



JBoss Developer Studio 4.0
Hands on Lab

SEAM AND HIBERNATE

Table of Contents

Introduction	5
Overview	5
Problem	5
Included Files	5
System Expectations	5
What is Expected of You	6
Database Schema	6
Lab Number 1: Installation and Running of HSQLDB	7
Starting HSQLDB	7
Loading Data into HSQLDB	7
Lab Number 2: Installation of JBDS	9
Get the File	9
Running the Installer	9
Lab Number 3: New Seam Project	19
Start JBDS	19
Select a Work Space	20
JBDS Start Page	21
Create New Page	22
Seam Web Project Wizard	23
Lab #4 - Exploring the Generated Artifacts and More	34
Explore the Tree View	34

Embedded Runtime	34
Access Control	35
Web Page Editing	35
Lab #5: Starting the Server	38
Start the Server	38
Lab #6: Editing Your First Page	40
Exploring Dynamic Publishing	40
Lab #7: Generate CRUD Application	41
CRUD Application in Seconds	41
Generate Entities	41
Lab #8: Running the Generated Application	45
Running the Application	45
Lab #9: AJAX in Moments	47
Why AJAX	47
Adding AJAX to the User List	47
Lab #10: Seam Security	50
Seam Security	50
Update the Authenticator	50
Lab #11: Security Continued	52
Rendering Tag Using Role	52
Selectively Rendering Menu Items	52
Lab #12: New Action	54
Create a New Action	54

How to Create a New Action	54
Lab #13: Testing	58
TestNG	58
Testing FirstAction	58
Lab #14: Style	61
Edit the Theme	61
Conclusion	62
What you learned	62

Introduction

Overview

In this lab we will show you all the steps necessary to create a blogging application. This lab leverages the embedded JBDS tools to quickly create a CRUD (Create, Read, Update, and Delete) web application. This lab will leverage an already created HSQLDB instance that you will be able to run from the command line. You will be provided a full installation of JBDS that is yours to use after this lab of JBoss Developer Studio 3.0 with an Embedded JBoss Enterprise Application Platform instance. This lab will take you through all the steps necessary to create a working prototype of the blogging application.

Problem

You were recently hired at ACME corporation and you have been asked to create a blogging application. For some reason the big bosses upstairs think that somehow your company will be able to corner the market on blogging. A little far fetched I know, but it does pay the bills. Since you are wanting to try out JBoss and Seam you thought you would try this integrated stack out, and try to see if you can at least come up with a prototype to drive some other Design Thinking sessions around how you will corner the Blogging Market. So this is a little far fetched, but it will at least be a fun exercise to see many of the parts of web development with JBoss Developer Studio

Included Files

In the user home directory a Downloads folder will exist, something like `/home/${user}/Downloads/Seam` you will find all the files needed to complete this lab. These files include the database directory, the lab guide, and a readme. We will walk you through installing all of these and creating a sandbox for you to play in throughout each of the labs. Please note that `/home/${user}` will now be known as `${USER_HOME}` throughout the rest of this document.

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. Please make sure you do this before running any of the labs. Two examples of what these settings might look like is below:

```
PATH=${Some Path}/jdk1.6.0_20/bin:${Some Path}/ant/apache-ant-1.8.1:${More Path Info}
JAVA_HOME=${Some Path}jdk1.6.0_20
```

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.

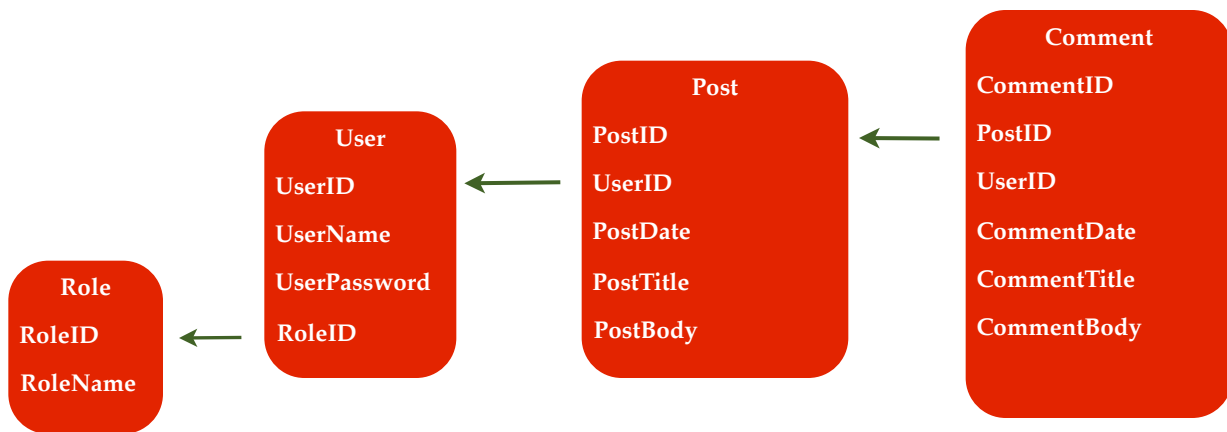
Please note that having an existing CLASSPATH environment variable set may cause odd issues with jar class loading, it is recommended to have this empty and not set. Please make sure to back up this value for when the lab is over. You are welcome to not do this, however weird things may happen when you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

Database Schema

A few things to note about what is in the picture. The first row/column in all of the tables is an auto number/identity/primary key field. This field will be incremented with each new row that is created in the database automatically for you. This gives us a nice easy way to reference every row in side of the database. Also notice the the odd layout below. This was done on purpose so you can see the Foreign key relationships in the database and how they are inherited/referenced by other tables. This “inheritance” and the ability for Hibernate (an Object Relational Mapping tool) to read these relationships greatly speeds up your application development. Later on we will see how the generated web application used these relationships to create the web UI for us. Some arrows are below to show this Foreign Key relationship



Lab Number 1: Installation and Running of HSQLDB

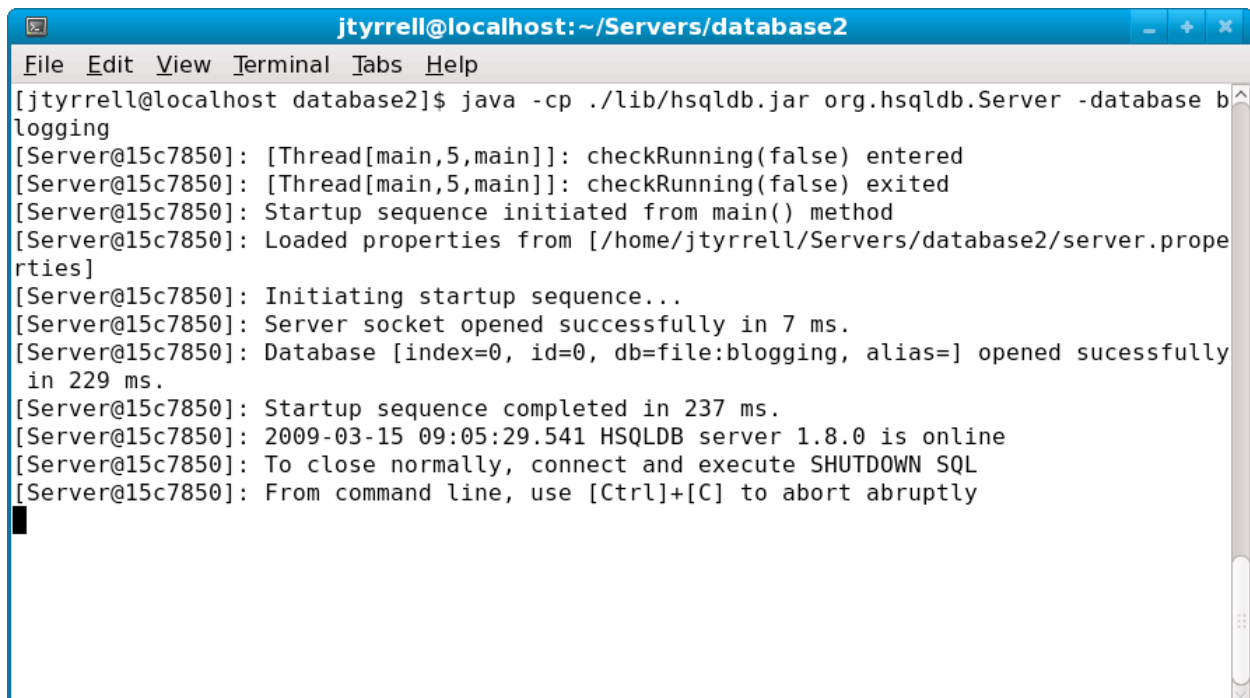
Starting HSQLDB

Look in the Seam directory and open up the readme.txt file. You should see two commands in there. One will look like the one below, but the below command may not work, just cut and paste what is in the readme.txt file it works for sure, this command should be ran from the database directory

Open some sort of Command Prompt, DOS Box, or Shell and run:

```
cd ~student/Downloads/Seam/database
java -cp ./lib/hsqldb.jar org.hsqldb.Server -database blogging
```

Please run this command in a command window as shown below:

A screenshot of a terminal window titled "jtyrrell@localhost:~/Servers/database2". The terminal shows the execution of the command "java -cp ./lib/hsqldb.jar org.hsqldb.Server -database blogging". The output shows the server starting up, including messages like "checkRunning(false) entered", "Startup sequence initiated from main() method", "Server socket opened successfully in 7 ms.", and "Database [index=0, id=0, db=file:blogging, alias=] opened successfully in 229 ms.". The terminal ends with "2009-03-15 09:05:29.541 HSQLDB server 1.8.0 is online" and instructions on how to shut down the server.

```
jtyrrell@localhost database2]$ java -cp ./lib/hsqldb.jar org.hsqldb.Server -database blogging
[Server@15c7850]: [Thread[main,5,main]]: checkRunning(false) entered
[Server@15c7850]: [Thread[main,5,main]]: checkRunning(false) exited
[Server@15c7850]: Startup sequence initiated from main() method
[Server@15c7850]: Loaded properties from [/home/jtyrrell/Servers/database2/server.properties]
[Server@15c7850]: Initiating startup sequence...
[Server@15c7850]: Server socket opened successfully in 7 ms.
[Server@15c7850]: Database [index=0, id=0, db=file:blogging, alias=] opened successfully
in 229 ms.
[Server@15c7850]: Startup sequence completed in 237 ms.
[Server@15c7850]: 2009-03-15 09:05:29.541 HSQLDB server 1.8.0 is online
[Server@15c7850]: To close normally, connect and execute SHUTDOWN SQL
[Server@15c7850]: From command line, use [Ctrl]+[C] to abort abruptly
```

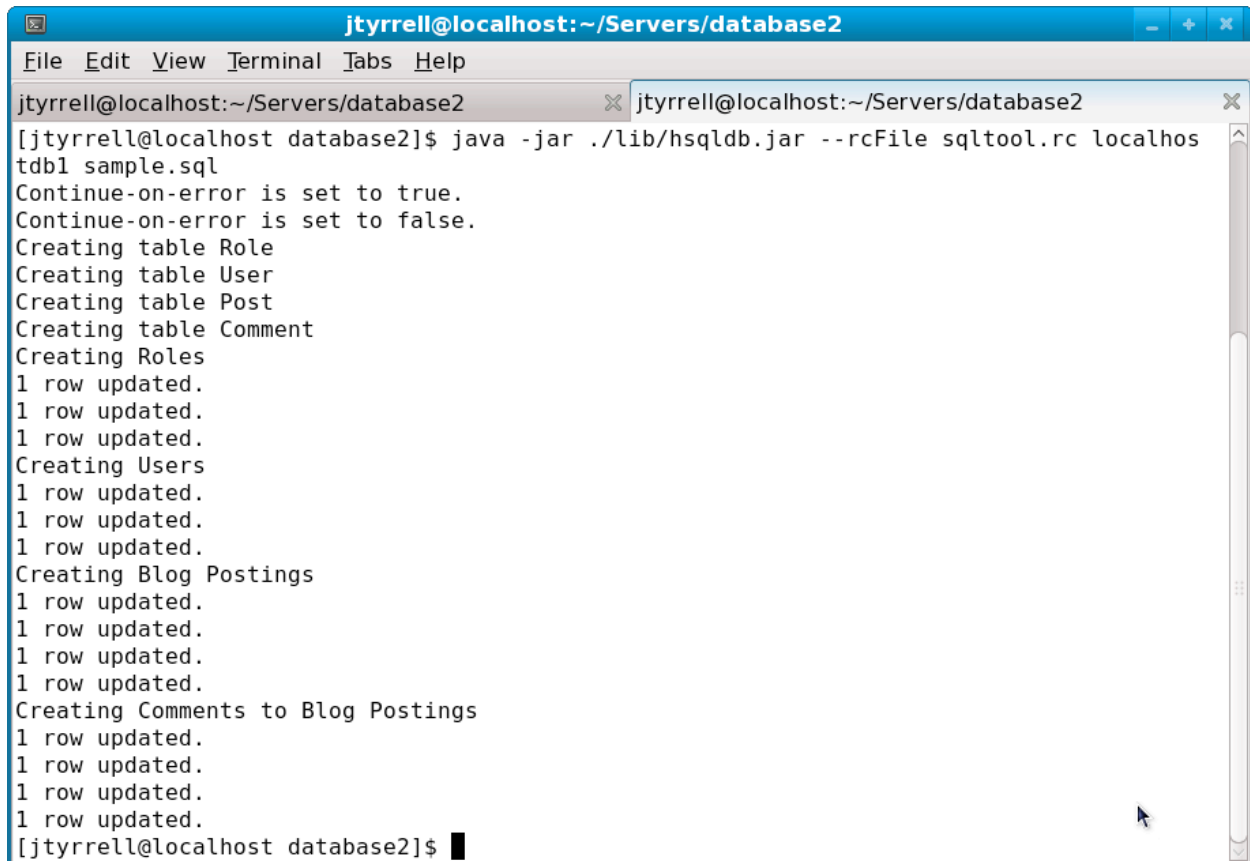
Leave this window running for the entirety of the lab.

Loading Data into HSQLDB

Since this database does not have any information in it, we will have to load it. If during the lab you wish to reset your database you can also rerun this command. You can hold down Shift-Control-t to open a new shell in the same directory. It might be recommended to run this in a separate command tab if one is available on your OS. Again look in the readme.txt to see how to run this command and it should look like this:

```
java -jar ./lib/hsqldb.jar --rcFile sqltool.rc localhostdb1 sample.sql
```

You can see a sample run of it attached below:



```
jtyrrell@localhost:~/Servers/database2
File Edit View Terminal Tabs Help
jtyrrell@localhost:~/Servers/database2 x jtyrrell@localhost:~/Servers/database2 x
[jtyrrell@localhost database2]$ java -jar ./lib/hsqldb.jar --rcFile sqltool.rc localhos
tdb1 sample.sql
Continue-on-error is set to true.
Continue-on-error is set to false.
Creating table Role
Creating table User
Creating table Post
Creating table Comment
Creating Roles
1 row updated.
1 row updated.
1 row updated.
Creating Users
1 row updated.
1 row updated.
1 row updated.
Creating Blog Postings
1 row updated.
1 row updated.
1 row updated.
1 row updated.
Creating Comments to Blog Postings
1 row updated.
1 row updated.
1 row updated.
1 row updated.
[jtyrrell@localhost database2]$
```

Again it would be recommended to keep this window open the entire time, however it may not be needed. The HSQLDB must be up and running through out the entirety of the lab.

Lab Number 2: Installation of JBDS

Get the File

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

```
jbdevstudio-product-eap-linux-gtk-4.....
```

Running the Installer

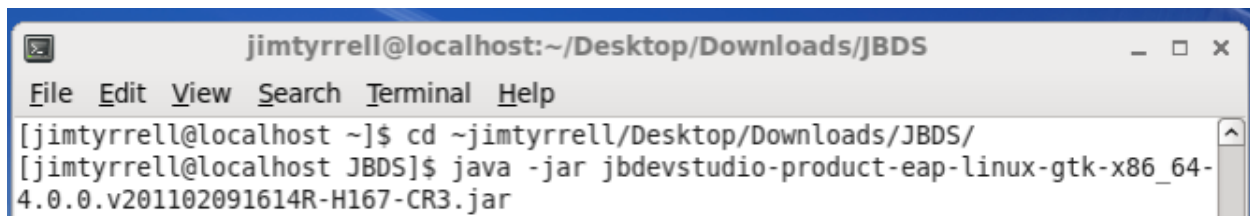
The next step is to run the installer. The command to do that is very simple:

```
cd ~/student/JBDS
```

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

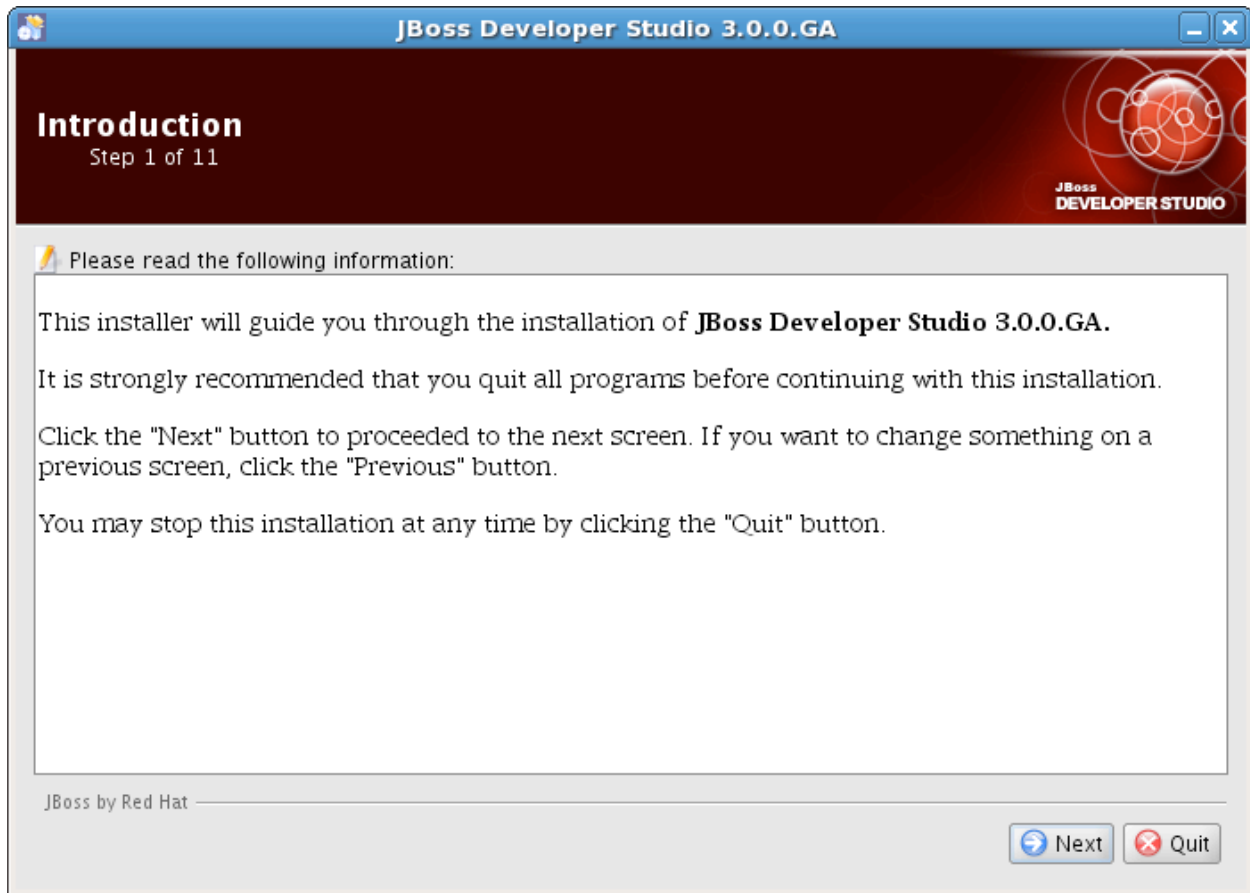
```
java -jar ${The_File_Available_For_Linux_Above}
```

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

A screenshot of a Linux terminal window. The title bar reads "jimtyrrell@localhost:~/Desktop/Downloads/JBDS". The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal content shows the following commands and output:

```
[jimtyrrell@localhost ~]$ cd ~/jimtyrrell/Desktop/Downloads/JBDS/
[jimtyrrell@localhost JBDS]$ java -jar jbdevstudio-product-eap-linux-gtk-x86_64-4.0.0.v201102091614R-H167-CR3.jar
```

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



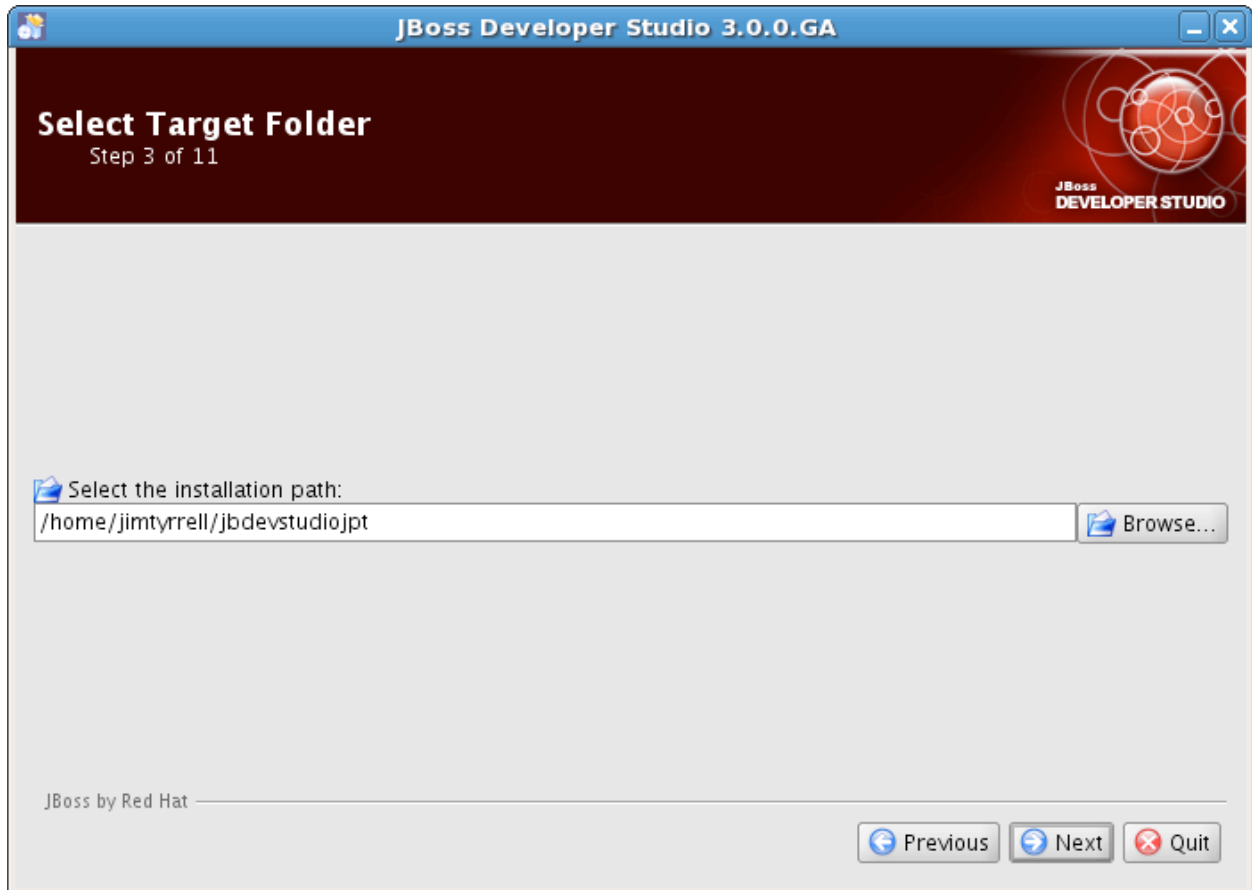
Please Select Next

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



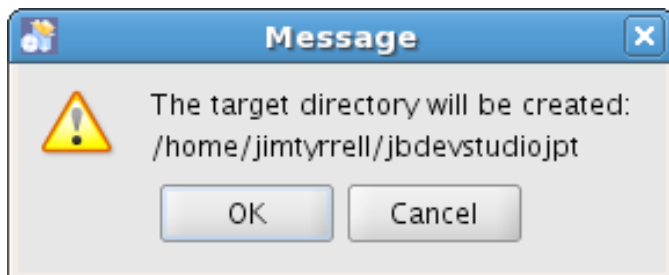
Please select the "I accept...." radio button to enable the next button. Click the next Button.

