

Mark Stevens  July 19, 2020

A Journey through Java, MySQL, Apache, and the MVC Architecture with Eclipse IDE

A Synopsis of learning how these development tools work together

**The Beginning**

**07/19/20**

This is my memoirs of developing my full stack web development skills. This journey started back in 1986 when I saw my first computer up close and personal. It was during a tour of Pittsburgh Technical Institute before registering for classes in Specialized TechnologyArchitectural Design/Autocad. I was amazed by the huge dual screens, digi-pad(as they were called back in the day) and the whole workstation. Back then you had a table with the pad on it, a mouse, and a menu underneath which needed calibrated from time to time. The leading software packages at that time were called MicroStation and Autocad. I don’t recall the versions, 1.2 or something. I knew right away what my calling was; hooked for life on computers!

When first starting out in my calling, a major engineering firm hired me on. Through the years I dabbled in many different computer technology’s in my free time, including building computers, reformatting hard drives from scratch, building websites with Macromedia products before Adobe bought them, databases, networking, servers, domains, etc,, etc. During that time my interest in learning how everything worked together on the internet(and now in the cloud) became my next passion. When an opportunity arose to go back to college and further my studies I jumped on it! The curriculum path I chose was Associates in Cyber Security. It was during those classes that my interest in databases, Java, Servers, SQL and all things in between peaked again. So I find myself really diving into it again instead of just a hobby, you might say.

I finally have my skills honed enough to attempt a full development cycle of Design, Develop, & Deploy an application to the Cloud. It hasn’t been easy, many frustrating days and nights of learning to pull everything together through online tutorials, books, and classes through the Udemy website among others. Reformatting my laptop several times, getting Maven and Spring installed correctly, where and when to use dependencies, and how to find them. Finally was able to figure out how to properly install Maven also. Most of the tutorials on the internet only get you so close, you need a descent understanding of how everything comes together between Eclipse, MySql, Apache, Maven, and Spring. Going through all that gave me a good foundation to continue my studies. I found the easiest way to install Maven and Spring is to import the folders into an Eclipse project and run the mvn command files from there. Some guys prefer to use the console window pane in Eclipse and navigate to the directory where the files are and run it that way. To check if everything’s installed correctly just right click on a project and go to the “Run As” pane. You will see a bunch of maven options to choose from. I recommend playing around with those in the console window to get a feel for what it’s doing. Also, when you install Apache, sometimes it sets the HTML/1.1 port to 8080 instead of just 80. I found that out the hard way when I was getting “*Server could not start,* *another application is using the port”* Error. That drove me nuts for a few days then I opened up the server.xml file in Eclipse, and changed the port back to 80 and un-commented out the thread pool settings for future use.

# Bibliography

Gaddis, T. (2016). *Starting Out With Java: From Control Structures Through Objects 6th Edition.* Pearson Education.

Kulkarni, R. (2018). *Java EE 8 Development with Eclipse 3rd Edition.* Packt Publishing.

**Ramash Fadatore**

Java Guides (2020)

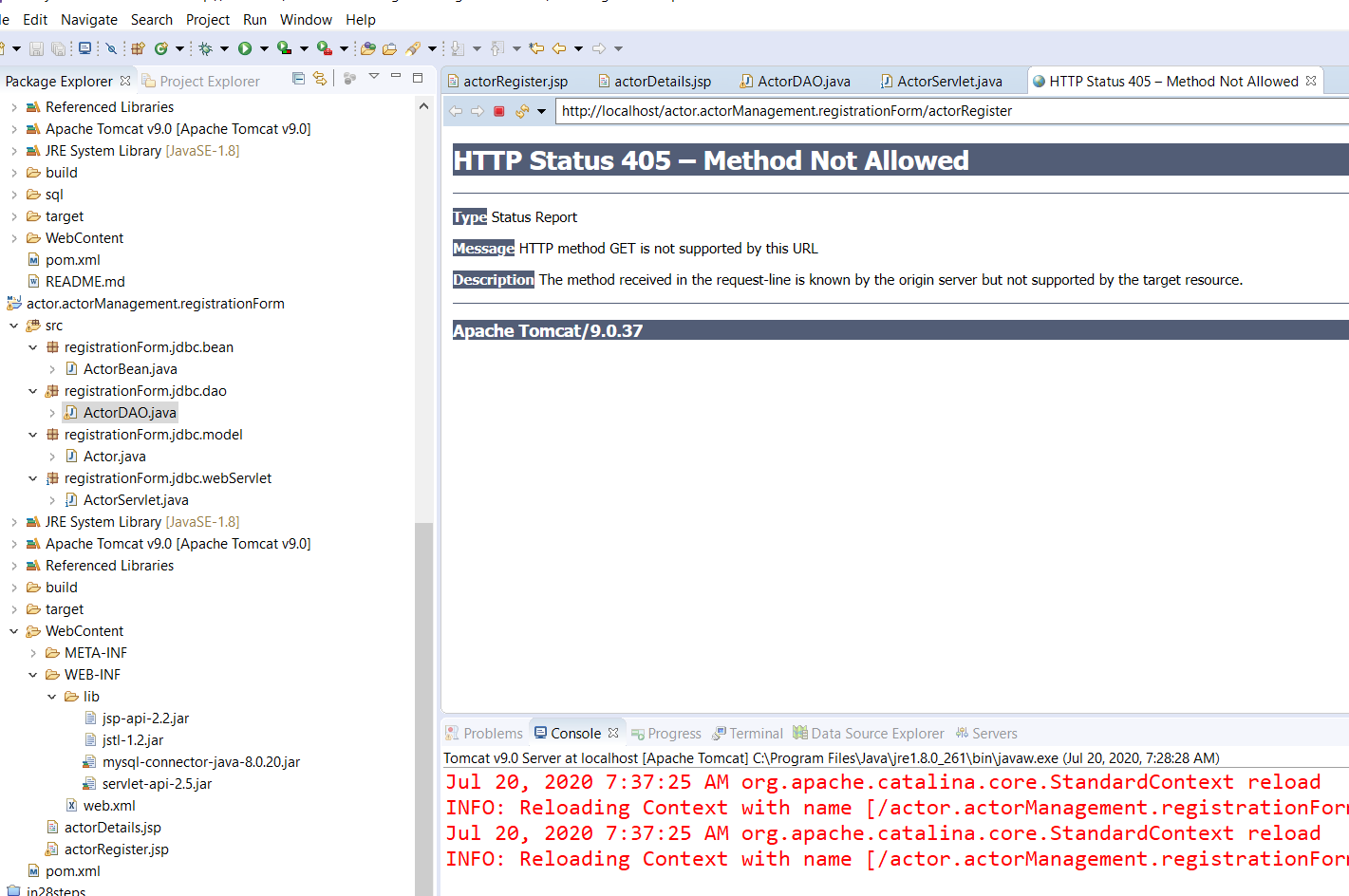


Figure 1: Error Registering New Actor

I suspect there isn’t a GET method in the actorRegister file, or it’s pointing to the wrong method in a file. The plan is to study Ramesh Fadatare’s Tutorial over at JavaGuides.net. He has an excellent one called *“Registration Form using JSP + Servlet + JDBC + Mysql Example”.* I have done this one already, so I’ll go over it again and figure out why my file is not posting.

**07/20/20 4:00PM** – Well I spent most of the day messing around with it. So close!! I’m getting a file not found error, something’s wrong with a path somewhere! LOL Taking break for awhile and regroup, in the mean time I really need to study this file closely, then modify it to suit my project needs:

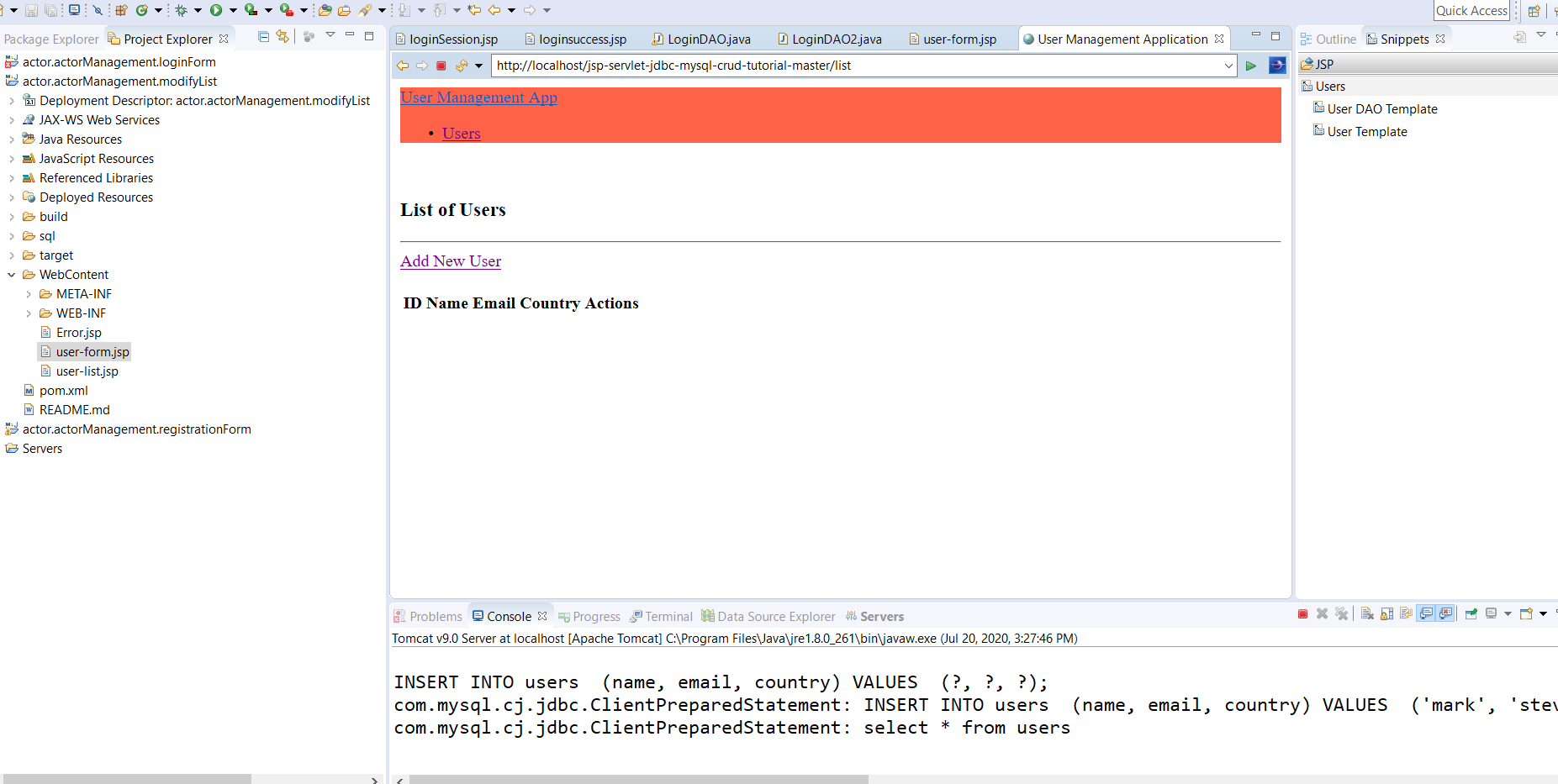


Figure 2: A working Insert file.

I’m not liking how the address bar path is pointing to a different project. Oh well I’ll figure it out! This could be my new starting base for the project.

**07/21/20 – 10:00AM** I decided to go another route. I was having a hard time getting that setup to work correctly, so I deleted everything in the project folder and started over. The other tutorial I’m using for the User Management files has the hibernate jar files referenced in it, along with a persistence file. So I’m going to try it again along with the Java EE 8 Development book, so my current directories are like this:

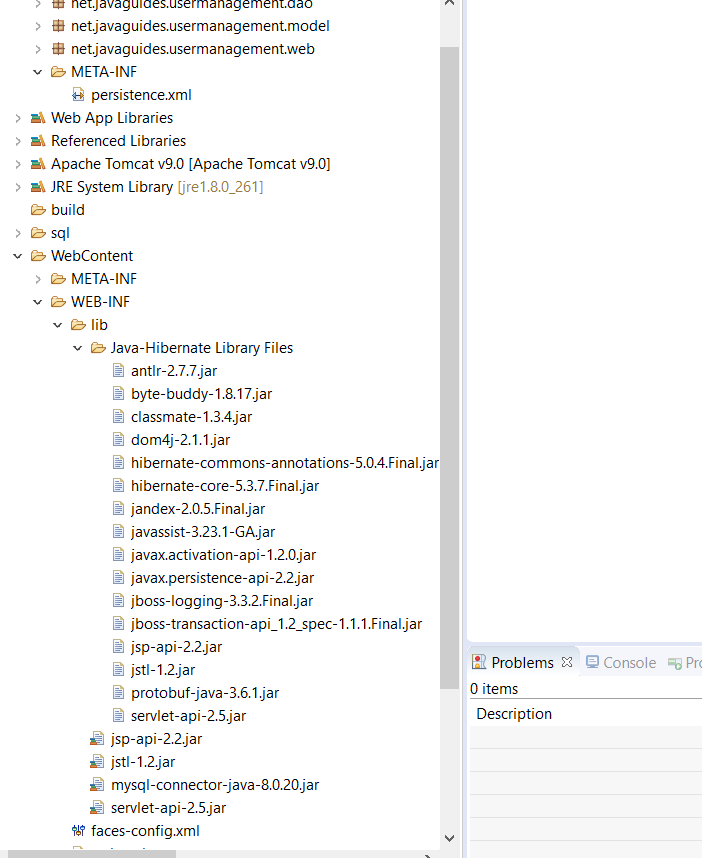


Figure 3: User Management Project directory structure

Notice the persistence.xml file, we’ll see how this goes….I added the jar file in the xml documents:

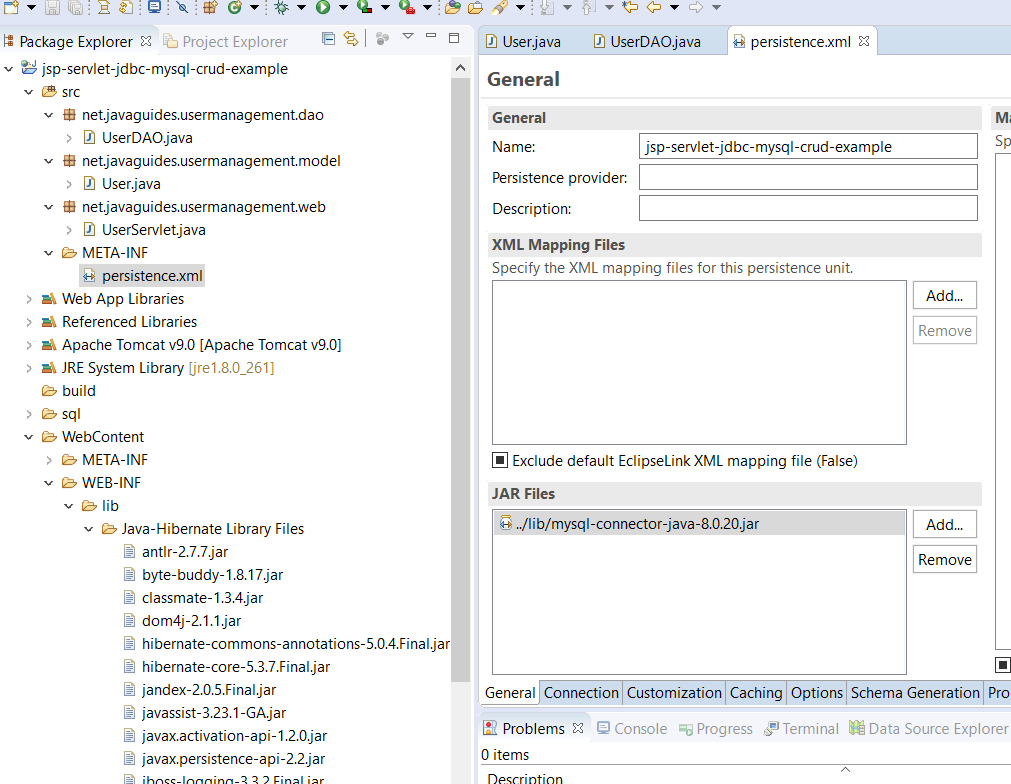


Figure 4: 8.0.20 Jar file added into persistence.xml file

The ping to the demo DB is successful, also I used the default JTA option and JDBC batch writing, with native SQL box checked. I’m getting a jar file error saying it can’t find it when it’s right there:

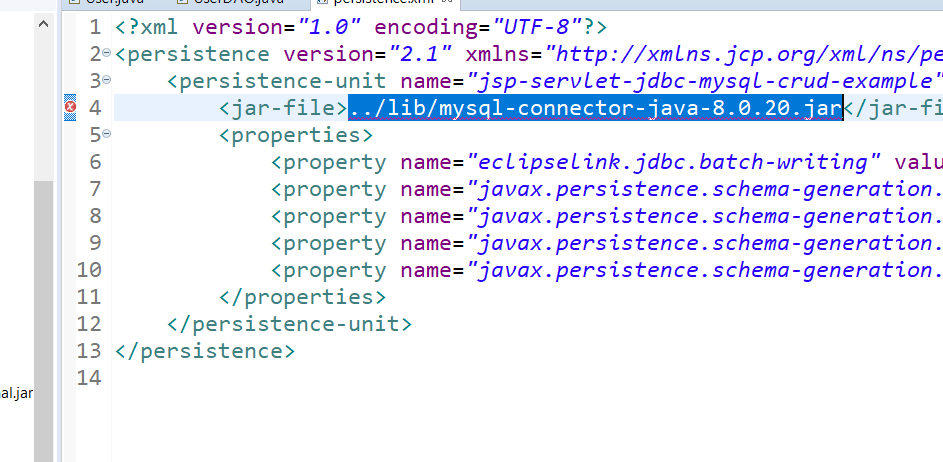


Figure 5: Jar file error

I went ahead and removed the jar file, and it still pings, so,,,maybe because it’s already mapped in the reference library and WEB-INF library?? Hmm… Oh well we’ll find out soon enough! The web xml file looks interesting:

