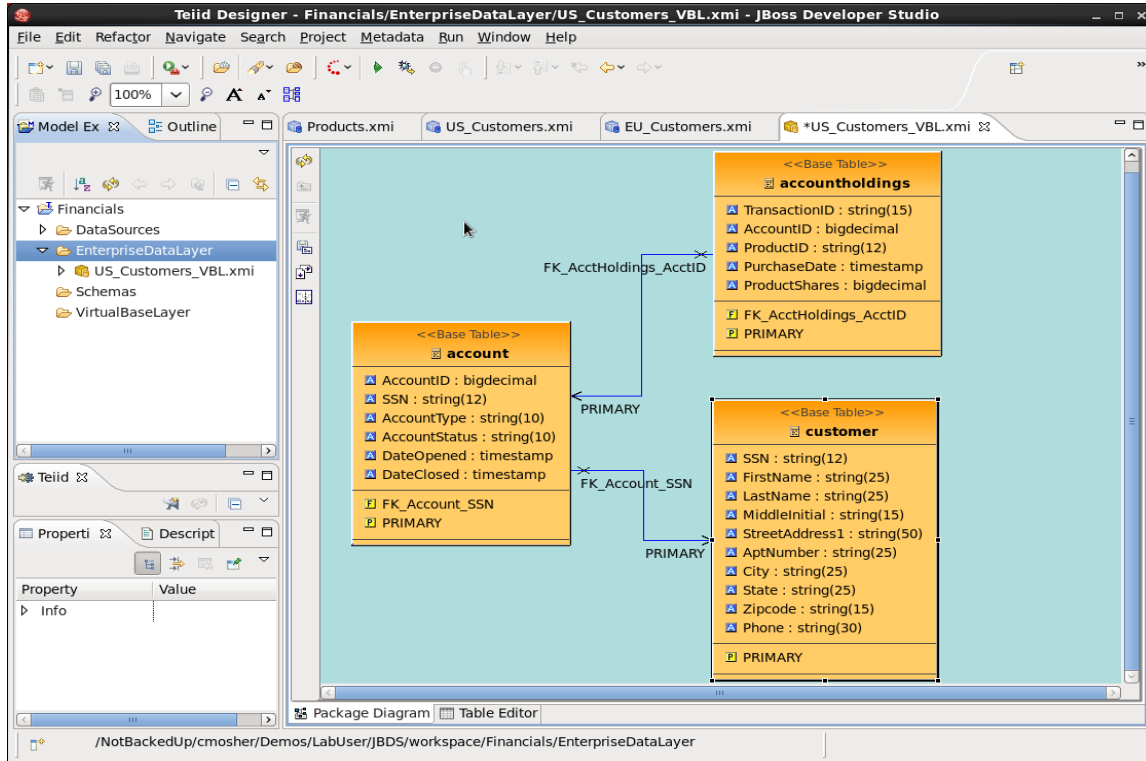


Illustration 63: Creating a VirtualBaseLayer built from US_Customers

The Package Diagram and Model Explorer for the US_Customers_VBL (virtual) model will now open in the workspace. Observe that 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:



Double-click on the **customer** table in the US_Customer_VBL model. This will open the Transformation Editor. Note that 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 EDS is because the Query Engine in the Teiid 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 EDS.

Note that 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 Tranformation Editor further on in the lab.

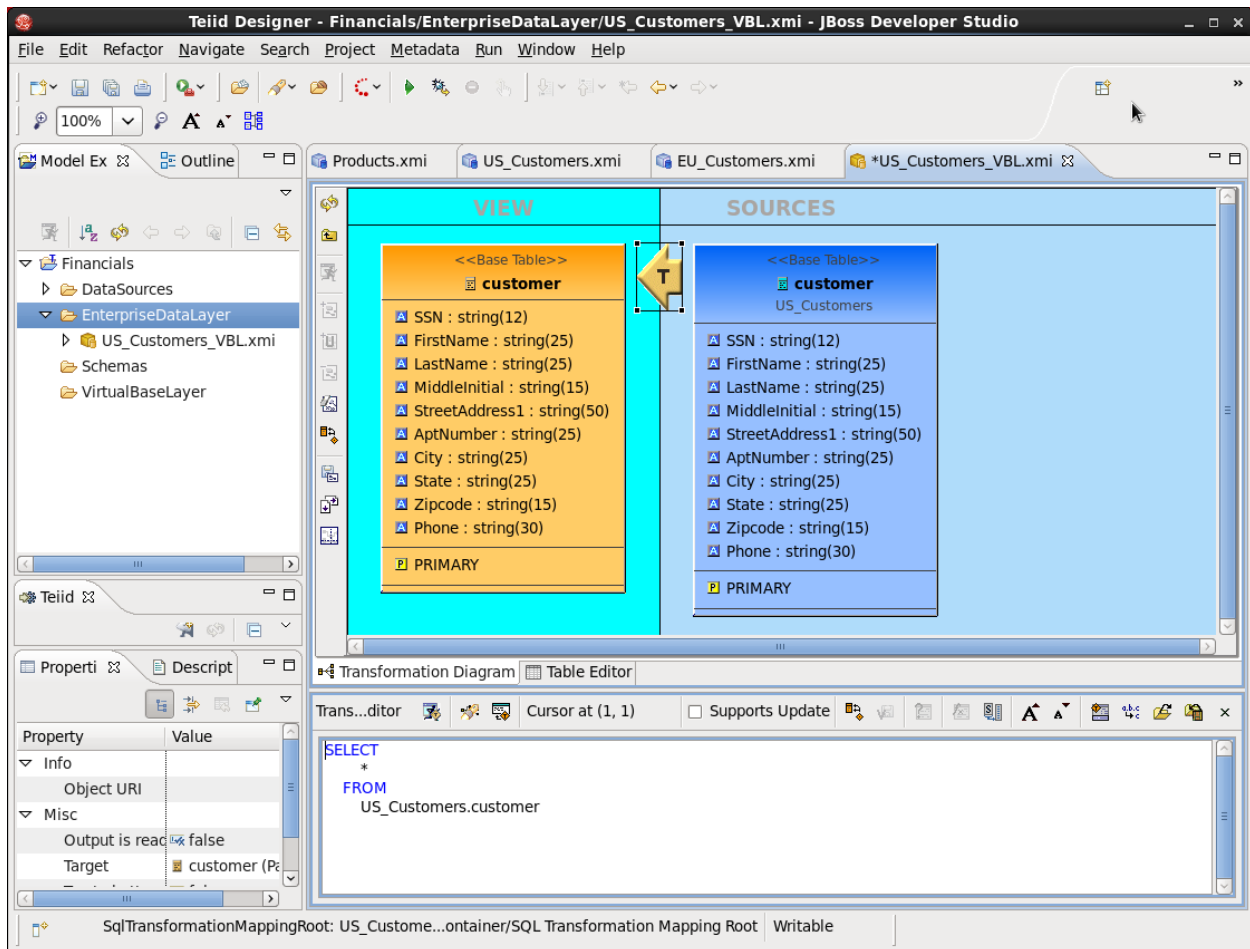


Illustration 65: Transformation Editor

Create EU_Customers_VBL and Products_VBL Models

Create VBL's for the other two physical models you have imported (Products_VBL, EU_Customers_VBL). Be sure the 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 this:

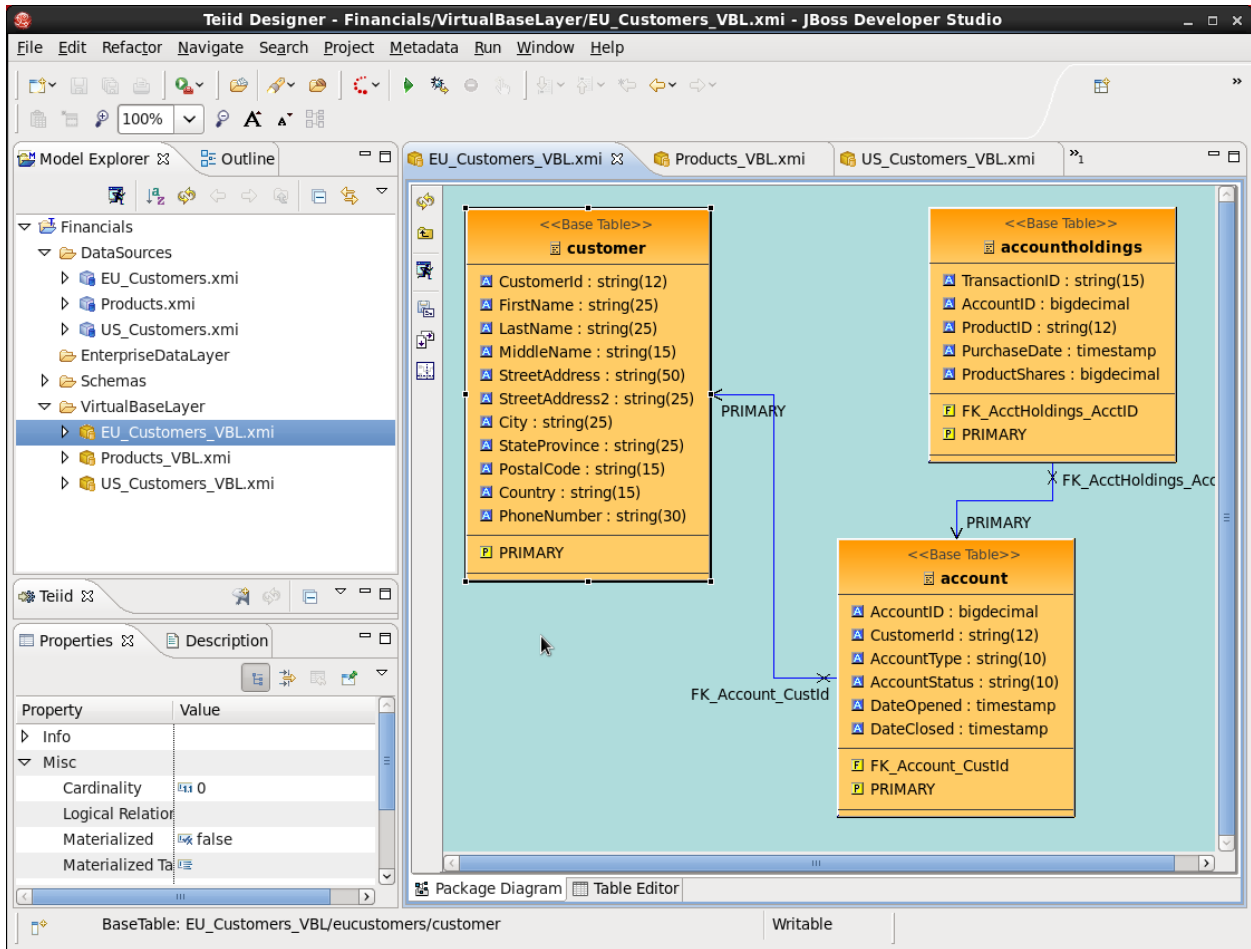


Illustration 66: Completed VBL Layers

Congratulations, you have finished Lab 4.

Lab Number 5: Create an Enterprise Data Layer

Rationale

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_Customer and US_Customer 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 these 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 EDS'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 with custom datatypes within EDS. In fact many of our customers do not 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 EDS 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.