The next screen prompts you on where to install JBDS. Make the name unique and don't just select the default. A few recommendations. 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 so that someone else after you will be able to install this lab without any problems.



Please Select Next

No changes probably will be made to this setting 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.

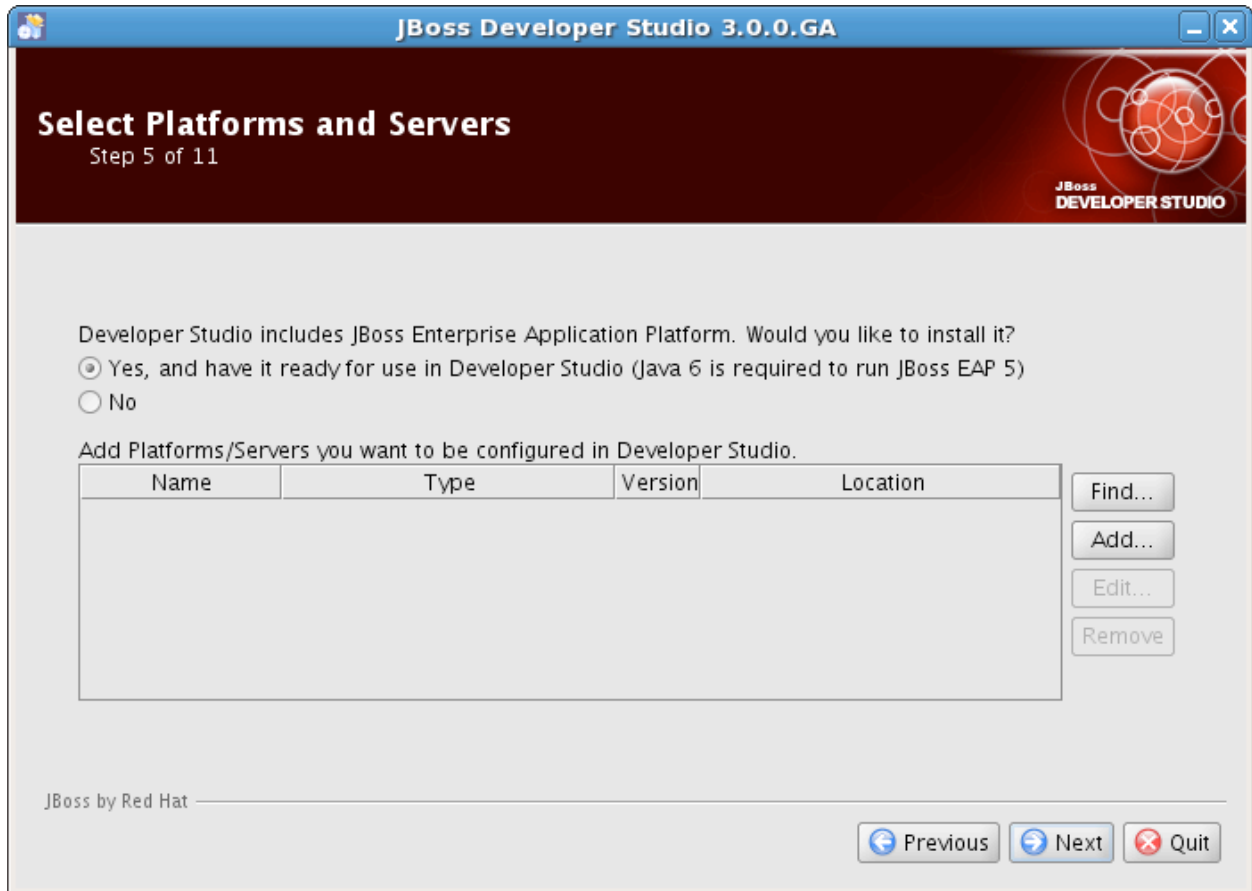


Next you will be prompted to use the JDK that was used from your path/java_home properties. No changes are needed with this setting.



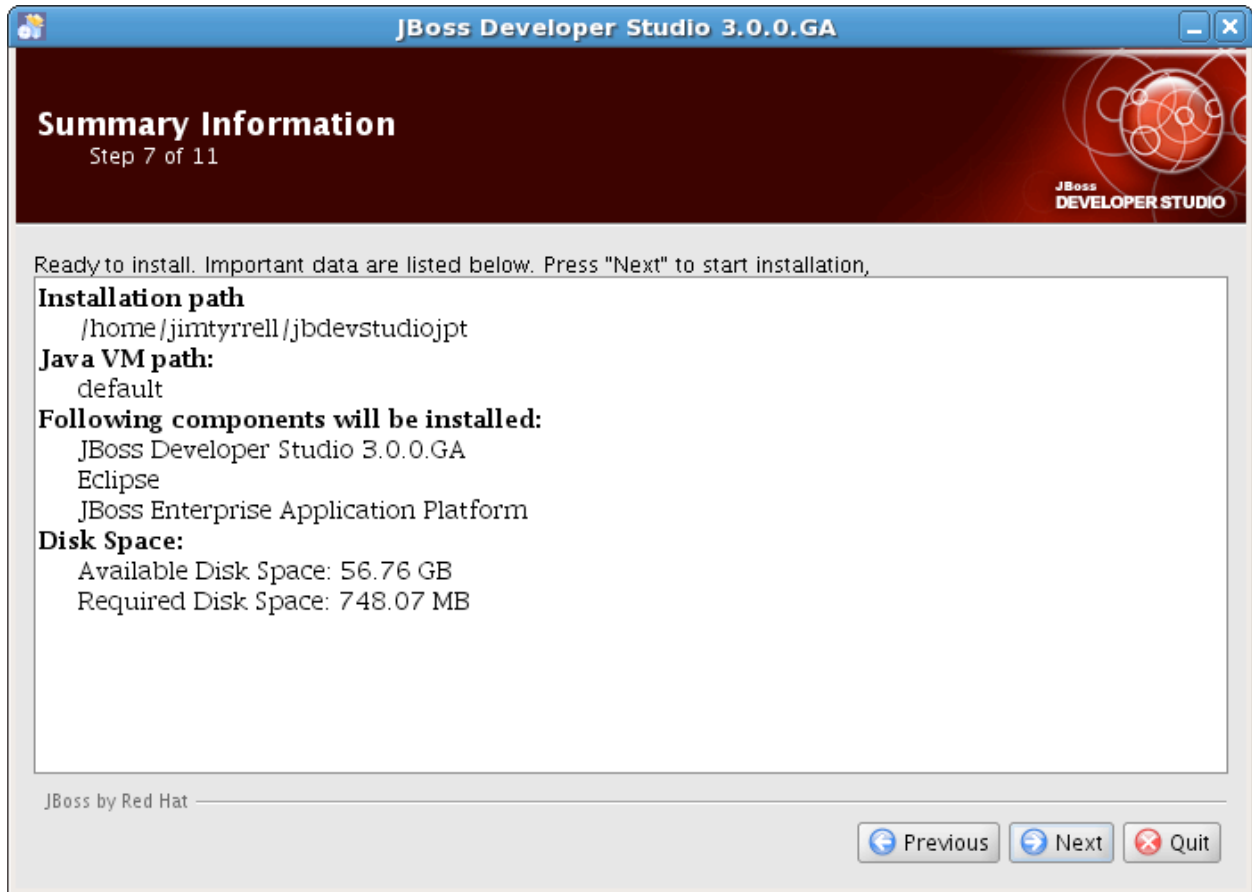
Please select Next

The next box will prompt you to install and make available the embedded EAP instance. Please leave this section alone and click Next. If this was the SOA-P (Service Oriented Architecture - Platform) Lab then you could add a SOA-P instance that would already have to be installed. You also could add another JBoss EAP/AS instance, but for the purposes of this lab this is not required and not recommended.



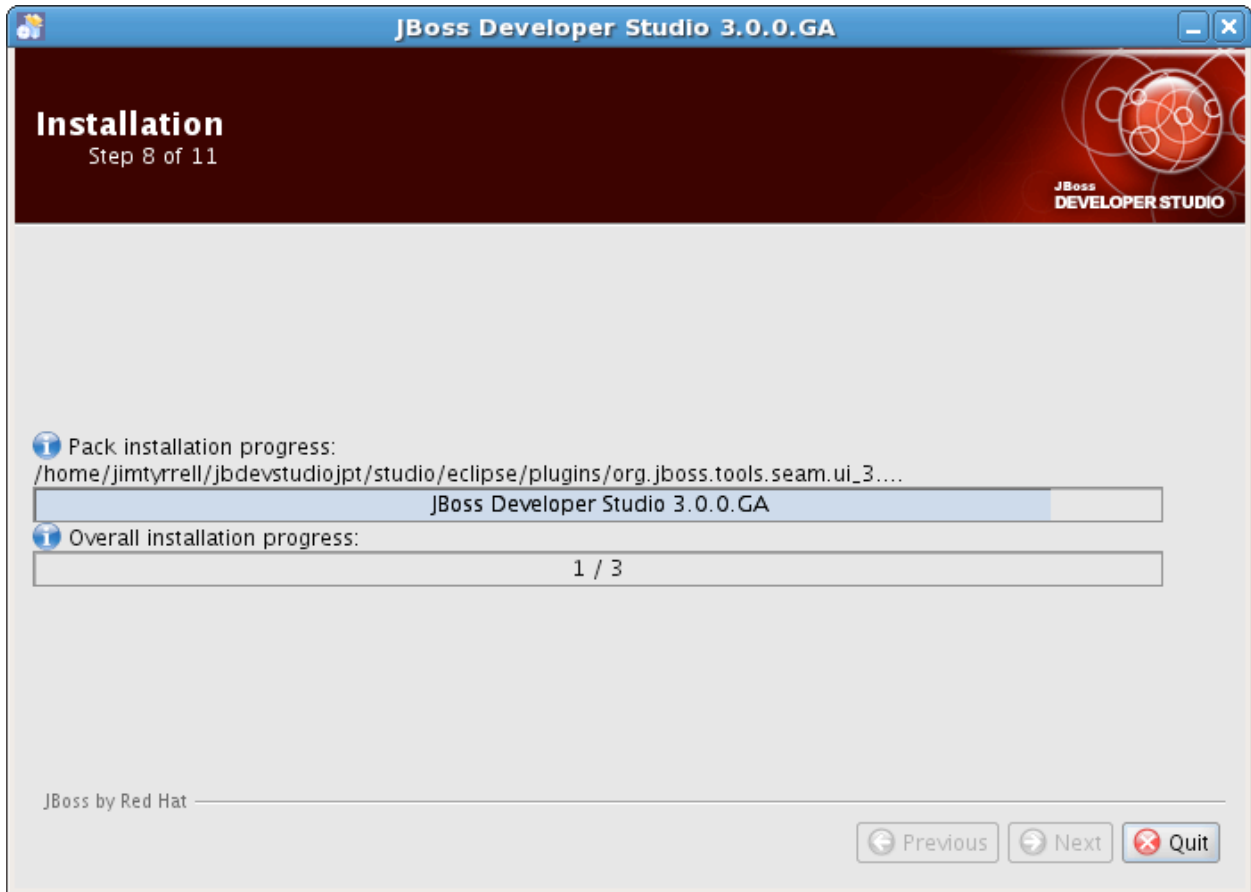
Select Next

The next screen just shows you what will be installed:



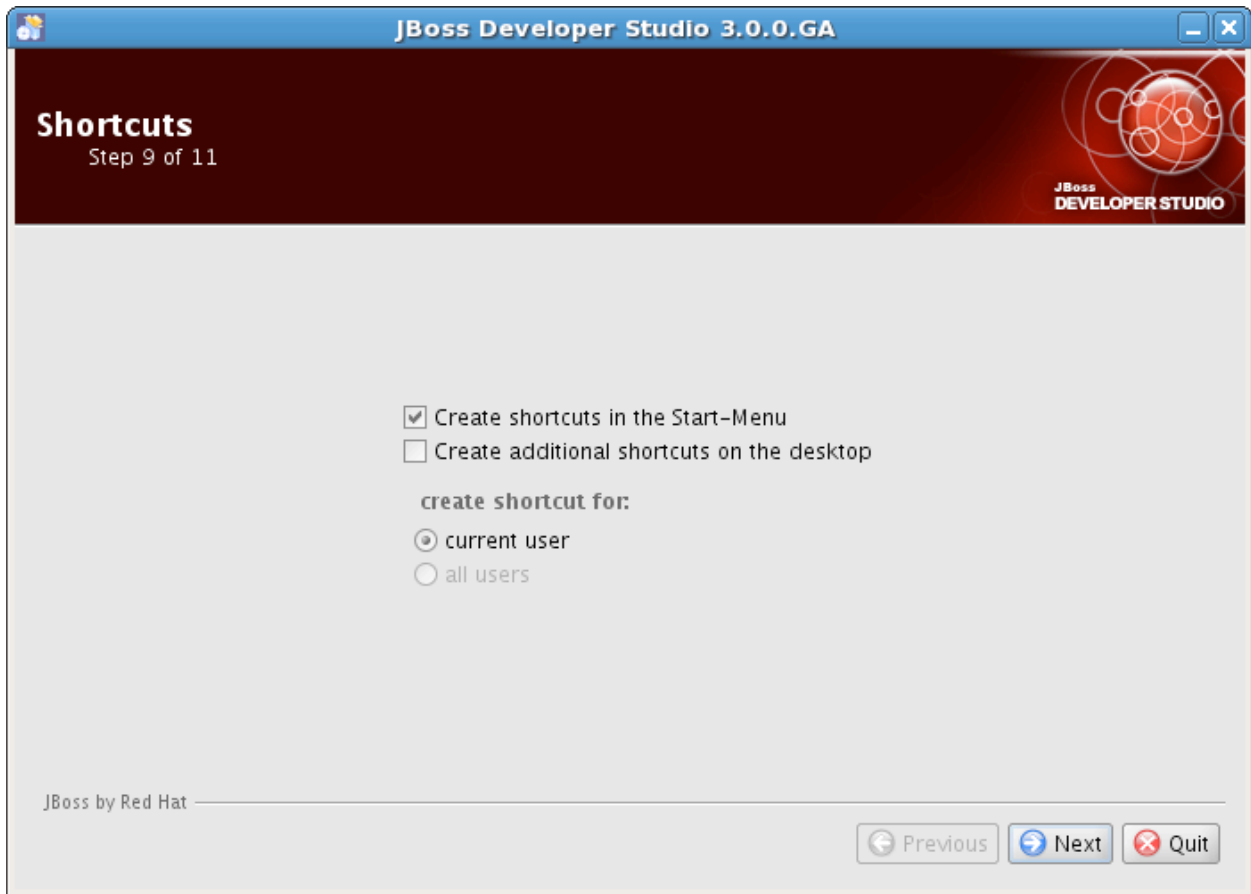
Click Next

The installer running



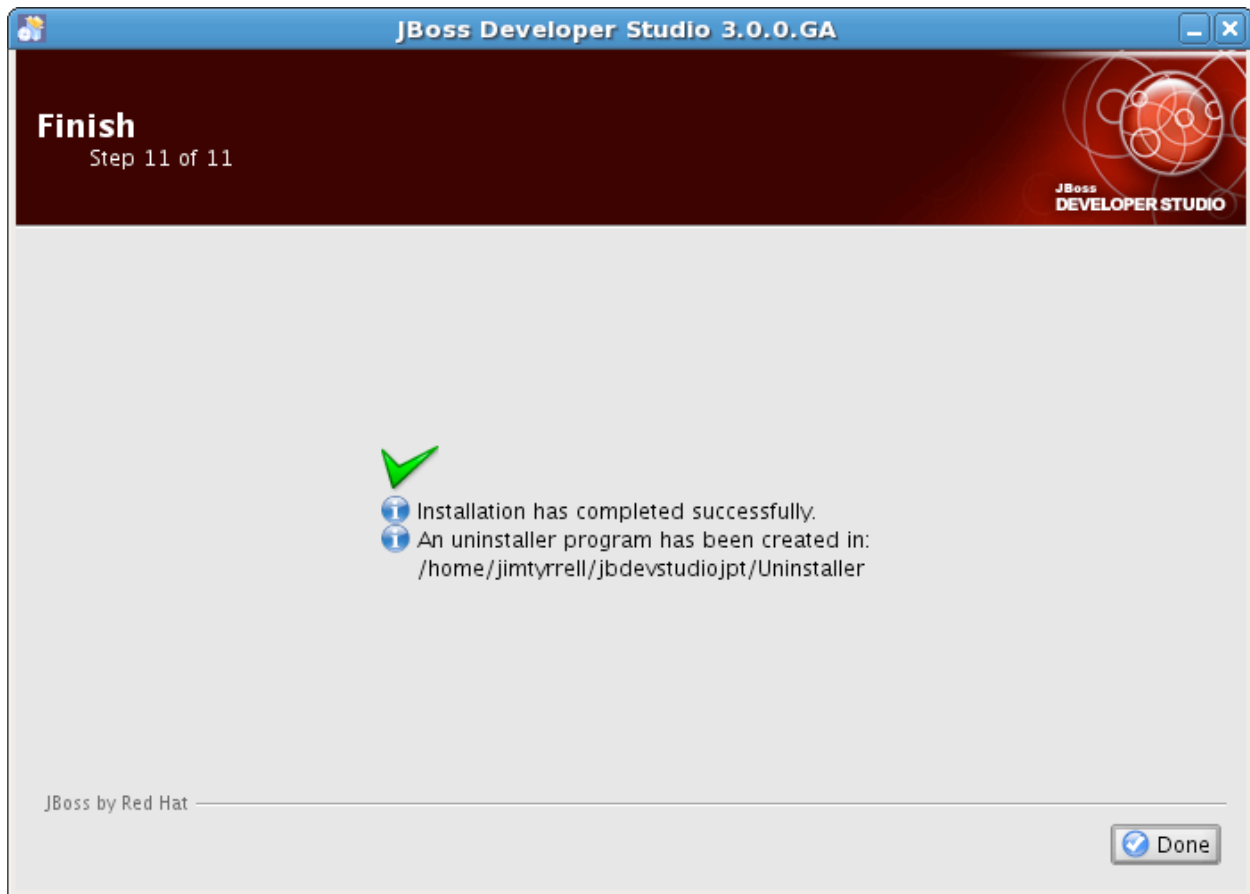
Wait for Next to be available to you once the installer is completed

The next step will show you option on where to place short cuts and the like. The defaults are fine for this lab.



Select Next if you want to make any changes, feel free.