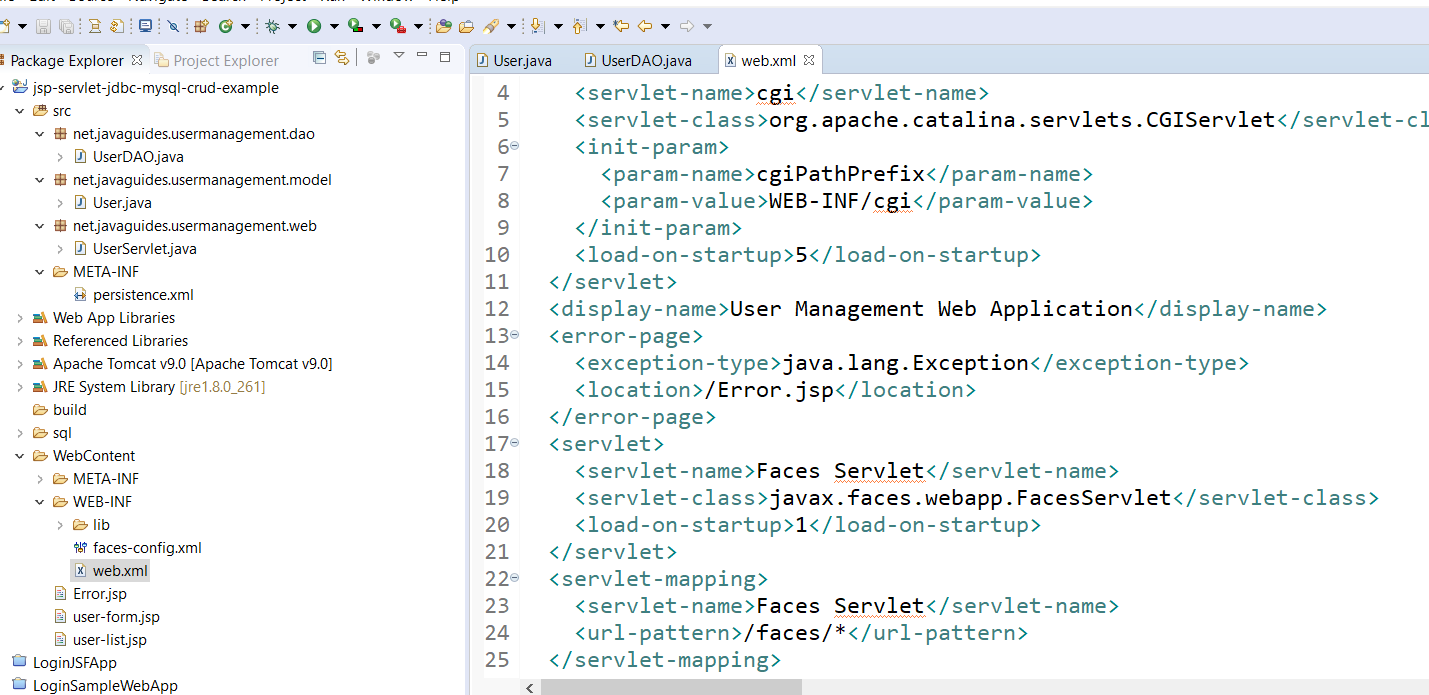


Figure 6: Web Xml file configuration

I’m not sure why he uses a book reference for a new user here:





Figure 7: Book new user reference

Right click on the project in Package Explorer🡪 Run As 🡪Run on Server:

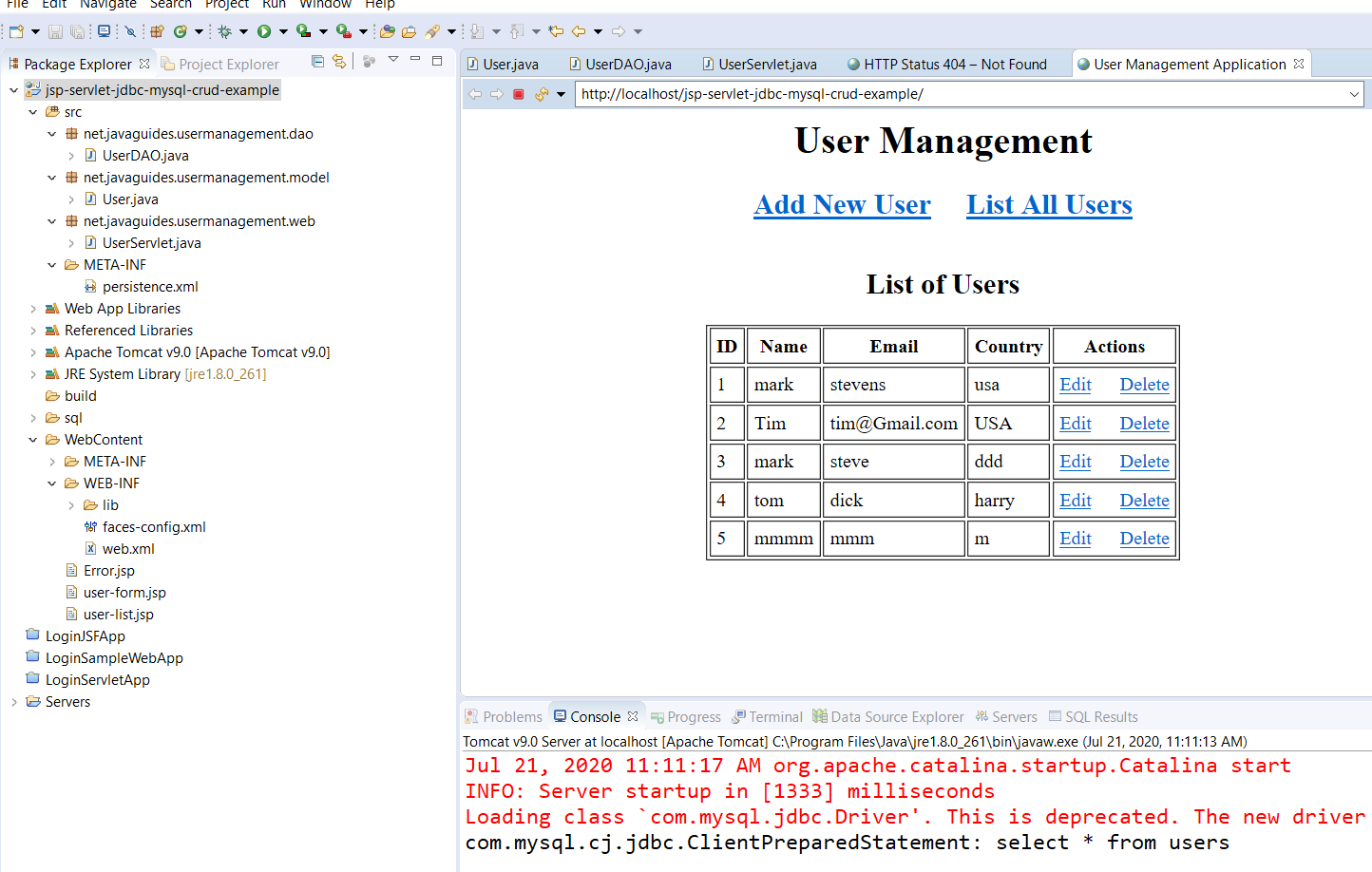


Figure 8: Runs correctly on server

I should change the Driver class to reflect the com.mysql.cj.jdbc reference! Here’s another little thing:

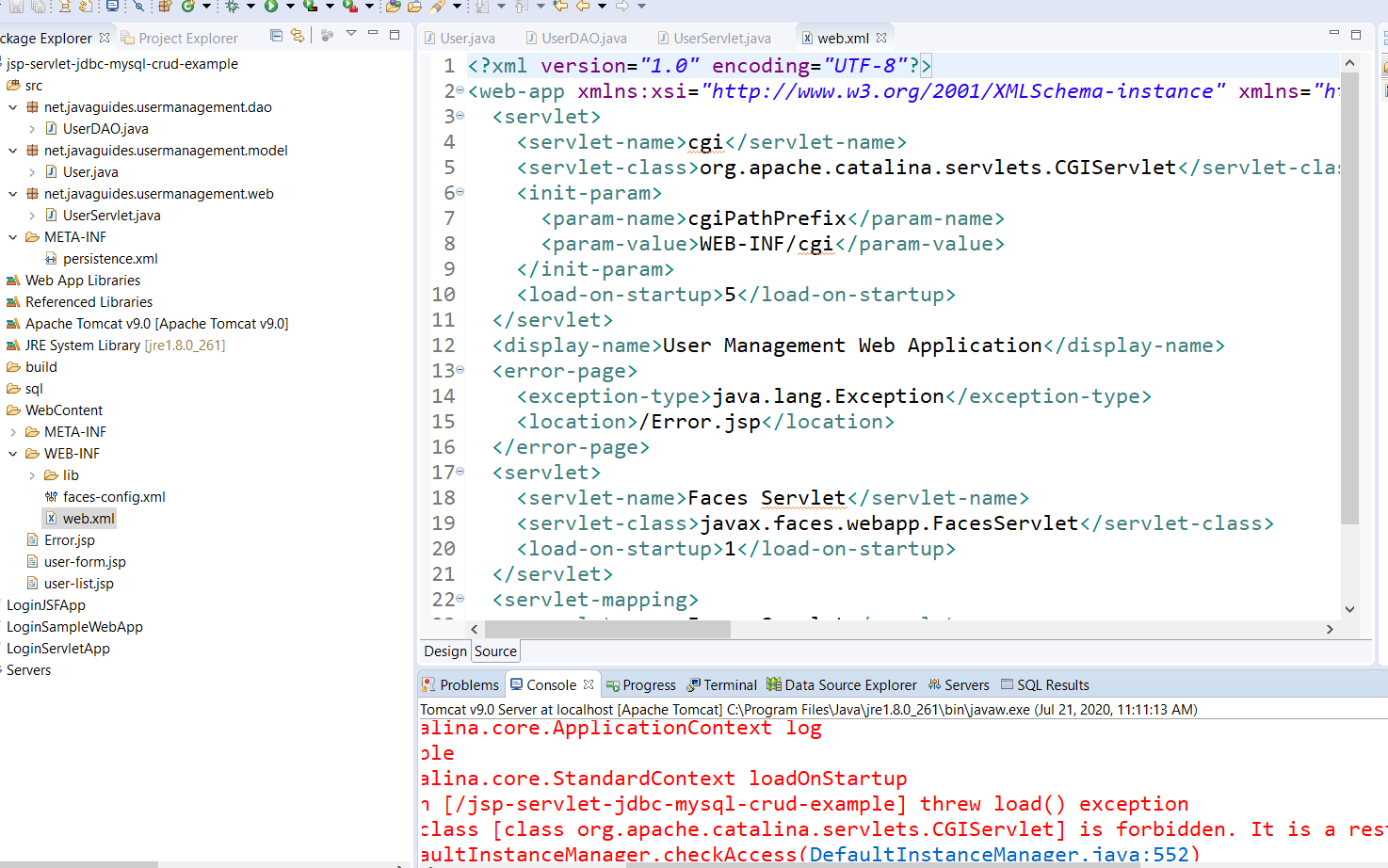




Figure 9: CGI forbidden here error

Mapping the User class as the managed Class in the persistence unit:

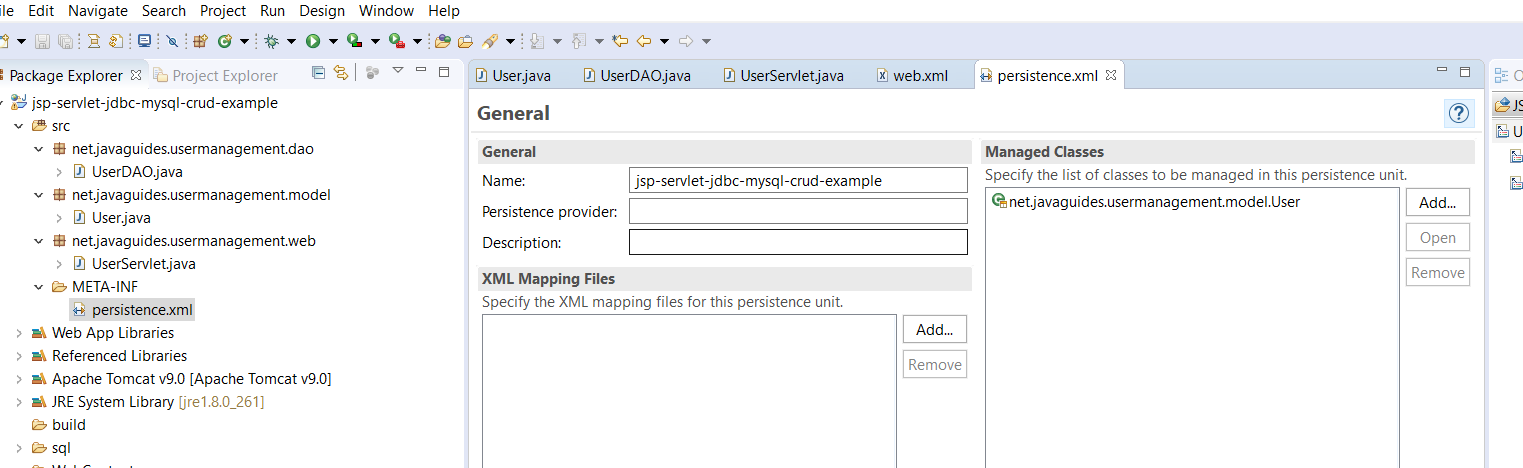


Figure 10:Adding User class in the persistence unit as the managed class

Open the persistence xml file and add User.

No arg-constructor is required so Hibernate can create the instance of the Persistance class by *newInstance()* method like this: *public user() { }* Initializes the method.

**A helper class should be created to help Bootstrap Hibernate. Should only be used once during initialization to be used by all code.**

**See HibernateUtil class.**

The UserServlet file is the page controller for the application, handling all requests.

This was a perfect project to go through, it has a lot of techniques to use as a foundation. You can find this and others at javaguides.net.

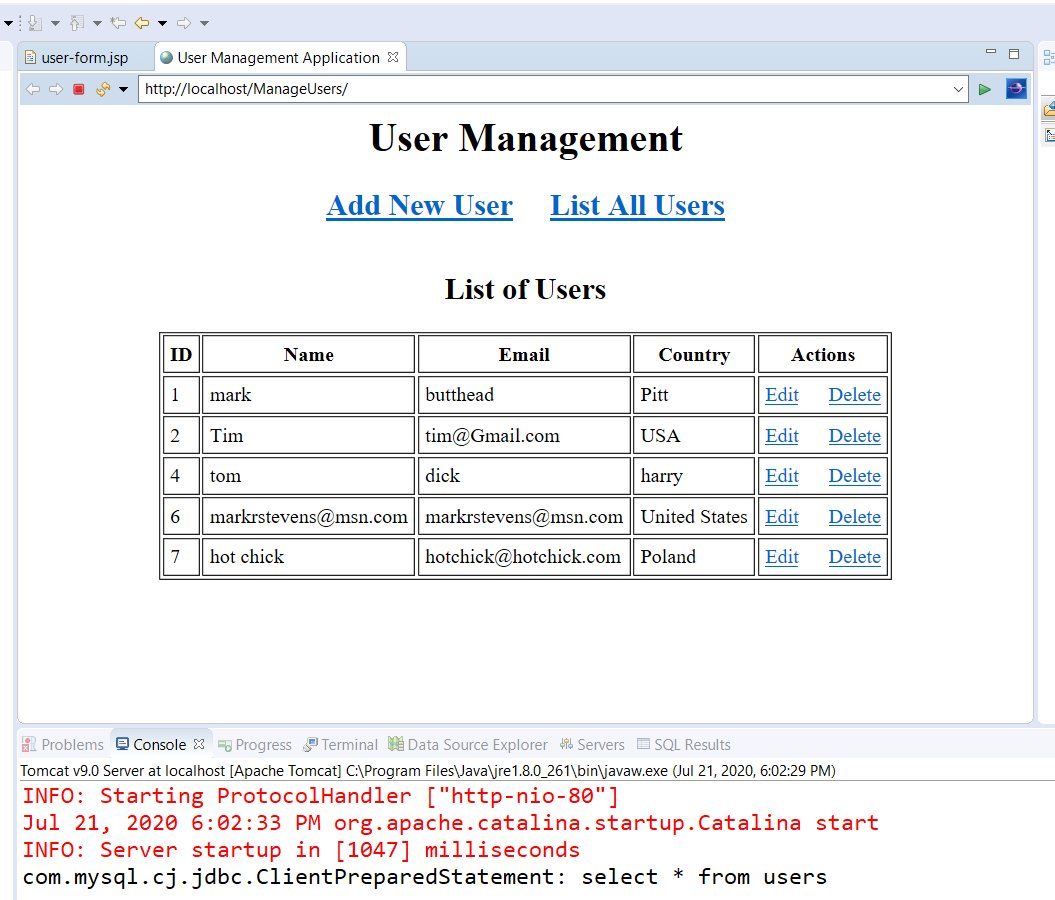


Figure 11: The finished project

Everything works as intended, Edit/Delete/Add/List from the UserServlet file. I love how it prints out the SQL statement in the console, good for troubleshooting. Before I go back to the Java EE 8 with Eclipse book and take another look at the Course Management Project, I’m doing one more tutorial, a 3-part To-Do list. It should go a lot smoother now that I have a pretty good grip on the process. I’m sure I’ll be revisiting the UserManagement files again to see how he did that! 😊

**Part I To Do List**

07/22/20 8:00AM – The to-do list project is in three parts, the Model, Data Access Object, and Controller. First part is done with no problems:



Figure 12:Phase 1 ToDo Tutorial

One thing about this tutorial, the file names don’t match. For example it calls for a LoginBean file, when mine is LogInDao. The references to the files in the tutorial don’t match as you progress through it. For example it talks about a LoginBean file, but the file is actually LogInDao.I had to go back through the code and figure out what was wrong. I think this is right:



Figure 13: Verify user credentials

User is the file that does verification, we will see if I’m right! Although I think it should be UserController?



Figure 14: LoginDao validates User credentials

I think it is LogInDao that validates. Yes I was correct. Another annoying thing, Eclipse likes to put the login file under WEB-INF even though I said WEB-CONTENT, so just copy it over.

**Part II**

This part addresses management features such as Add, Update, List, and Delete. First is to create a Model Class that represents a ToDo Entity called,, well, ToDo!

Next we create a ToDo Dao interface object file in the dao folder.

Next is a DAO class tat provides CRUD operations for the table todo’s in the database. This code is very complex, it iterates through all the scenario’s too, depending on what method is called upon. TodoDaoImpl file. This file is key to making everything work as intended. I suspect I’ll spend a lot of time here! LOL!

I was wrong! The TodoController file is the Page Controller for the application. It handles all page requests, Process all the Http request handlers, and direct to the appropriate JSP page after data is stored in database. I’ll definitely have to go over this page many times.