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 the \${USER_HOME}/Downloads/EDS folder. We want to import the schema into the Schemas folder you created in Lab 2. Perform the following steps:

- 1) Right-click on the Schema folder in the Model Explorer and choose Import -> Teiid Designer -> XML Schemas:

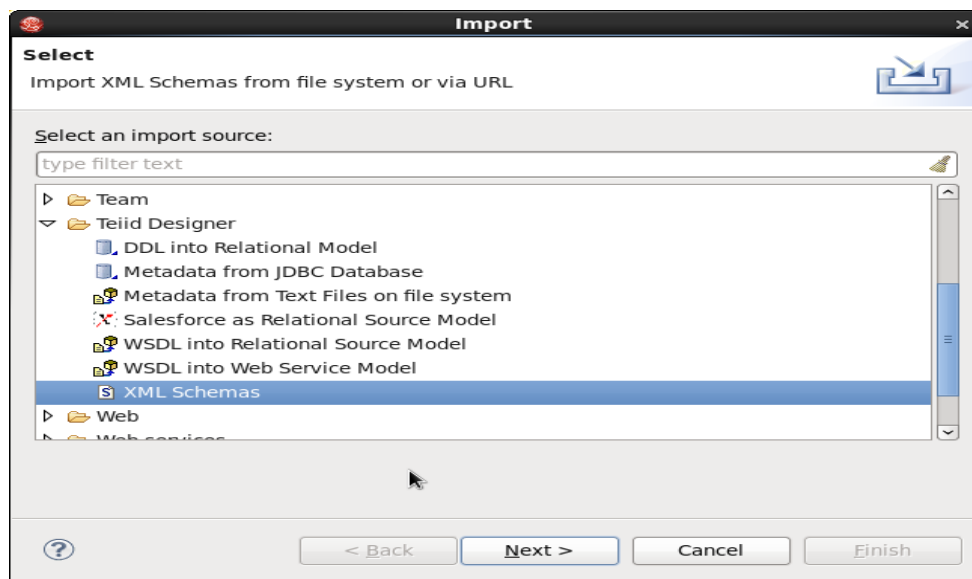


Illustration 67: Import XML Schema

JOSS by Ken Hat EDS

2) Choose to import the schema from the local file system

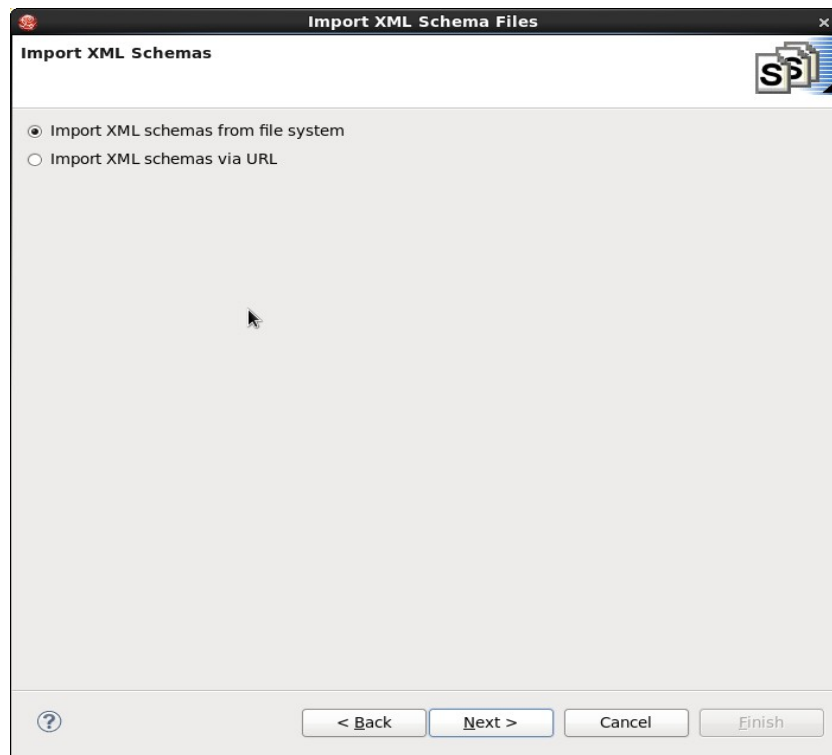


Illustration 68: Import Schema From File System

3) Browse to the `${USER_HOME}/Downloads/EDS` 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:

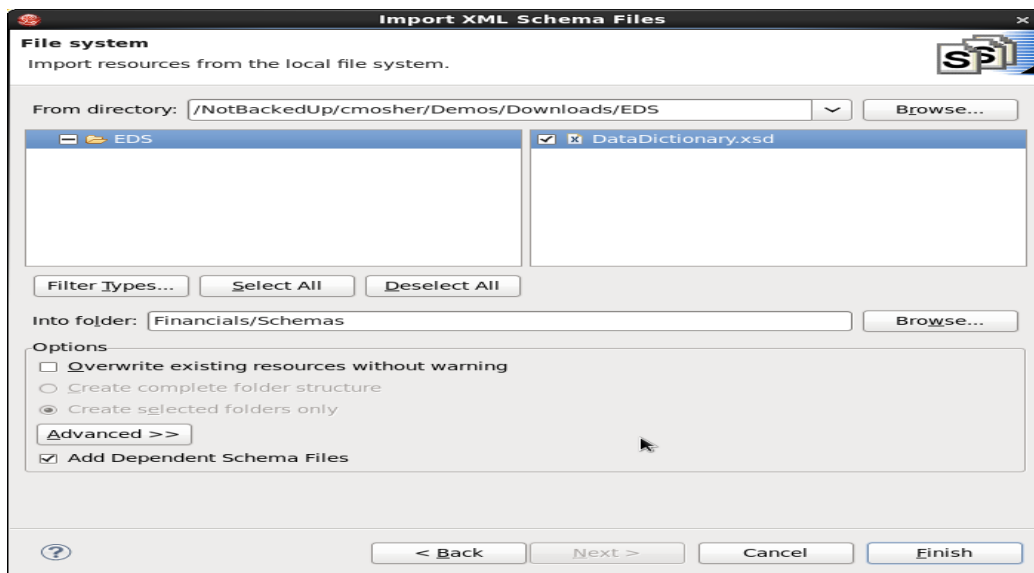


Illustration 69: Choose DataDictionary.xsd in the Downloads/EDS directory

4) You may get a warning that Financials/Schemas already exists. Choose Yes to overwrite:

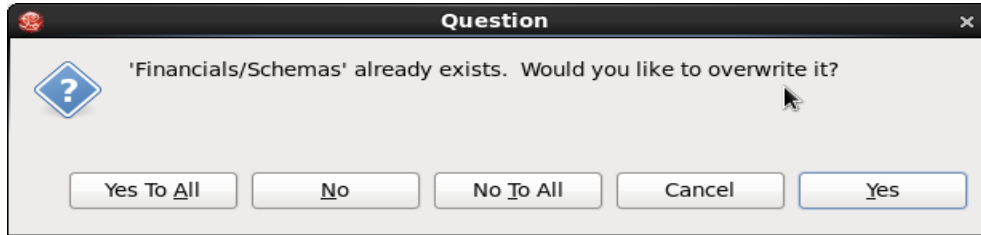


Illustration 70: Choose Yes to Overwrite

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 its 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.

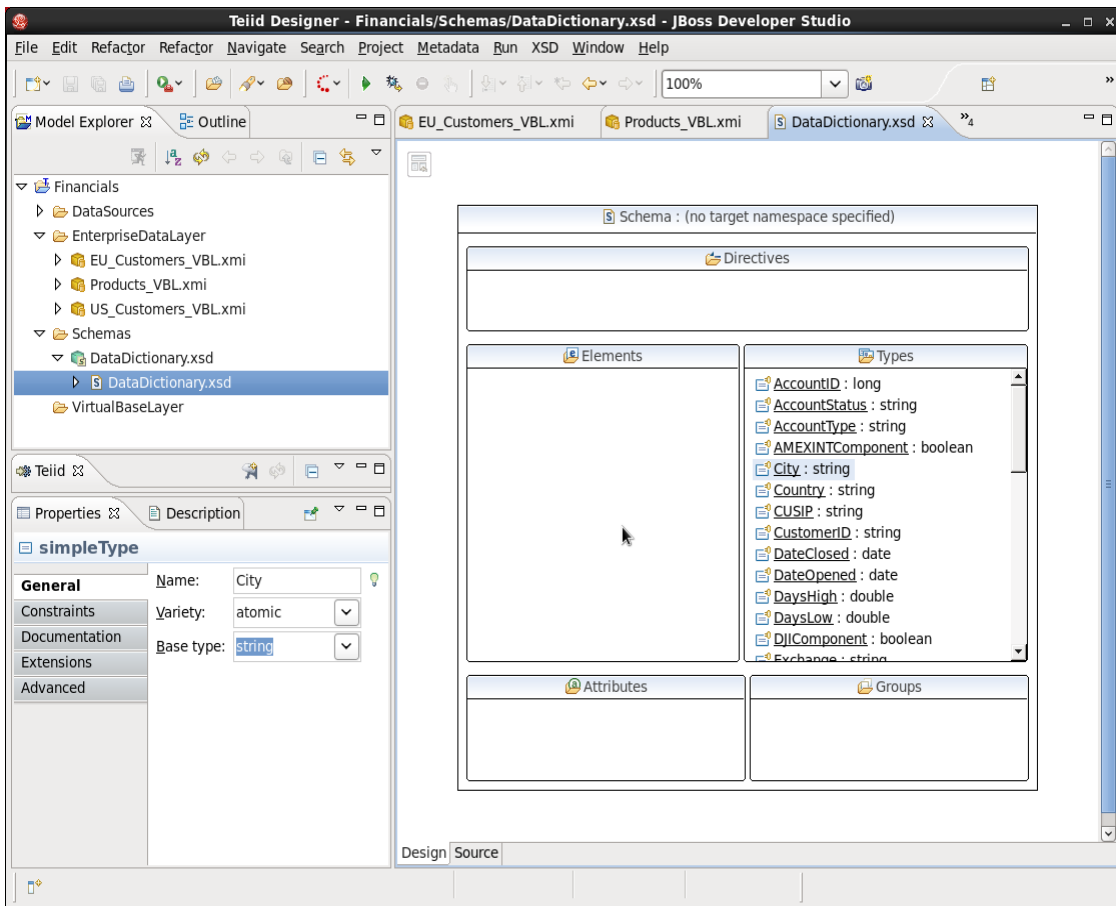


Illustration 71: XML Schema Browser

Create the EU_Customers 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 and select New -> Teiid Metadata Model to bring up the New Model Wizard and make the following selections:

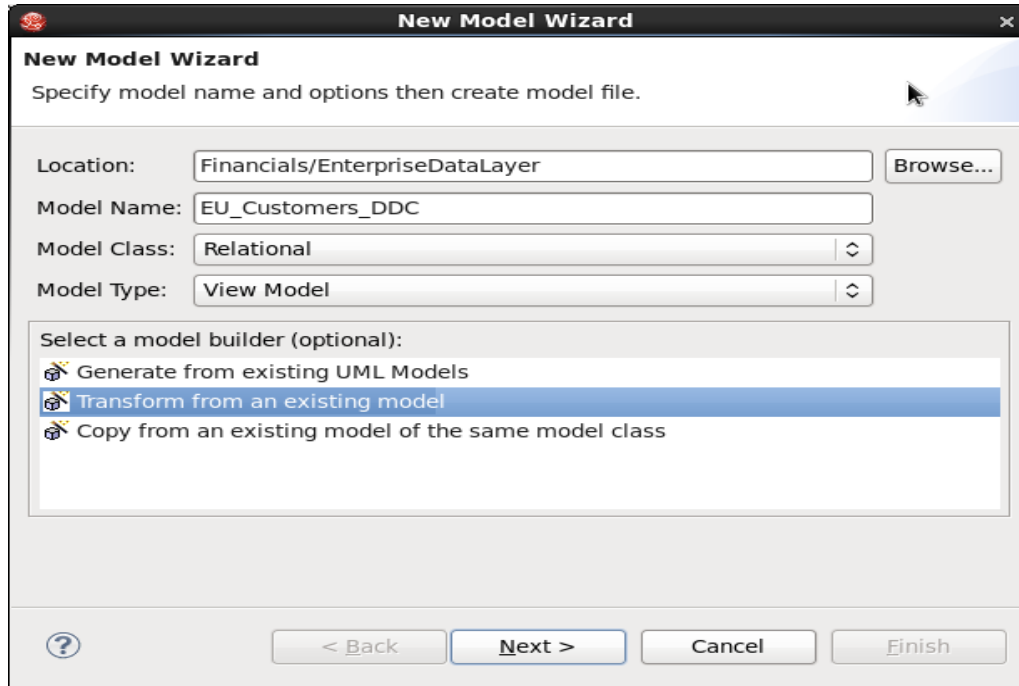


Illustration 72: XML Schema Browser

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

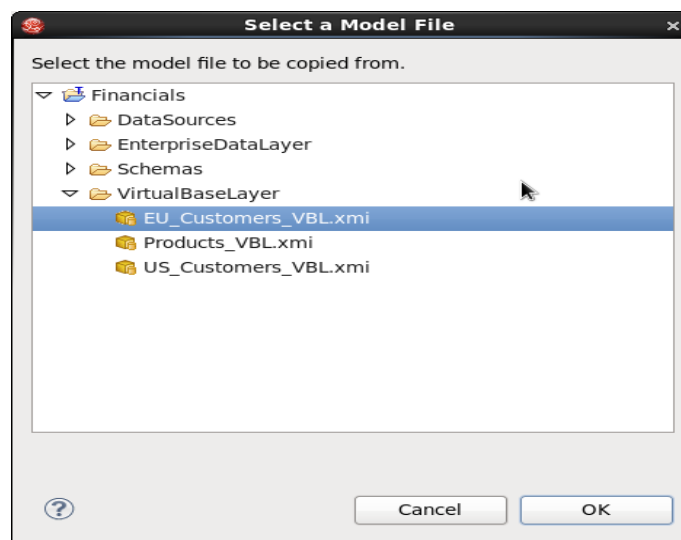


Illustration 73: XML Schema Browser

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 APAC_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:

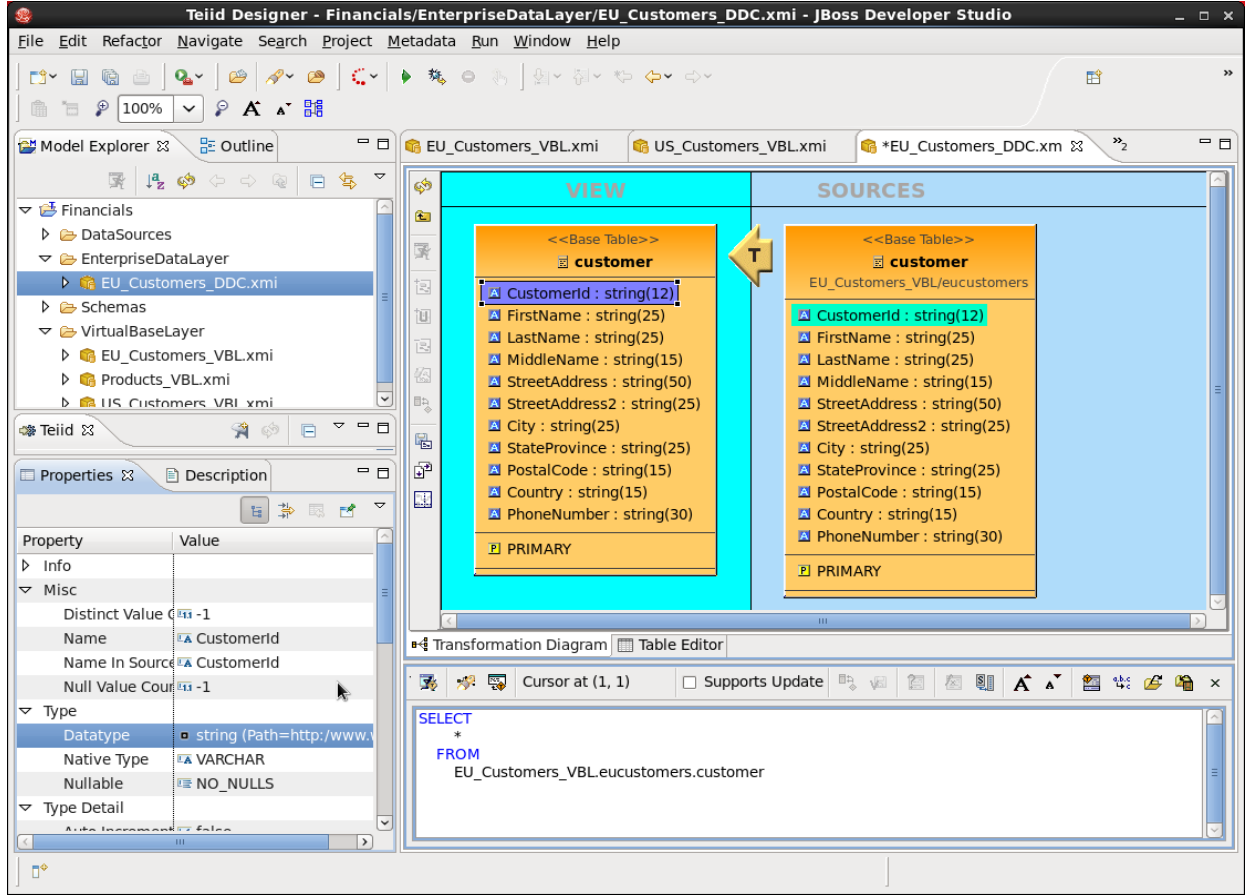


Illustration 74: Transformation Editor for EU_Customers.customer

Click on the Value field to the right of Datatype in the Properties Panel to bring up the Select a Datatype wizard. Note that the wizard has both built-in datatypes and User-Defined Types. 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:

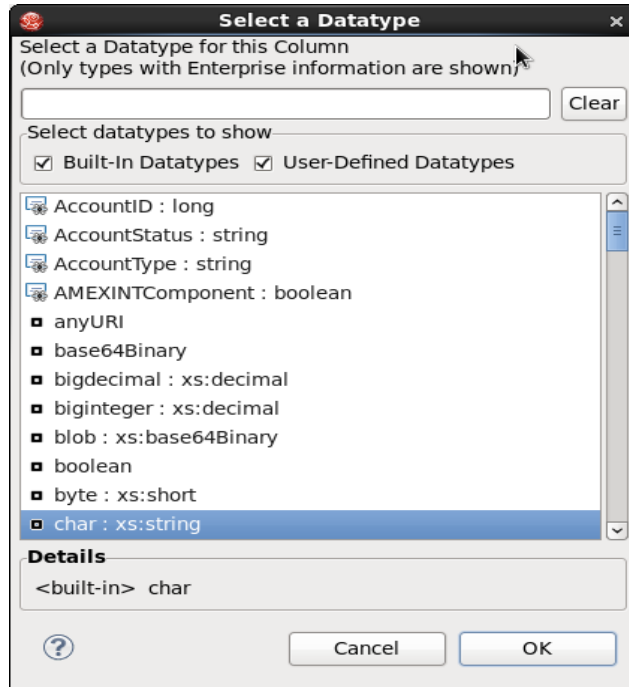


Illustration 75: Datatype Selection Wizard

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 narrow the selection of types displayed.

Go through this exercise with every column in the Customer table, choosing the appropriate datatype for each. Save your work. It will look like this:

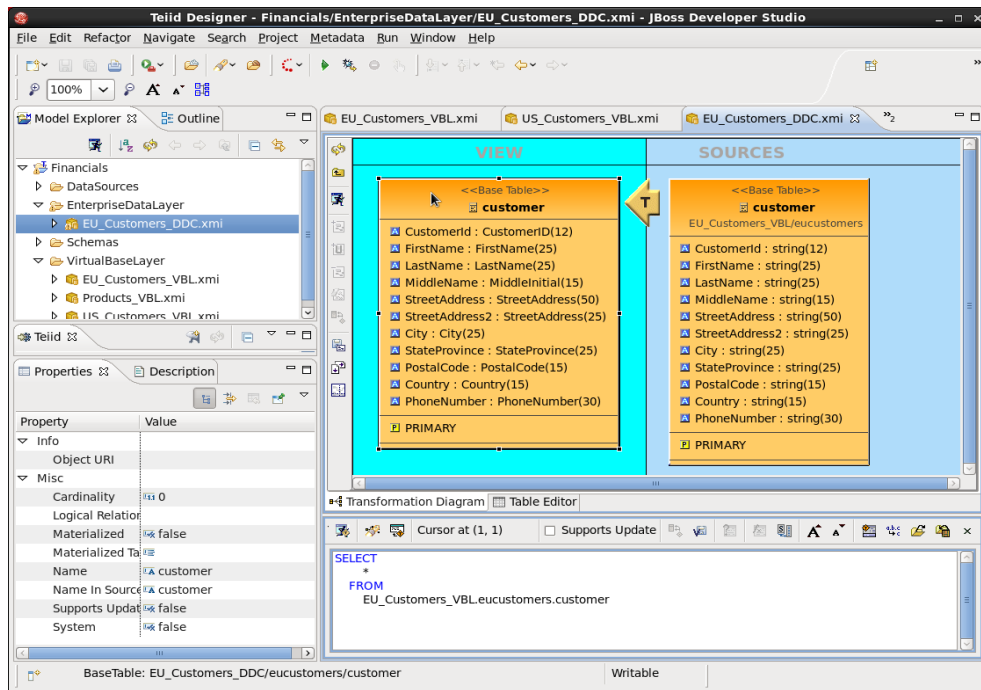


Illustration 76: EU_Customers_DDC.Customer Mapped to Enterprise Datatypes

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:

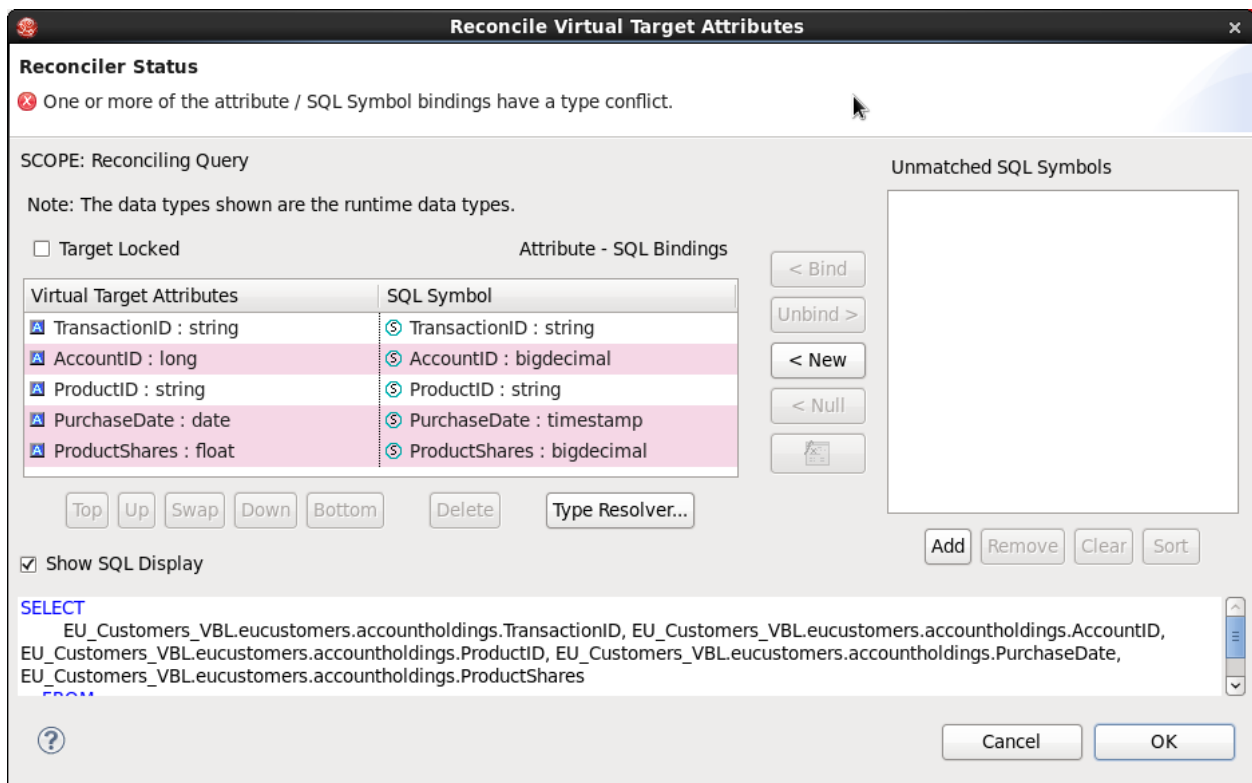


Illustration 77: Reconciler Opened for EU_Customer_DDC.AccountHoldings

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

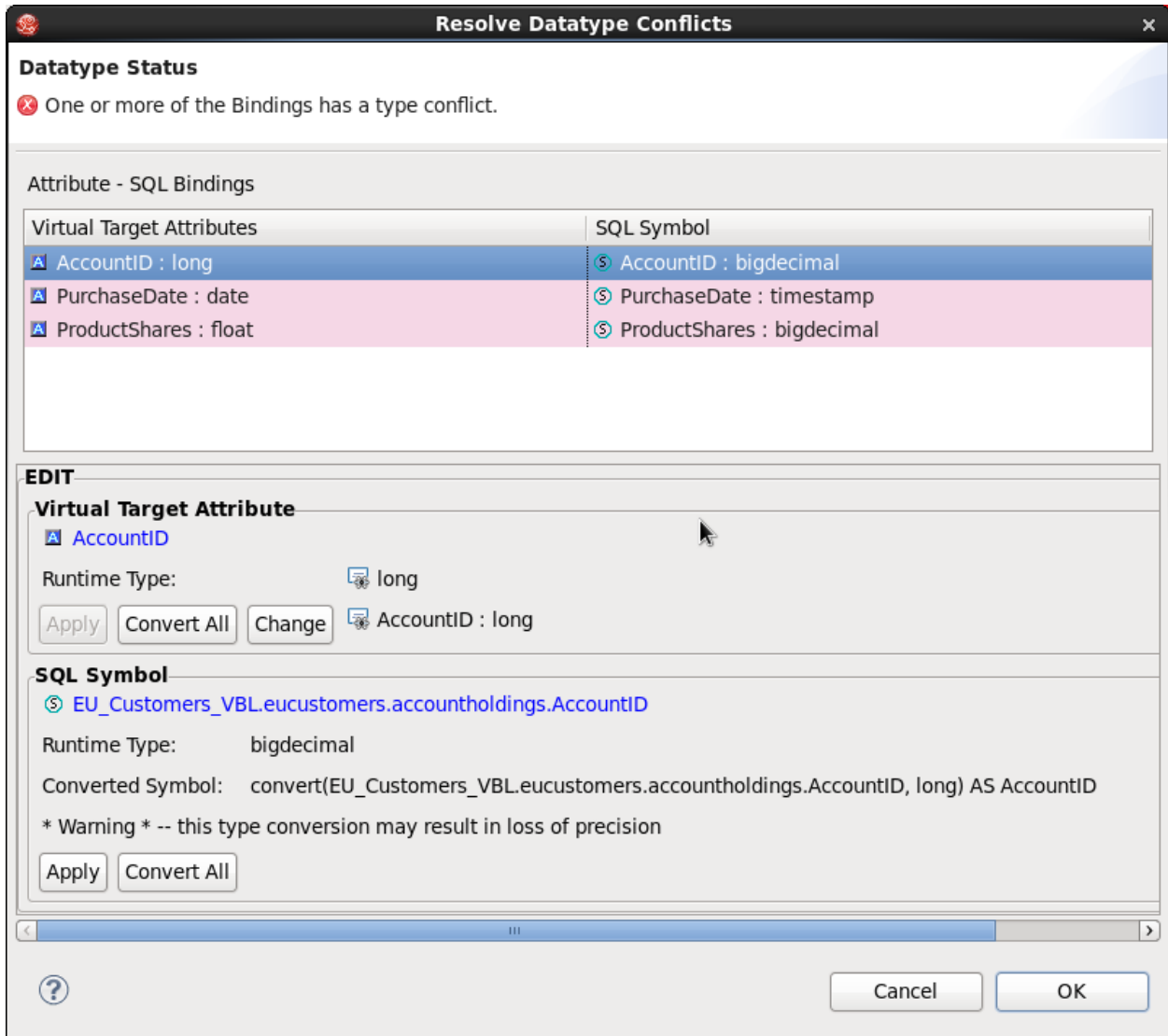


Illustration 78: Type Resolver

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 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 this:

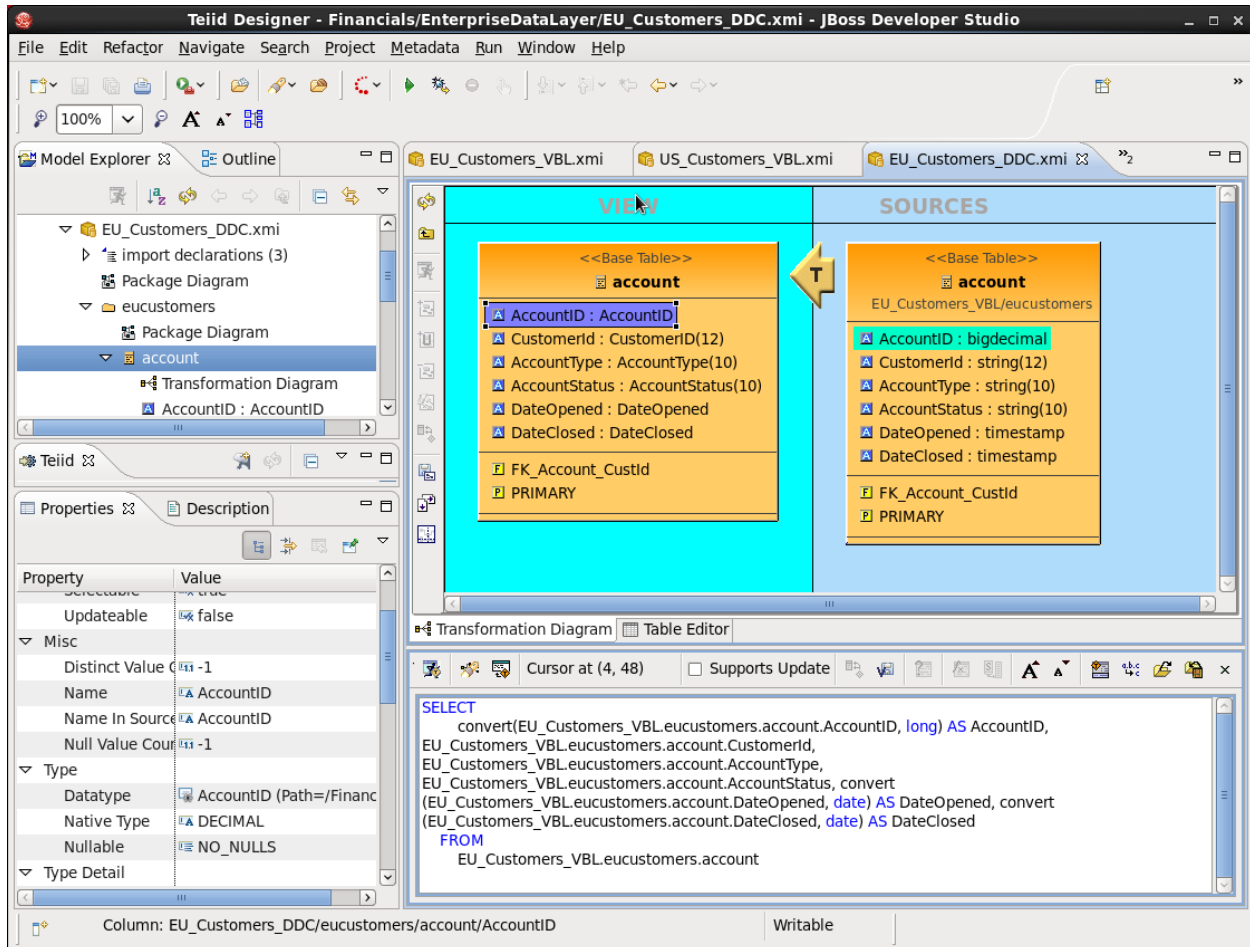


Illustration 79: EU_Customers_DDC.Account, with Reconciled Datatypes

Create the US_Customers 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:

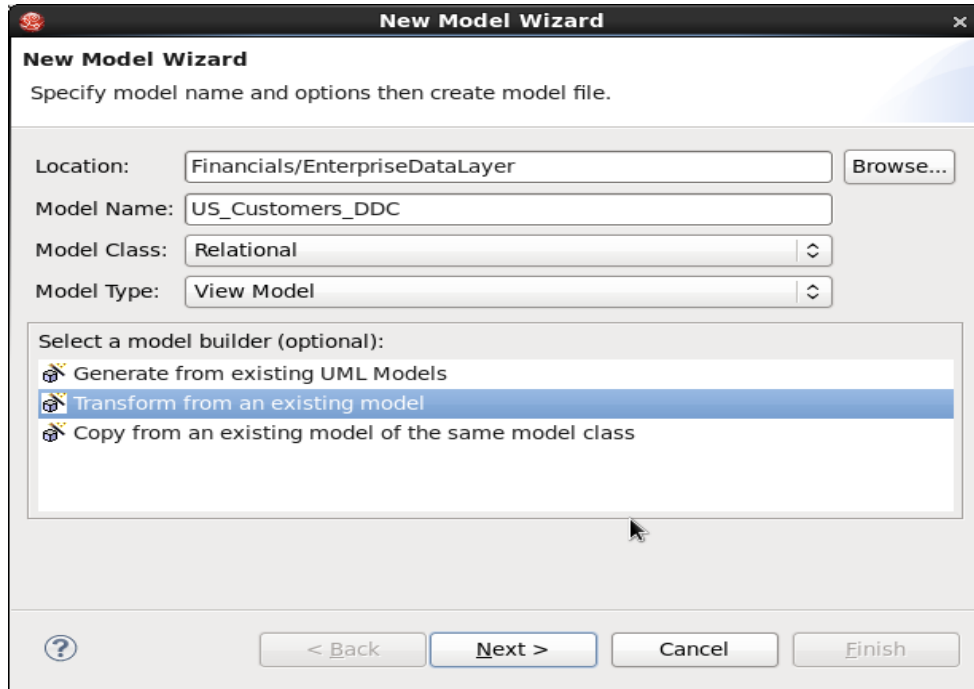


Illustration 80: Creating US_Customers_DDC Model, Based on EU_DDC Model

In the Select a Model wizard choose the EU_Customers_DDC model in the EnterpriseDataLayer folder and click OK, followed by Finish:

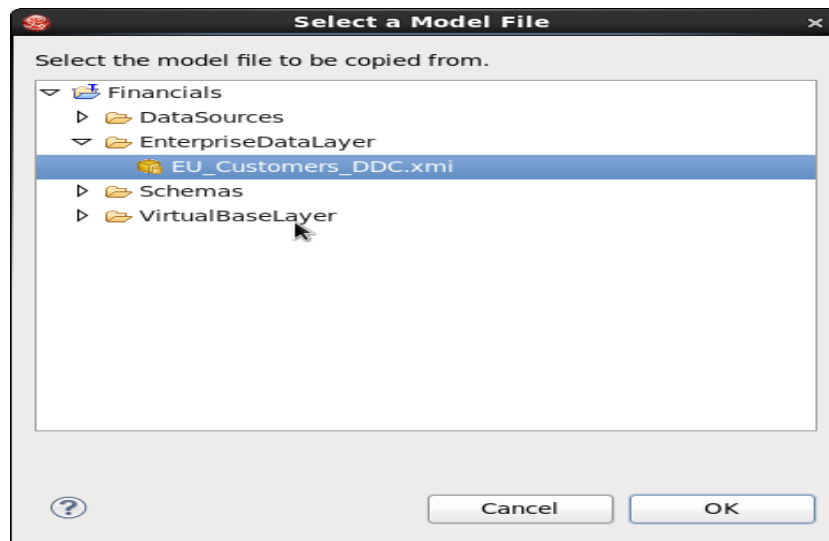


Illustration 81: US_Customers_DDC Model based on EU_Customers_DDC Model

Open up 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)". The following window will pop up:

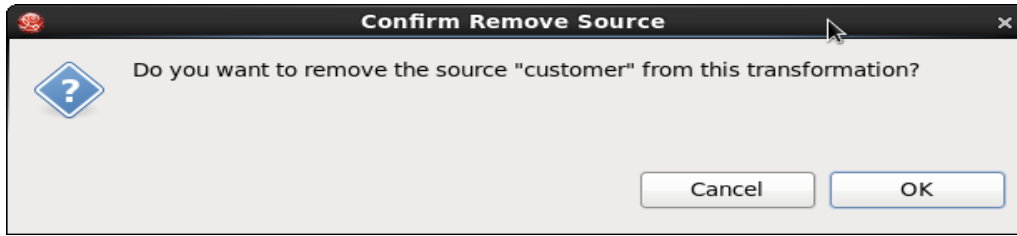


Illustration 82: Confirm Source Removal

Click OK.