Congratulations you have installed JBDS 2.0



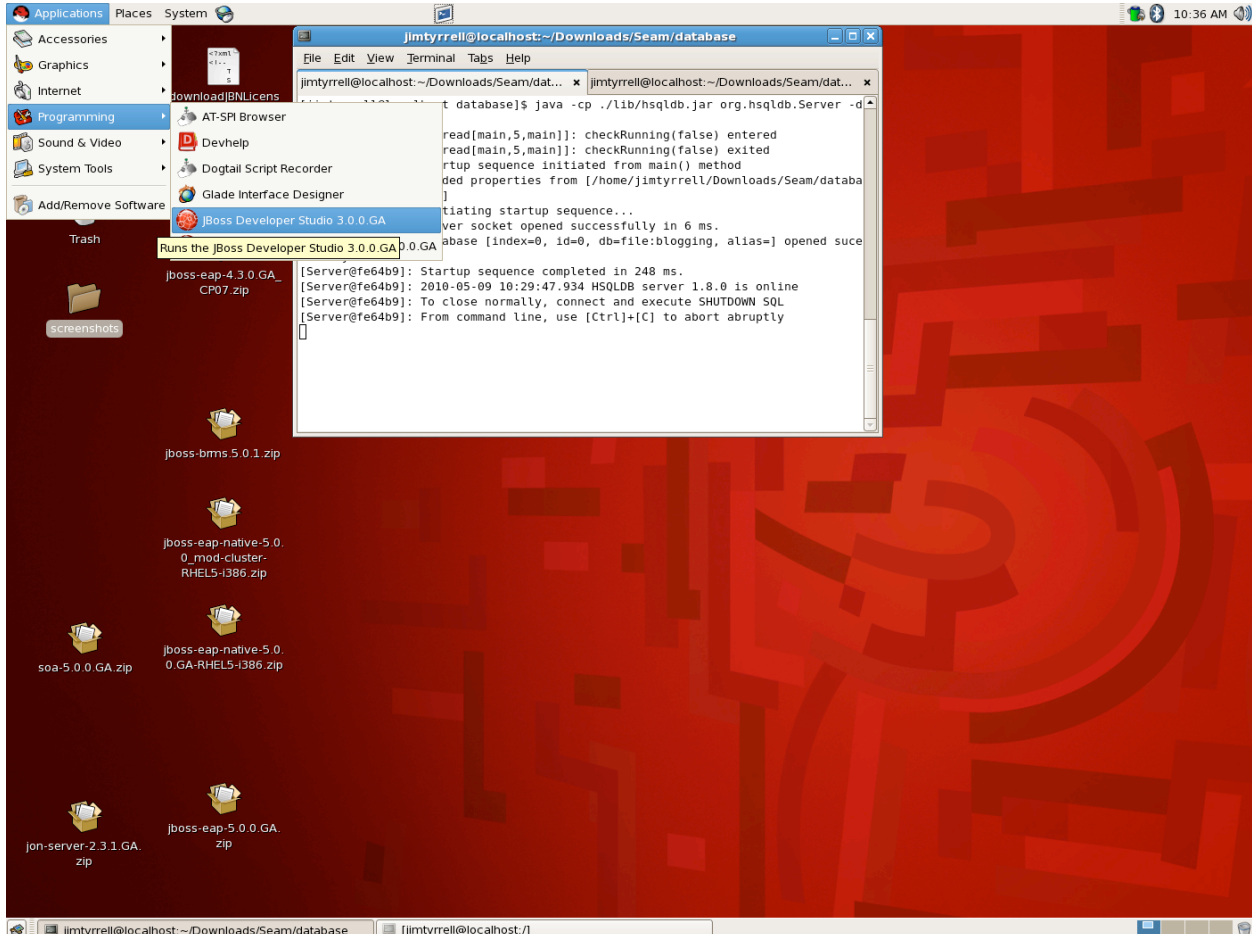
Please Select Done.

Congratulations you are now done installing JBDS 3.0 on your local workstation. You have now completed this lab and we will next create a new Seam Project using the already running HSQLDB.

Lab Number 3: New Seam Project

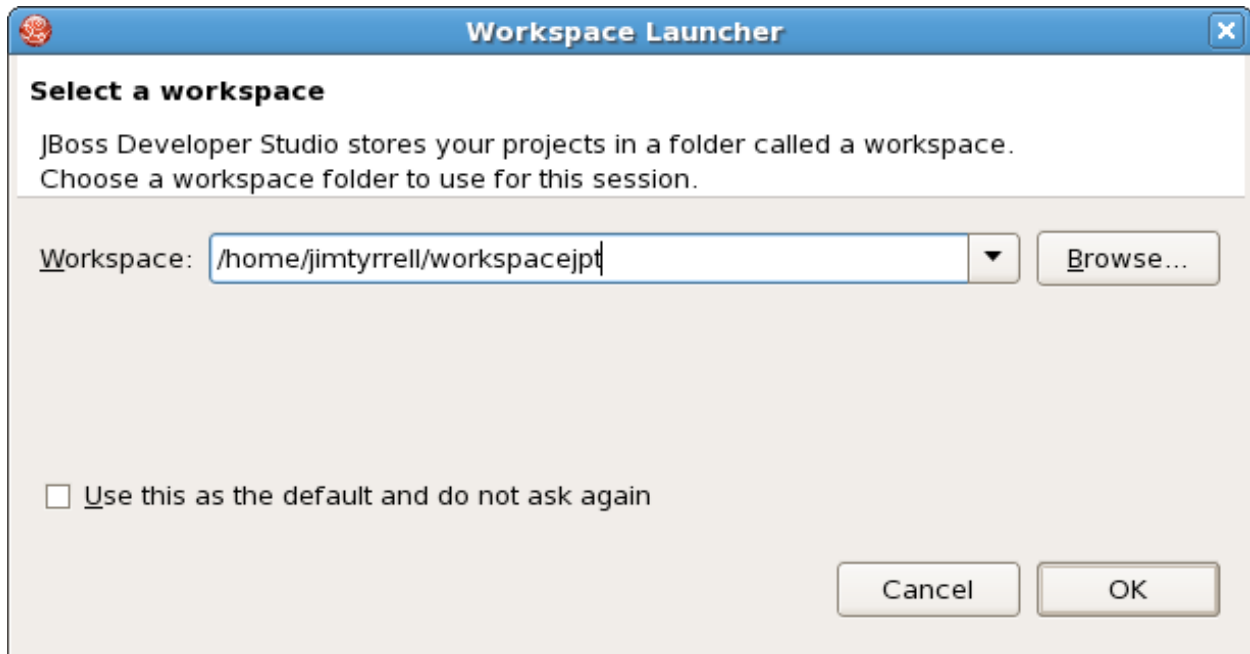
Start JBDS

To start JBDS you will have to find the JBDS instance we just installed. There should be short cuts in your programs menu or maybe even on your desktop depending on what you selected above when installing. On linux it will be installed Applications -> Programming -> JBoss Developer Studio 4.0.0.GA as seen below:



Select a Work Space

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



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. Select the Create New Icon Above



Create New Page

This brings up several options as seen below:



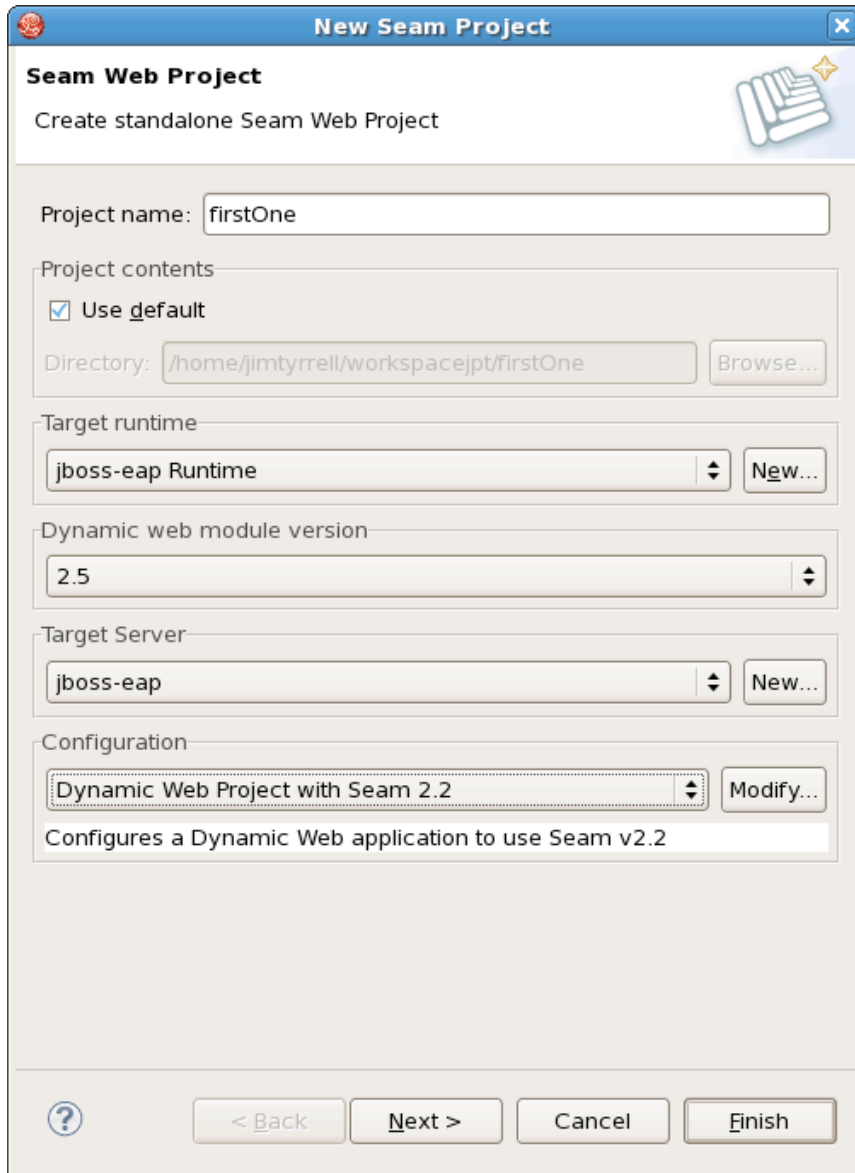
Please select the "Create Seam Project"

Seam Web Project Wizard

The Seam New Web Project Wizard allows us to rapidly create all the artifacts needed to run a Seam Application on the embedded EAP instance. Very few things will need to be changed in this Wizard please make note of any changes to the defaults.

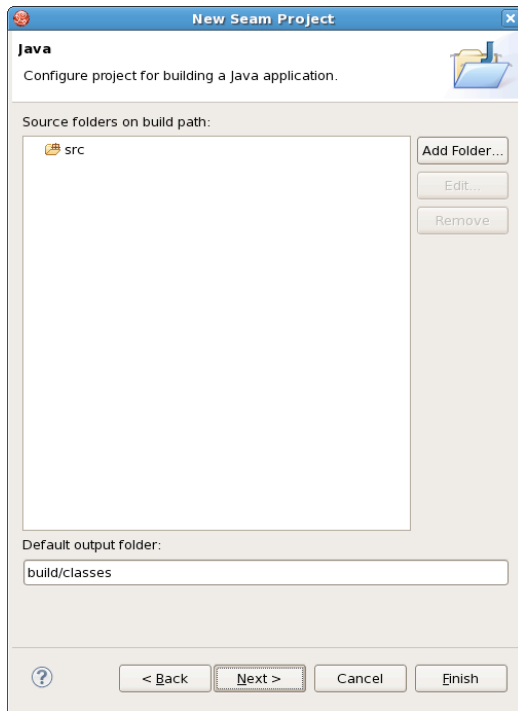
Step 1 in the Wizard you have one thing to do.

- 1.) Type in a Project Name, "firstOne"

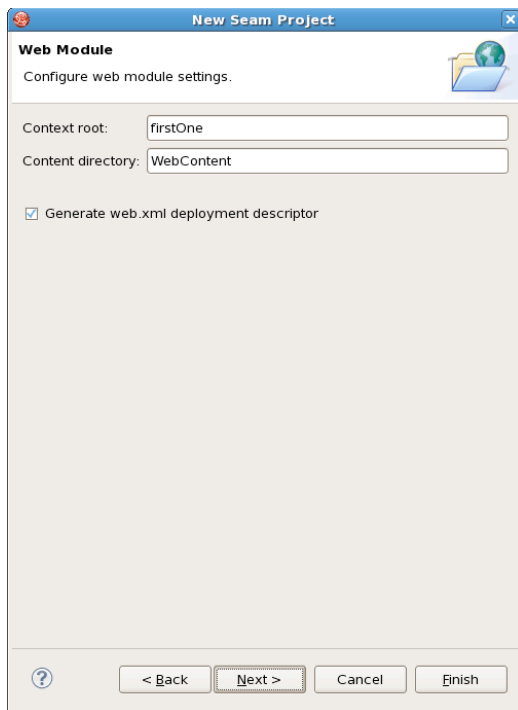


Please make sure you have named the project as above. Select Next

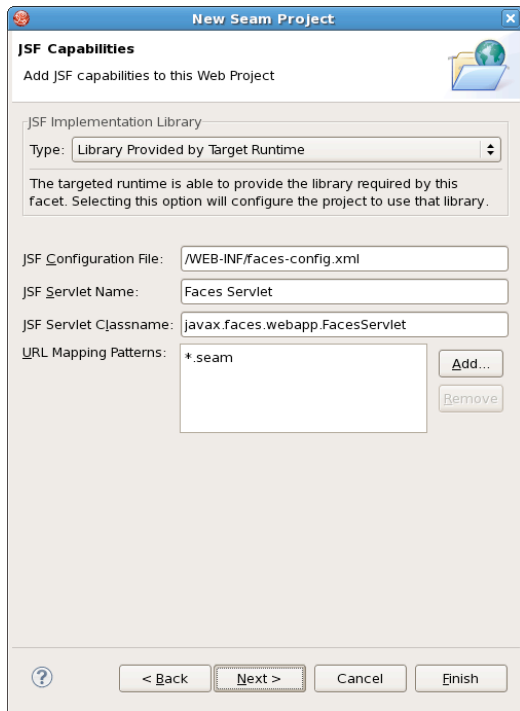
The next three pages in the wizard require no changes:



Please Select Next



Please Select Next



The next pages requires us to make available the HSQLDB tables and values that we created earlier in this lab.

New Seam Project

Seam Facet
Configure Seam Facet Settings

General

Seam Runtime: Seam 2.2.EAP5 Add...

Deploy as: WAR EAR

Database

Database Type: HSQL

Connection profile: Def Edit... New...

Database Schema Name:

Database Catalog Name:

DB Tables already exists in database:

Recreate database tables and data on deploy:

Code Generation

Session Bean Package Name: org.domain.firststone.session

Entity Bean Package Name: org.domain.firststone.entity

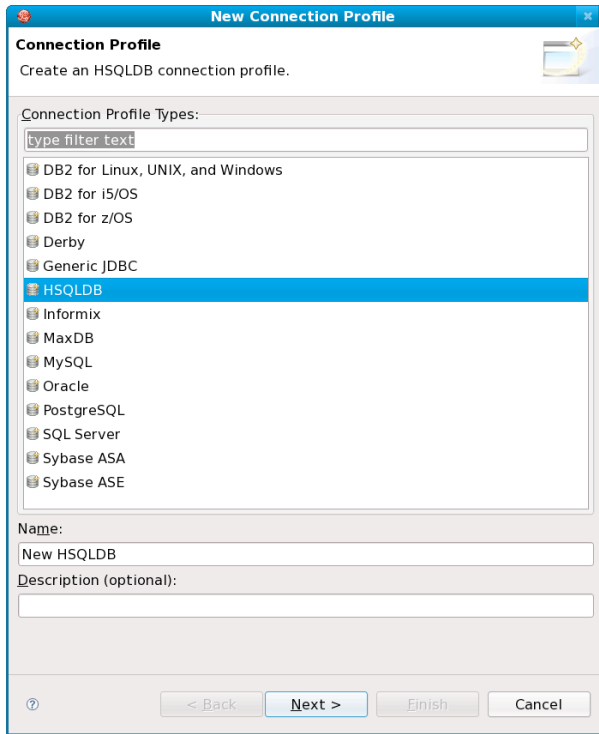
Create Test Project:

Test Package Name: org.domain.firststone.test

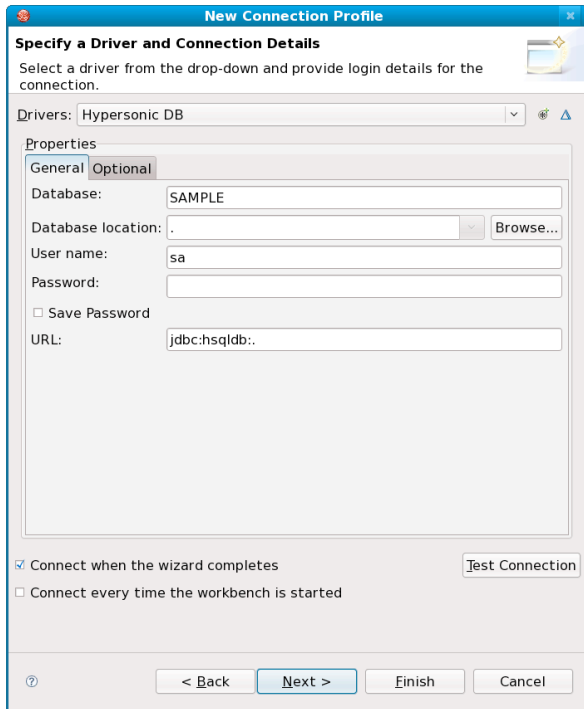
? < Back Next > Cancel Finish

Please select "New" above near where it says HSQL. You will now need to create the correct settings to connect to the already running database.

After selecting New across from the Connection Profile you will be prompted with this screen. Please select HSQL:

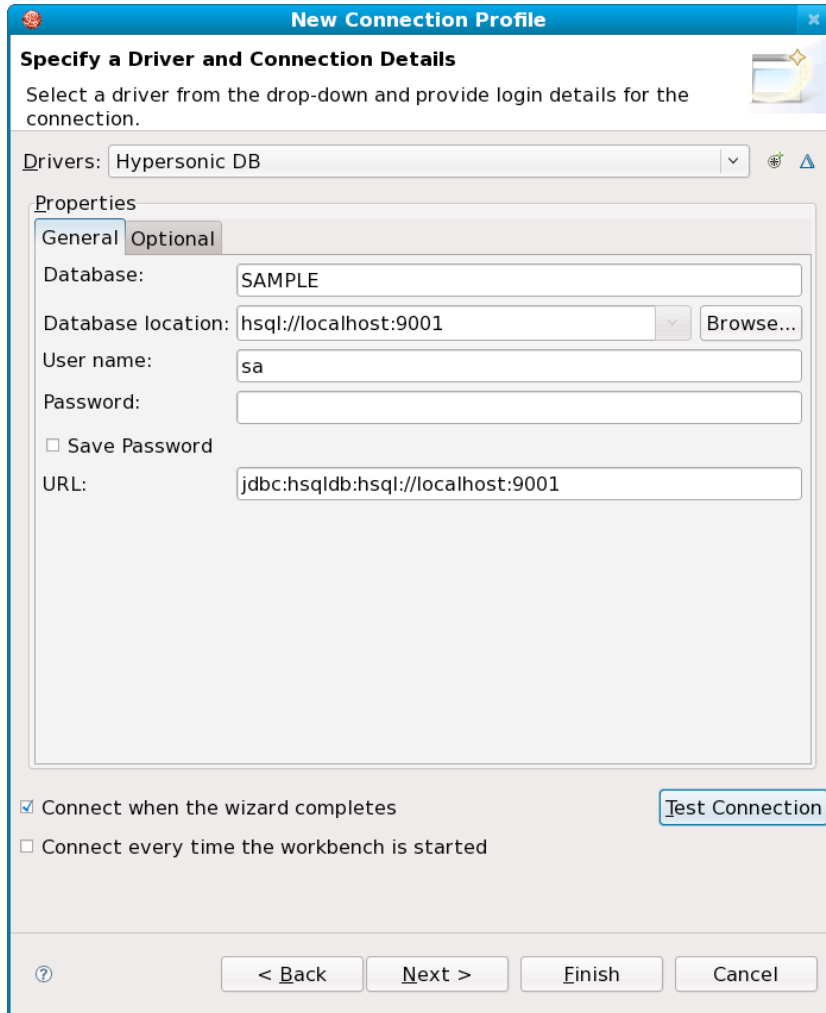


Please Select "Next"

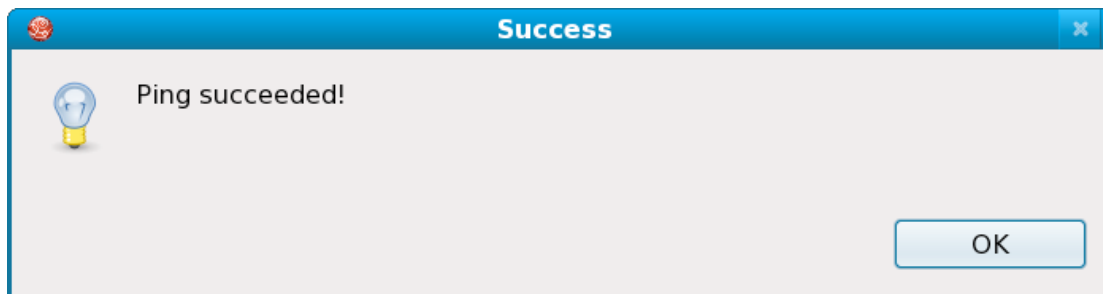


This is the panel before making any changes, now update it to look like the next screen grab.

You will now have to type in the correct Database location as indicated in the next screen shot. Please note that hsql://localhost:9001 was used in the Database location box.



Once you are done putting in the correct Database Location, please select the “Test Connection” button above. You should see the next screen shot pop up. If you do not please raise your hand and ask for help. Please do not go past this step if the “Test Connection” button click does not result Ping Succeeded Pop Up.



Select “OK”

The next step is to review the setting for the database, but since the ping succeeded there is no need to go back and make any changes at this point.

New Connection Profile

Summary
Information gathered from previous pages.