07/23/20 8:30AM – Going over Part II before I move on to Part III. There’s a lot going on here! If things go sideways(which they always do when testing a new project) that’s the first place I’ll look! I have to todo-list and error page files to do yet. The todo list gather’s up all the todo’s the user selected.

**Part III**

This part is just configuring Tomcat to display the pages. Register page works, database was updated with new user info, although it does not redirect you to a todo list, I’ll figure that out later. I think I’ll try the User Management Project again using Hibernate as the JPA implementation and see what the differences are after the XML Hibernate Configuration

**Hibernate XML Configuration Example**

**(Fadatare, n.d.)**

**3:30PM**

Well I got sidetracked again! LOL 😊 But.I managed to fix up the User Management files. Apparently my dumbass mixed up demo, user, and users databases! HAHA. I figured it out though, and even have Hibernate going on it:

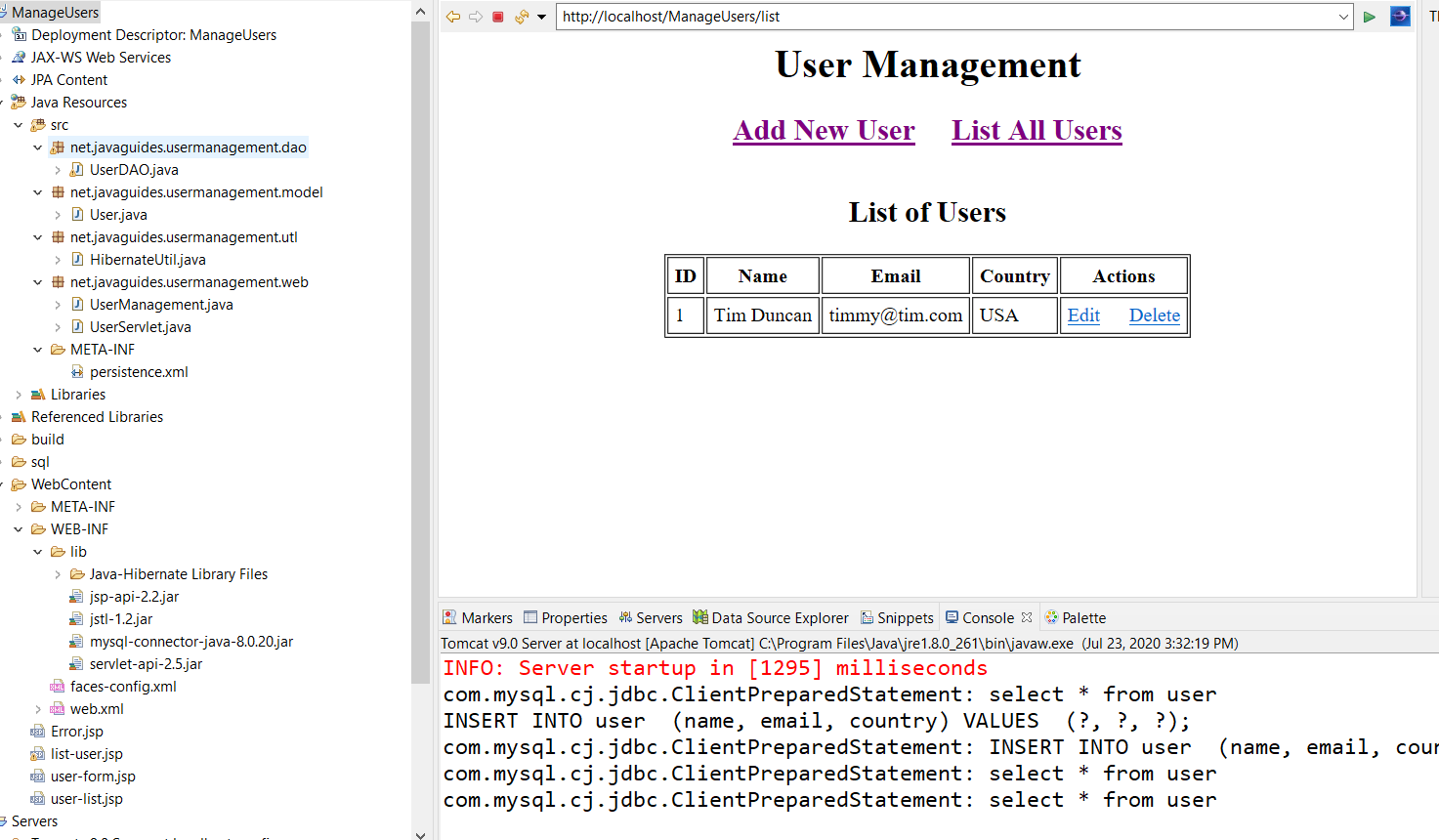


Figure 15: User Management converted to Hibernate

Sweet! Let the fun begin! Now to check the ToDoList files.

It appears to be working ok, except for one thing, I don’t have a database set up for it! HAHA! Maybe later I’ll get to it.

**07/24/20 9:00AM**

**Practice Projects available for learning:**

**Java EE 8 Development with Eclipse, 3rd Edition:**

* **Course Management –** A Domain Model based on Courses, Teachers, Classes, & Students. This project starts at the beginning, making an EER diagram, MySql, Tomcat, JDBC, & JPA with Hibernate. I’m going to start here since the book involves Eclipse.

**Java Persistence with Hibernate, 2005:**

* **Caveat Emptor** – A domain model based on an auction site,much like E-Bay.

**Hibernate In Action, 2005:**

* **Caveat Emptor** – Another domain model based on the same domain Model.

**Java Server Pages, 3rd Edition, 2003:**

* A variety of small examples/projects demonstrating JSP 2.0, JSTL, Java Beans, Enterprise Java Beans(EJB), and JDBC. Also good information on Generating Dynamic Content and JSP page creation development cycle.

**JSP- The Complete Reference, 2001:**

* Thorough JSP reference book, with a Customer Support Project

**Pro JSP 2, 2005**

* Another thorough JSP reference book, Chapter 9 has excellent JDBC information and Object Model Mapping as well as JavaServer Faces.
* **Java Database Bible**

Another good tutorial for a membership website Domain Model

**07/25/20 10:00AM** – The User Management project works great now:

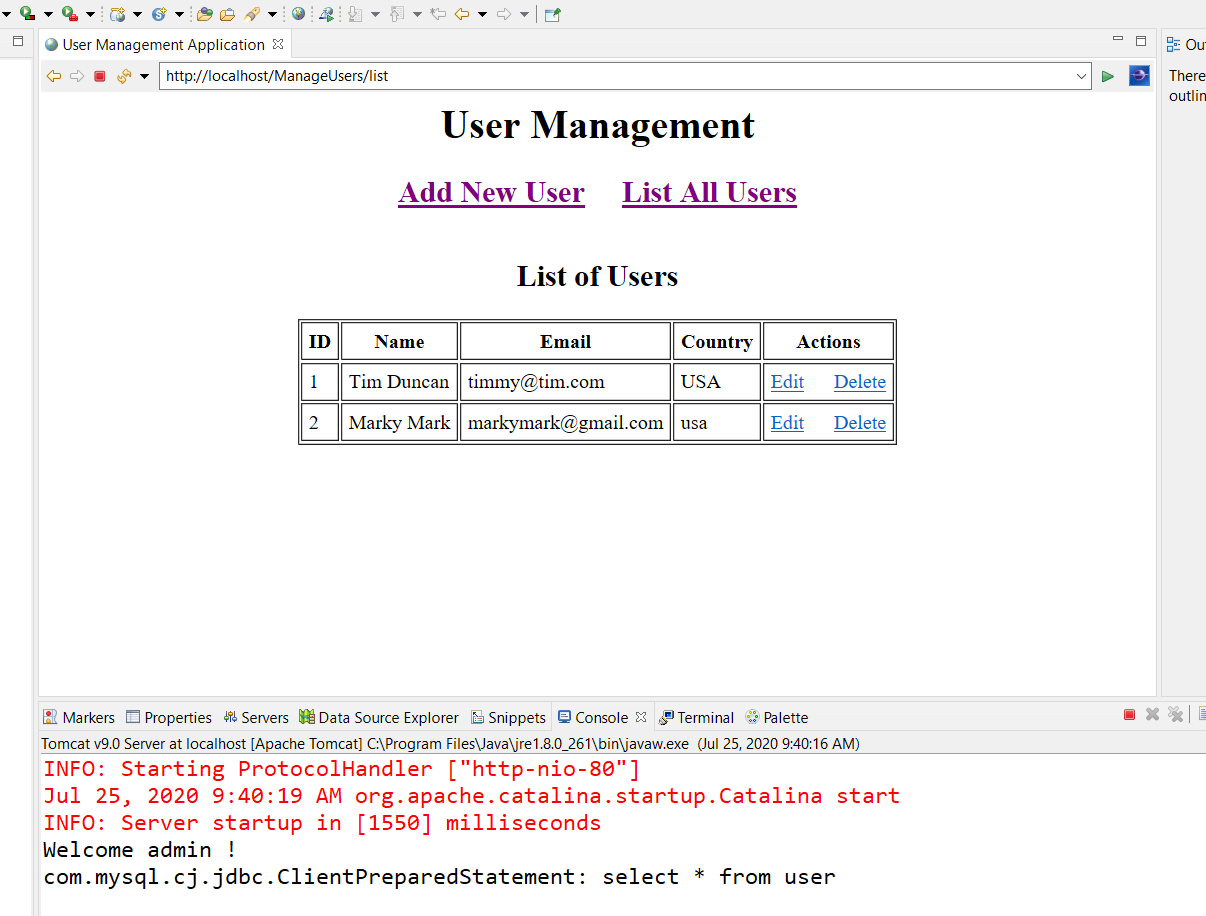


Figure 16: User Management before changing to Course Management

What I’m going to attempt is to take the lessons from this project, and modify it for use with the Course Management Project as the addCourse portion.

**12:00PM** Guess there is no easy way of doing this! As usual,,the coursemanagement import from the master directory wasn’t working out to well,,,It keeps reverting back to JSE-1.5, the Maven Reference Library is messed up,, blah blah..Looks like I will be starting from scratch again after all! Oh well, I need the practice.

4:00PM Well that was pretty brutal, gave me a headache! I deleted the whole CourseManagementJPA project and started a new one. Created the entities Person, Student, Teacher, & Course from scratch, easy enough. Then created the StudentDAO very carefully, now to track down the errors! LOL

**StudentServlet needs work!**

**07/26/20 9:00AM –** I need a new strategy, Nothing seems to be working, I’m missing something! Think I tried to run before I could walk. I may just close the CourseManagement project for now, build another tutorial with a combination of ManageUsers & ToDoList. So,,I’m going to start new documentation also.

# Using the Persistence tools in Eclipse

08/02/20 8:00AM – After many hours of frustration, scraping, & re-working everything, my knowledge of using Eclipse for data manipulation got a lot better! LOL Even though the book I’m using is very good, it lacks the details of where to go and what to do once you are in the Persistence level of your data. For example, this is how it actually looks in this version of Eclipse:

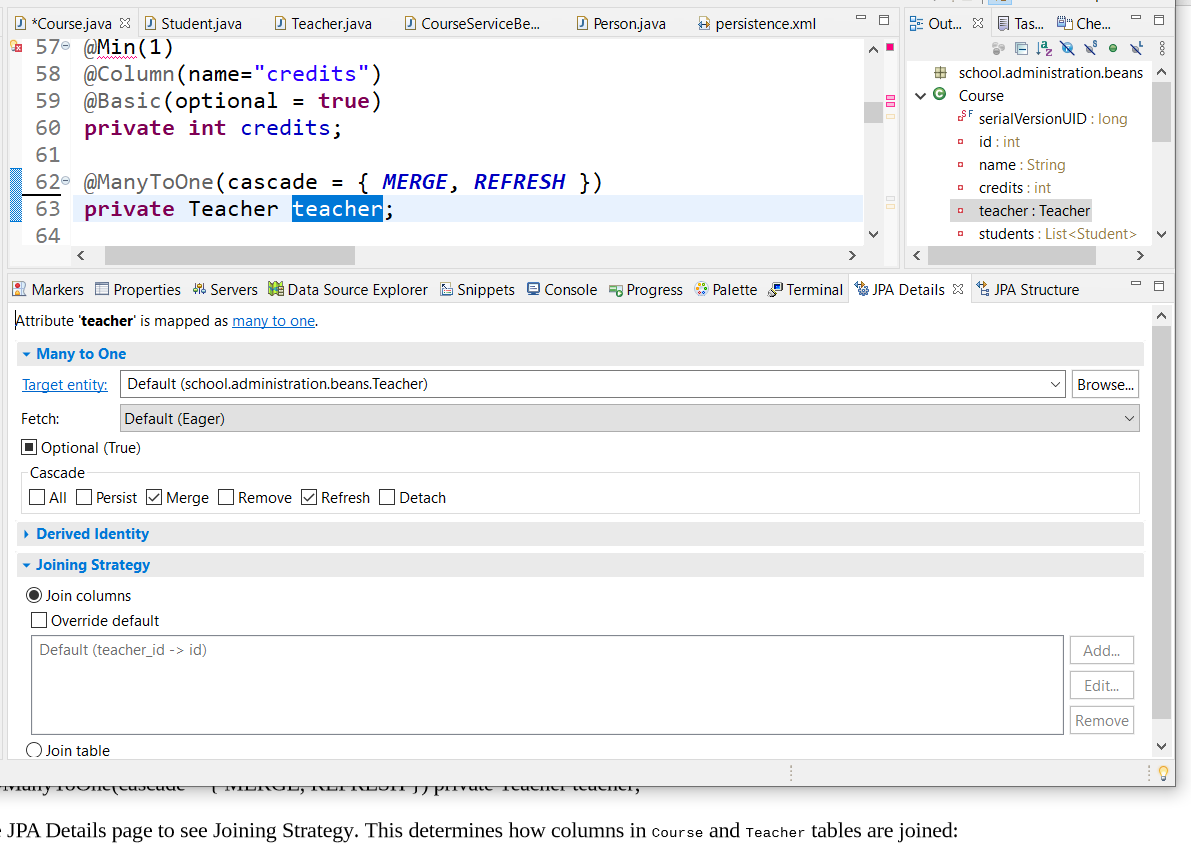


Figure 17: Teacher/Course join strategy

Notice Teacher field is selected in the Course file, then you see the mapping, target entity, Cascade and Join Strategy. Click the override default button, then highlight the teacher\_id default, and click edit,I believe this is correct.

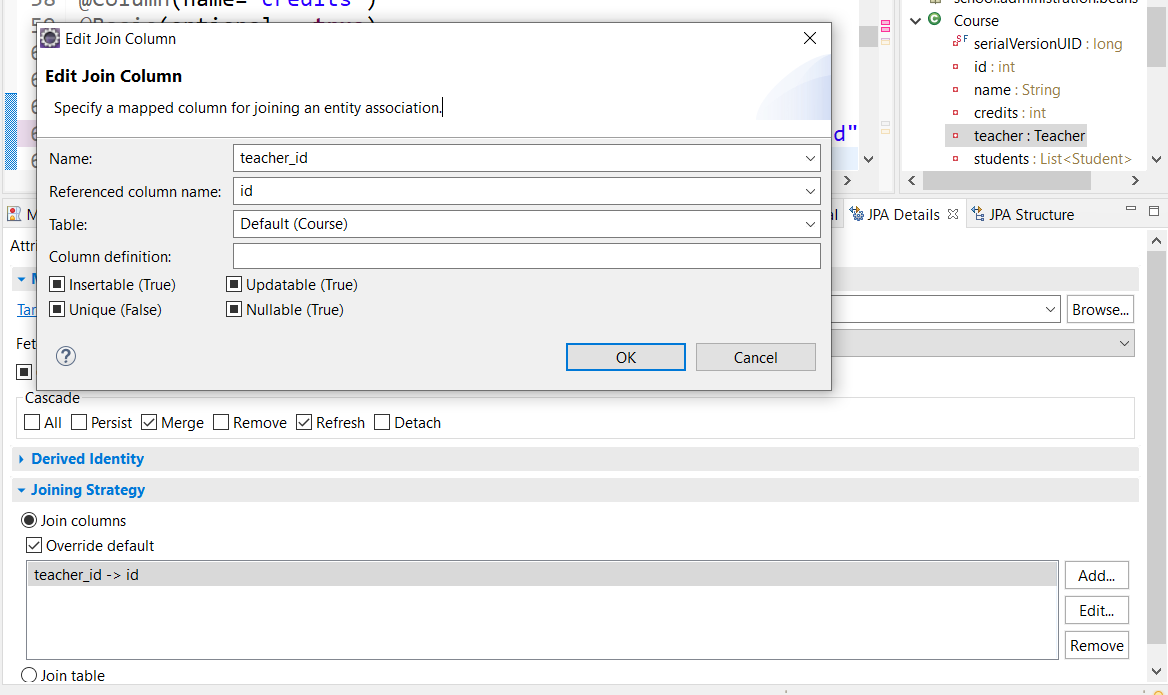


Figure 18: Correct Mapping settings

This just makes it a little clearer, as I said, the book is a little behind but that’s to be expected in any book or tutorial you use.