This is what you will see:

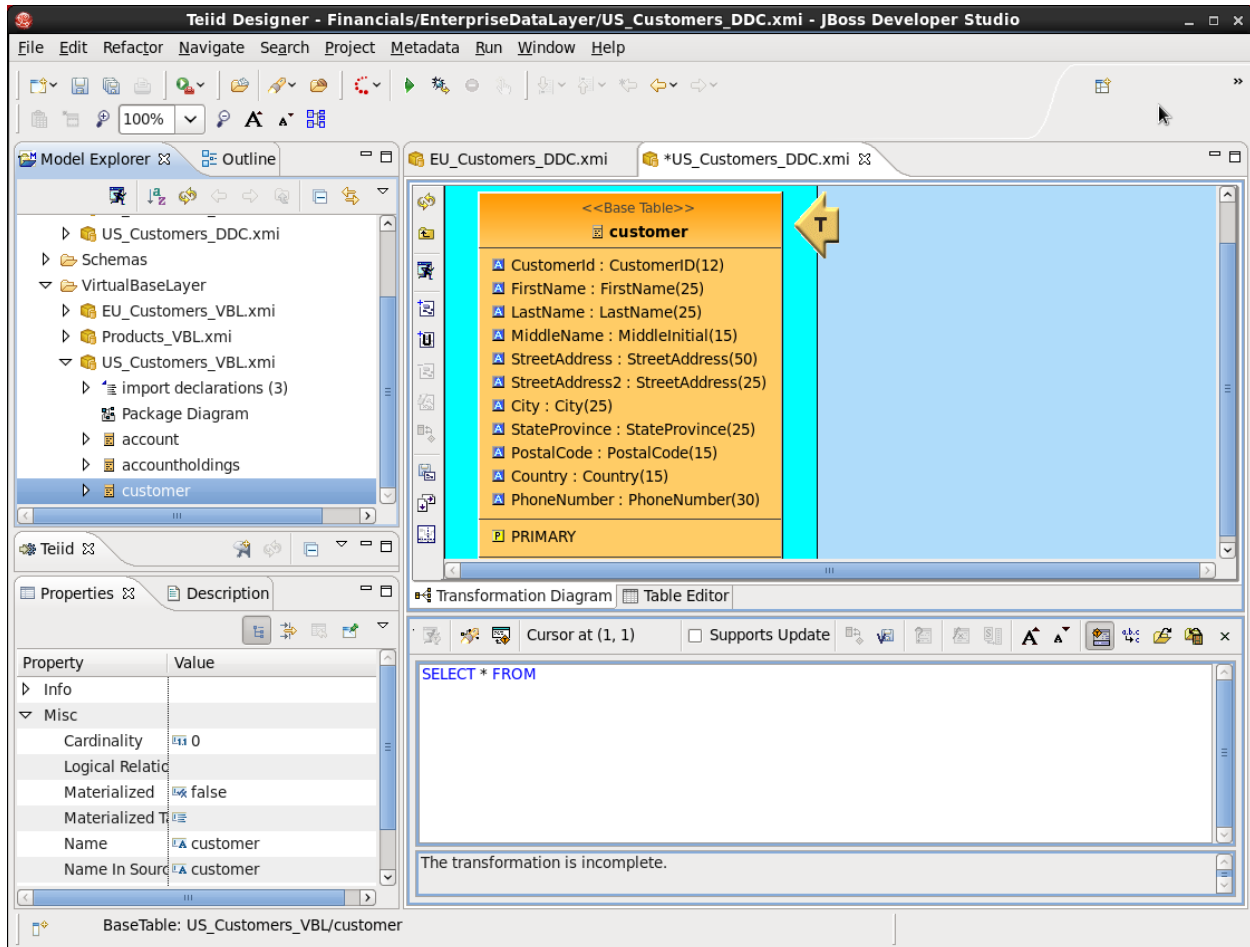


Illustration 83: Empty Transformation (but the View Table now has the correct schema)

Select the VirtualBaseLayer->US_Customers_VBL->customer table (it is highlighted on the left above) and drag it onto the Source window of the transformation to the right.

Click the Save/Validate SQL button (the little disk icon with a check-mark next to it, just to the right of the Reconciler button). There are many errors. Click on the Reconciler to bring up the wizard. As you can see from the screenshot below we have a lot of work to do:

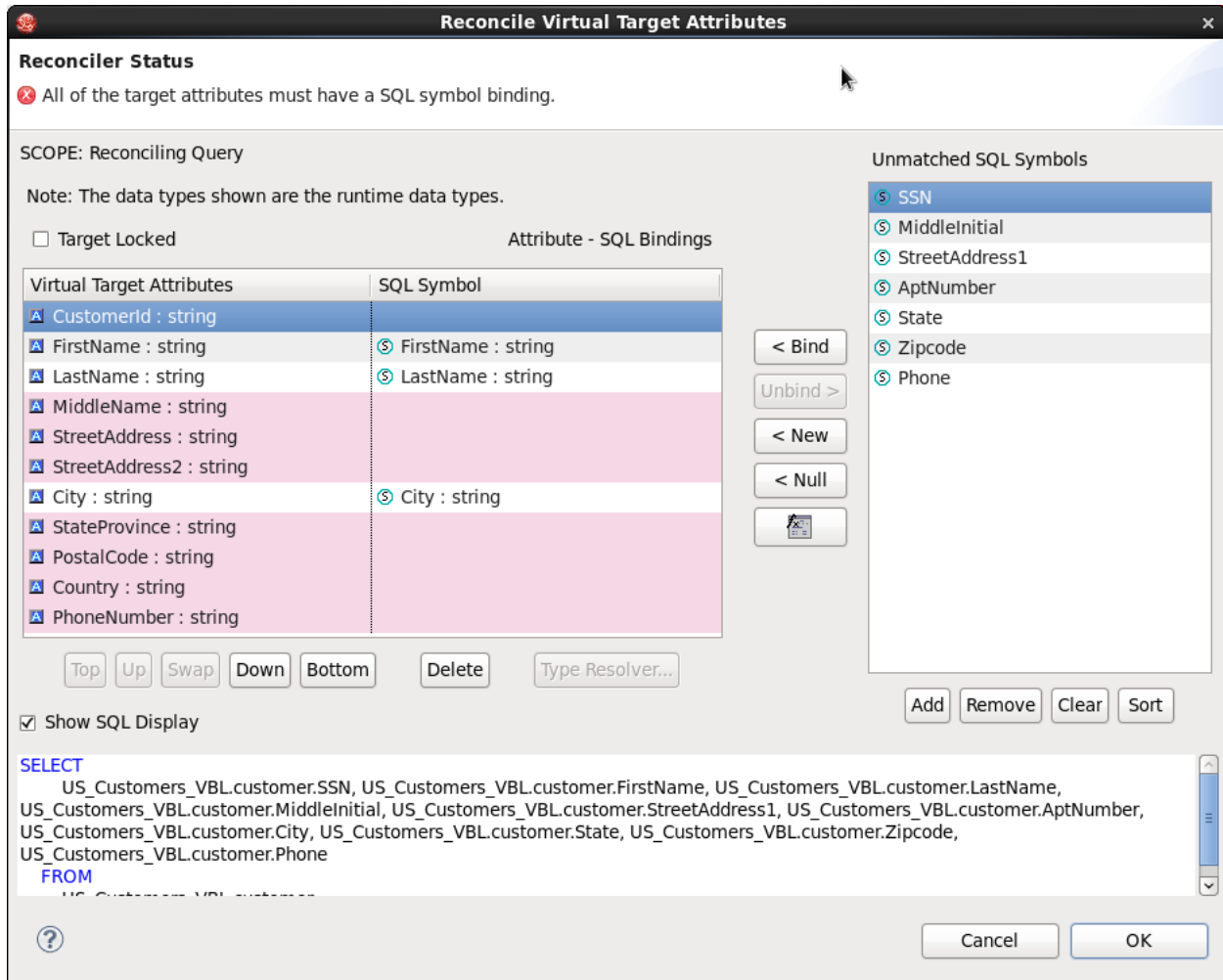


Illustration 84: Reconciler on US_Customers_DDC.customer table transformation

There are two things we need to do to fix this transformation:

- 1) Assign (Bind) variables that did 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
 - Apt 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 with Step 1 the Reconciler will look like this:

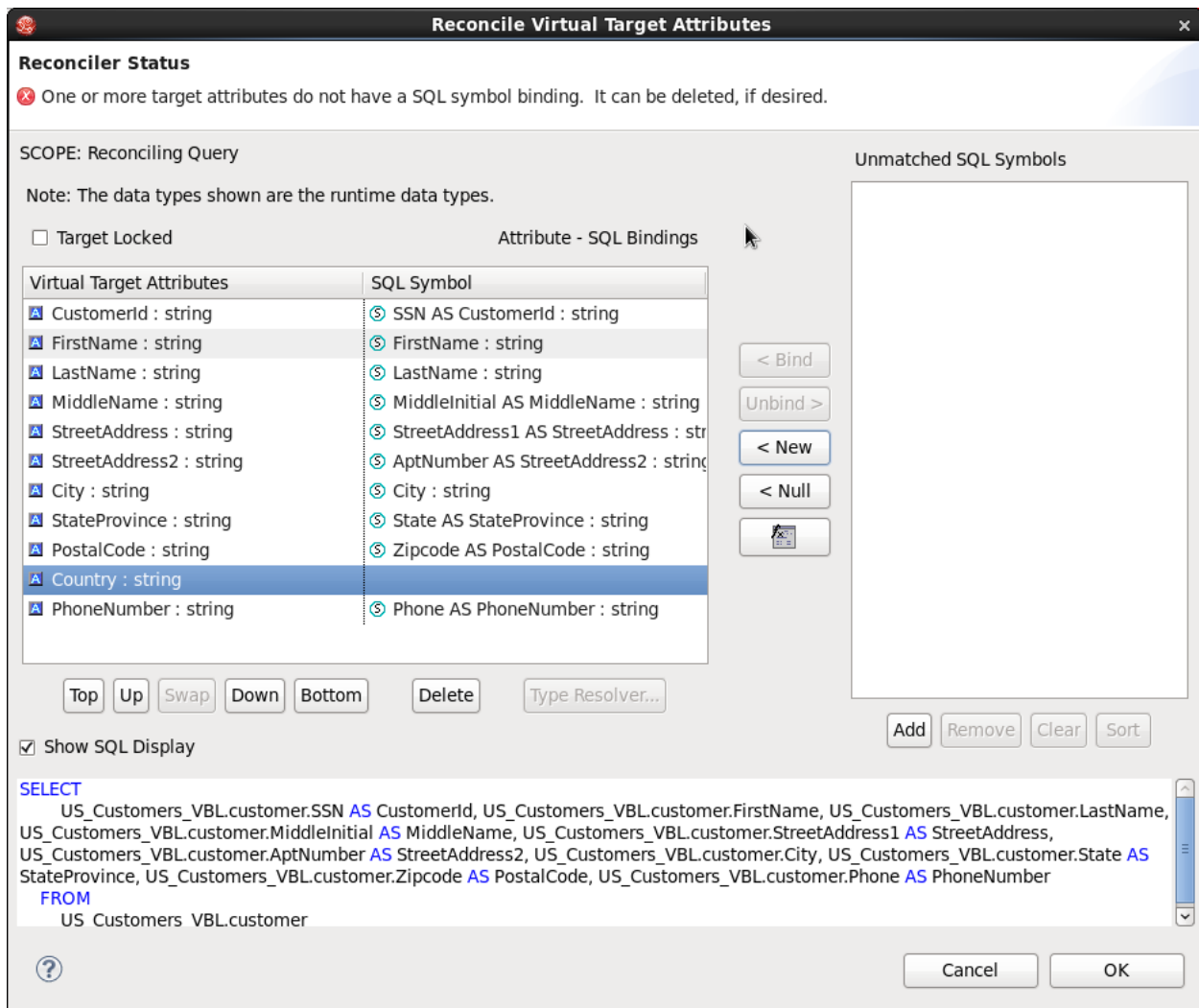


Illustration 85: *US_Customers_DDC.customer* table mostly Reconciled

We still have one more step:

- 2) 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 (right under the "< Null" button in the middle of the wizard). Since all that is needed is a simple (static) assignment The Expression Builder comes up with the following screen:

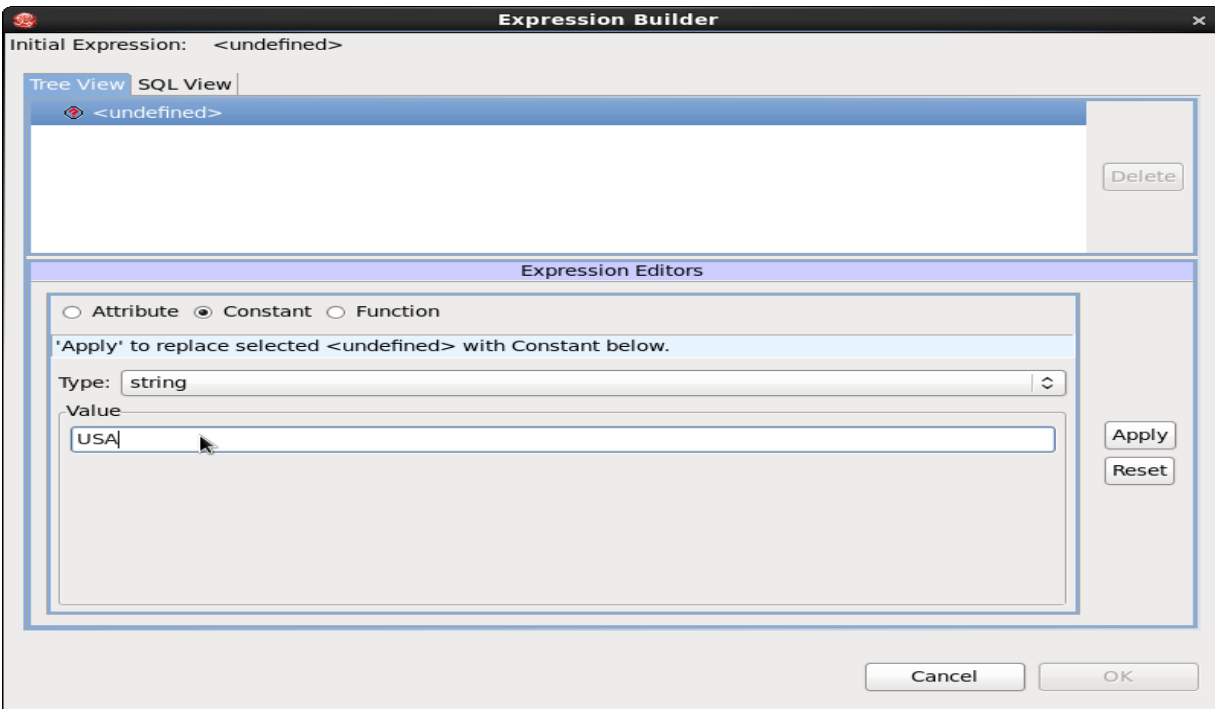


Illustration 86: Expression Builder

All we need to do is type "USA" into the Value field, click Apply, then OK. However, while you are in the 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 EDS. When you are finished though be sure to set it back to Constant and complete the instructions as 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_DDC source, replace it with the appropriate US_Customers_VBL source, and perform any necessary reconciliations. You can always validate your work using the Preview button.

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 finished Lab 5.

Lab Number 6: Data Federation and VDB Deployment

Data Federation

Now that we have created our VirtualBaseLayer (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 EnterpriseDataLayer that captures those mappings, we are ready to integrate some different data sources. And since we have done that ground work it could not be simpler.

Create a new folder under the Financials project 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 as follows:

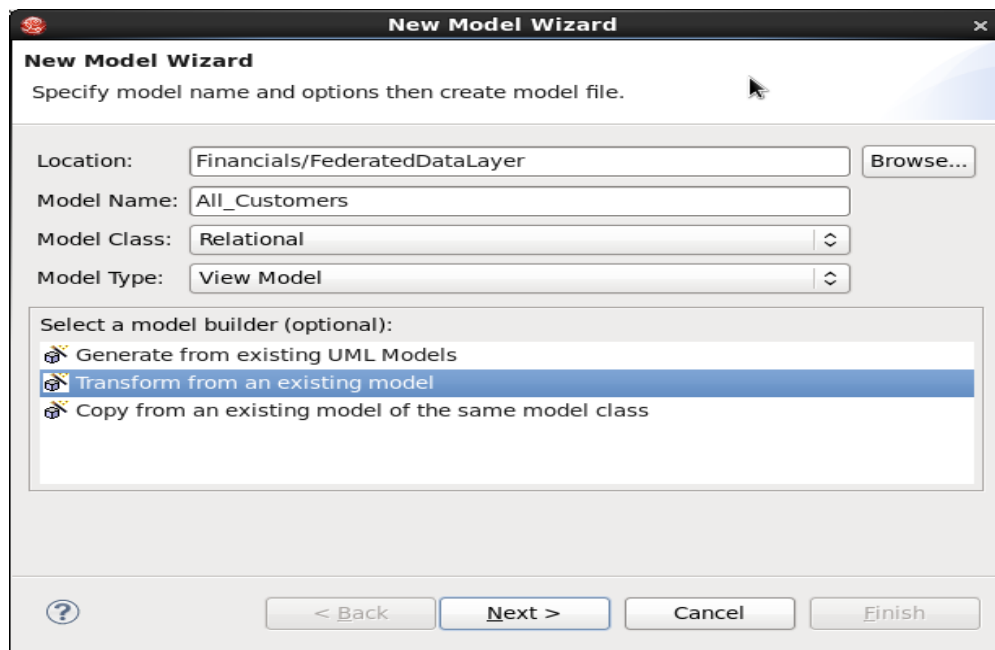


Illustration 87: Create an All_Customers Model

And:

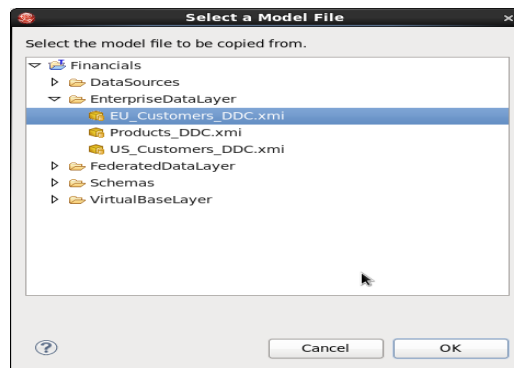


Illustration 88: Source it from EU_Customers_DDC

Now open up the Transformation Editor for All_Customers.customer. In the Model Explorer on the left, open up EnterpriseDataLayer -> US_Customers_DDC and highlight the customer table, as shown below:

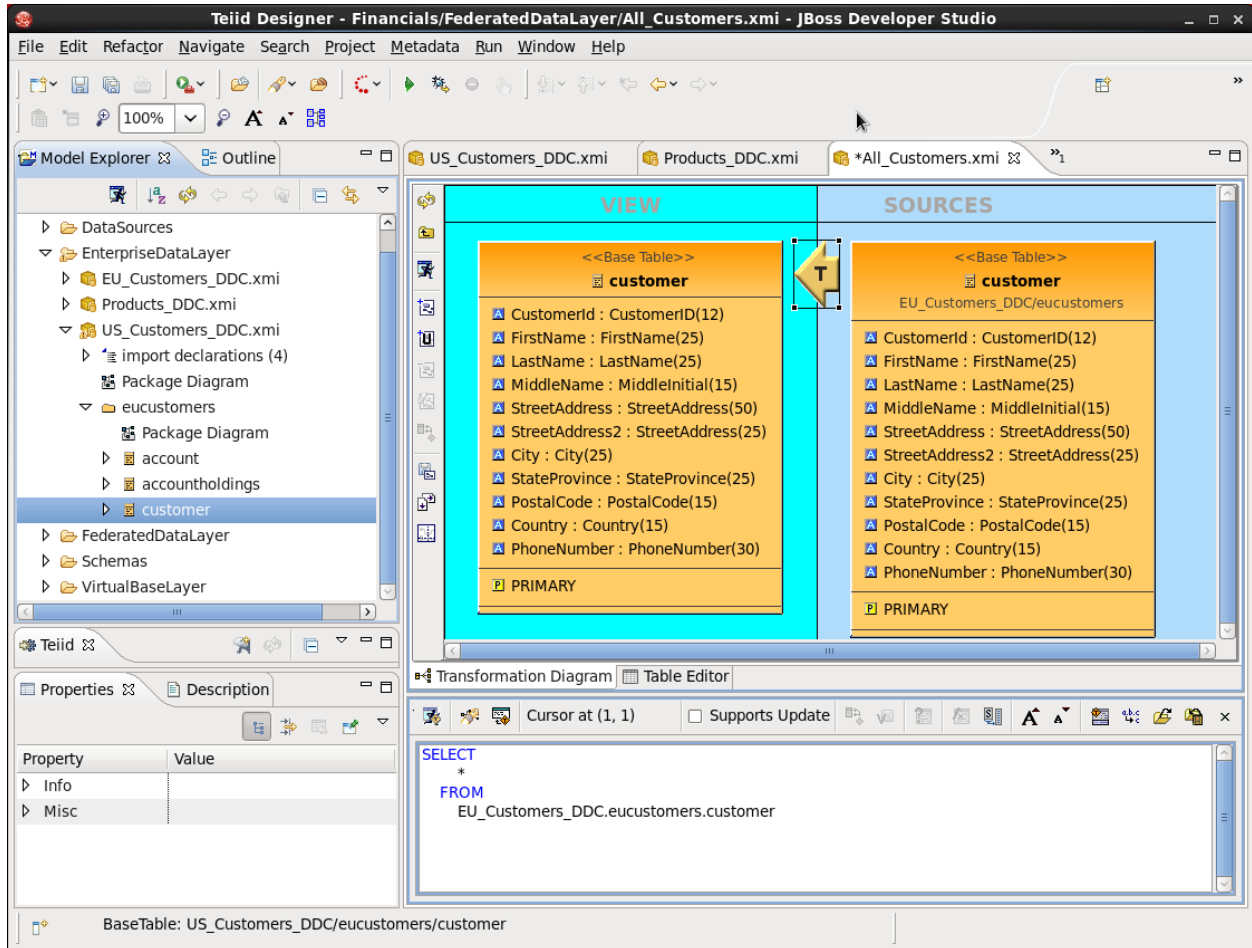


Illustration 89: Highlight EnterpriseDataLayer->US_Customers_DDC->account

Note that several new icons have been activated in the middle tool bar. Click on the second one under the Preview (running man) button. If you hover your mouse over the label is "Add Union Transformation Source(s)". (You may have to press the button twice.) That's it! Save your changes a test with Preview. Create unions of the other two tables.

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 EDS is capable of as sophisticated a set of modeling and transformation that you care to throw at it. There has not been time in the scope of this lab to cover many other exciting features. (For example, right-click on one the All_Customers tables, and choose Modeling -> Create Web Service.)

VDB Deployment

We have been doing all of our querying directly through the Preview interface of 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 .ear file; once created it can simply be dropped into the deploy/ directory of a running Server (with EDS installed) and the Teiid Server process will hot-deploy the data services modeled within it. The process is very simple.

Create the VDB Metadata Model

Right-click on the Financials project (top layer folder in the Model Explorer) and select New -> Teiid VDB. This will open the New VDB wizard. Enter a name, "Financials", and click Finish:

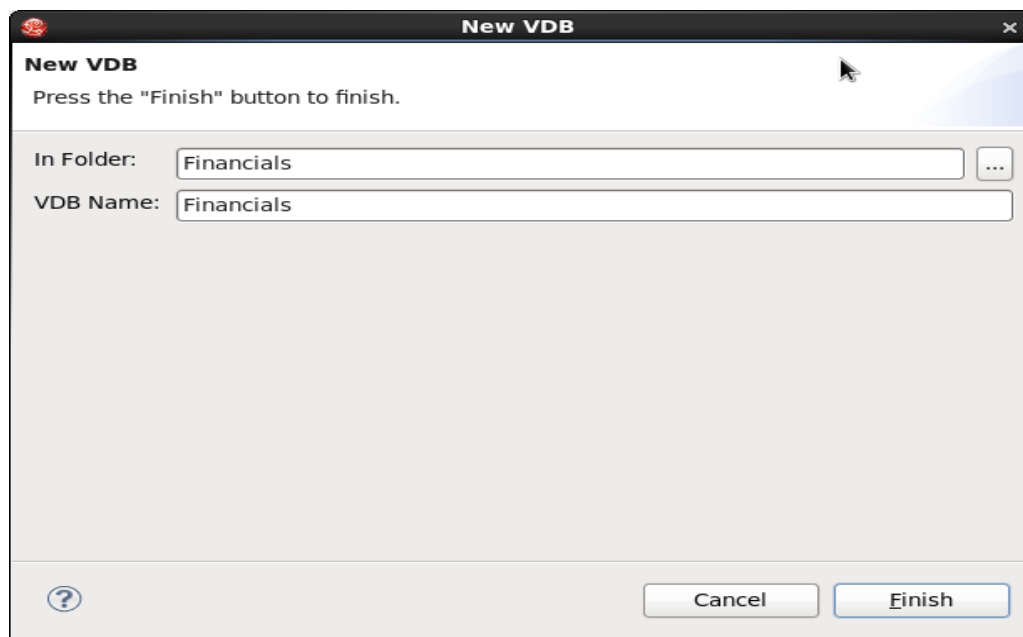


Illustration 90: New VDB Wizard

This will open the VDB viewer in Designer. Click on the Add Model button to add data service definitions to the VDB:

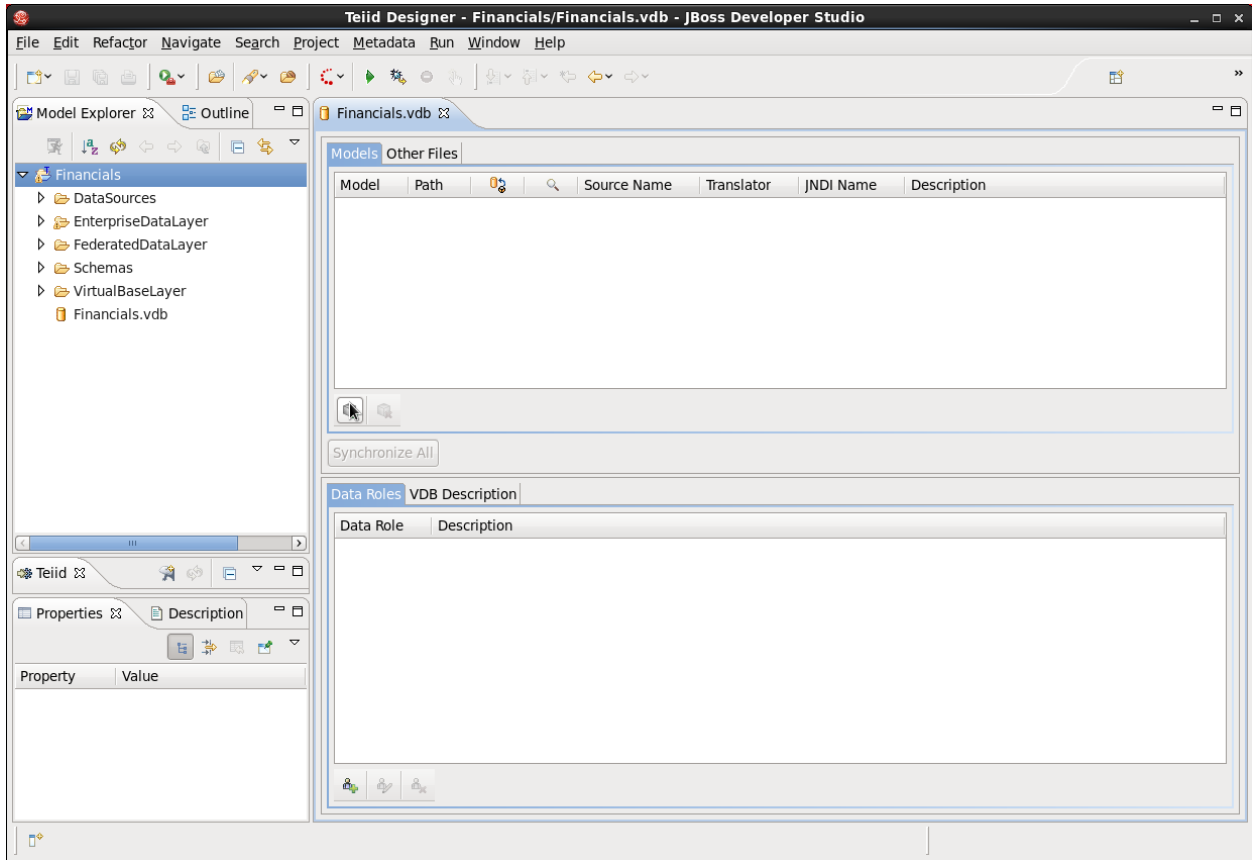


Illustration 91: VDB View. The Add Model button is right above "Synchronize All" (greyed out)

Select the top-level (most granular) data service model in the "Add File(s) to VDB" Chooser:

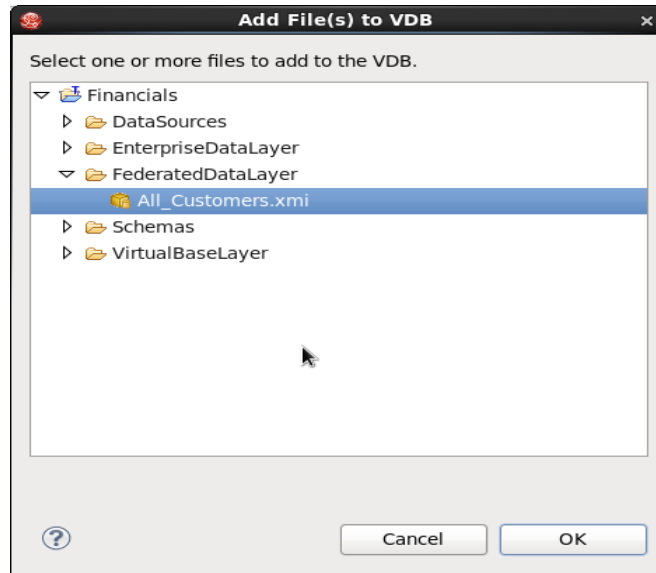


Illustration 92: Add the All_Customers.xml Model, then click OK.

Notice that **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 EDS 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 invisible/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:

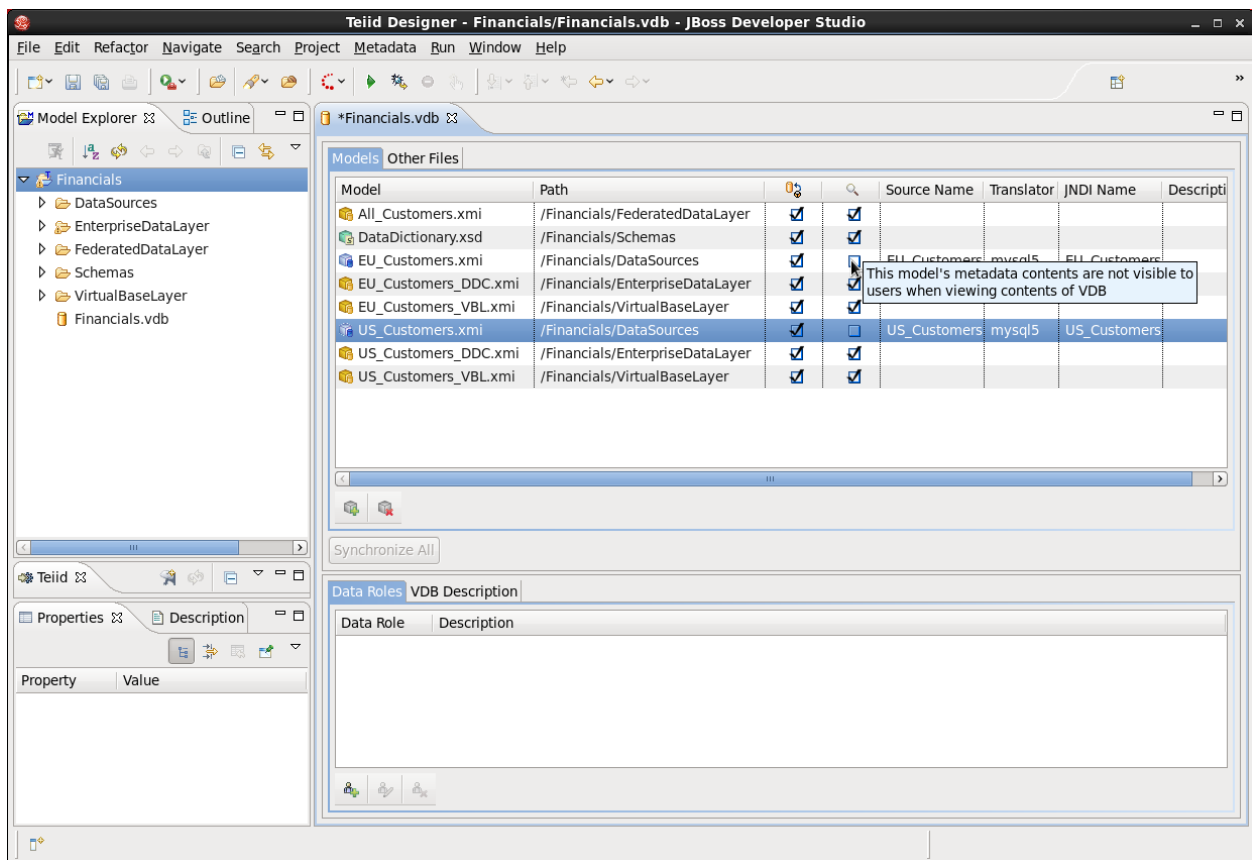


Illustration 93: Direct Source Access Hidden/Prevented in VDB

Save All to save the changes to the VDB.

Create Data Sources on the Teiid 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 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 (see the *JNDI Name* column in the VDB View in Designer, Illustration 94 above) must match. We can also use 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.

Right-click on the Data Sources folder under the Teiid Server instance in the Teiid Sever panel:

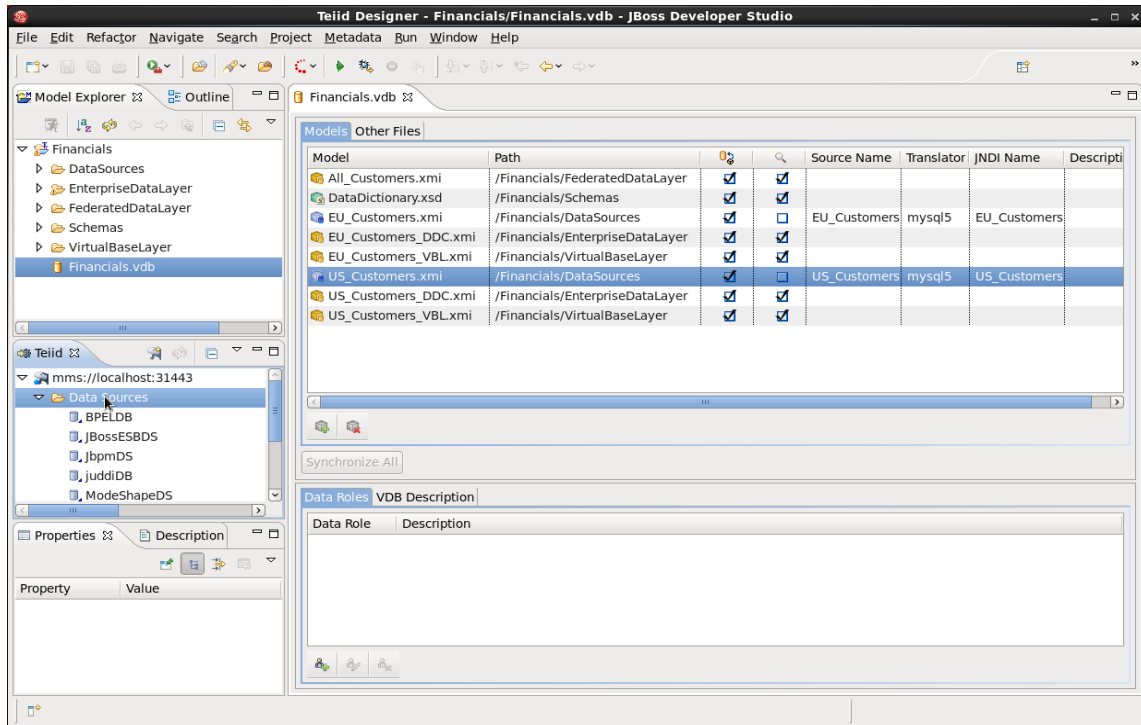


Illustration 94: Right-click on Data Sources (highlighted above) and select "Create Data Source"

This will open the Create Data Source wizard. Name the data source (US_Customers, for example) and choose the appropriate Connection Profile from the drop-down menu:

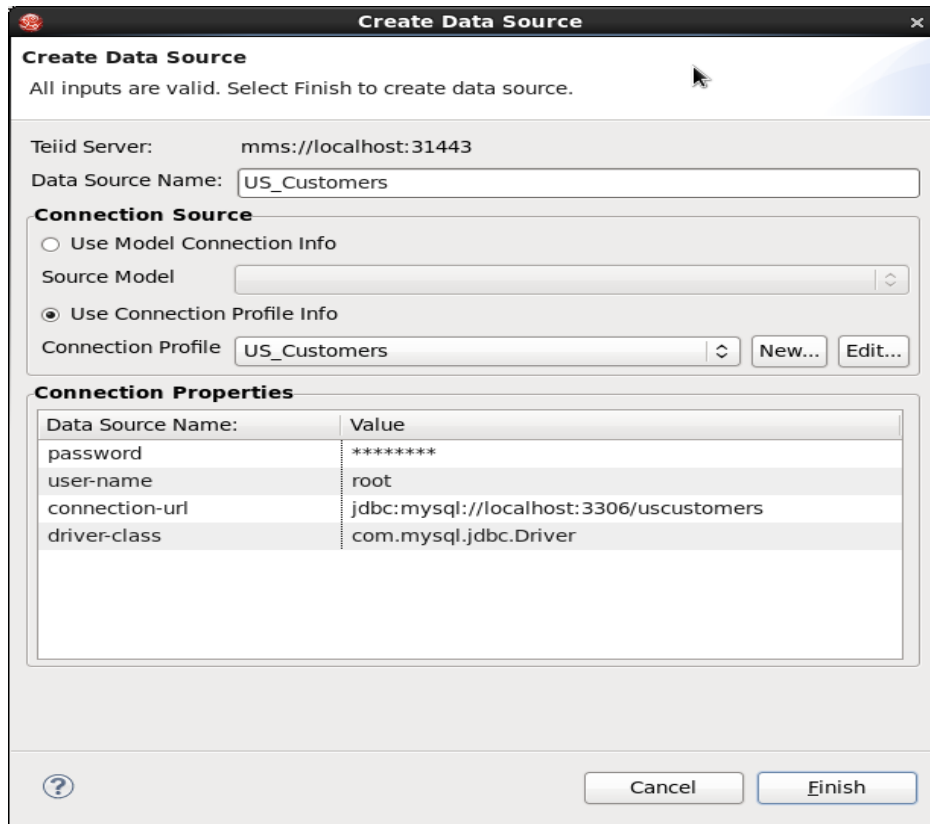


Illustration 95: Create Data Source on the Server wizard

Click Finish. Do the same for the other sources (EU_Customers, Products). Note that the sources all appear (along with some other internal sources used by Teiid) under the Data Sources folder of the Teiid instance:

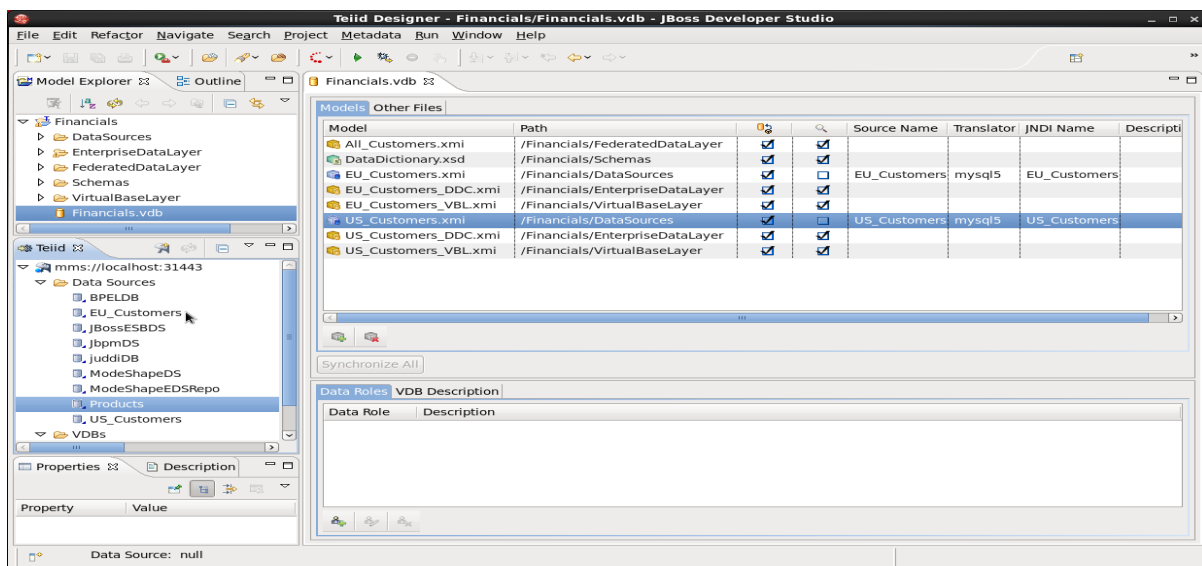


Illustration 96: Data Sources Visible under Teiid Server Instance

Deploy the VDB

We can now deploy the VDB to the Server, and it is as simple as right-clicking on the Financials.vdb model and selecting Modeling -> Deploy. Do this and then open the Admin Console Data Services -> Virtual Databases view to see that the VDB was deployed:

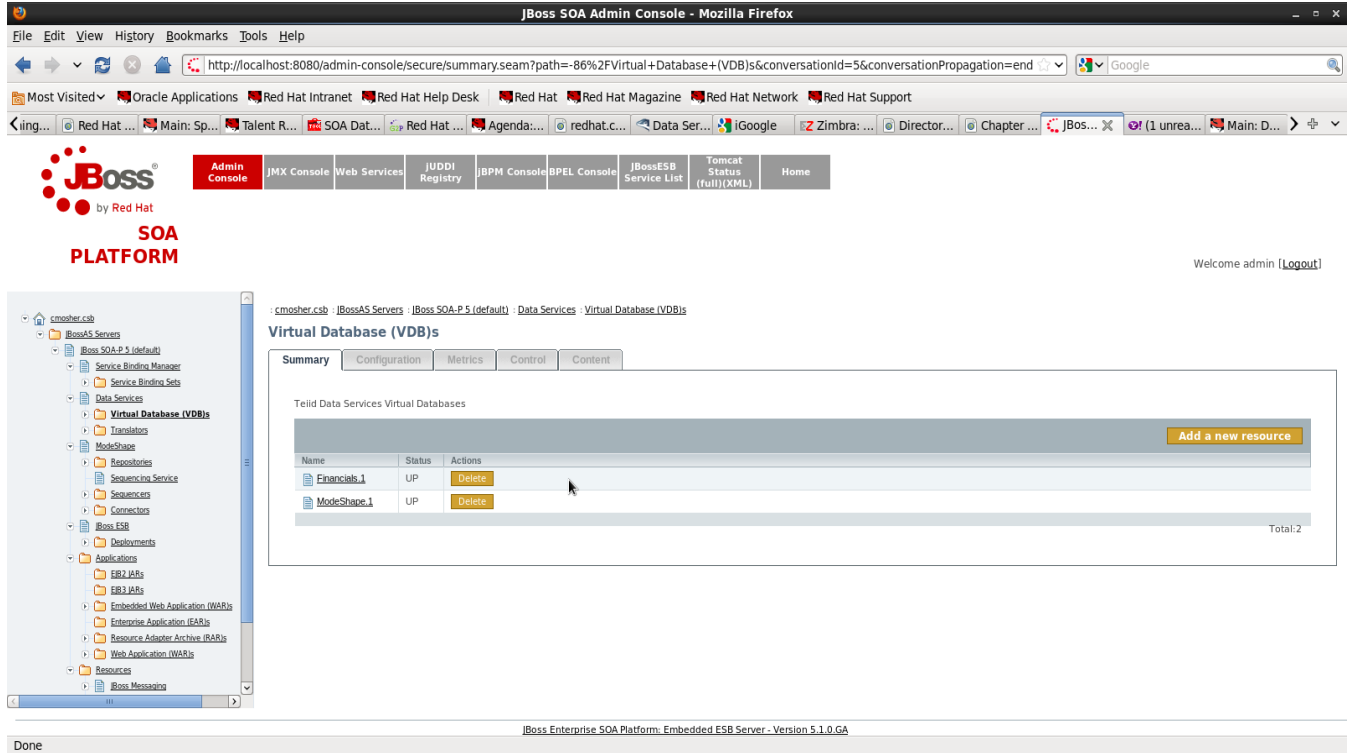


Illustration 97: Admin Console, Showing Deployment of Financials VDB

Dependency Diagrams

Here is a useful modeling feature: right-click on the All_Customers customer table and select Modeling -> Show Dependency Diagram. Then choose the Outline view (the tab next to the Model Explorer). Finally, select the "Show Diagram Overview" button (right under the Outline tab). You will see something like this:

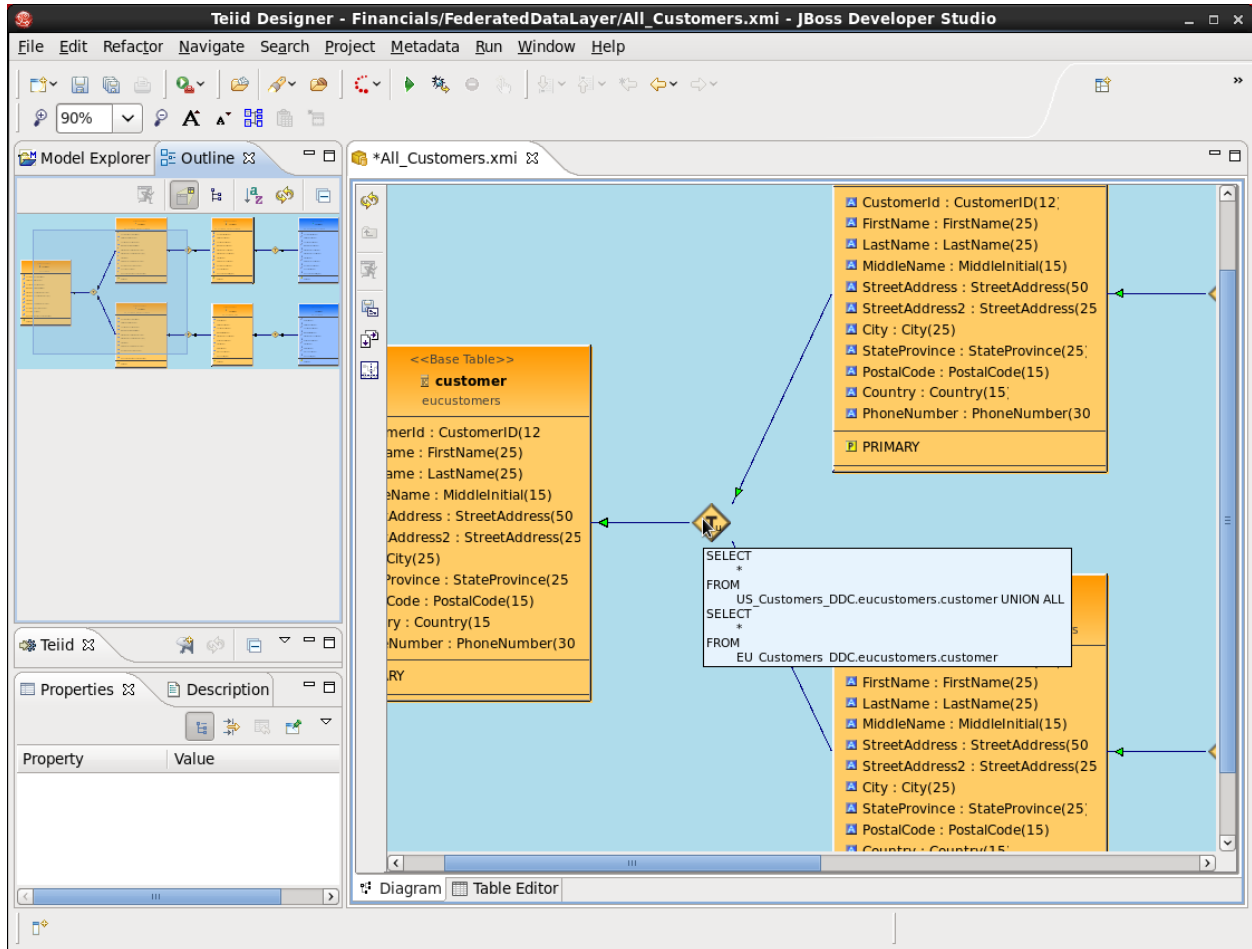


Illustration 98: Dependency Diagram of All_Customers Customer

Notice that 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 reusability that results from the ability to create data service layers without a performance penalty.

Congratulations, you have completed Lab 6.


```
[java] Body (row): 4-data d d d d  
[java] Body (row): 5-do not consume  
[java] Body (row): 6-data last record  
[java] [ESBTeiidServiceSyncClient] End of Contents
```

To undeploy the quickstart, execute the following from the `samples/custom_action_edss/` directory:

```
ant undeploy
```

Congratulations, you have finished Lab 7 and the complete set of lessons! But don't let this be the end of your exploration of the JBoss Enterprise Data Services Platform. Many more materials, quickstarts, documentation, whitepapers, etc. are available at jboss.com/products/platforms/dataservices/