Property	Value
Name	New HSQLDB
Description	
Auto connect at start	false
Auto connect on first use	true
Driver name	Hypersonic DB
Database	SAMPLE
Database location	hsql://localhost:9001
User name	sa
URL	jdbc:hsql:hsqldb:hsql://localhost:9001
Save password	false
Connection properties	

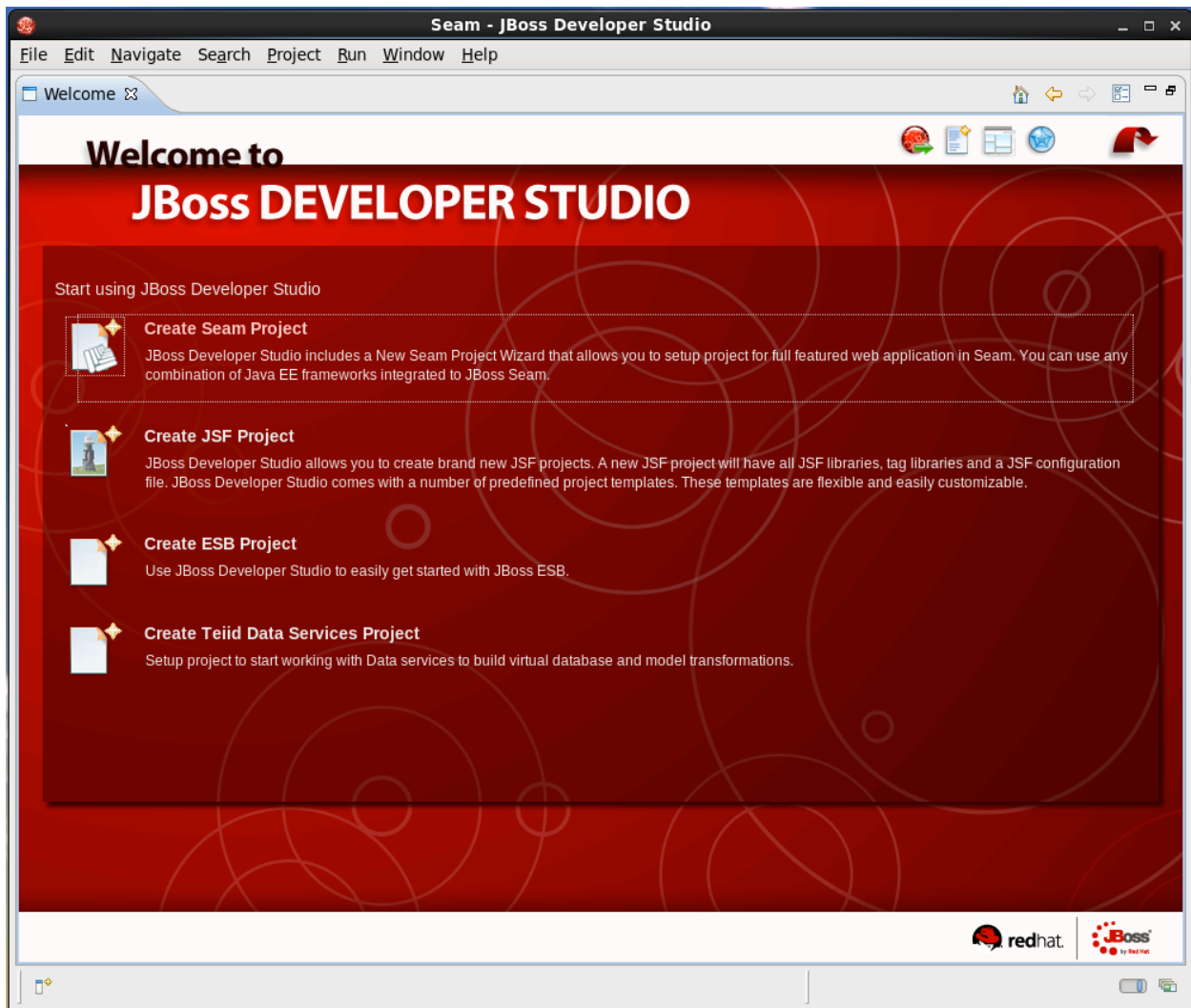
Buttons: ? < Back Next > Finish Cancel

Please select "Finish"

At this point you are done and Seam and SeamGen are ready to create your Seam Project.

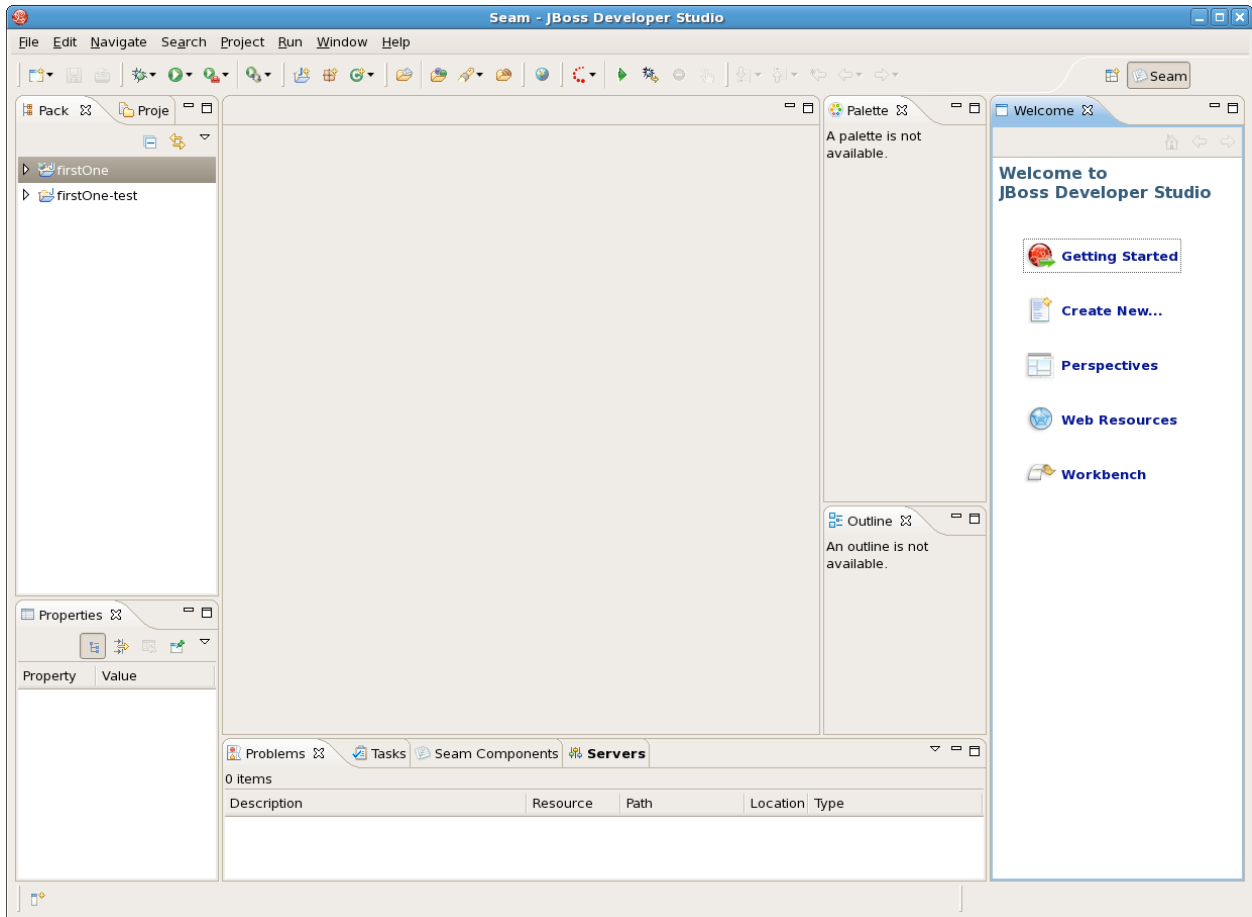
Please select “Finish” and you Seam Project will be created for you.

You will now be brought back to this landing page:

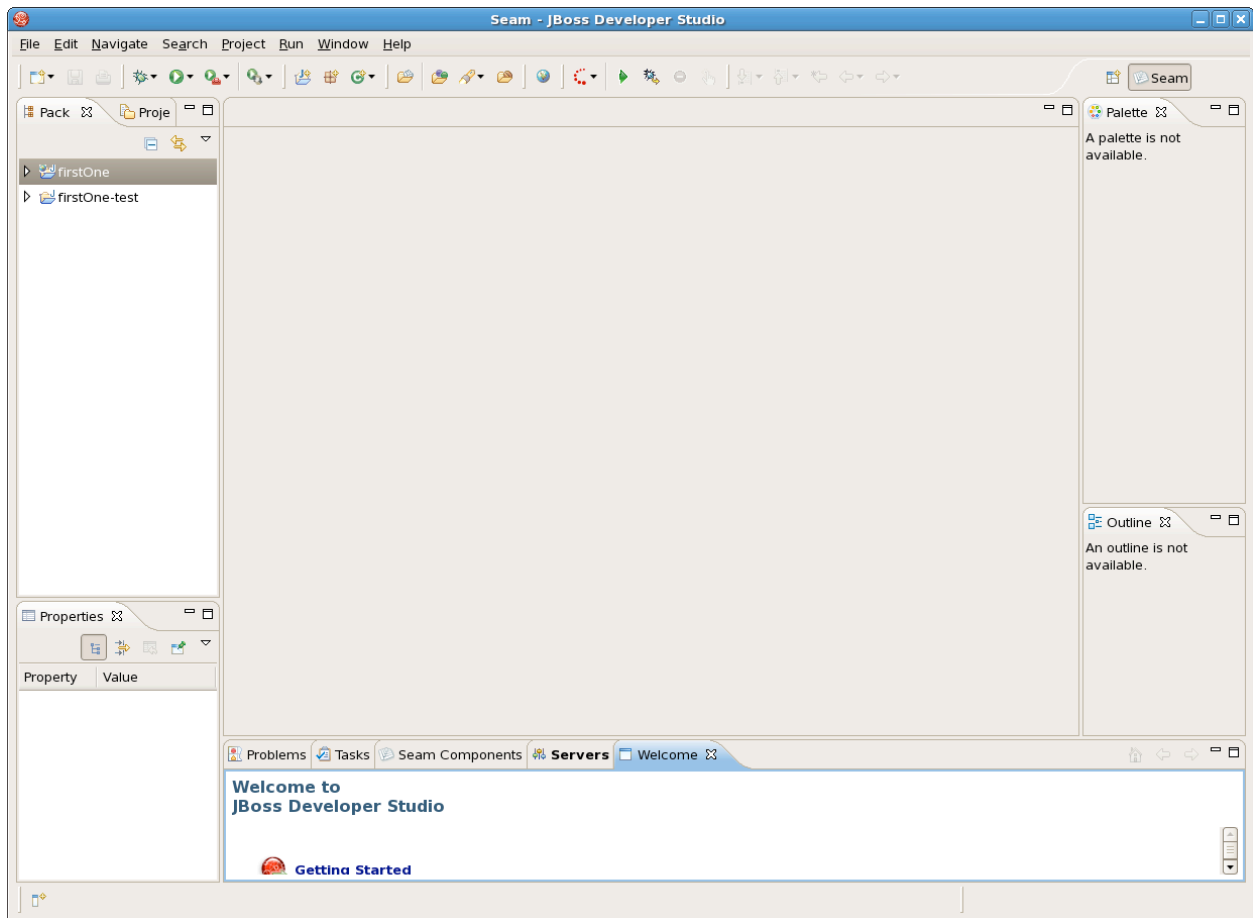


Please double click on the “Welcome” in the upper left hand corner.

I have submitted this bug and in a future version it should come to this landing page. After double clicking on “Welcome” in the upper left this is the way this should look.



I will usually drag the welcome Screen that is on the right hand side to the bottom middle pane as show here:



You are now done with this lab. We have successfully created a Seam Application. I am sure at this point you are looking to start looking at and working with the artifacts created. We will do that in the next lab.

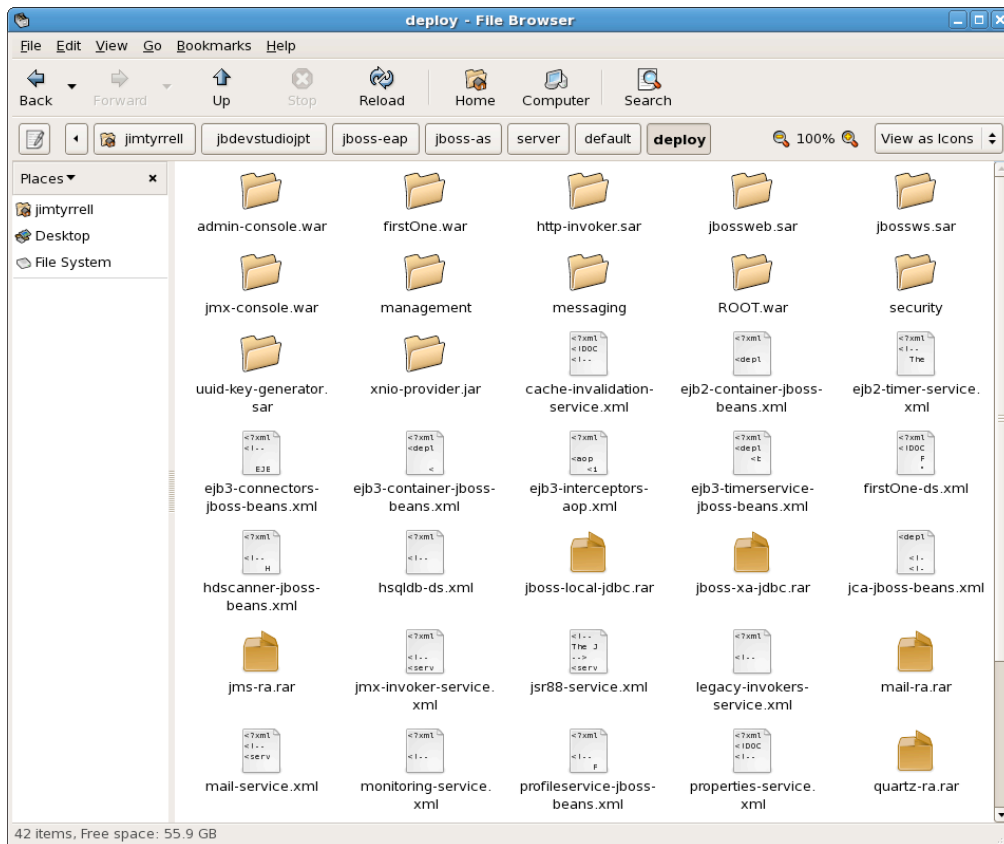
Lab #4 - Exploring the Generated Artifacts and More

Explore the Tree View

- firstOne - This is the top level project that holds of the generated artifacts
 - src/hot - This is the home for java classes which will be loaded automatically or hot w/o a JBoss EAP Server restart. Please pay attention to the kinds of files that are generated in here.
 - src/main - This is the home for java classes which will not be loaded for you automatically and you will have to restart the server to get these loaded into the application. Pay attention to the files that are placed in here and remember they are not reloaded for you at code writing time.
 - WebContent - This is the home for the facelets that drive your web application. These will also be loaded/ deployed for you automatically as you make changes to them.
- firstOne-test - This is where your TestNG test cases are created and placed for automatic testing of your objects.

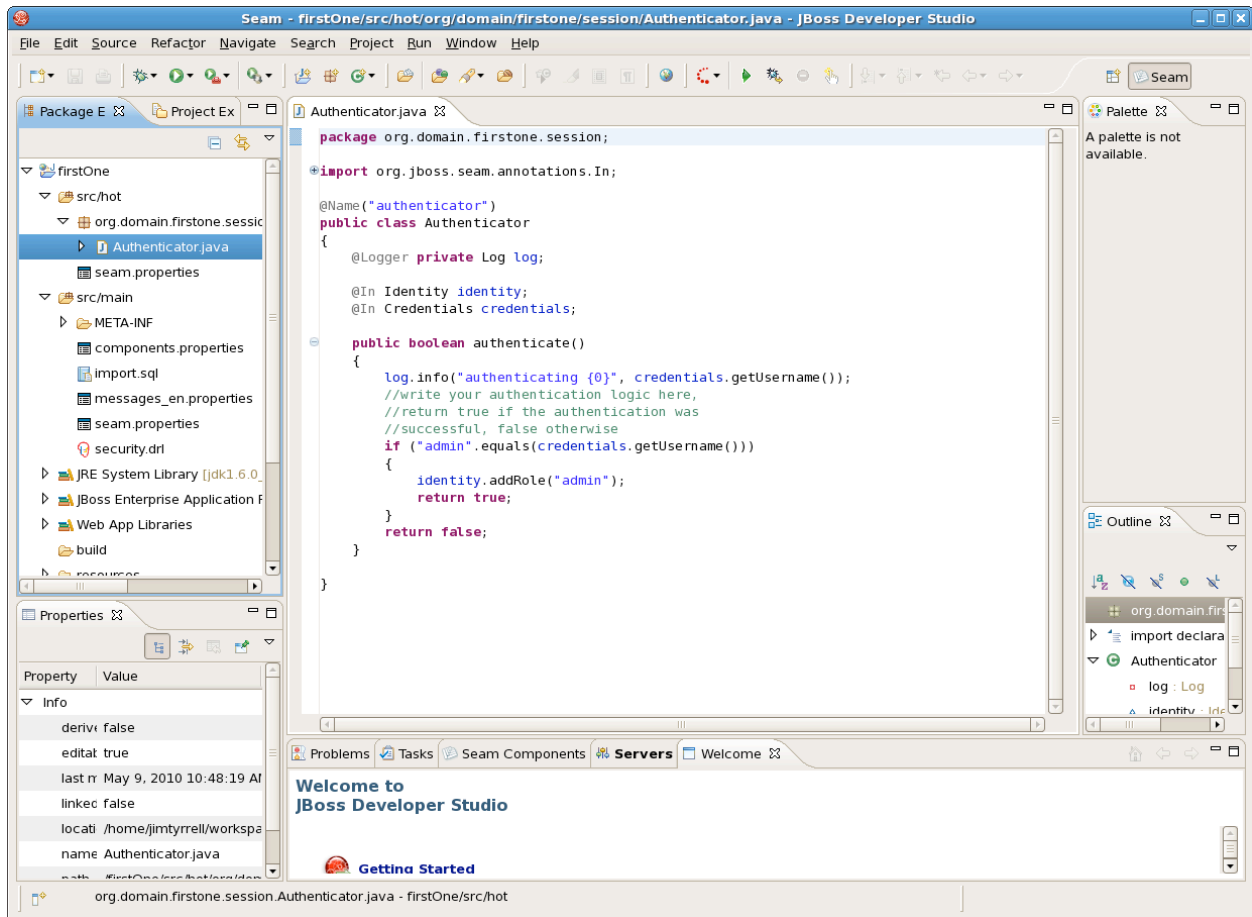
Embedded Runtime

Another area to explore is the embedded JBoss EAP instance and some of the directories and locations of files that are placed out there when you are working with JBDS. As you can see in this attached screen shot an firstOne.war and firstOne-ds.xml were created for you. Both of these files are in `$(JBDS-Home-Dir)/jbdevstudio/jboss-eap/jboss-as/server/default/deploy`



Access Control

When exploring the firstOne project, you will have to select or click the arrow/chevron to the left of the project name to expand the view of available files in the bucket, you will do this for any folder, package, etc you need to open. The first file to open is the double click on Authenticator.java (firstOne -> src/hot -> org.domain.firstone.session -> Authenticator.java) this file is used to grant access to the Seam Application. We will explore this file in a bit. Notice that an `addRole("admin")` method is called in this class. We will make some changes to this class later on in another lab. Your screen should now look like this:

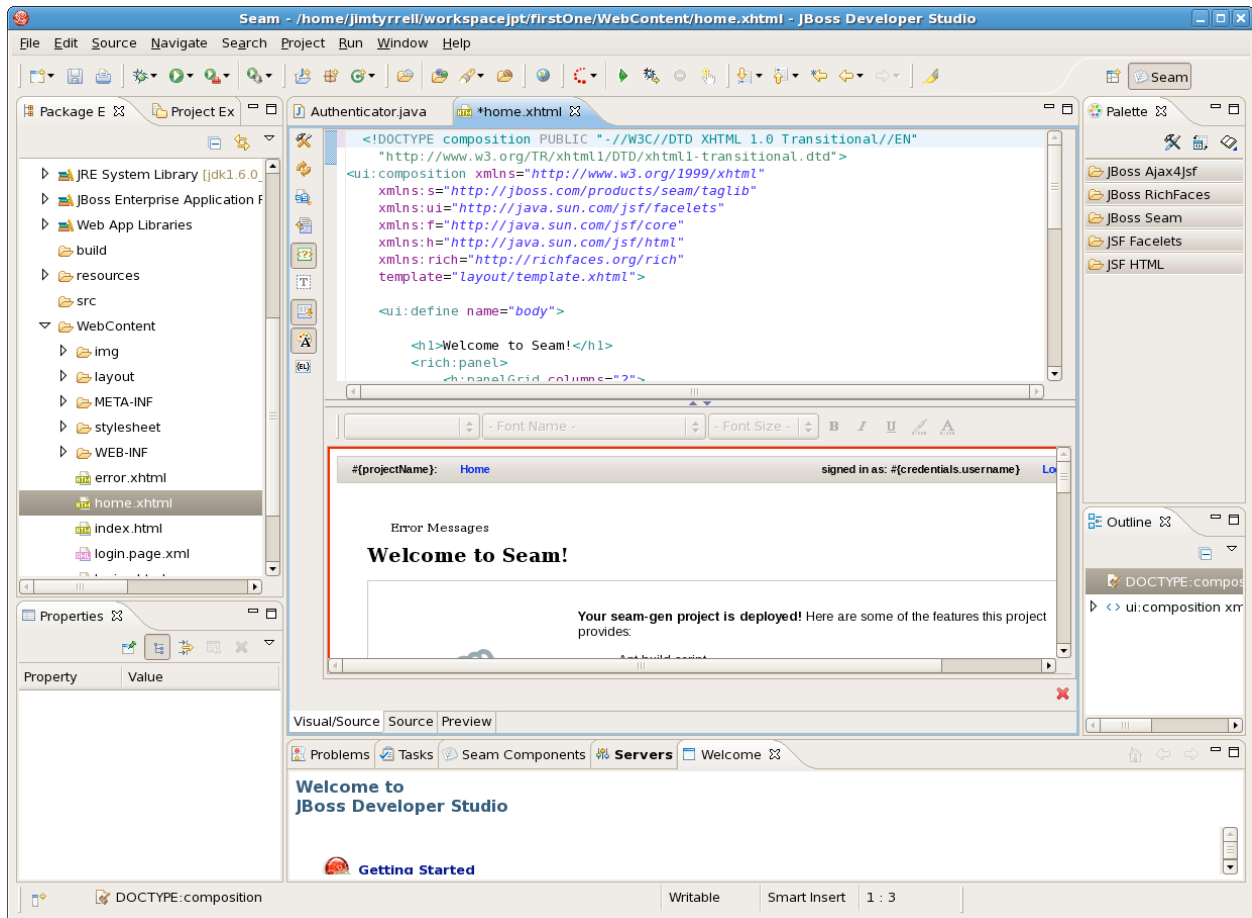


Web Page Editing

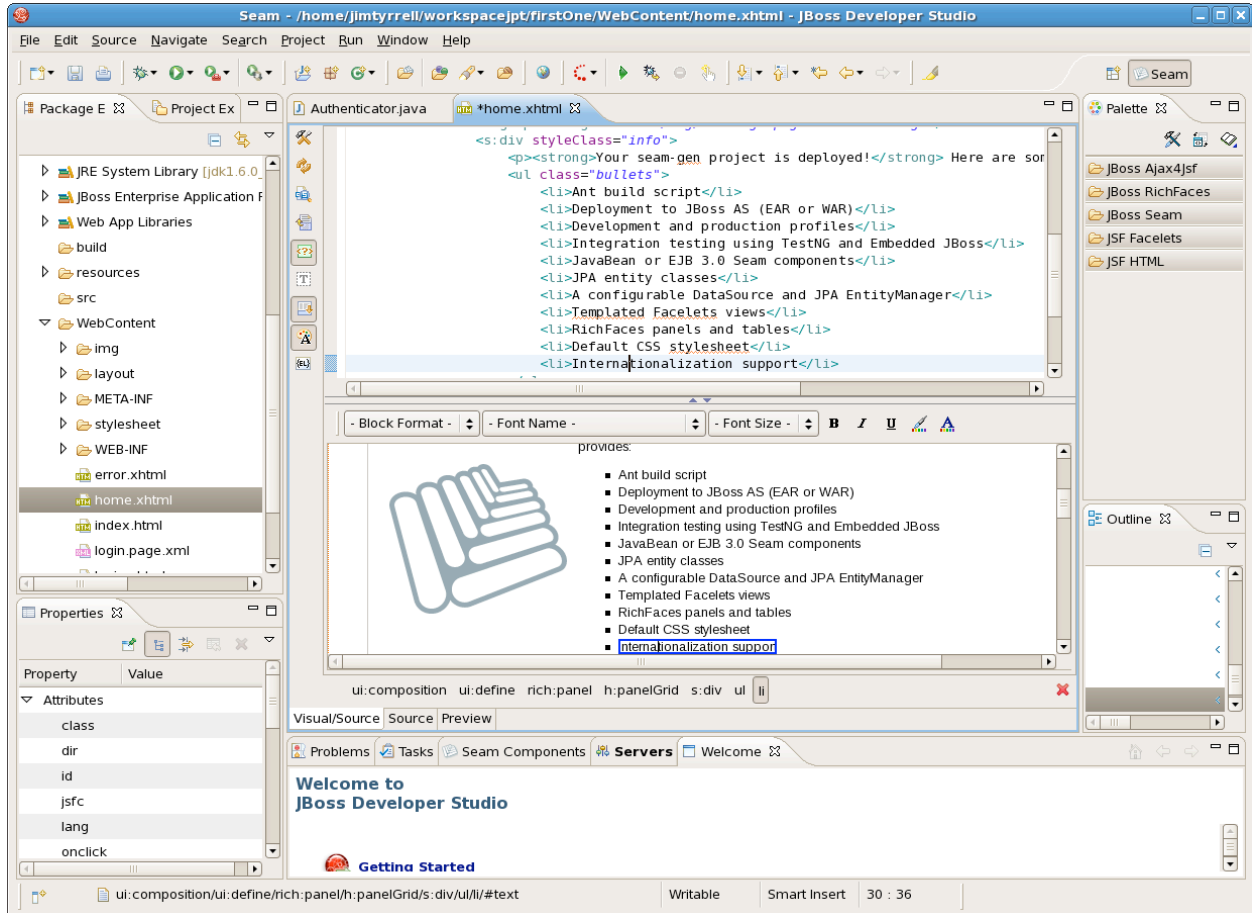
The second file to open is the home.html file (firstOne -> WebContent -> home.html). A second way to find this file is to use the menu at the top of JBDS. Go to Search -> File and in the File Name Patterns: put a "home*" w/o the quotes in the box. In the bottom pane of the IDE you will see a result of the search you can double click on the file there and it will be opened for you. There are various other ways to search for assets in your project. Feel free to search for other things as you work your way through the lab.