Here is a final look at the Course and Student files in Eclipse:

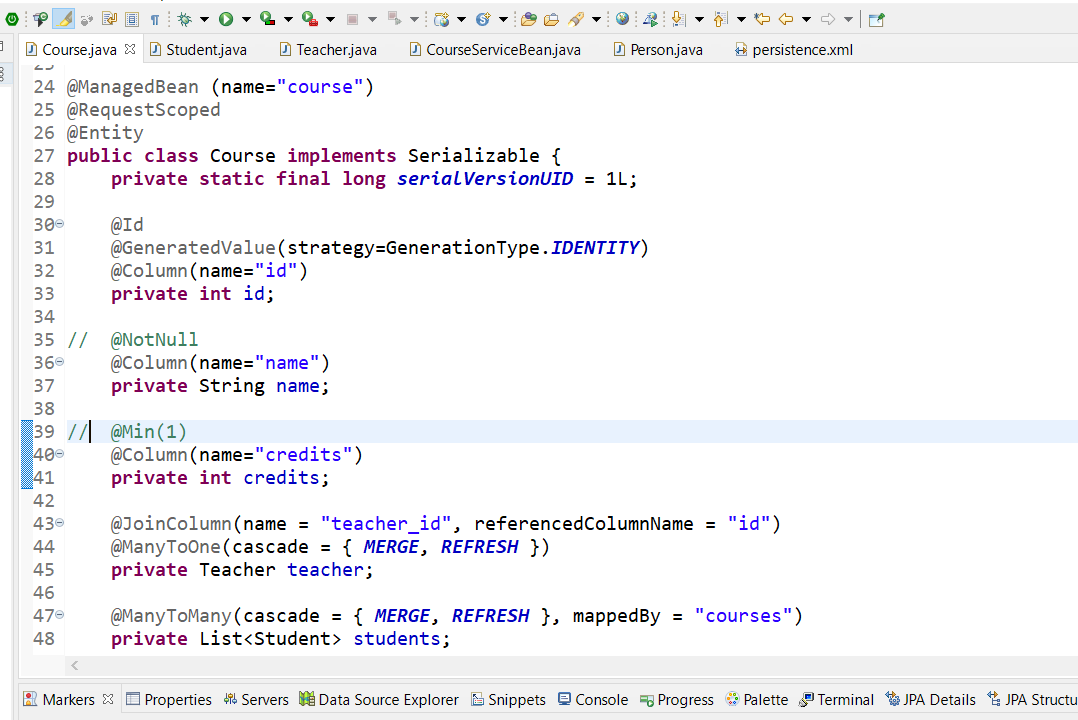


Figure 19; Course Final mappings

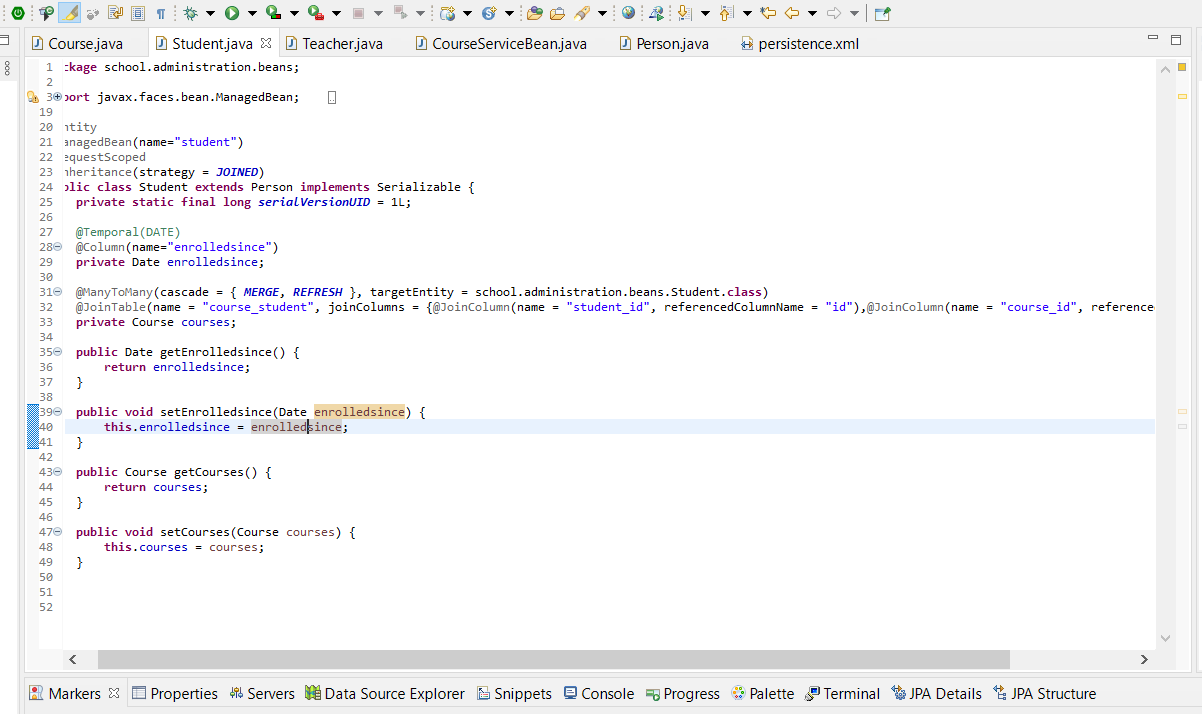


Figure 20: Student final mappings

Notice Course\_Student table is mapped by student\_id, reference column ”id” , and course\_id referenced by “id”. Seems to make sense since many students goes to several courses, mapped by the ID’s of each to keep track? Hmm,,

We shall soon see if this is right. The next section deals with Creating Database Tables from the Entities we just mapped. I still did not get the “NotNull” annotation to work, nor the MIN(1) in course credits yet.

08/03/20 11:00AM – Not going well! LOL Crashed the server as usual, think I tracked it down this time though. I closed all the projects in the eclipse-workspace and the server came up fine. So I opened them one by one and tested after each time. It seems the projects with Hibernate/Persistence enabled were causing it. If I run those projects when everyone else is closed in the workspace, there’s no problem. That kinda makes sense since the persist data is always talking to the server so, I assume that’s one reason why.

Spent the morning working on the CourseManagementJPA Project. Got the index, welcome, and LoginBean going, except the index page is giving me problems at this line: <rendered="#{loginBean.errorMsg != null}" style="color:red" /></rendered> Guess I should track that down huh? Think I’ll just attempt to use the Login Tutorial from javaguides.net. I know that works, just have to modify it a bit.

The best way to do this I figure is to make a login package, and put the files in there. Then modify them to match up with usernsme/password in the course\_management.student table. Let’s see how this goes! LOL

**08/04/20 6:00AM** – OMG! Reading up on JSF pages to get more insight into how they work and the different tags and such, I ran across this little blurb: *“****if you have several JSF applications, you can put the JAR files into a common location. For Tomcat, that location is %TOMCAT\_HOME%\common\lib. When the JAR files are located in the common directory, then every application in the application server has access to them.”*** I knew there had to be an easier way to organize the different dependencies in Maven for different applications. That’s why I get classpath errors when exporting files, the directory they point to is different from the actual Tomcat Container the application runs in. So I went ahead and copied the usual files I use from the folder into the Tomcat lib folder, also made a hibernate folder, and copied the hibernate jars in there, let’s see how this goes! Probably have to change the build paths in my projects to reflect the new file/folder locations.

8:30AM – This is fucked up, two of the files I’m working with keep reverting back to JSE-1.5, even though I clearly changed the build paths to JSE-1.8. Manage Users works flawlessly, and it uses 1.8, think there’s a connection? Should I download 1.5? Or start over again from scratch instead of copying them from the original files? I don’t like the idea of having 2 different Java Runtime Environments on one machine you’re just asking for problems. If this was a real production environment, I’ld have two machines, one with JSE1.5 for legacy applications, and an updated 1.8 environment for new applications.

**11:30AM** – Seems as though if you cut/paste everything, you end up with problems! So starting over with a Course Management JDBC project first.

# 08/08/20 8:00AM

Well after 4 days of fucking around with the project servers/dependencies/reference libraries, think I have it straightened out this time..LOL 😊! Apparently there is another obscure runtime setting under the Window--🡪 Preference drop down. Even if you delete the old server (Tomcat versions) in the project itself, it still gets referenced in this area. I knew there was something silly missing!! So I got rid of all the Tomcat references(or so I thought) and downloaded the J-Boss RedHat EAP 7.3 Server configuration. Then I ran across that setting! Oh well,I’m rolling with the JBoss for now, it seems easier to deal with so far. Except, I need to download the correct dependency files for the DataSource/DataPool that correspond to JBoss. I’m getting errors in the DatabaseConnectionFactory, the pom points to this:

*<dependency>*

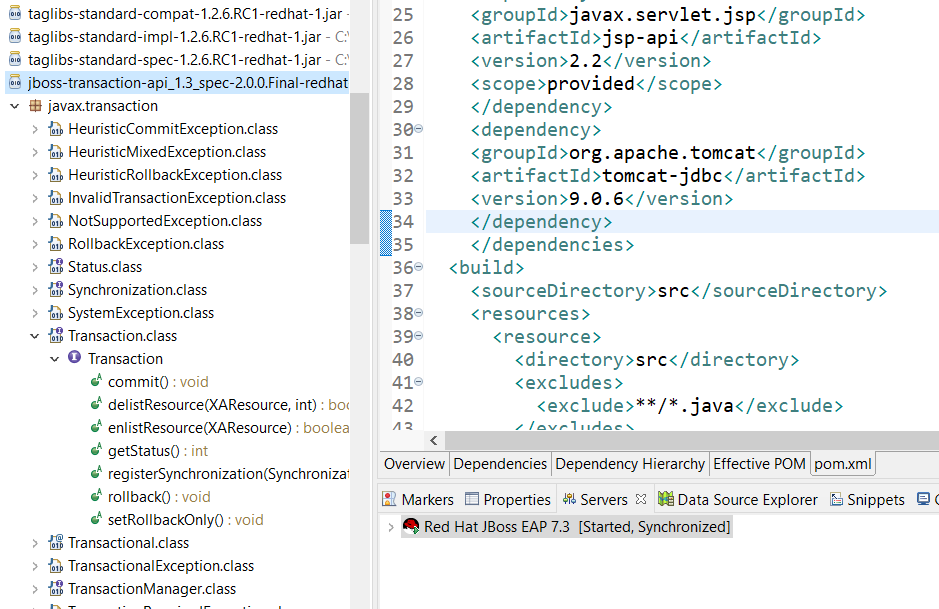
*<groupId>org.apache.tomcat</groupId>*

*<artifactId>tomcat-jdbc</artifactId>*

*<version>9.0.6</version>*

*</dependency>*

So I should just be able to replace that with the one for JBoss JDBC one right? HAHA We’ll see how that works out. I found this little ditty too:

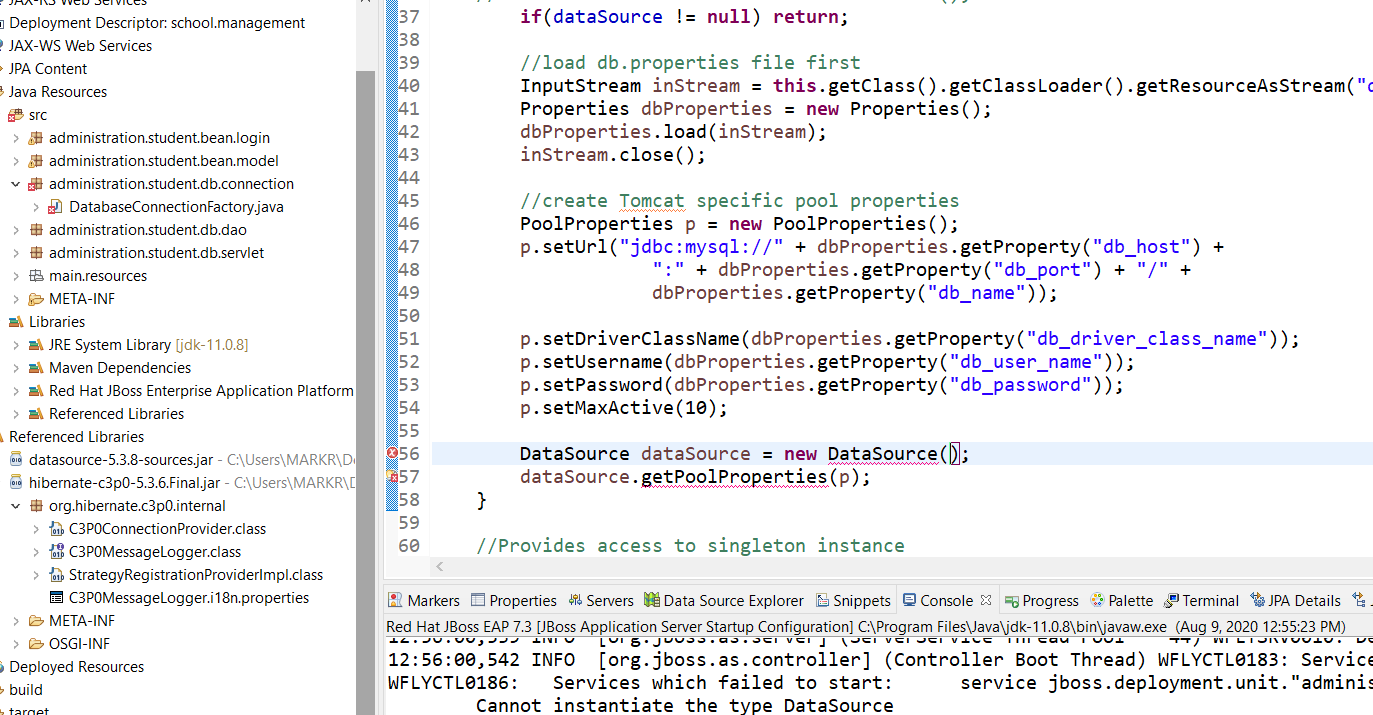
**



*Figure 21: JBoss Transaction Commit & Tomcat JDBC Error*

I may need to remember this Transaction Class later.

08/09/20 1:00PM – Still at it! HAHA! I kinda got them working but still have the missing datasource:



*Figure 22: DataSource Pool Error*

Figured I’ld try the Hikari Drivers, still no luck. Cannot instantiate when clearly it is. I found some tutorials on Hikari, I guess I need to look at those. At least I’m not getting a server failed to start error like on Tomcat! Port in use.I messed with that so much, used netstat -o -n -a and port 8080 never showed up so I have no clue WTF.Maybe I should scrap this project and start over with a fresh one since I dicked this up so bad! LOL

# Yet Another Approach 08/10/20

So,, here we go again, from the beginning. This time though, I’m going to install JBoss AS7.1 and EAP7.3, and just add the Runtimes to the files/project instead of actually installing a TomCat Server. I can add that as a runtime later if this experiment works out. So when the project gets tested, you just go into the server tab under properties dropdown, and add the runtimes from there. I’m pretty sure that’s how it goes. You really don’t need a server at this point, just a simulator. At least I hope so! LOL

So the three beans are started (Student, Teacher, Course), I decided to start small with just the basics, leaving out the Person file and Enrolled Since Date. I’m adding the @ManagedBean and @Scope annotations. I believe I may need each file converted to a DAO object in a separate folder, not sure yet.

I can see one mistake I made attempting this already. Not putting the @ManagedProperty value and the listCourse method:

*“The ManagedProperty annotation tells the JSF implementation to inject another bean (specified as the value attribute) in the current bean. Here, we expect CourseServiceBean to have access to theC Course bean at runtime”*

This makes sense, last time I didn’t do this, somehow I missed it! **This is part of** **Dependency Injection (DI) framework.** This is an example of what I mean by the tutorial doesn’t always give you the exact way to do things, it might just say *“install this”* but most of the time **HOW** you install it makes a big difference!

Here are my persistence settings so far:

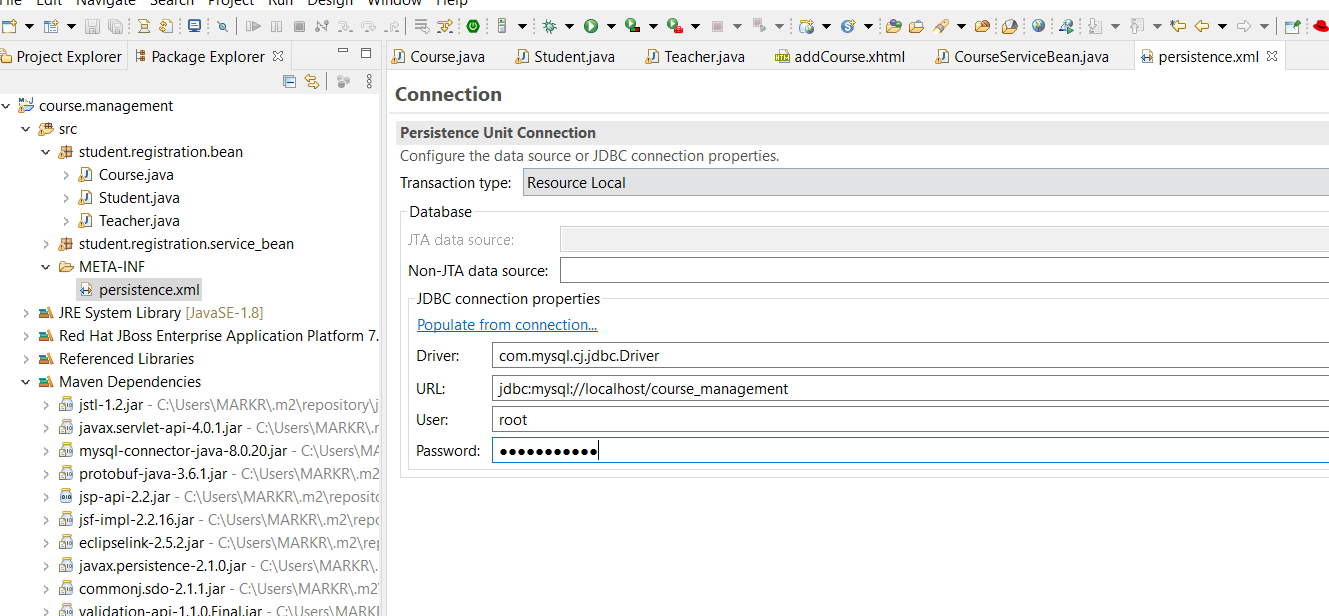


Figure 23: Persistence file settings

Notice the Resource Local setting for now. Also the ‘cj” in the Driver file designation, hopefully it picks up the 8.0.20 driver.