When you double click on the file opening it, did you notice the two paned editor. On the top half is a source code view and on the bottom is JSF rendering of what that page will look like when rendered in the Web Browser. This makes it very easy to click around and either look at the rendering or the source code view and make changes in ei-

ther. Click around and see how the cursor is updated. It should look something like this:



Leave this page open and we will come back to this in a second. At this point your JBDS instance should look like this: Authenticator.java is open and home.xhtml is open and maybe you have moved the cursor and clicked near the last bulleted item and noticed how both views are at the same place ie they have scrolled automatically to the same place. Do not make any changes just yet, we will get to that in a second.

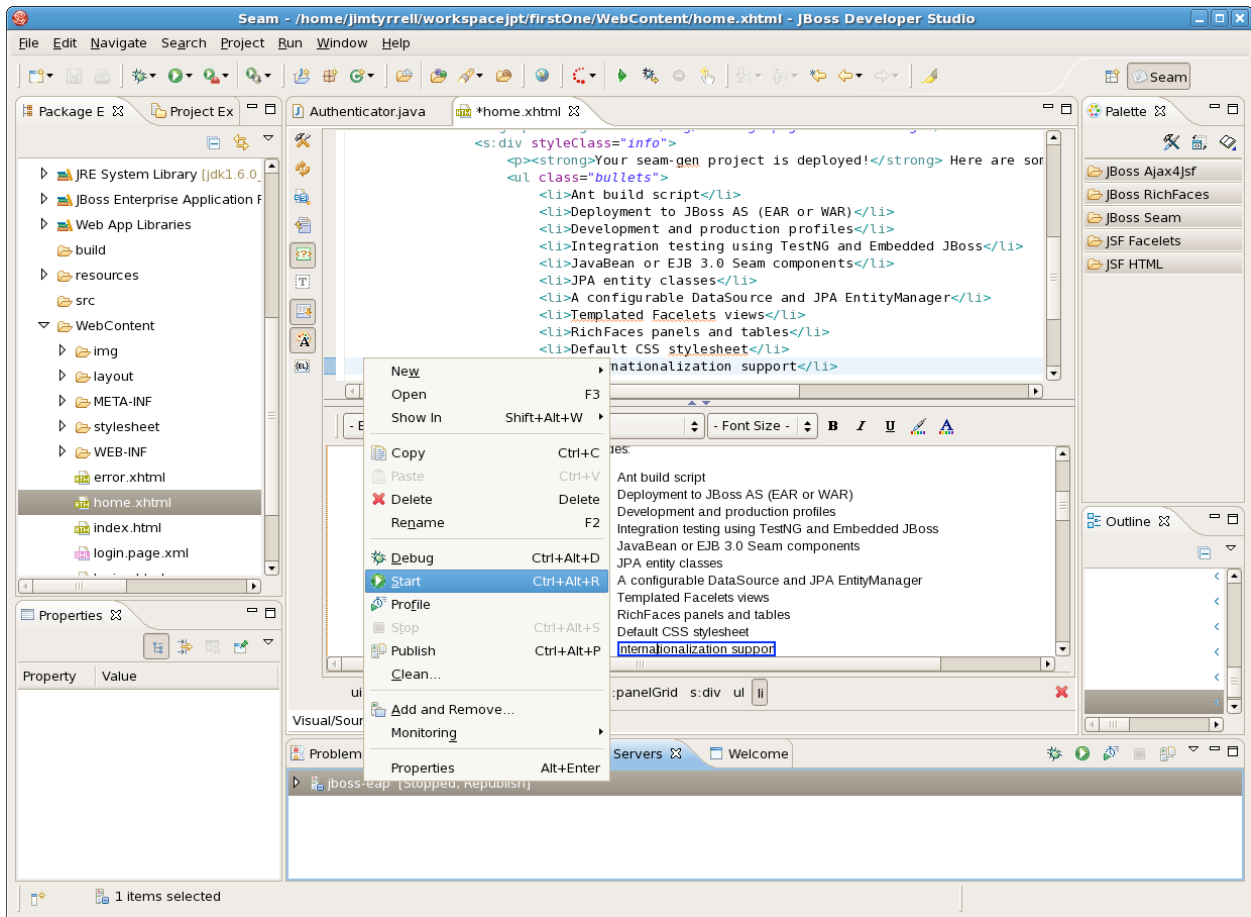


You have now completed this lab. You have an idea on some of the files that were created for you, where they are placed, how to get around the project view, and you are now ready to start the server in the next lab.

Lab #5: Starting the Server

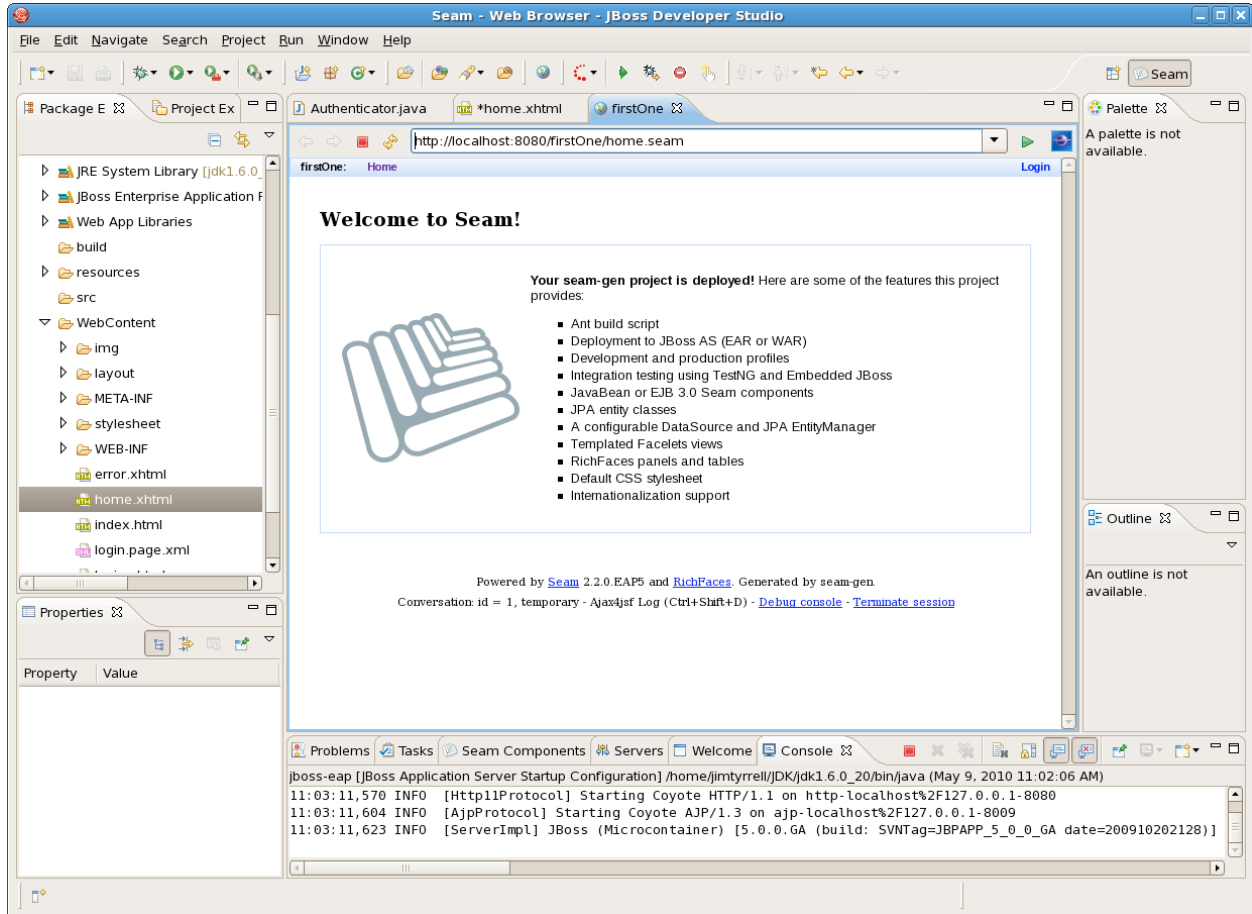
Start the Server

The easiest way to start the server is to go to the bottom pane in the middle of JBDS and click on the tab: “JBoss Server View”. Then either right click on the jboss-eap and select start. Or highlight the jboss-eap line and click the green play button that is enabled once you have highlighted the server runtime you wish to start as seen below. You are also able to view the status of the deployed projects as to whether or not they are in sync with the project view you are working with in JBDS.



In the bottom Console you will see a last line that indicates that JBoss has started in about 30 seconds or so.

Clicking on the Earth looking icon (just below Window and the Help in the main Tool Bar) Left most icon in the below screen shot. This allows you to open a web browser inside of JBDS. Please click that icon. Once you do please type the URL <http://localhost:8080/firstOne> into this newly opened tab. Then hit either the play button to the right of this bar, or press enter and you should see the Generated Seam Application rendered in this view as you see below:



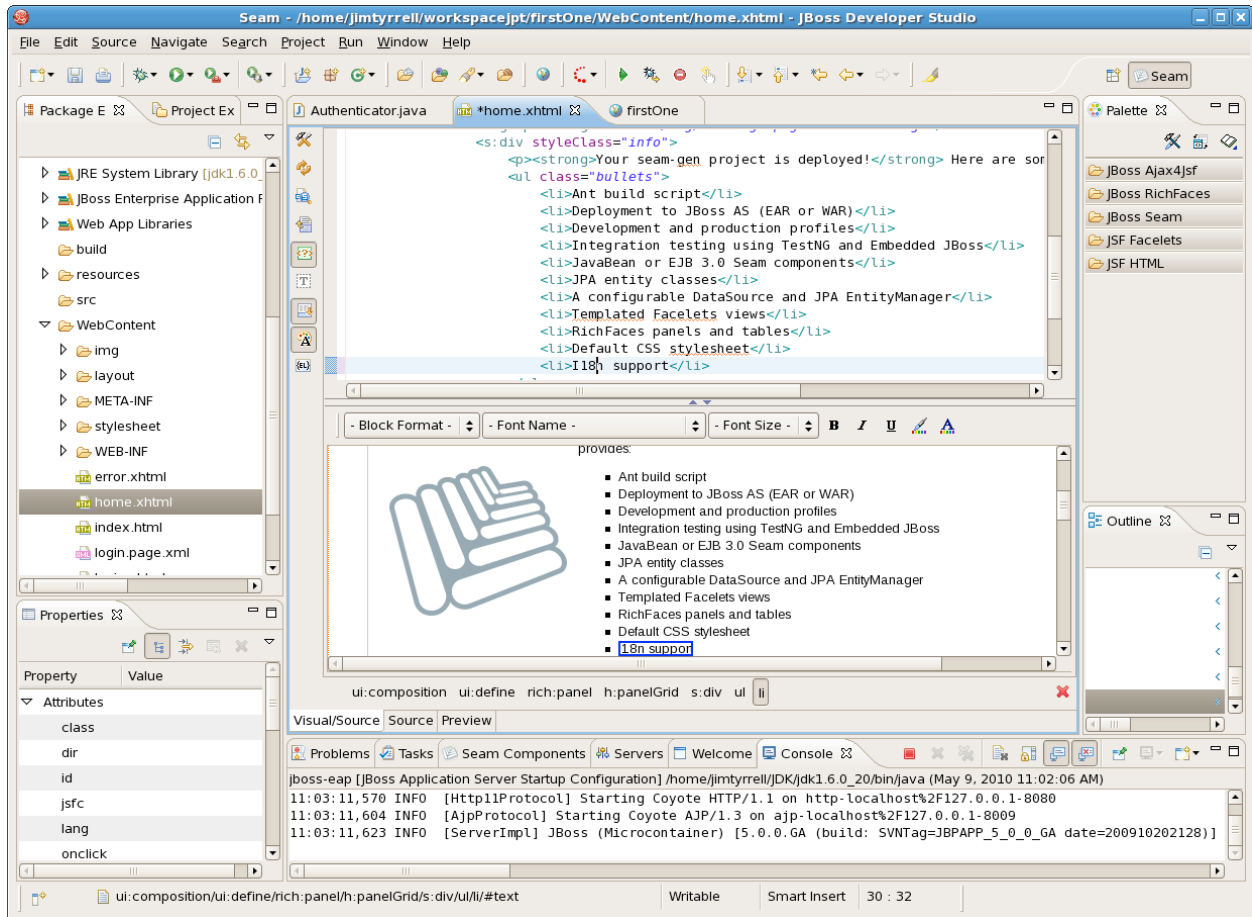
You have now completed starting the application and viewing it in the embedded web browser. Instead of starting the server this way, you could have right clicked on the home.xhtml file and selected the Run As -> Run on Server and worked your way through the wizard to deploy the file. This type of deployment or startup is not needed since JBDS will watch for changes in your Application during development and will push them to the running instance.

You have now completed this lab.

Lab #6: Editing Your First Page

Exploring Dynamic Publishing

At this point you have a web browser tab open that is rendering the home.xhtml file for you and another tab that shows the home.xhtml file. Notice that in the last bulleted list “Internationalization Support” is there. Go to the tab with home.xhtml open and change “Internationalization” to `l18n` the common way of abbreviating that term. Note that you can edit it in either of the two panes, and that they both stay in sync. Also notice that at the top near the menu bar on the left hand side the disk representing saving this artifact, editing this file activates the control and allows you to save the file. You can also use the various key strokes to save the file. Make sure you save the file so it can be automatically published to the embedded instance.



Switching back to the browser tab allows you to click the refresh icon (the two arrows in a circle to the left of the URL) to see you change published in a few seconds. This allows and empowers your developers to create/edit/view application look and feel changes very quickly and rapidly speeding up application development.

This lab is now completed.

Lab #7: Generate CRUD Application

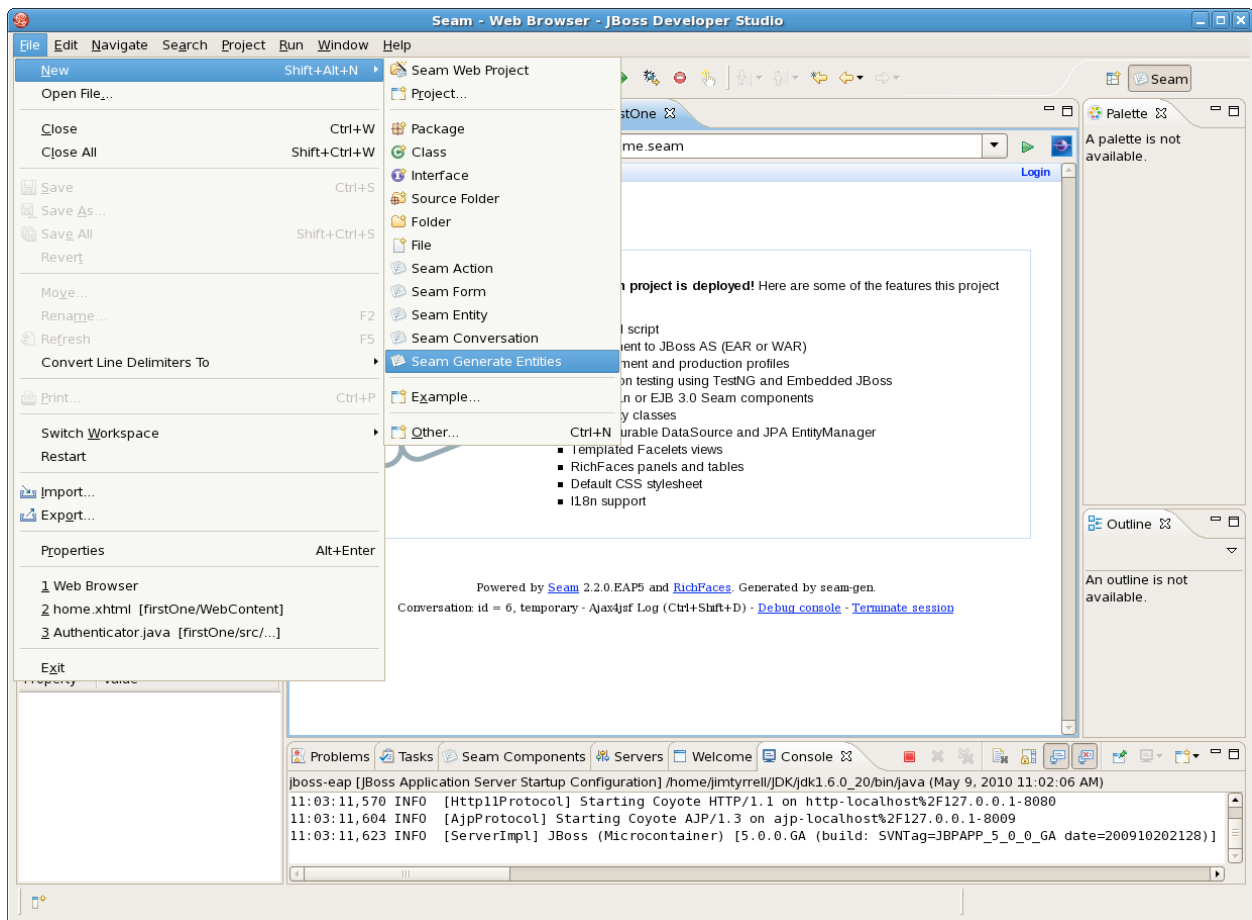
CRUD Application in Seconds

CRUD - Create Read Update Delete.

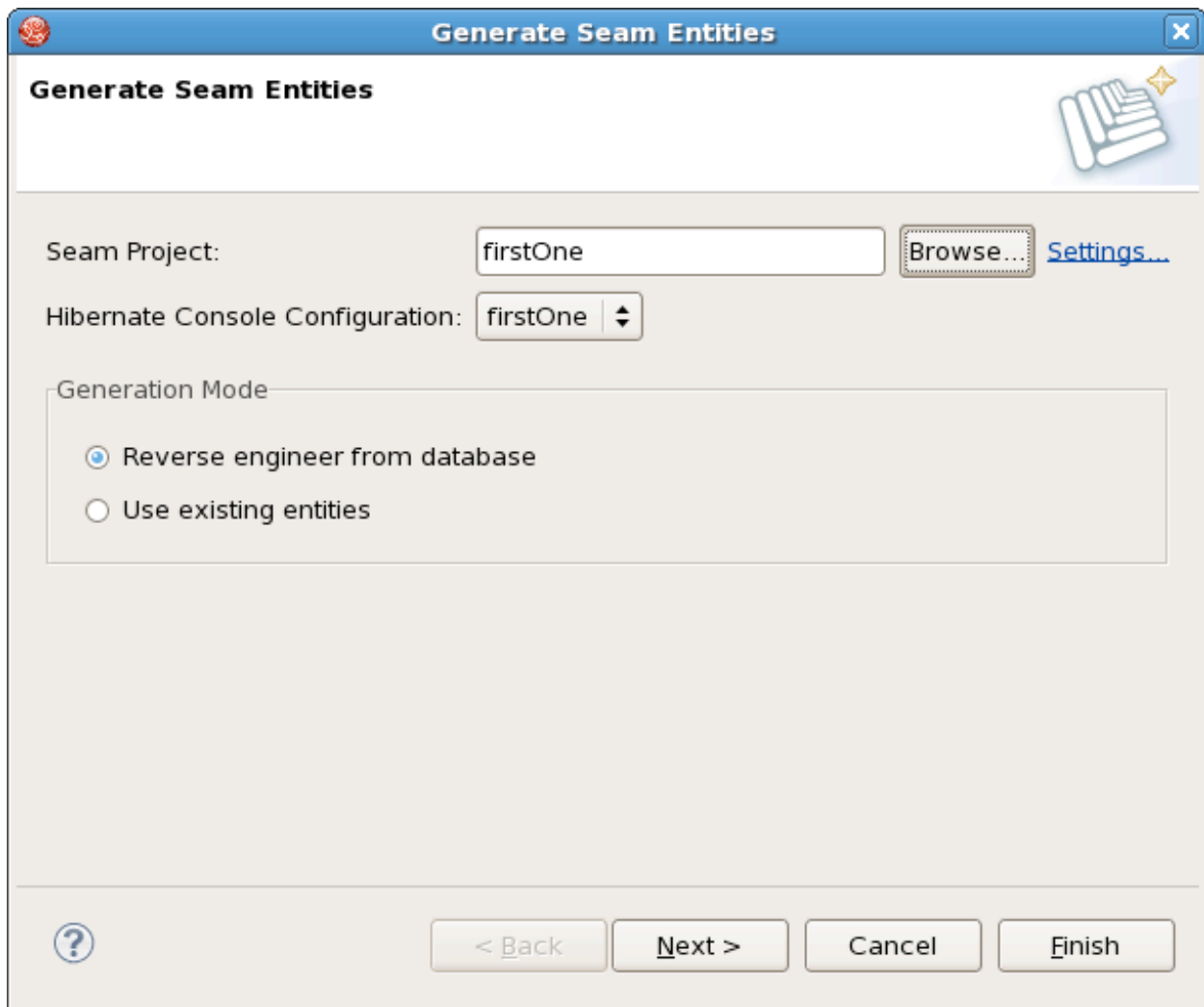
At this point you have spent time getting comfortable with the various parts of JBDS and starting your Seam application. For developers the above steps will become second nature to you, and you will very quickly jump to creating the CRUD application in just a few minutes once the above lessons become second nature to you. The benefits of this CRUD in minutes are: Rapidly prototyping an Application, playing with the data model to identify weaknesses or errors early on, and getting out of the office on time or making that vacation.

Generate Entities

With the correct database configuration from when we created the project firstOne we are now able to select File -> New -> Seam Generate Entities as shown below:



The “Generate Seam Entities” Wizard shows up. Make sure that Seam Project: is set to “firstOne” and that the Hibernate Console Configuration is set to: firstOne as seen below, click Browse or the drop down to select the right project name, and make sure that “Reverse engineer from database” is selected as shown below:



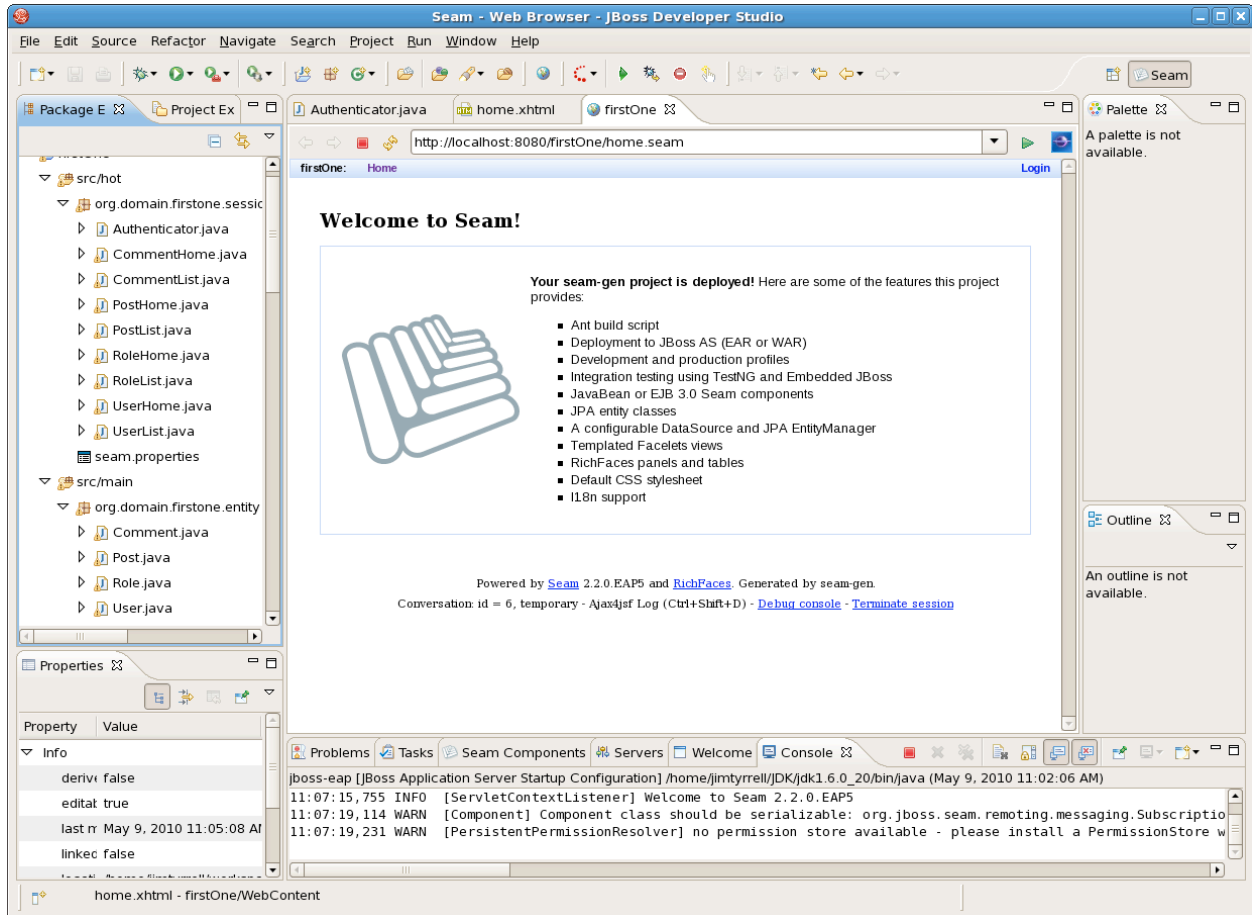
Select “Next”

The next page allows you to pick the Tables from the database that you wish to generate. For the purposes of this lab nothing needs to be selected as shown below:



Click "Finish"

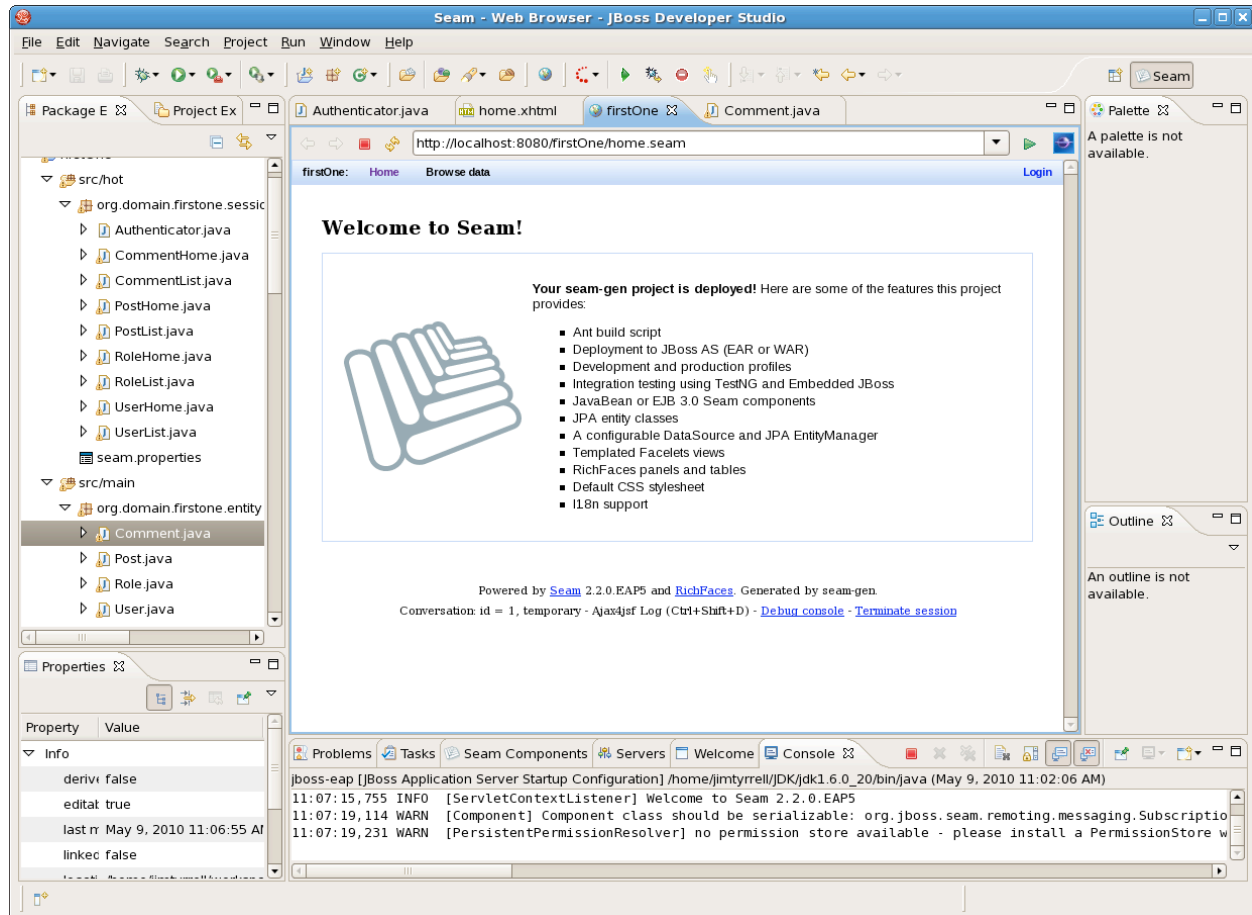
You now have a series of new artifacts (Comment.java, User.java, Role.java..., UserList.xhtml....CommentList.javaCommentHome.java.....) that will show up in your project. Notice the new packages and files created in the src/hot -> org.domain.firststone.session, src/main -> org.domain.firststone.entity, and the new files in the WebContent folder. In the below screen shot you can see many of the new hibernate objects that were created for you.



Lab #8: Running the Generated Application

Running the Application

After waiting for the application to publish you will be able to refresh the button in the browser and see a new Browse Data link menu item on the screen. This is what it should look like:



Now for some activities to explore what was created for you:

1. Click on Browse Data -> Role List:
 1. Click on "Create Role". Notice a login screen comes up, type in "admin" per the instructions on the screen and login. We will deal with this in a second.
 2. Put the Cursor in the Rolename box, and tab out of it. Notice that the Not Null constraint from the Database was inserted into the application automatically w/o us having to write any code.
 3. Type in the Rolename box "Test Role" and select Save. Note the Test Role and the Automatically Generated RoldID on the conformation screen. Select "Done" to save the change.
 4. Note that the new role has been added to the list.

5. Click on "View" to the right of the Admin Role name in the Roles List. Notice that the User List members of admin is already mapped to the role admin. This relationship was injected and inherited from the database and made available to us without any coding.
 6. Feel free to click around the Role pages a little more. Also feel free to edit the files RoleList.xhtml or RoleEdit.xhtml in the WebContent folder to change column names etc.
2. Next Select the Browse Data -> User List
1. Notice that we have a lot of Users in system and that the User list is already pageable without writing any code. Note however that this list requires a browser refresh and is not a AJAX web call. We will address this in a future lab.
 2. Click on "Create User"
 3. Click on Save, notice it is not hot, enabled, is virtually grayed out, and not selectable, or whatever definition you want to use. The foreign key mapping for the Role is required to be selected below before the save button is enabled or made available to the end user. You will notice the * in the bottom tabs on any value that needs to be selected, or a Foreign Key mapping not null that is enforced in the user interface.
 4. Click "Select Role" in the Role tab below, to the right of a role you would like to apply to your new user, and notice you are brought back to the page to input a username and password. Fill in those, if you want to tab off an empty Username or Password and notice the NOT NULL constraint was enforced for us again with an AJAX (no browser page refresh) call. Use a password of "password" our system is very secure :), and any Username and Password you would like.
 5. Click on "Save", review your changes, and then click done, notice that new user name in the view.
 6. Make sure your new user was added, select the "Lst Page" in the lower right corner of the browser.
3. Next Select the Browse Data -> Post List
1. Click "Create Post" and notice the "User" Tab has a Foreign key relationship that is enforced. Select a User
 2. Then click the calendar icon to the right of the Postdate and notice the Calendar/Date chooser that pops up enabling you to pick a date. Select a date and fill in a Posttitle and a Postbody. Clicking on Save and Done when you are happy with your changes.
 3. Click the back button the arrow. Click the refresh icon. Notice that page did not ask you resubmit the data with a popup box. Seam is managing this for you. Click on the "Save" button and then the "Done" button when you are completed with this. Notice you did not end up with double posts, or some of the other odd things that will happen when hitting/selecting/using/enabling the back button.
4. Select the Comment List
1. Click "Create" Button. Notice that two tabs (Post and User) at the bottom are required to be filled in. Select the User and the Post you will be commenting against and then select a date, title and body as you did before.

You are now done with this lab. Notice that in just a few minutes you have a working CRUD application that leverages many features of Seam, uses constraints from the database, and injects behavior into the pages.

Lab #9: AJAX in Moments

Why AJAX

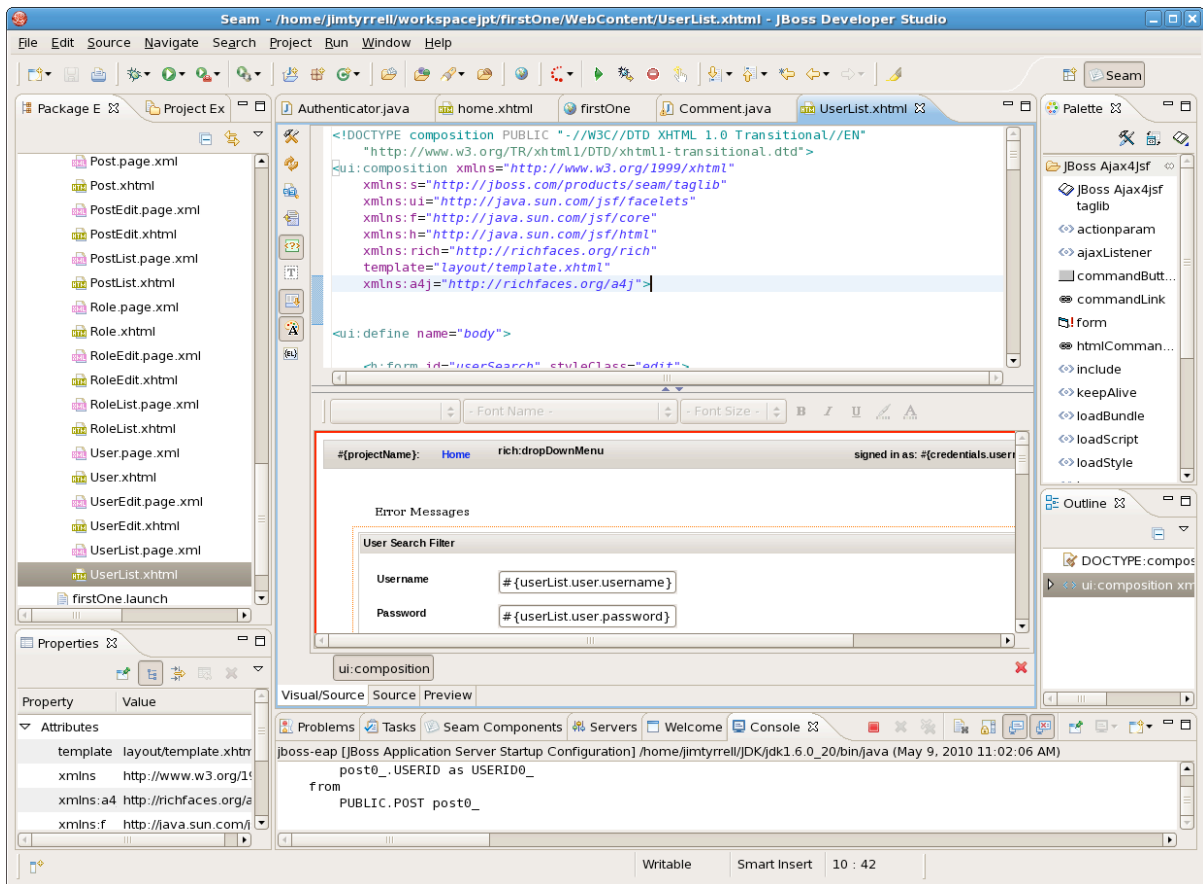
AJAX stands for Asynchronous Javascript and XML and it is this idea that you should not have to have browser refreshes of the entire page to enable the updating of content. This requires the user to wait while the page loads. It would be nice to enable the User List to not require a page refresh every time you switched pages you wished to view. This lab will show you how to update the generated facetlet and have it show dynamically the search as more letters are added in the username search box.

Adding AJAX to the User List

The Steps are:

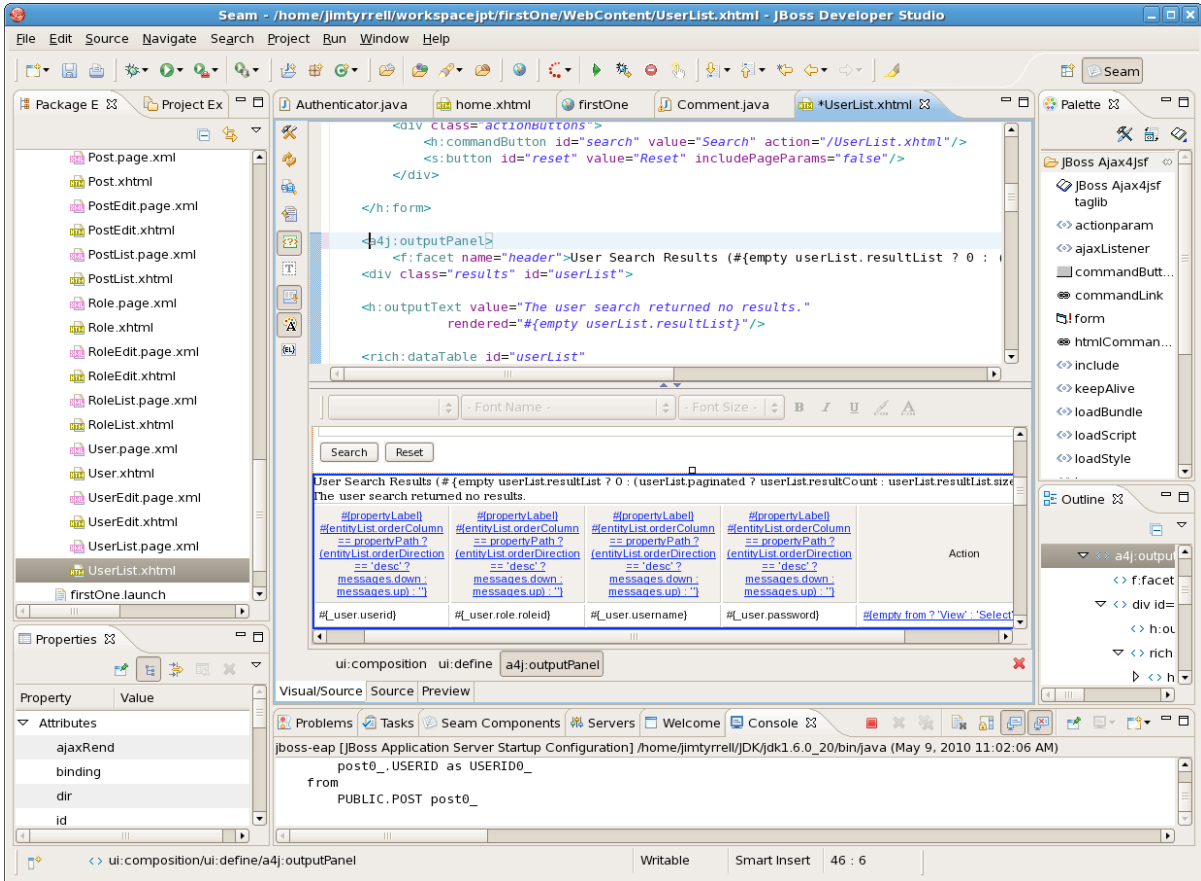
1. Open the UserList.xhtml

You will have to add in the line `xmlns:a4j="http://richfaces.org/a4j"` as shown below in the `ui:composition` tag below:



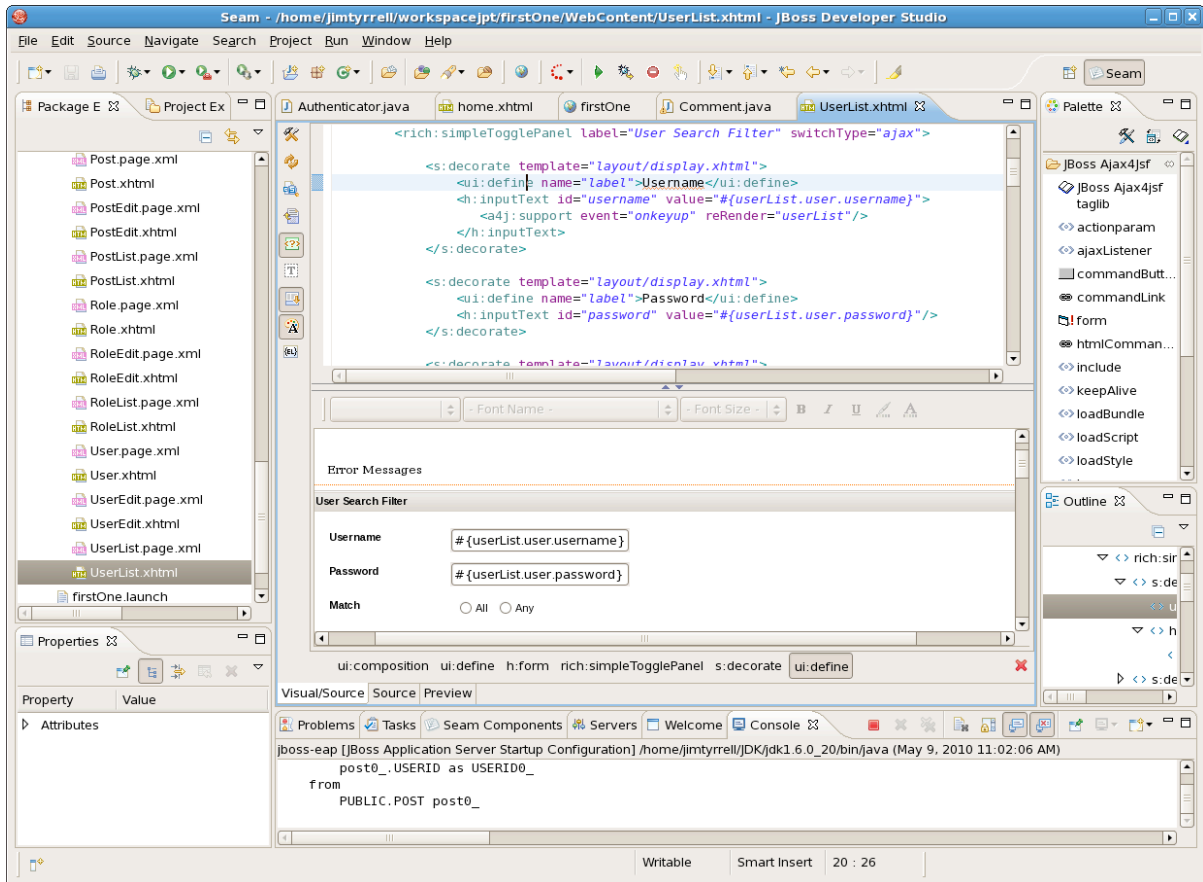
Notice this adds in the AJAX library for RichFaces.