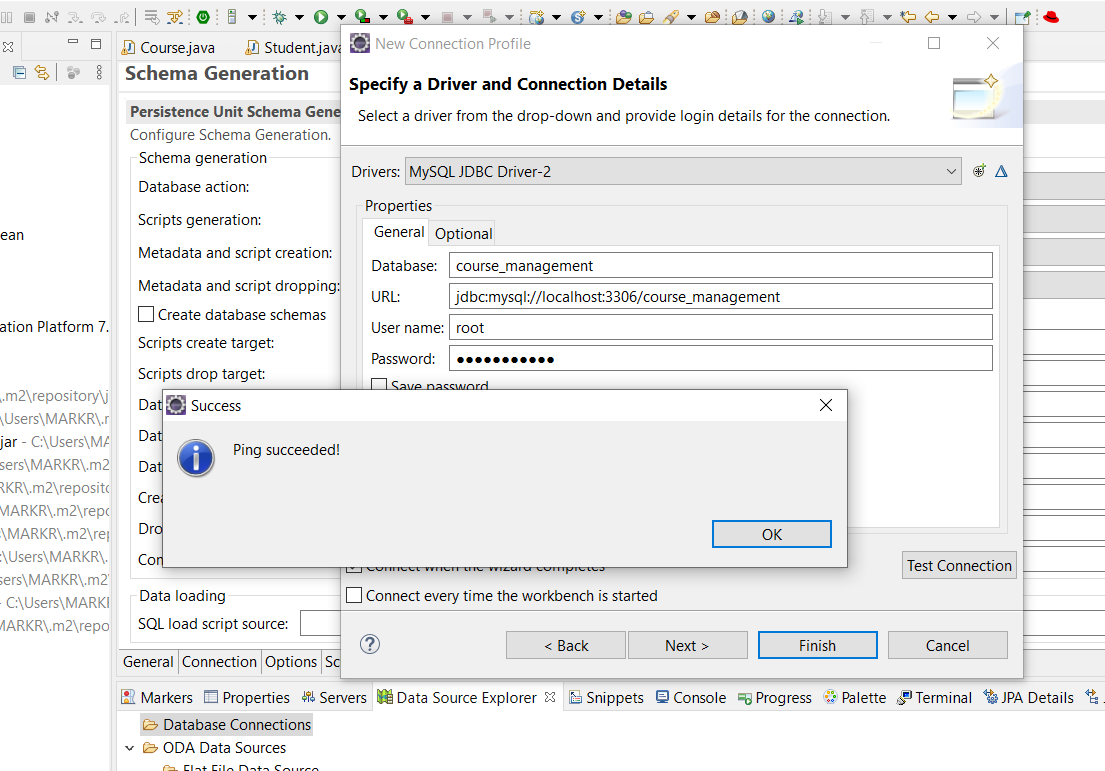


Figure 24: Driver works, and database in selected

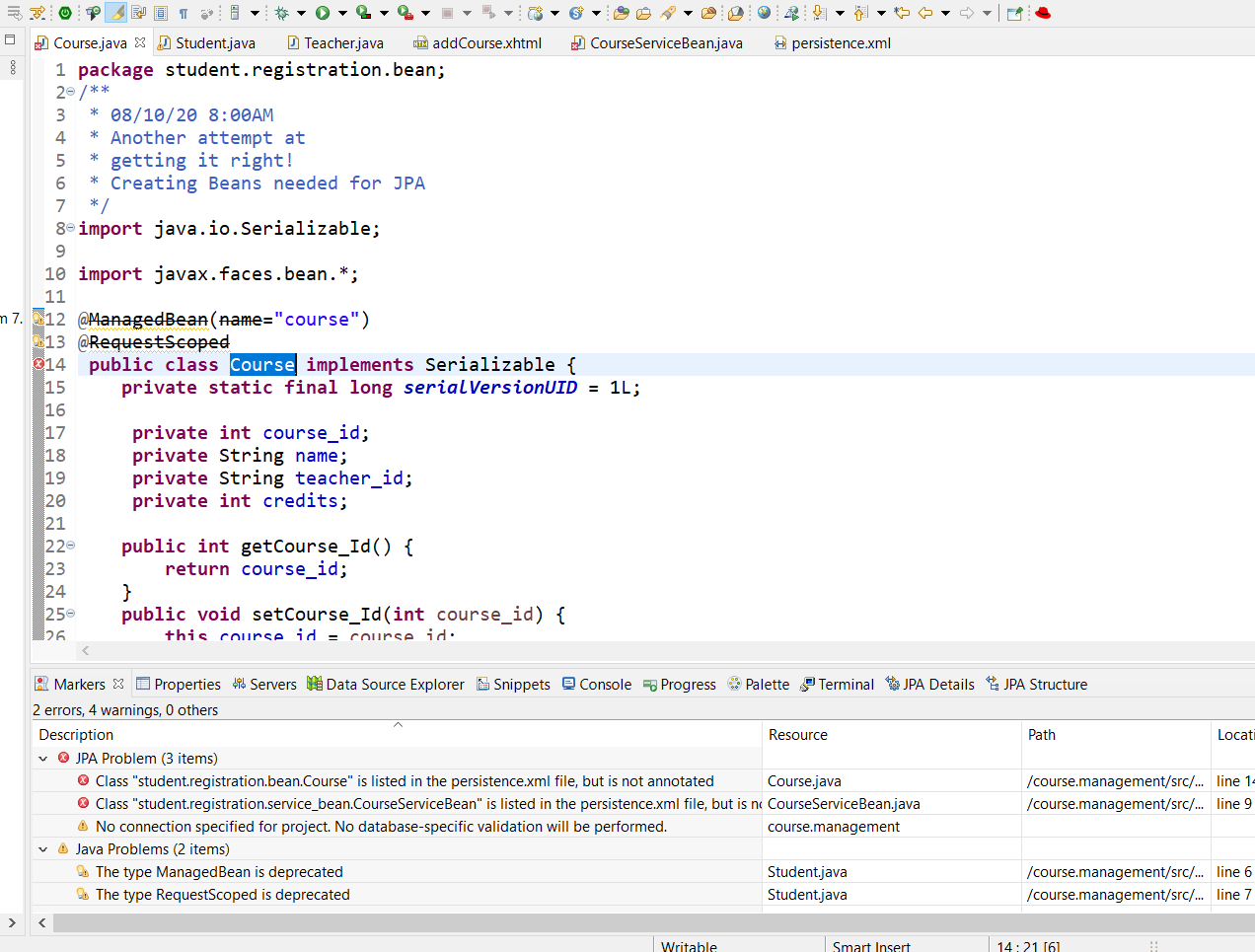


Figure 25: Persistence errors

I’m pretty sure this is because I didn’t annotate the @Id yet? NO! Need to be Entity’s!

It gets tricky here, the DB column name is course\_id, so need to note that in the entity name and column name “I think”



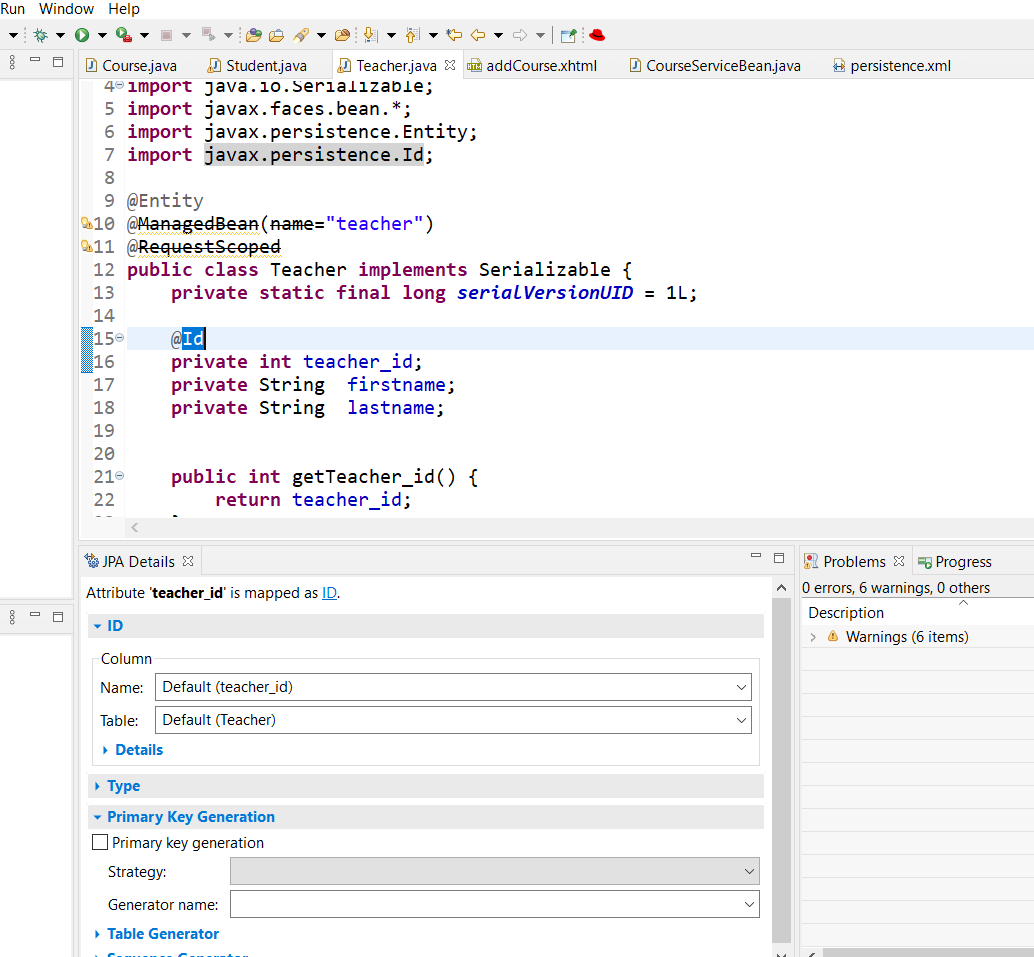
Figure 26:column notations

This way appears to be right. If I get errors, I’ll know where to find them! LOL Another bit of information here for future use:

*“We have now added annotations for all tables and their fields that do not participate in table relationships. We will now model the relationships between tables in our classes.”*

**Do not participate in table relationships** is the key here.

To get the JPA Details pane to open, you need to have the Id of the Entity mapped:



*Figure 27: Id Key Mapping*

I think this is right…..

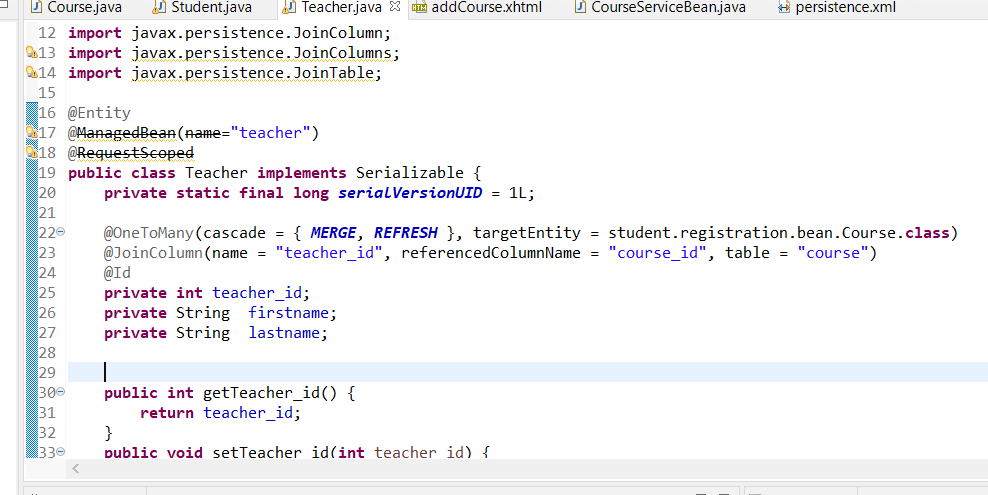


Figure 28: Current mapping entities-8/10/20 1:00PM

Figured I better note this for the future fuck ups!

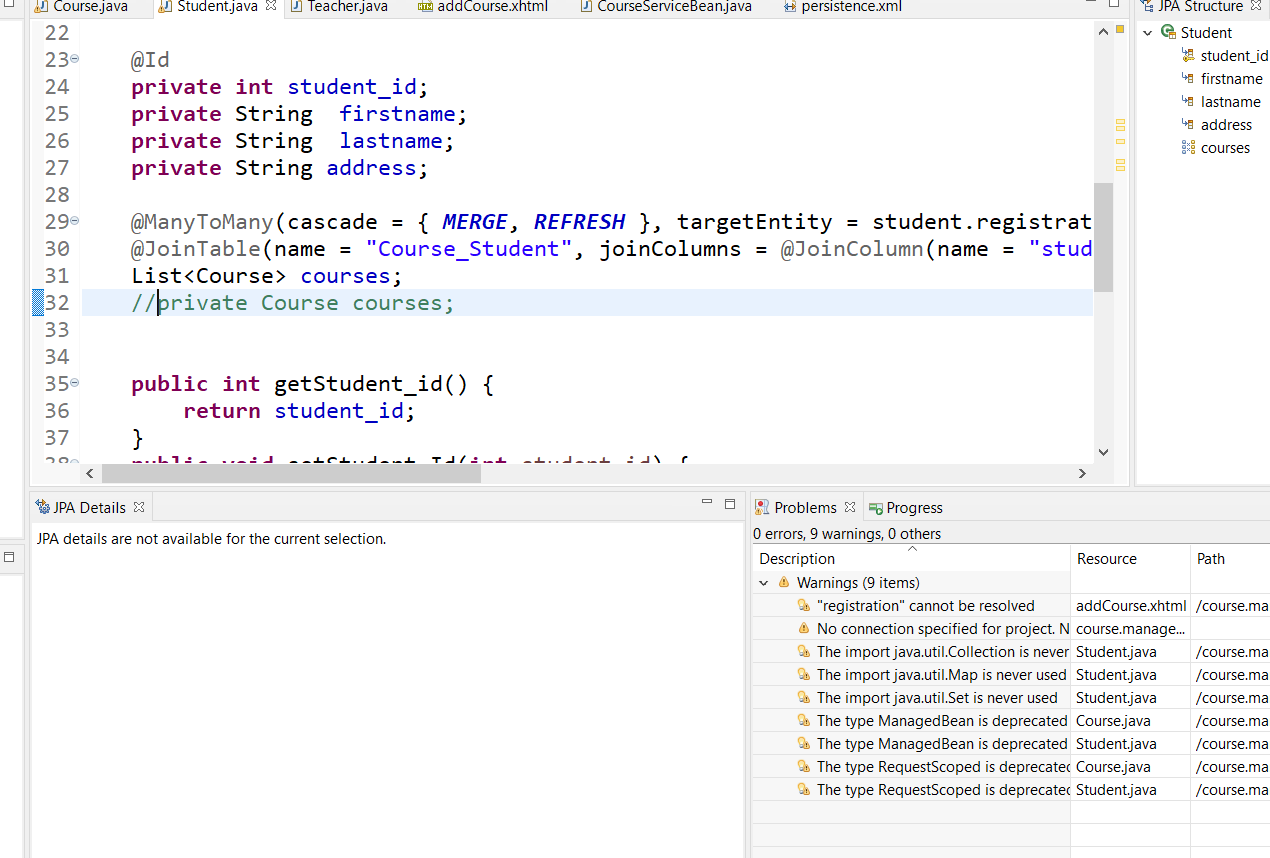


Figure 29: Correct course/Student mapping

At least I think this is right! Had to mess around with it a bit. Page 255 in the book. I got the errors to go away! HAHA! According to the book I’m really close:

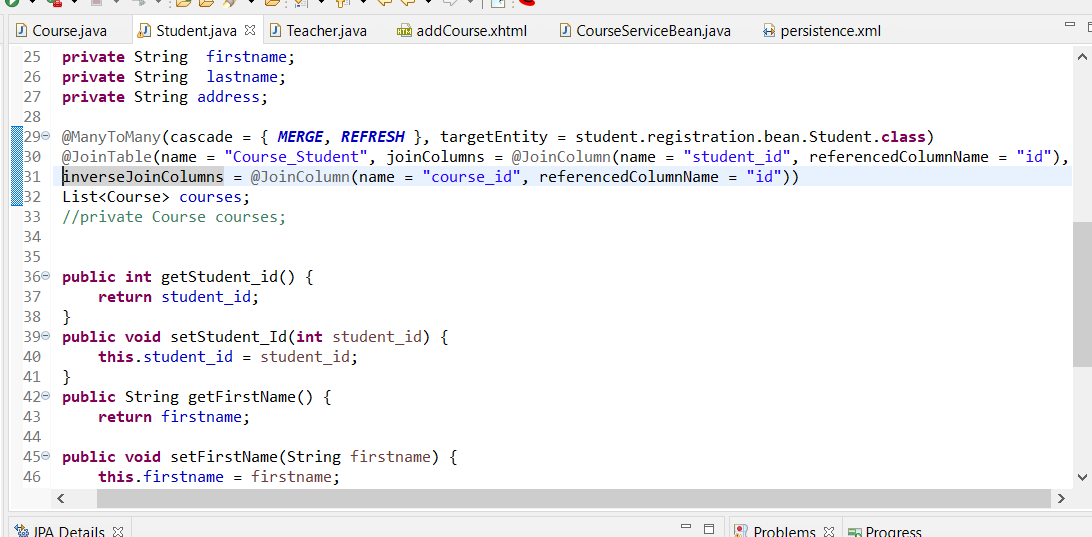


Figure 30: student-course mapping annotations in student file

Here is the pom mappings:

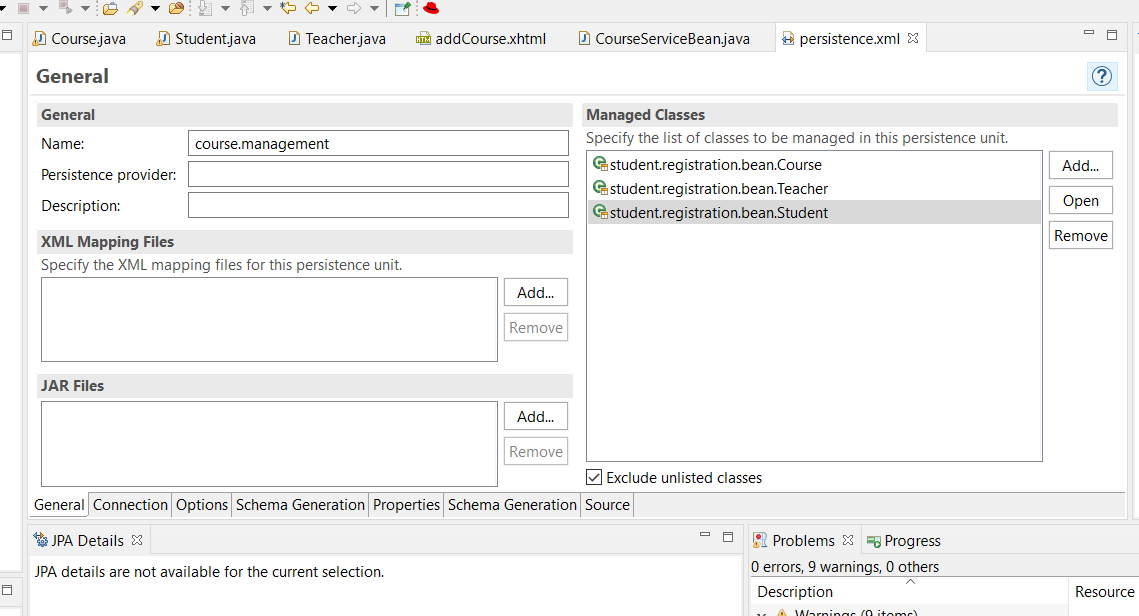


Figure 31: Managed Entity's

Well I give up for tonight, still getting an error:

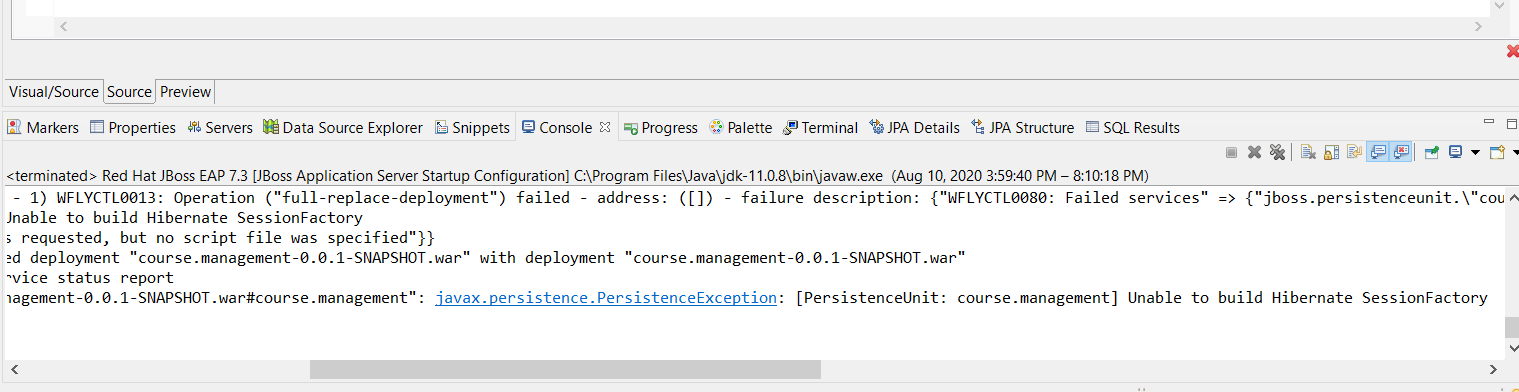


Figure 32: build error

I’ll figure it out tomorrow.