- Find the rich:panel and replace it with a4j:outputPanel. Note you will have to do this twice ie the opening and closing xml markup. Remember you can click in the bottom rendering to move the cursor to the correct place, by clicking below the Search Button. The first change is shown below:



- The last step is to update the the username text box to add in Ajax functionality. You will do this by clicking the Username text box. This will move the cursor to the correct place in the file. Next add in a closing `</h:inputText tag>` and removing the trailing slash in the `<h:inputText />` remove the slash. Next add in the line

`<a4j:support event="onkeyup" reRender="userList"/>`. All the changes are shown below:



- Save the file, make sure there are no errors. Look in the Problems tab in the lower middle section of JBDS. More than likely you will have 50 some showing up, that is okay. They will all be warnings in yellow and will not effect what you are trying to do. Anything in red needs to be addressed. Wait for deployment and then test out the new Ajax functionality by searching on Adm or user in the Username text box. See how the richtable is updated w/o having to click the search button.

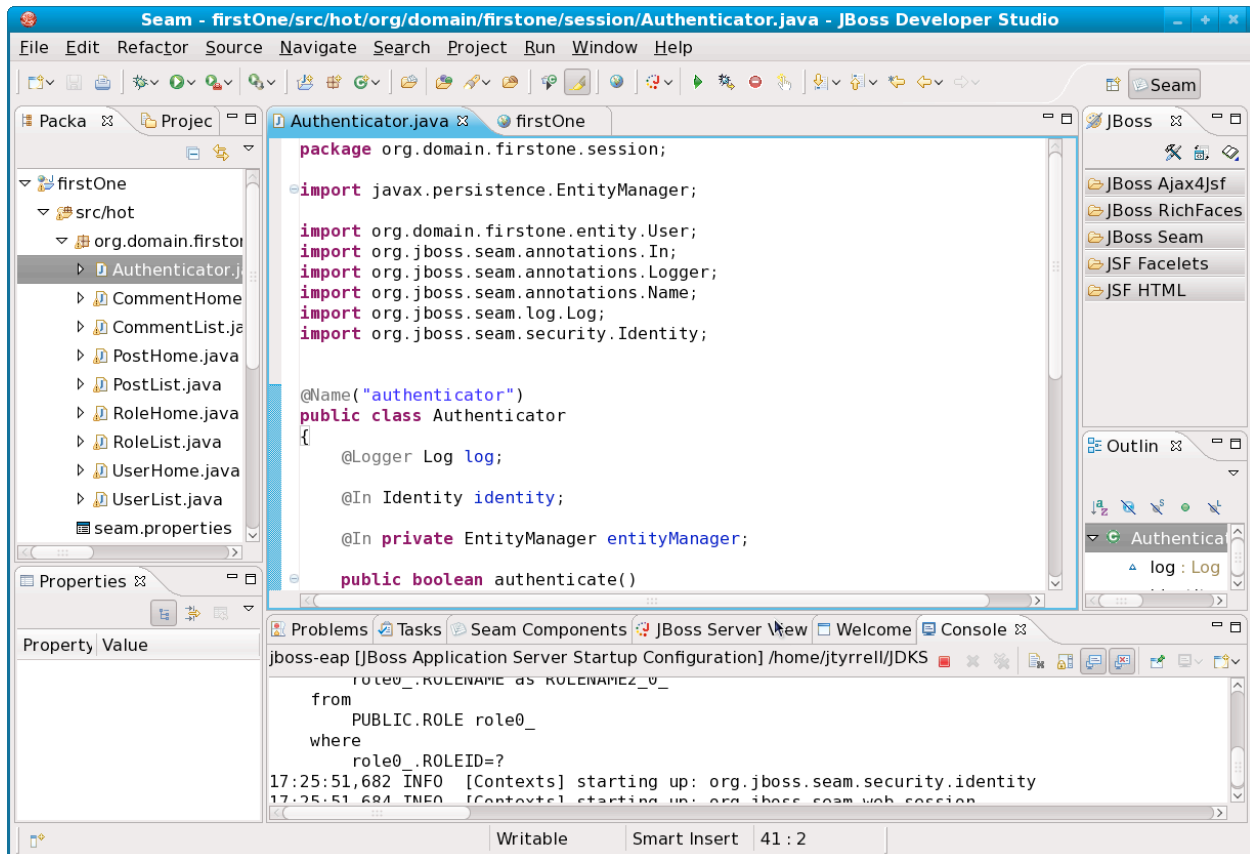
Lab #10: Seam Security

Seam Security

There are many things to secure in a Seam Application and various ways to go about this. The first of these is updating the Authenticator.java to use user and password information to inject a role into your web application.

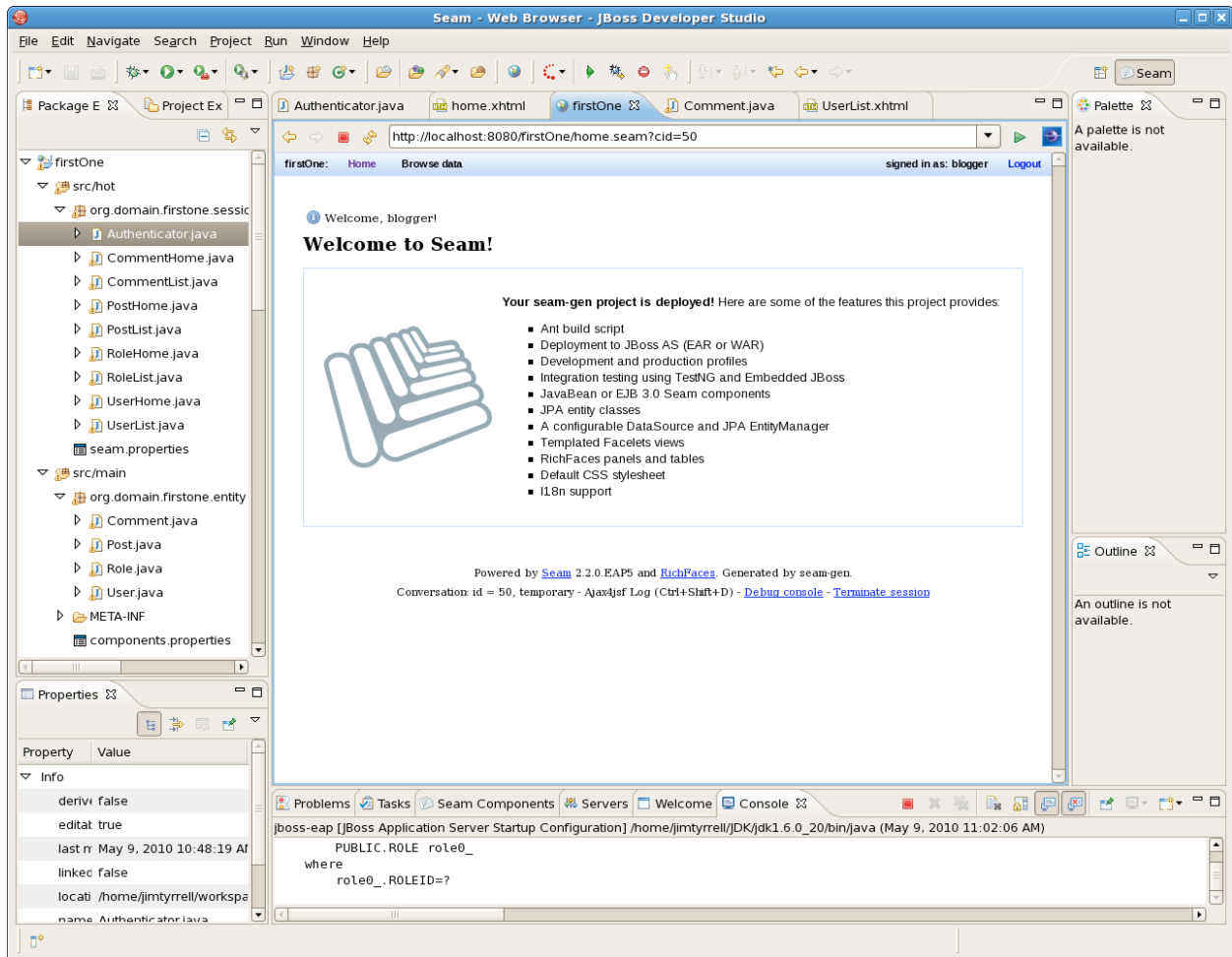
Update the Authenticator

Locate the readme.txt file and look for the third entry in it which includes an update to the firstOne -> src/hot -> Authenticator.java file that you can cut and paste in its entirety into your Authenticator.java that may already be open. If it is not open please open it. When are done cutting pasting the file should look like this:



Note the EntityManager this is an Object that is injected into the runtime by the Seam Framework and it gives us access to the database. In the authenticate method we check against the username and password, and if we find a match we insert the Role into the Identity context. This is an example of what Seam calls Bijection.

At this point you should switch over to the web application and wait for the application to finish being deployed to the embedded EAP instance. Try out a few usernames and passwords and see how the username shows up automatically in the upper left hand corner and how you are welcome in the screen below:



Feel free to try out any of the usernames you created in any of the earlier labs.

At this point you have explored updating the `Authenticator.java` file. Of course a lot more logic and code can be placed into this file, but this gives you an idea how to inject a role into the context of a logged in user.

Lab #11: Security Continued

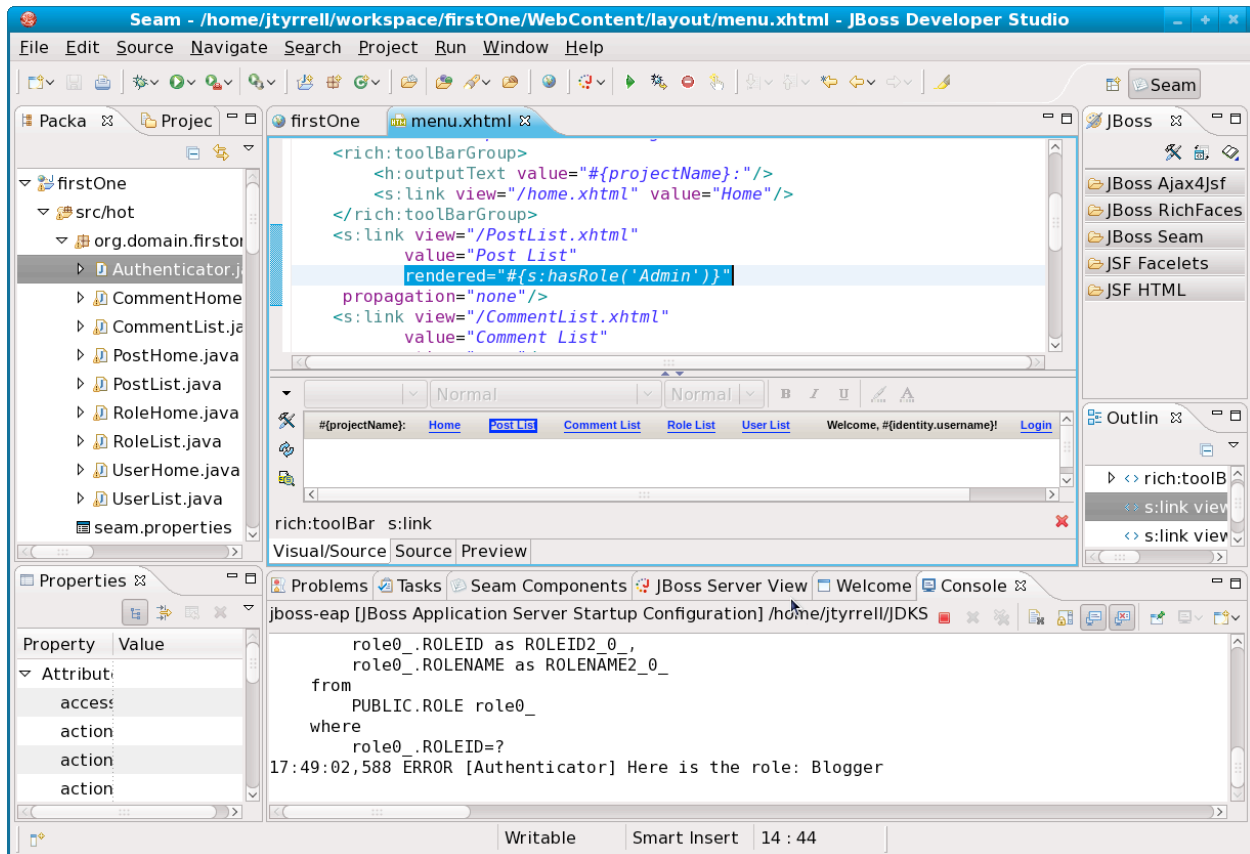
Rendering Tag Using Role

Occasionally when you are working on an application you wish to show various links and other components to a user based on the permissions that they may have. Seam also enables the application developer to lock down directories and other things like pages, but that is a topic that is outside of the scope what is possible in the time allotted for this lab. Lets look at how to conditionally show items in the menu.

Selectively Rendering Menu Items

You can conditionally show the various menu items across the top of the page based on the user logged in role. This will be a contrived example to show what is possible, however it shows the idea on how to lock down a particular web page component.

Open up the firstOne -> WebContent -> layout -> menu.xhtml file. In the readme.txt is the fourth section that has a rendered tag in it. Cut and paste that line and place it in any of the <s:link tags in the menu.xhtml file as seen below, you are welcome to configure each link separately as you see fit:



Now go back to the web browser tab and see how the link you secured does not or does show up based on the user you logged in as. Remember the usernames/passwords are (admin, password), (blogger, password), and many others. Login and Logout a few times to see the various behaviors. Also you can view the Role List to see the available Roles and Users that are mapped to a particular role.

You have now completed this lab.

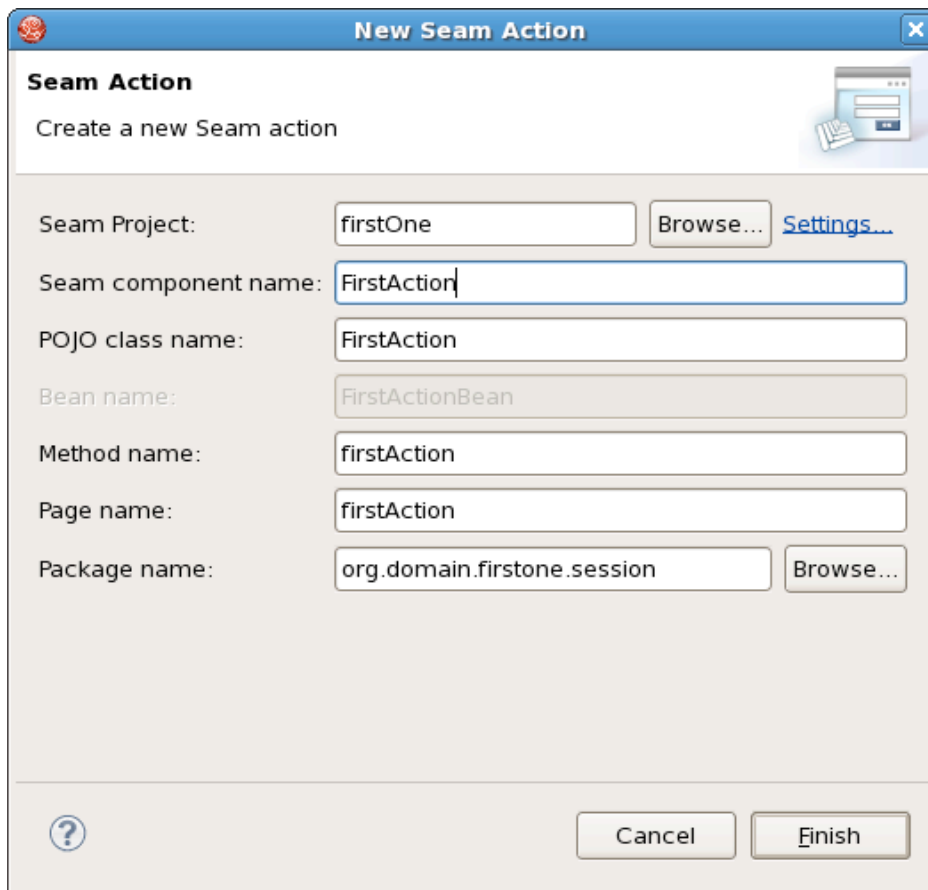
Lab #12: New Action

Create a New Action

Of course all applications need the ability to add new pages to the application. This is accomplished by creating a new action. This results in the creation of a New Action java class, a New Action xhtml and a New Test Case in the other Project firstOne-test.

How to Create a New Action

Select File -> New -> Seam Action. This will pop up a new Action wizard box where you can type in the name of the Seam Component Name. For this example you can use the name: "FirstAction" as shown below, you might need to click the browse button to find your "firstOne" project:



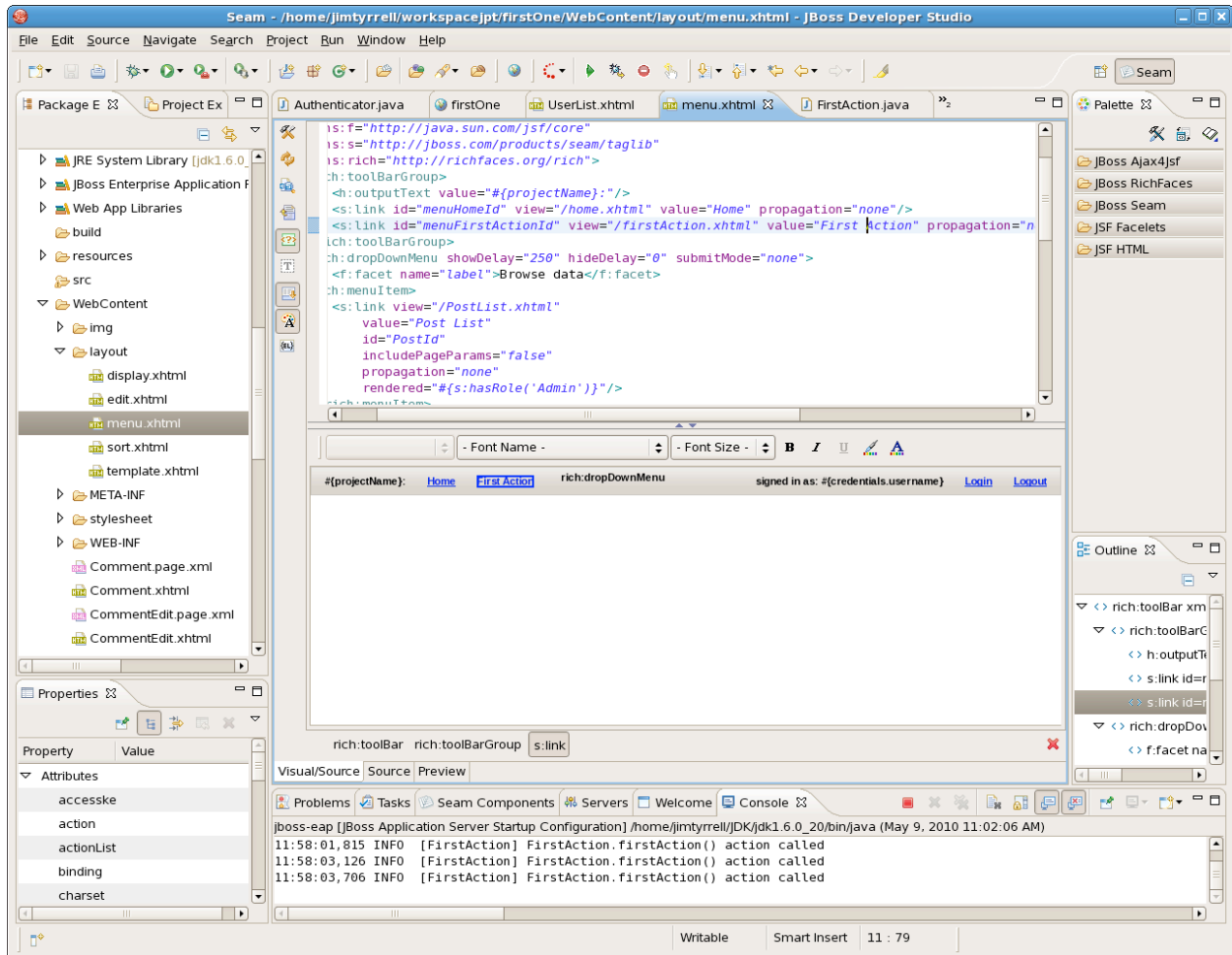
This will result in the creation of the FirstAction.xhtml and the FirstAction.java file. Open the two files. Do not forget about the search functionality we explored at the beginning of the lab.