**O8/11/20 6:30AM** – I believe my error issue comes from the here:

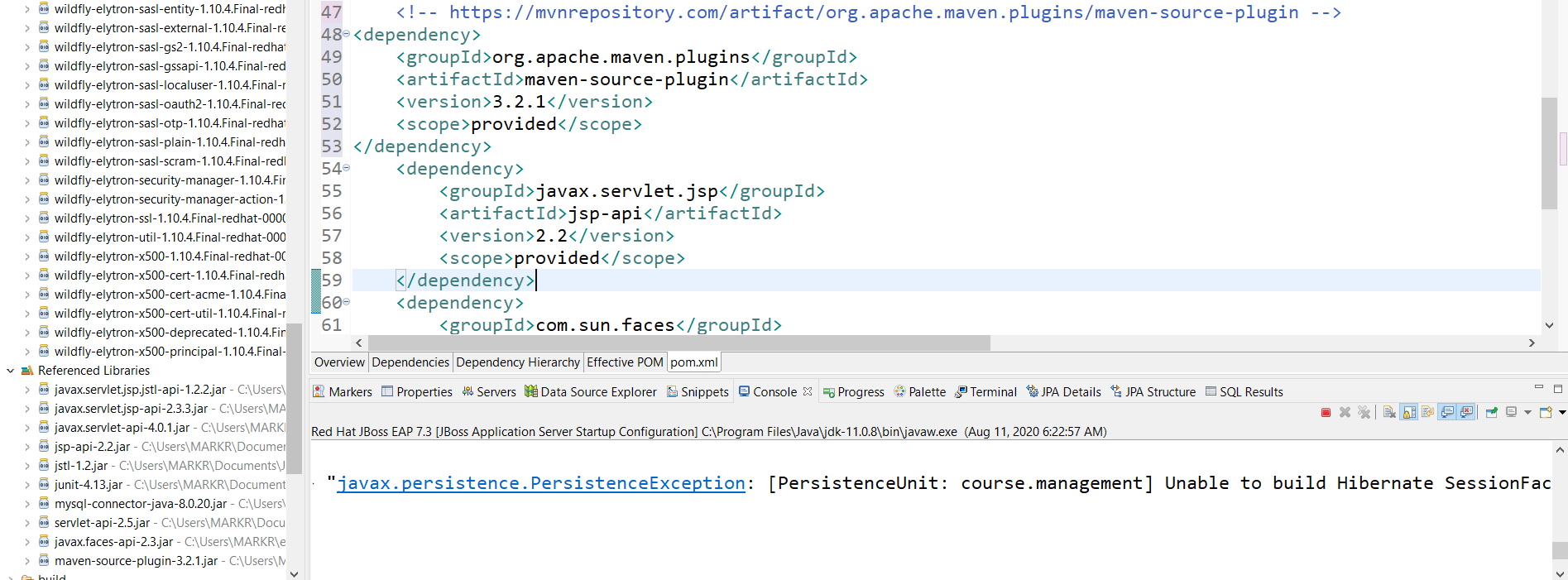




Figure 33: Build session factory error

So I’ll be researching this error, the plug-in is pointing to org.apache, when I’m using the jboss server, that could be an issues among others,,LOL

UGGHH!! This is pissing me off! As soon as I go into the pom file and try and edit dependencies manually, Eclipse crashes! Maybe I should do it from the dependencies tab instead of the actual pom text?

Maybe here is part of the problem:

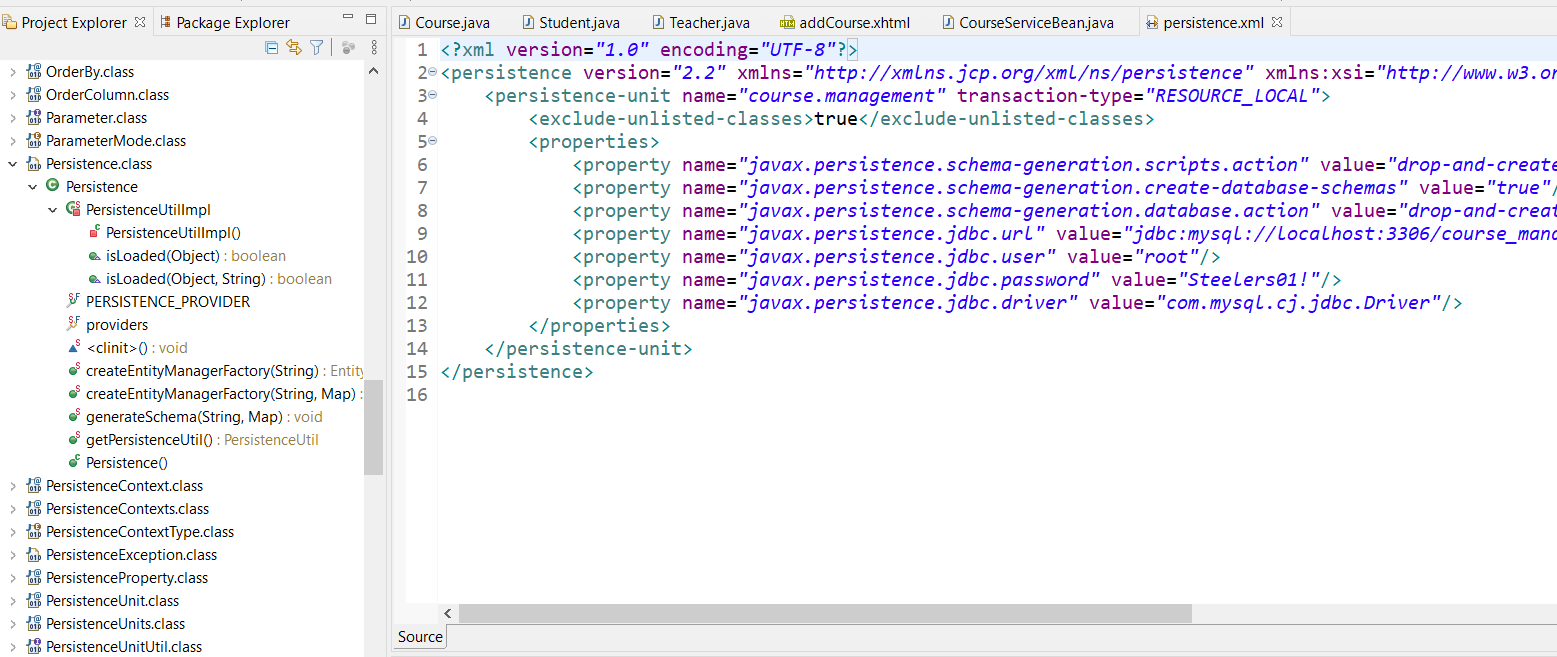




Figure 34: Persistence xml file

Maybe it’s the EntityManager not initializing?? Or change transaction type to JTA?

HA! I didn’t add the No-Arg Constructor for the Model classes! DUH!

***“A no-arg constructor:****It is recommended that you have a default constructor at least package visibility so that hibernate can create the instance of the Persistent class by newInstance() method.”*

The following are the typical steps in creating a JPA application:

***1. Create a database schema (tables and relationships). Optionally, you can create tables and relationships from JPA entities. We will see an example of this. However, it should be mentioned here that although creating tables from JPA entities is fine for development, it is not recommended in the production environment; doing so may result in a non-optimized database model.***

***2. Create persistence.xml and specify the database configurations.***

***3. Create entities and relationships.***

***4. Get an instance of EntityManagerFactory by calling Persistence.createEntityManagerFactory.***

***5. Create an instance of EntityManager from EntityManagerFactory.***

***6. Start a transaction on EntityManager if you are performing insert or update operations on the entity.***

***7. Perform operations on the entity.***

***8. Commit the transaction.***

(Kulkarni, 2018)

No Wonder nothing was working! LOL Here’s the current JPA configuration:

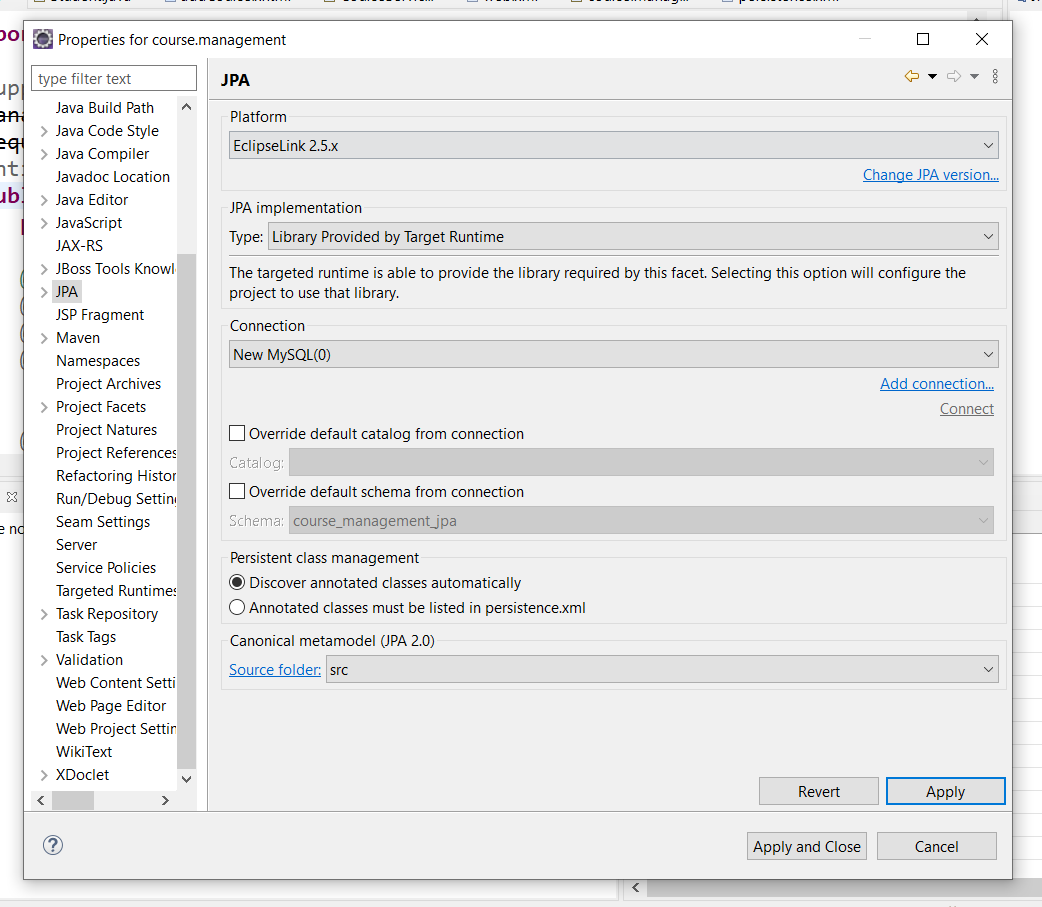


Figure 35: Current JPA configuration

Here’s another piece of info to remember: “*EntityTransaction Interface used to control transactions on* ***resource-local entity managers****. The EntityManager.getTransaction() method returns the EntityTransaction interface.”*

The key here is resource-local I believe. Still can’t seem to find the persistence pane to change it, somethings wrong with the file, might be corrupt or something.

After adding the no-arg and default constructors, I rebooted the server:

A screenshot of a social media post

Description automatically generated



Figure 36:server reboot after creating no-arg constructors

It appears to have started the driver, so that’s good! I still don’t have a persistence file yet though.

3:00PM – After monkeying around again with the persistence settings and setting it to JPA,, I finally have something:

Services which failed to start: service jboss.persistenceunit."course.management-0.0.1-SNAPSHOT.war#course.management": org.hibernate.AnnotationException: Illegal attempt to map a non collection as a @OneToMany, @ManyToMany or @CollectionOfElements: student.registration.bean.Teacher.teacher\_id

I changed the transaction type back to resource local and still the same error:

WFLYCTL0448: 278 additional services are down due to their dependencies being missing or failed

I’m taking a break and regrouping, maybe I should just leave the course\_student table out for nowand at least get the other tables made..LOL

After much monkeying around I might have something:

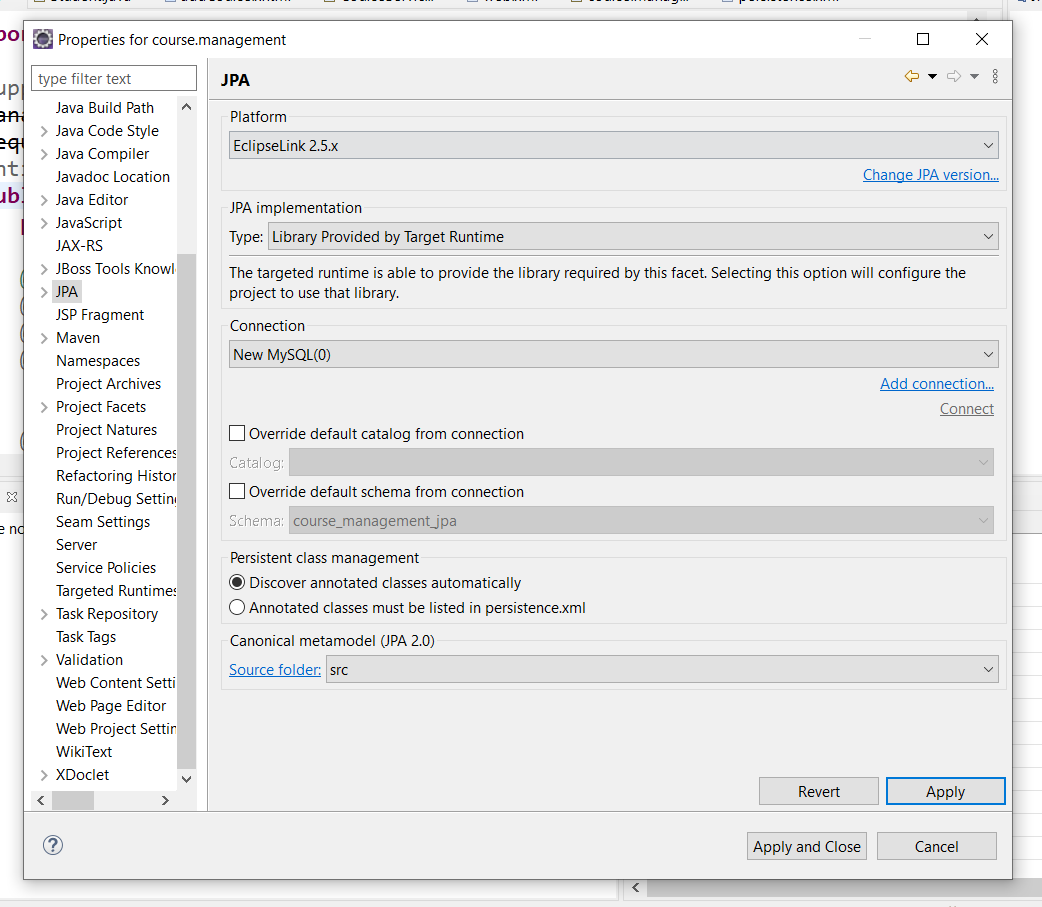


Figure 37: Project jpa settings

And of course I got an error:

Internal Exception: Exception [EclipseLink-7250] (Eclipse Persistence Services - 2.5.2.v20140319-9ad6abd): org.eclipse.persistence.exceptions.ValidationException

Exception Description: [class student.registration.bean.Teacher] uses a non-entity [class int] as target entity in the relationship attribute [field teacher\_id].

Should be an easy fix?? HAHA

**08/12/20 7:00AM** – After much tweaking and re-arranging the Entity Relationships, I might have something that’s workable! At lest the error’s went away,,HAHA! Now to see if the Tables get generated correctly from the Entity models.