In the FirstAction.java file add to the following line, you can find it in the readme.txt file:

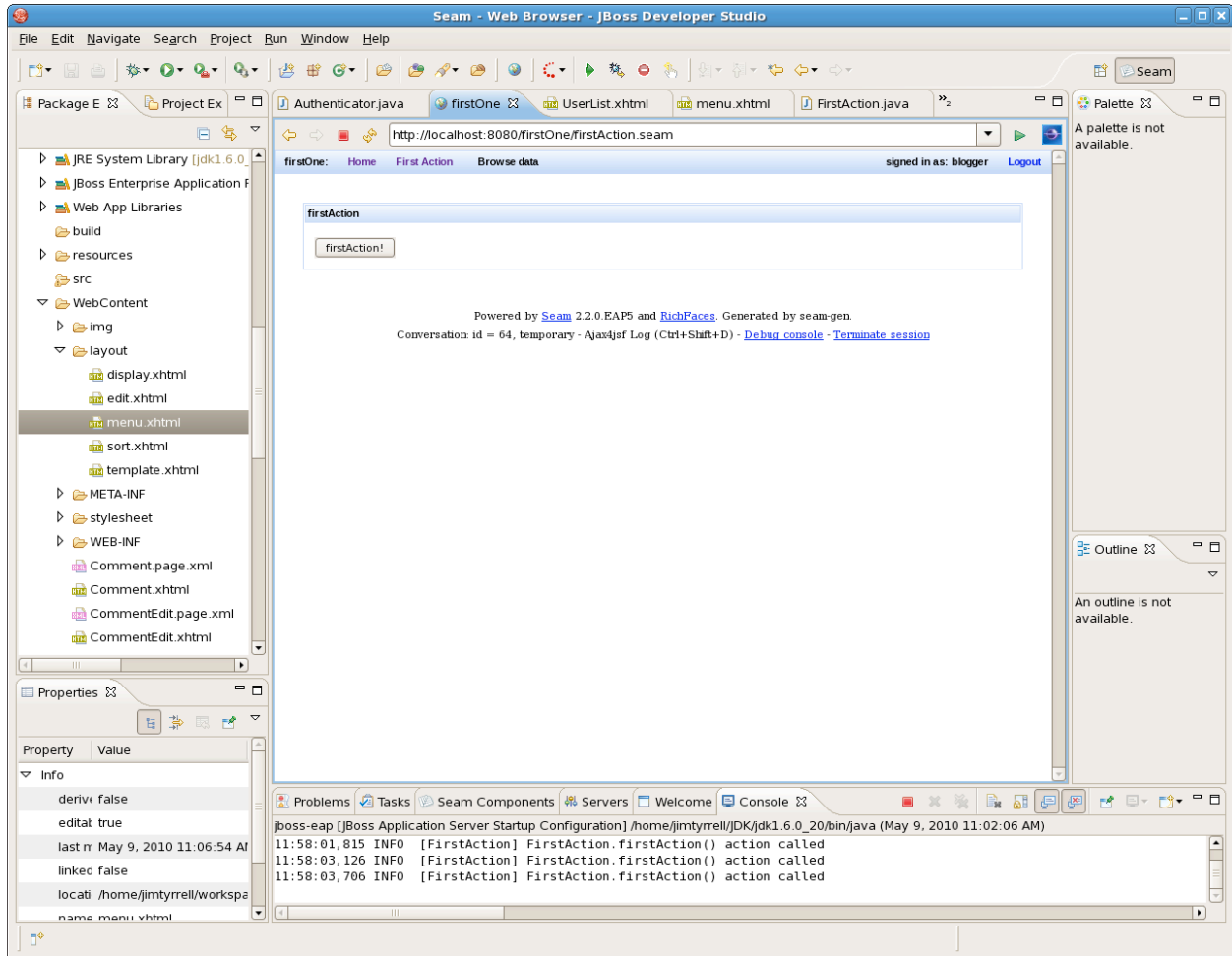
```
statusMessage.add("firstAction called now: " + new Date());
```

You will notice that to the left of the line is a red/yellow warning message. Double click on this and then make sure you select the - import 'Date' java.util from the pop up box to correct this error. This is an example of JBDS assisting you with writing the code. This will add in an `import java.util.Date;` at the top of the file.

Now open up the firstOne -> WebContent -> layout -> menu.xhtml and add in a link to the firstAction.xhtml file. Again you can cut and paste this from the readme.txt file. It should look like this, it an be added right below the existing Home link that was created:

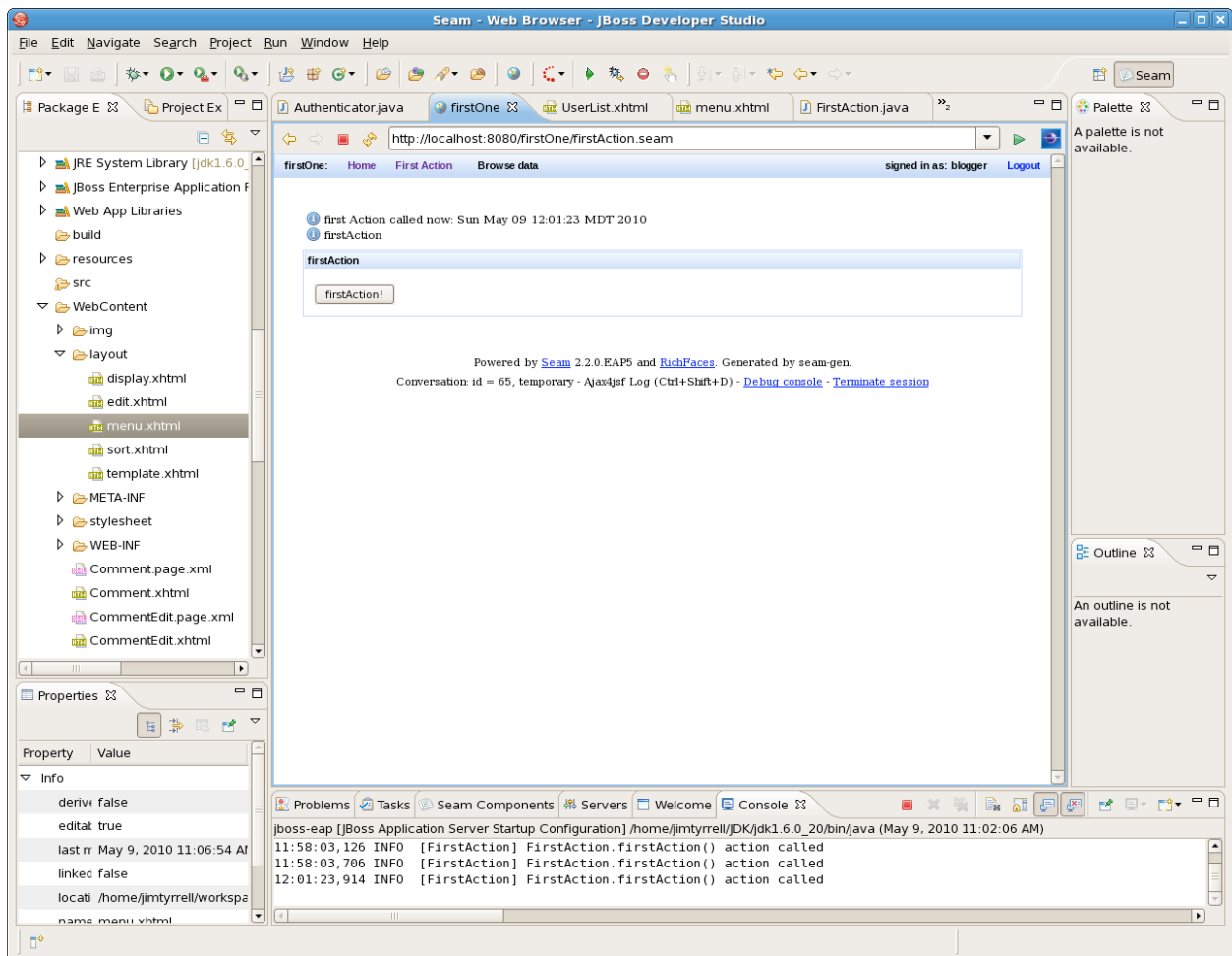


Save out all the files and wait for your updates to be published. Click the link to the FirstAction in the top menu and then



select the button on the page. Feel free to look at the generated `FirstAction.xhtml` and make some changes to this file and see the updates. Update the button to say: "First Action" in the browser. After clicking first action you should

see something like this:



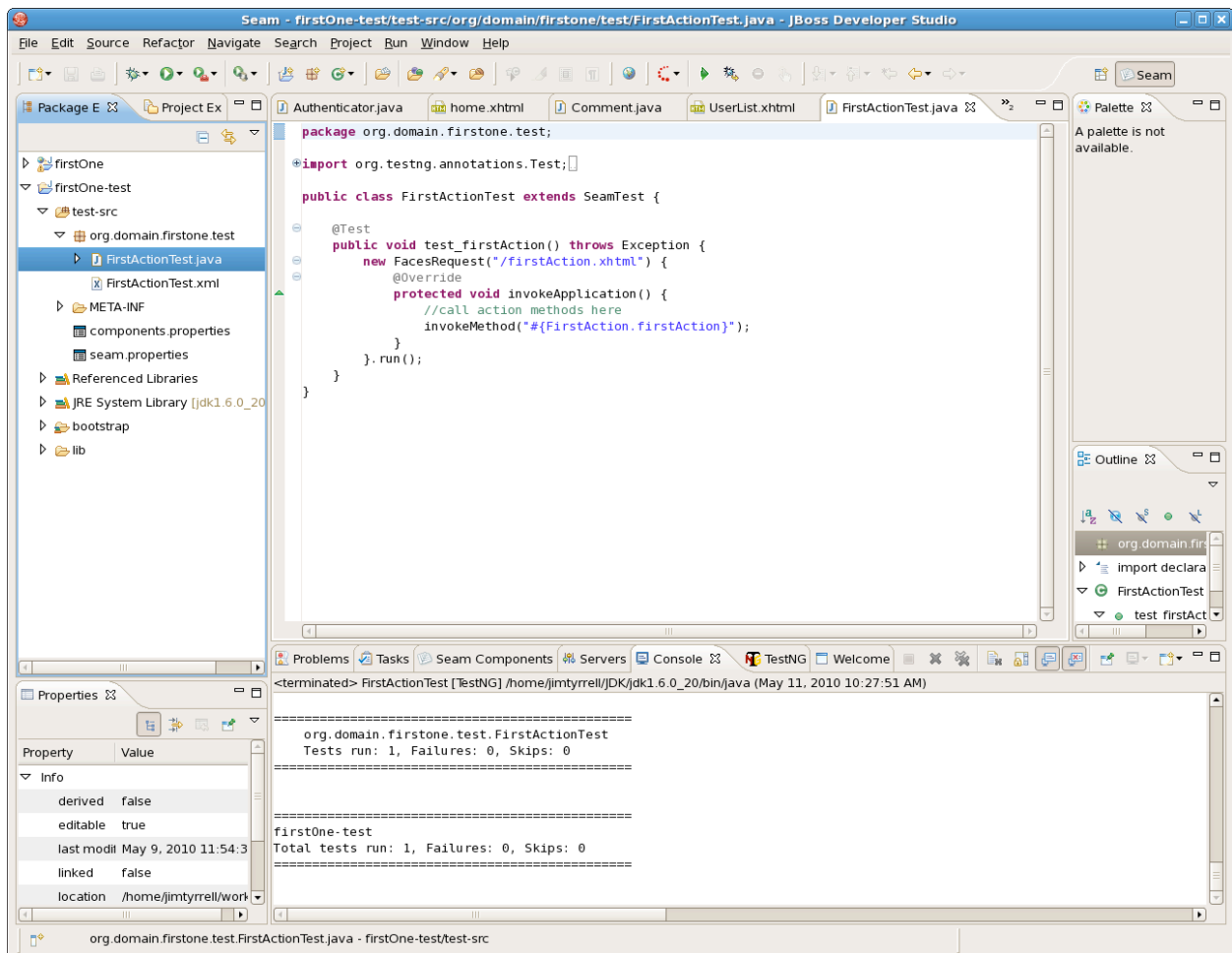
Lab #13: Testing

TestNG

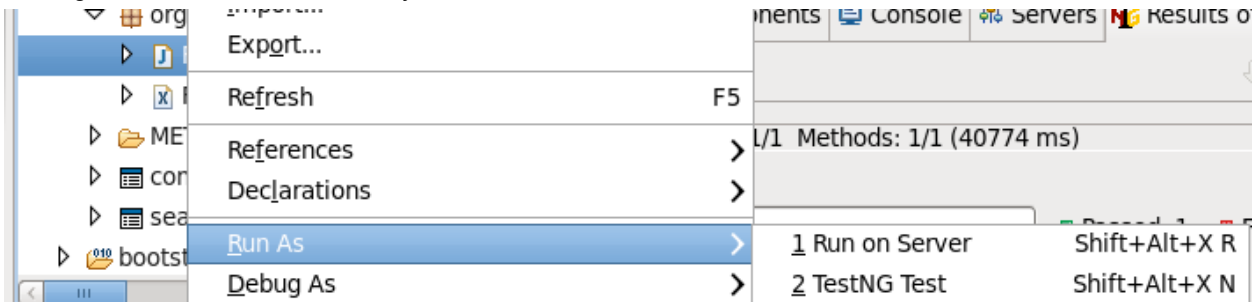
TestNG is included with JBDS and enables you to quickly create and write unit tests to explore functionality with in your applications. These test cases start up a standalone Seam runtime and allow you to emulate invoking your business objects as if you were a user invoking the button click as we invoked from the Web Browser in the previous lab. This greatly simplifies application testing and validation, of course you have to maintain and create the test cases.

Testing FirstAction

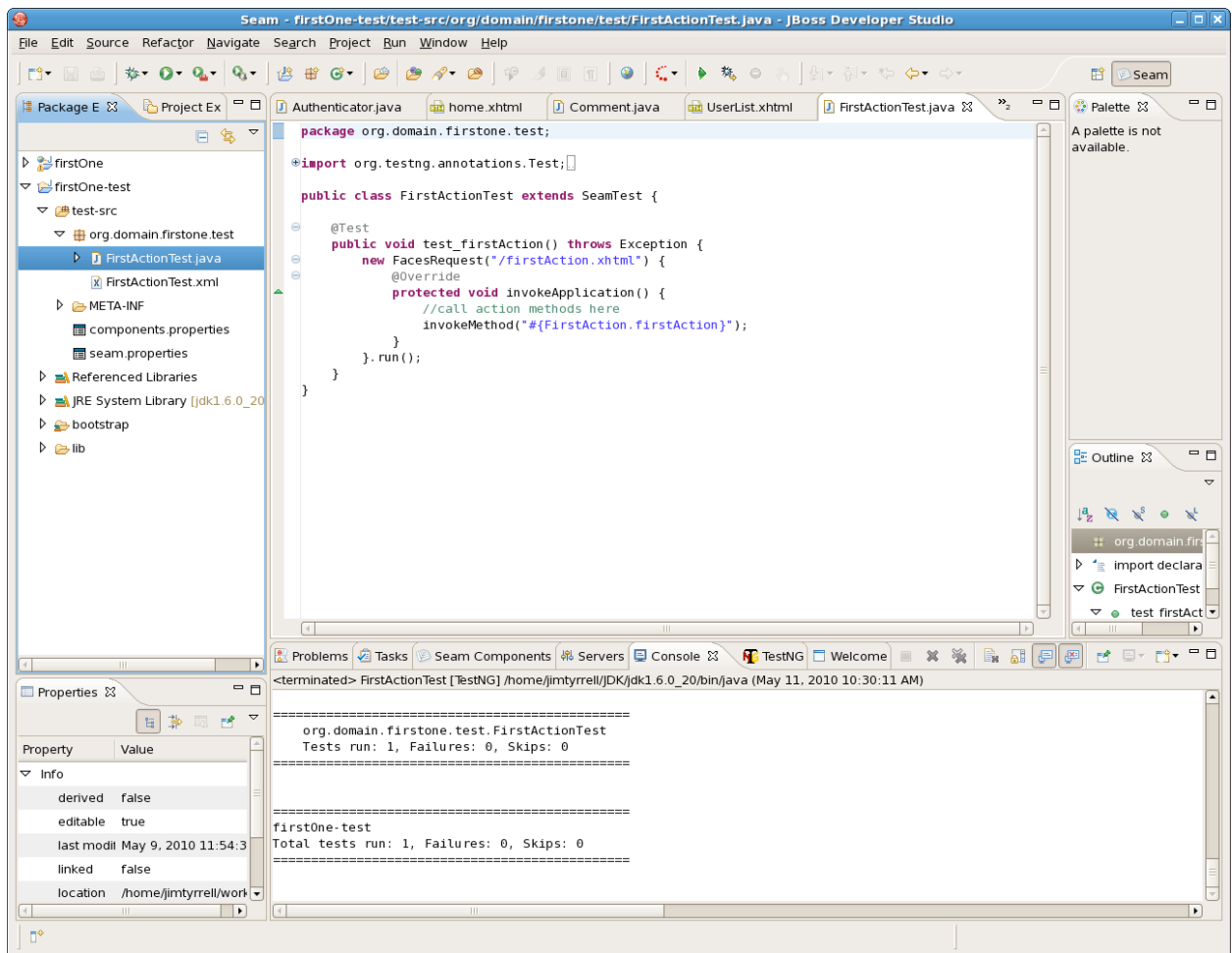
Open up the firstOne-test Project and notice the FirstActionTest as shown below:



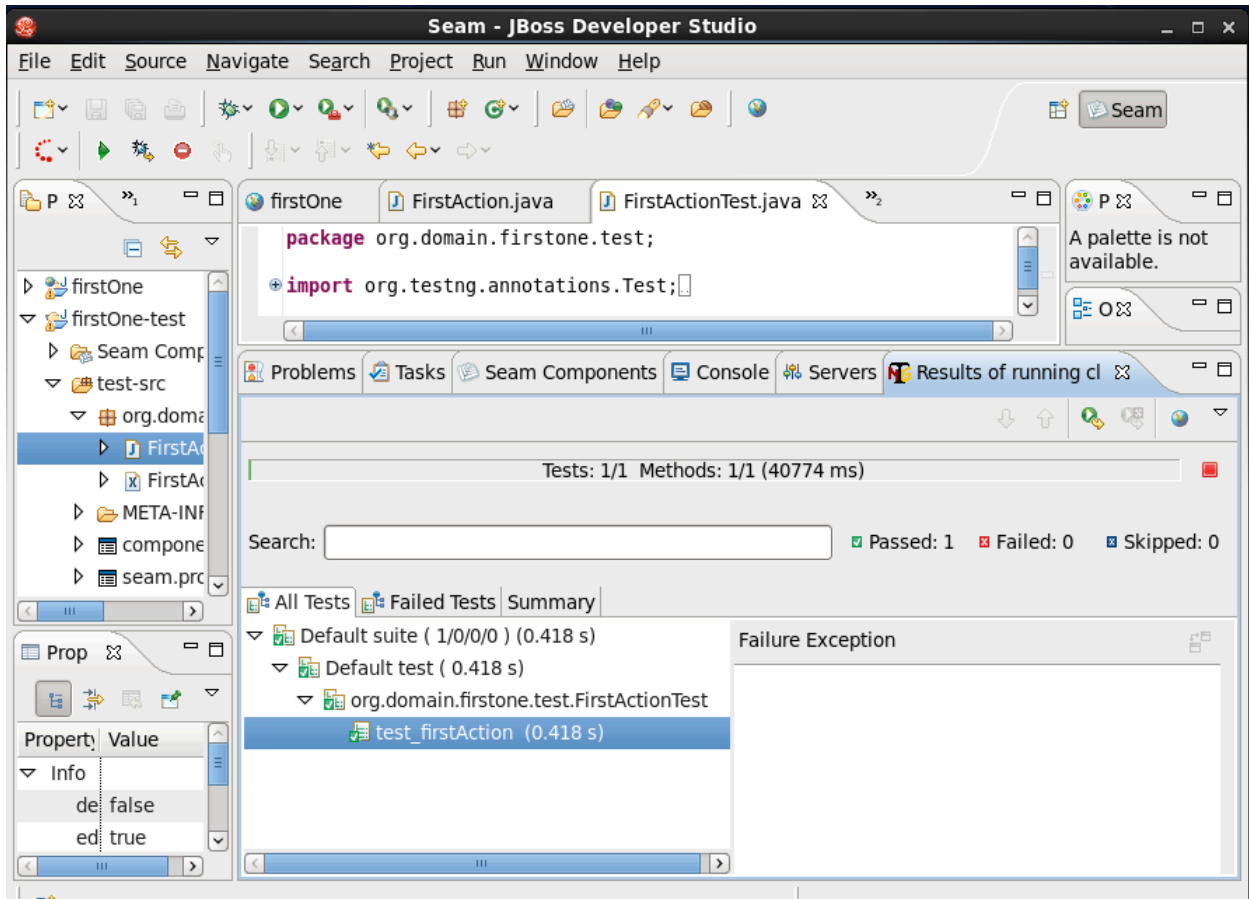
Next right click on the FirstActionTest.java and select Run As -> TestNG Test:



Click run, and after waiting you should see the following output:



And the TestNG tab should show a green bar showing success, as shown:



Congratulations you have now learned how to test a seam application, this is a very powerful feature, and one that we have only touched the surface of.

Lab #14: Style

Edit the Theme

Find the theme.css file in the firstOne -> WebContent -> stylesheet -> theme.css. Scroll down to the input, textarea section and change out the line:

```
color: black ; to
```

```
color:red.
```

Save out the file, wait for it to be republished and see how the input boxes text color has been updated to red when you type. If you are familiar with css feel free to make other changes. Can you change the buttons colors? What about border colors etc. See how beautiful you can make the layout with your css skills. Sadly your lab creator has none :(.

Conclusion

What you learned

- How to install JBDS
- How to create a New Seam Project
- How to create a CRUD application in moments
- Explore the various created artifacts
- Learn where files are placed
- How to create a new Action
- How to test the newly created action
- How to inject Ajax into your generated application
- How to work with Hibernate and view the data model