DAMNIT! Failed:

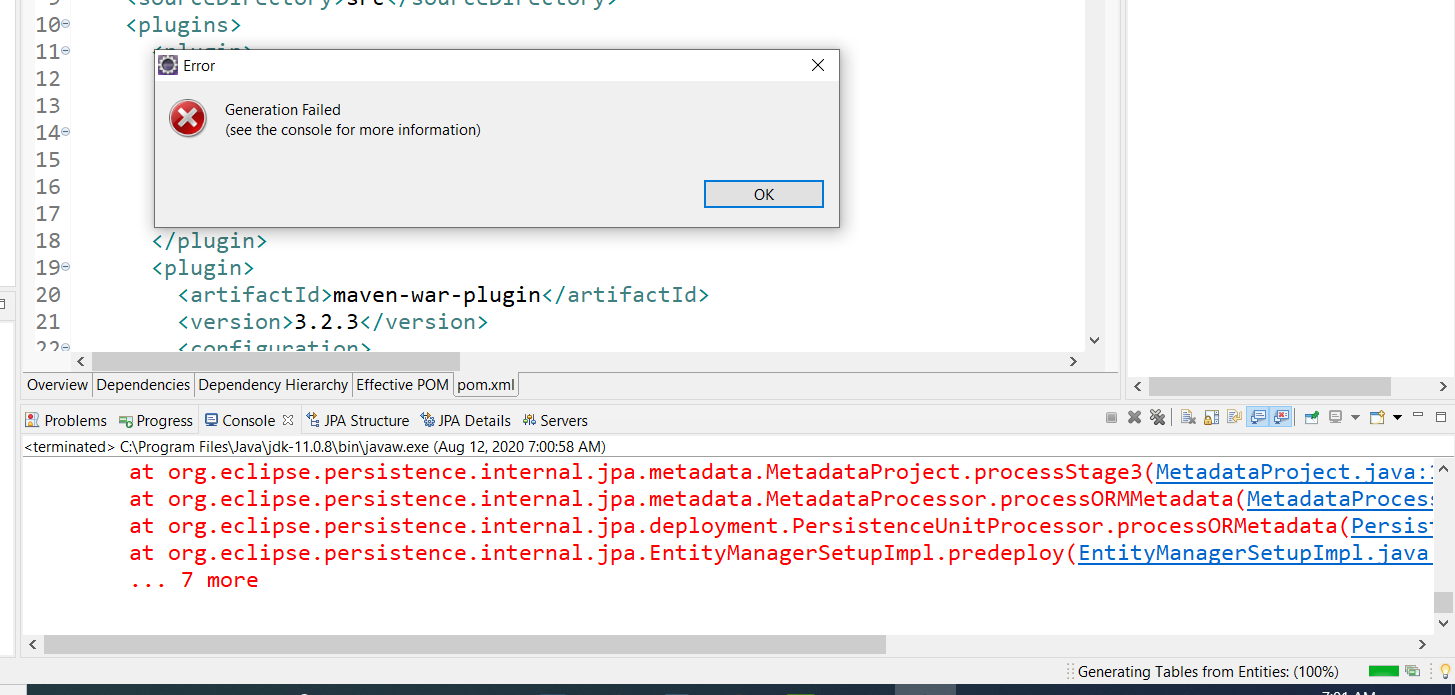


Figure 38 Table from entities fail

Here is the full error as far as II can tell:

“*Caused by: Exception [EclipseLink-7250] (Eclipse Persistence Services - 2.5.2.v20140319-9ad6abd): org.eclipse.persistence.exceptions.ValidationException*

*Exception Description: [class student.registration.bean.Course] uses a non-entity [class int] as target entity in the relationship attribute [field teacher\_id].”*

Let the investigating begin! LOL I might need to take a break from this project for awhile and do more studying. Maybe try this part of the project with a basic table creation with persistence, and add the columns to at least get a better feel for it. I kinda understand what’s going on here with the joins and such, it’s the syntax with the software that needs further practice.

9:30AM – After stripping down the Java Entities and re-ran the create table from entities, I got something at least:

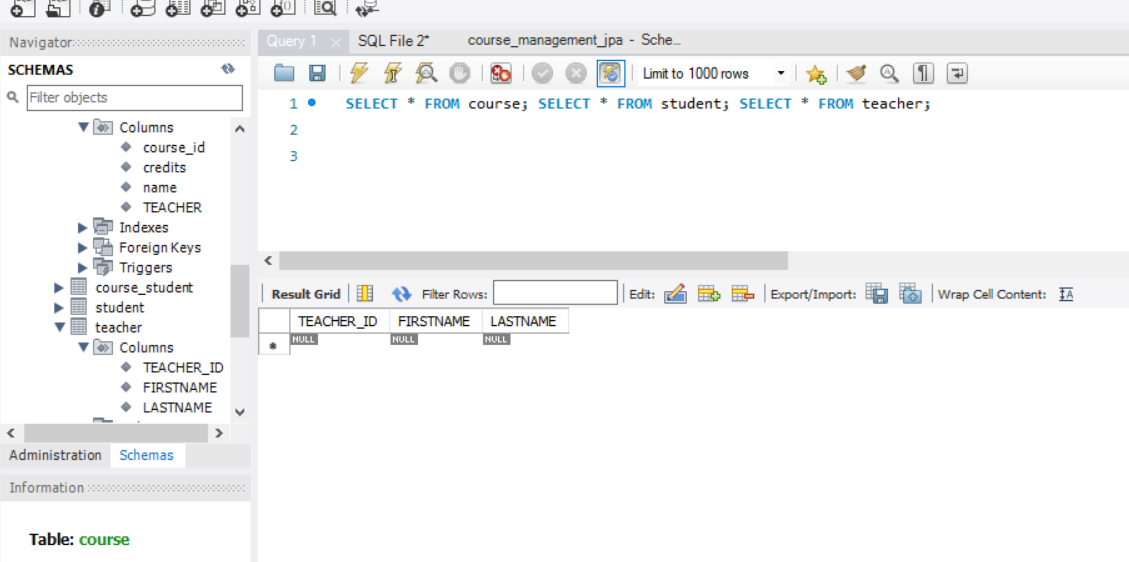


Figure 39 Generated tables from eclipse JPA

Have no clue why it turned the Columns into UpperCase though! Student and Teacher do it but course\_student and Course don’

sql: Connection(173738886)--CREATE TABLE COURSE (course\_id INTEGER AUTO\_INCREMENT NOT NULL, credits INTEGER, name VARCHAR(255), TEACHER INTEGER, PRIMARY KEY (course\_id))

[EL Fine]: sql: Connection(173738886)--CREATE TABLE STUDENT (STUDENT\_ID INTEGER NOT NULL, ADDRESS VARCHAR(255), FIRSTNAME VARCHAR(255), LASTNAME VARCHAR(255), PRIMARY KEY (STUDENT\_ID))

[EL Fine]: sql: Connection(173738886)--CREATE TABLE TEACHER (TEACHER\_ID INTEGER AUTO\_INCREMENT NOT NULL, FIRSTNAME VARCHAR(255), LASTNAME VARCHAR(255), PRIMARY KEY (TEACHER\_ID))

[EL Fine]: sql: Connection(173738886)--CREATE TABLE course\_student (course\_id VARCHAR(255) NOT NULL, student\_id VARCHAR(255) NOT NULL, PRIMARY KEY (course\_id, student\_id))

[EL Fine]: sql: Connection(173738886)--ALTER TABLE course\_student ADD CONSTRAINT FK\_course\_student\_course\_id FOREIGN KEY (course\_id) REFERENCES COURSE (id)

Well I guess this is why, it defaulted the columns to UPPERCASE:

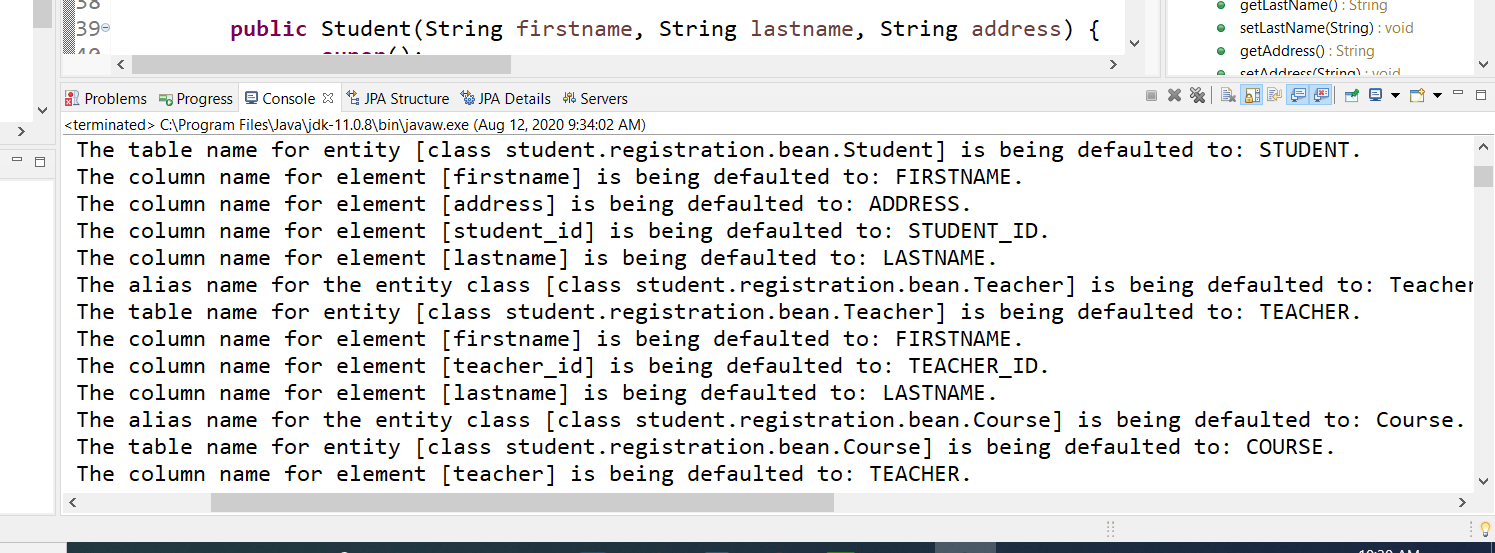


Figure 40 columns defaulted to uppercase

Just another problem to track down! ☹ Or maybe just let it go and move on.

After rebooting the JBoss Server I got this error:

failure description: {"WFLYCTL0080: Failed services" => {"jboss.persistenceunit.\"course.management-0.0.1-SNAPSHOT.war#course.management\"" => "org.hibernate.AnnotationException: Illegal attempt to map a non collection as a @OneToMany, @ManyToMany or @CollectionOfElements: student.registration.bean.Teacher.teacher\_id

Apparently it doesn’t like the teacher mapping..LOL I’m moving on to “*UsingJPA APIs to manage data” and using EntityManagerFactory.*

Use the name of the persistence unit in persistence.xml. “*Every service class will need access to EntityManagerFactory.”*

**7:00 PM** - So,,I got all the course service beans done, but the server isn’t displaying them,,Damit!

# Things they fail to mention

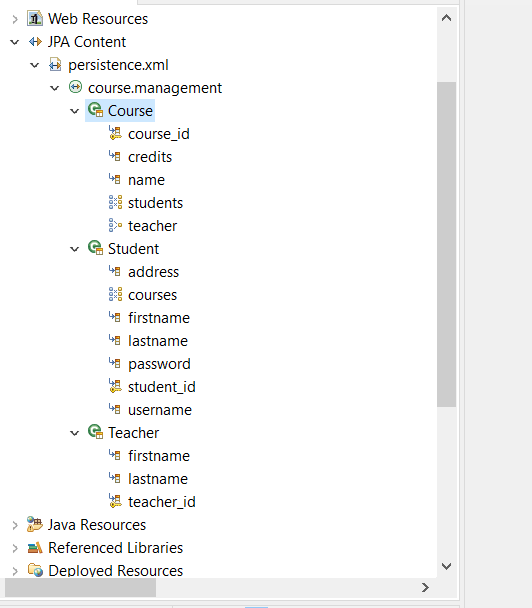
08/13/20 7:00AM – So this morning I’m going over everything before I move on into “Using JPA API’s to manage data” and discovered a few details left out. For instance, the code from “addCourse” in the book calls for “courseBean” file, but there is no course bean file. The course file is under the bean folder, so am I supposed to assume the software knows to look in that folder when you specify “bean”? It doesn’t mention anything like that in the book, I looked. I’ve done other tutorials where the files were actually called “\*bean” and in a bean folder. I guess it does! The files are listed in the persistence file as managed classes so,,,I don’t recall anyone mentioning that in any tutorials I’ve done.

Oh and another thing, not sure if I mentioned this yet. The server problems I was having for days at a time were caused because I actually **“*installed*”** the Tomcat, not just an **“*instance*”** of it, big difference!

# Modify an Existing Form

08/13/20 – There are many tutorials/books available to assist in JSP pages. I’m going to use the *javaguides.net* tutorial called: *“JSP Servlet Hibernate Web Application”* Basically it is a UserManagement App which will hopefully be suitable for my needs, communicating with a DB using a JSP form to Add/Delete users. Instead of users, I’ll use Courses. Sounds simple enough right?

08/14/20 7:00AM – This morning I discovered another interesting way to check on your dependencies in persistence files. I kept getting double columns as I stated previously, while investigating how it happened, this method to check what persistent columns are mapped in the table came to my attention:





*Figure 41: Persistence file entity attributes*

Notice in the Course entity, students and teacher have mapping indicators for many-to-one & one-to-many symbols. A nice way to verify your selections after using the JPA details in the console view!

9:30AM Here’s some weird shit! Check this out:

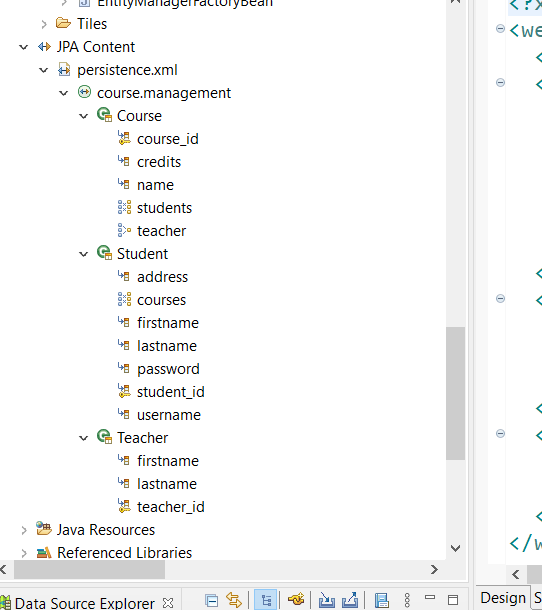


Figure 42: Persistence file columns and fields

This is correct, verified in the SQL Database. Now check this out:

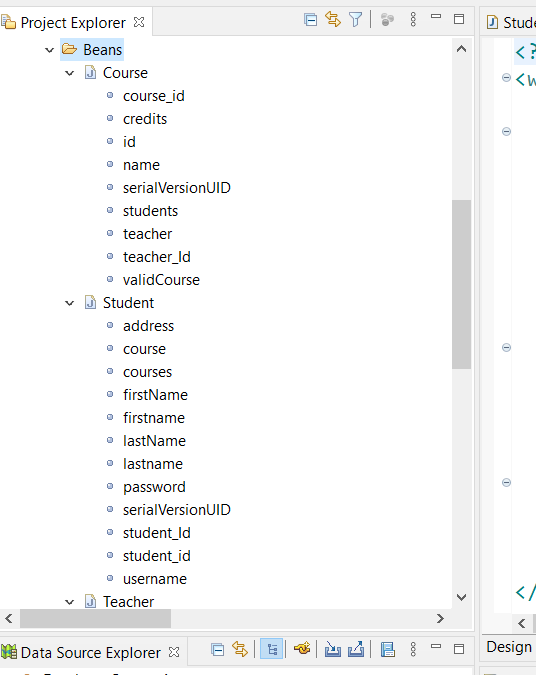


Figure 43: Beans fields under web resources

Both of these are under the web-resources bean directory. Another freaking thing to track down! Not sure if I’m up for it today.

11:00AM – Well there’s more than one way to fix the school\_management\_jpa database! 😊 I opened up the script file in Sql and corrected it. Got rid of all the double fields, saved the script as separate files for each table. Worked perfectly!

Finally! I imported the javax.faces.bean…..files and got a clean build for a change!

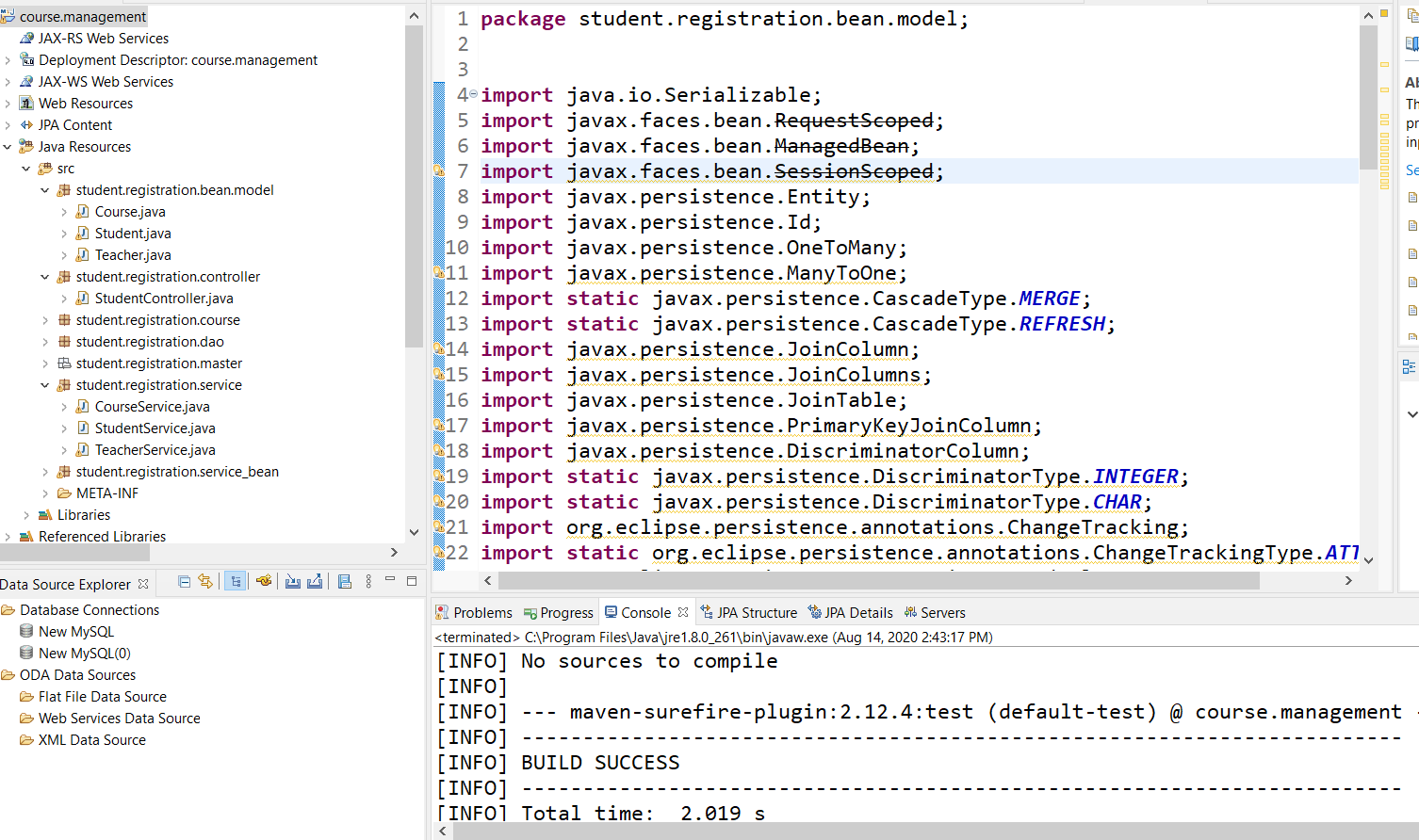


Figure 44: successful maven build

I spoke too soon apparently! Still getting an error:





Figure 45:illegal mapping error

Guess I’ll have to google that too! ☹

08/15/20 8:00AM – So, I get one problem solved and another pops up! I went and edited the relationships and joins so the last error got omitted, now I have this one:

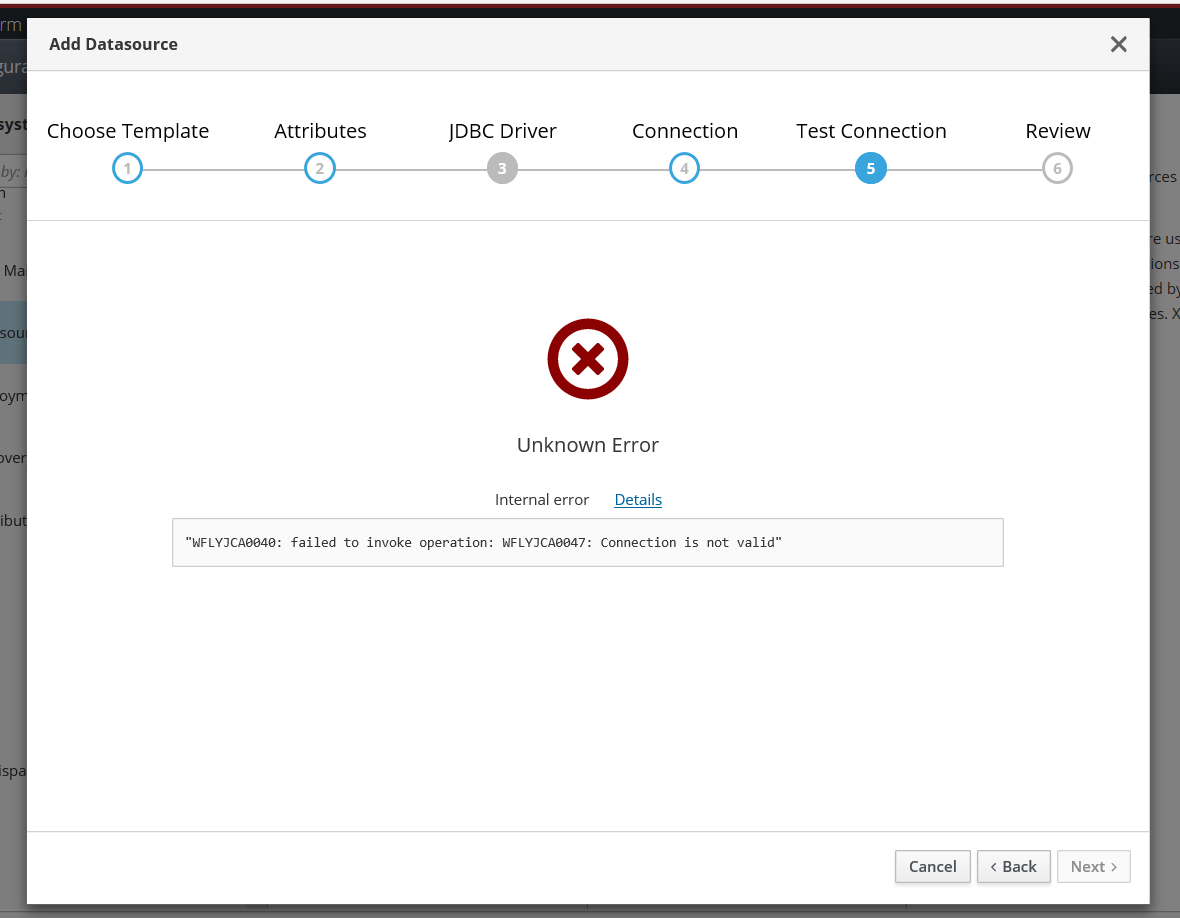


Figure 46: connection failure

So close! And yet so far! LOL I’ll guess my day will be googling that now.

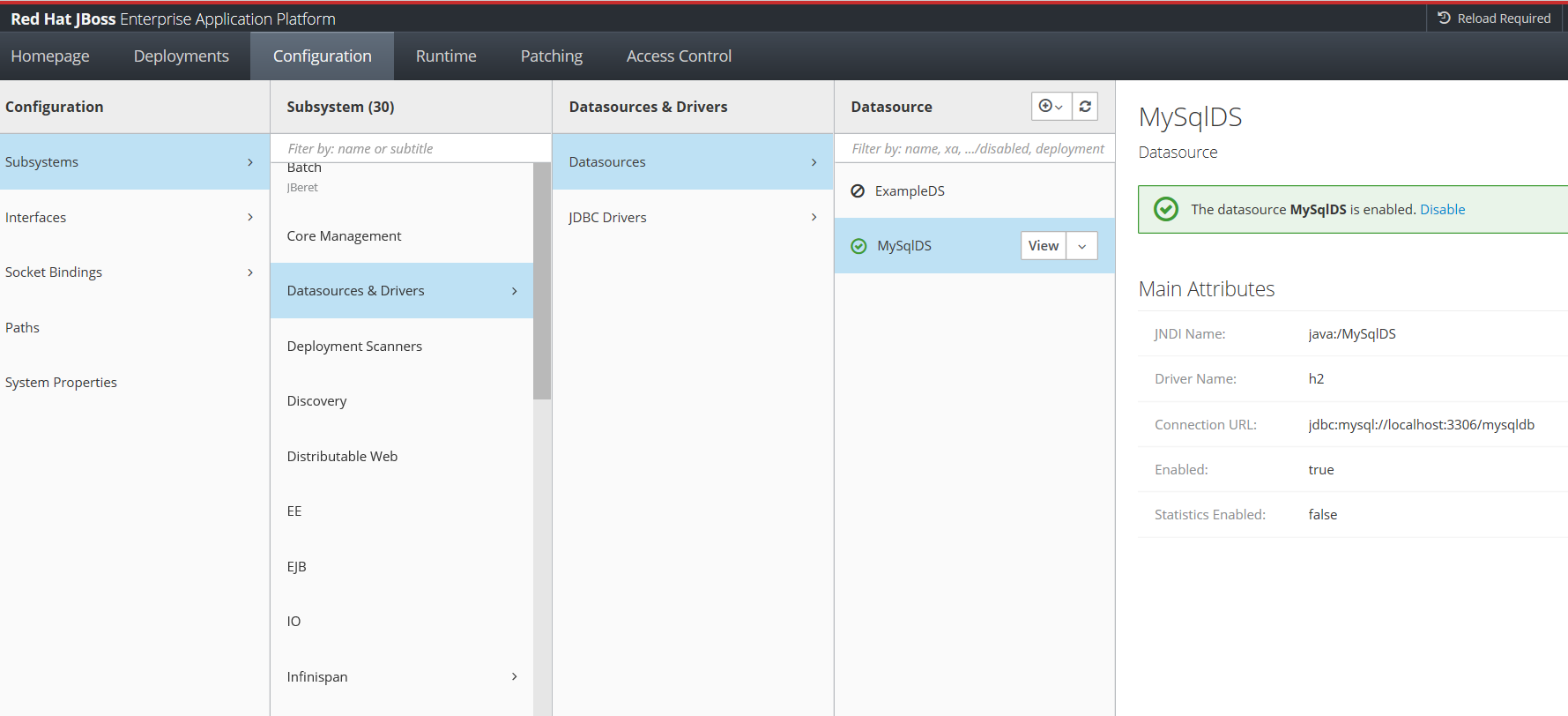


Figure 47: DataSource Error

So close! LOL

# Again with the Server Problems!

**08/19/20 9:00AM** - So,,here we go again! After much research and reading forums/user groups,I “think” my problems may be sorting out. Turns out it was user error after all! HAHA! Imagine that! Re-using code and copying whole projects and renaming them, probably not a good idea I’m finding out. All my problems start when I do that it seems. A wrong dependency here, missing dependency there, a wrong piece of code here, etc.. What finally caught my eye was an error that stated “illegal reference to another file or procedure” or something like that! I should have cut and pasted it! LOL 😊 Anyways, after freshly installing Java, Eclipse, Tomcat, etc,, I’m back up and running so far.

The Course Management Project is at Rev.3 now,,HAHA! Different approaches result in different obstacles & problems. My current approach is to have a Student, Course, and Teacher POJO. Pretty basic. Incorporating Maven, Hibernate, Apache 8.5 Server and Java 1.8 runtimes. Also the Context and Dependency Injection (CDI) nature. Start small and make sure the DB connections work is the first step with Student, Course, Teacher beans. Before I kept using student\_id, teacher\_id, course\_id as the beans. This time “student\_id” is just “id”, so that will be the model that ties everything together hopefully.

**2:30PM** – Just confirmed! If I put the forward slash “/” here:

*@WebServlet("/register")*

***public******class*** *StudentServlet* ***extends*** *HttpServlet {*

It works, if I leave it out, it doesn’t start! Weird! I have never learned that in a tutorial that’s for sure!

6:00PM – Here’s a bit of info to keep in mind:

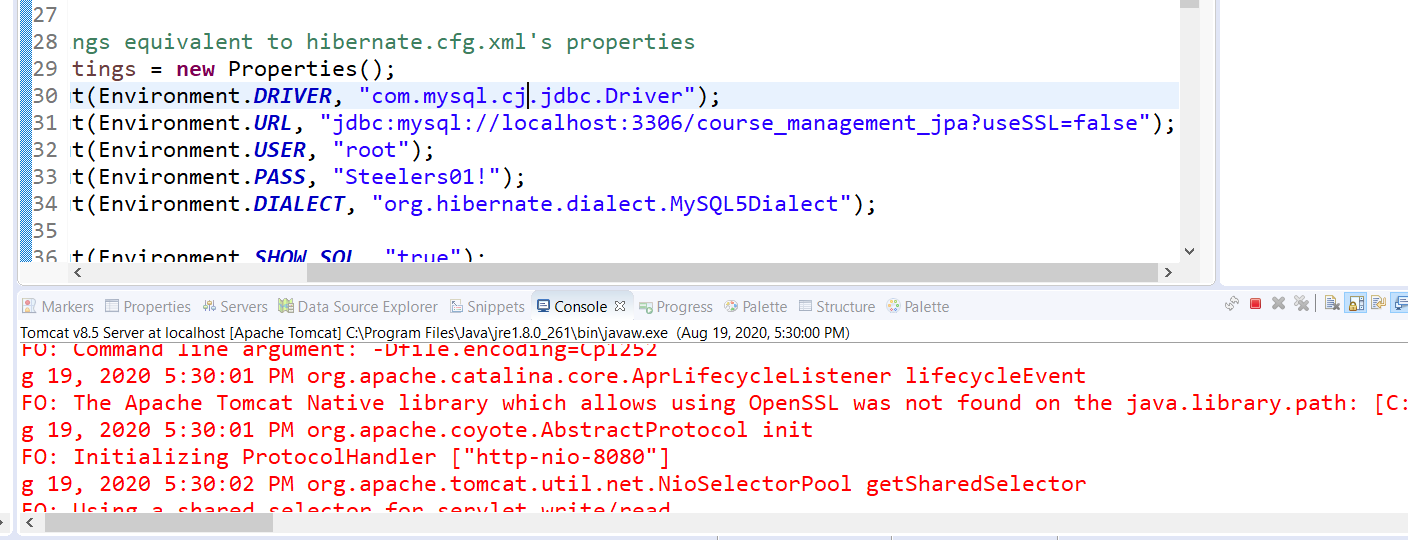




Figure 48:Connection errors

After rebooting the server after a few code changes, I got this error about SSL, I think I’ll have to turn it off in the server file, or make useSSL=true. I also saw this too in the error log:

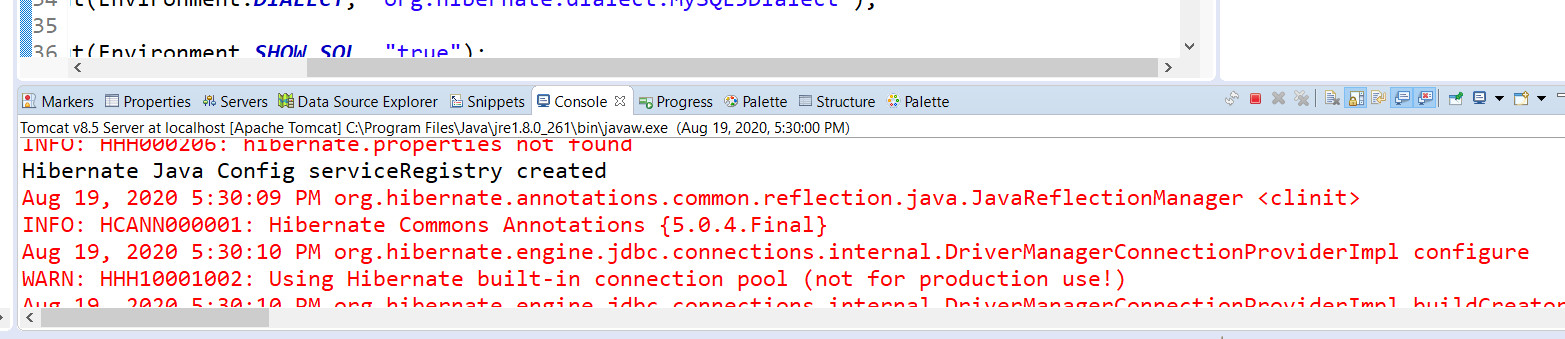


Figure 49:driver connection errors

At least I know I’m getting something again!

# I think I have it figured out!

**08/20/20 8:30AM** – After looking closely at the code the sessionFactory defined in HibernateUtil is not being called in the DAO class! It’s opening up a connection instead of a session from HibernateUtil like it should.

In StudentDAO the org.hibernate.Session was not imported, needs to be used with the org.hibernate.Transaction import.

# YES! I was right!

Success! Damn I’m getting good! Hibernate is working:

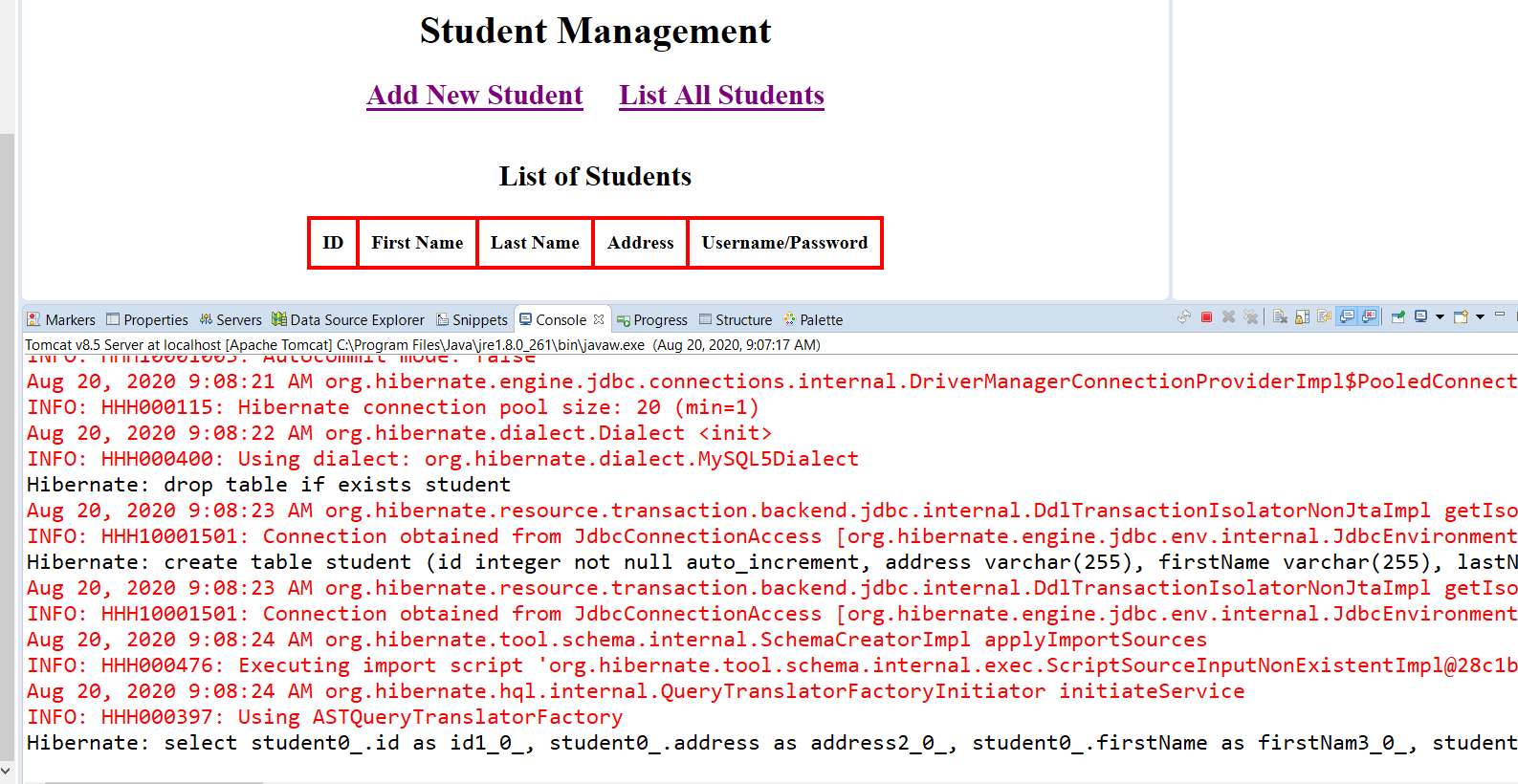


Figure 50:list all students query using hibernate

A thing of beauty! Add Student:

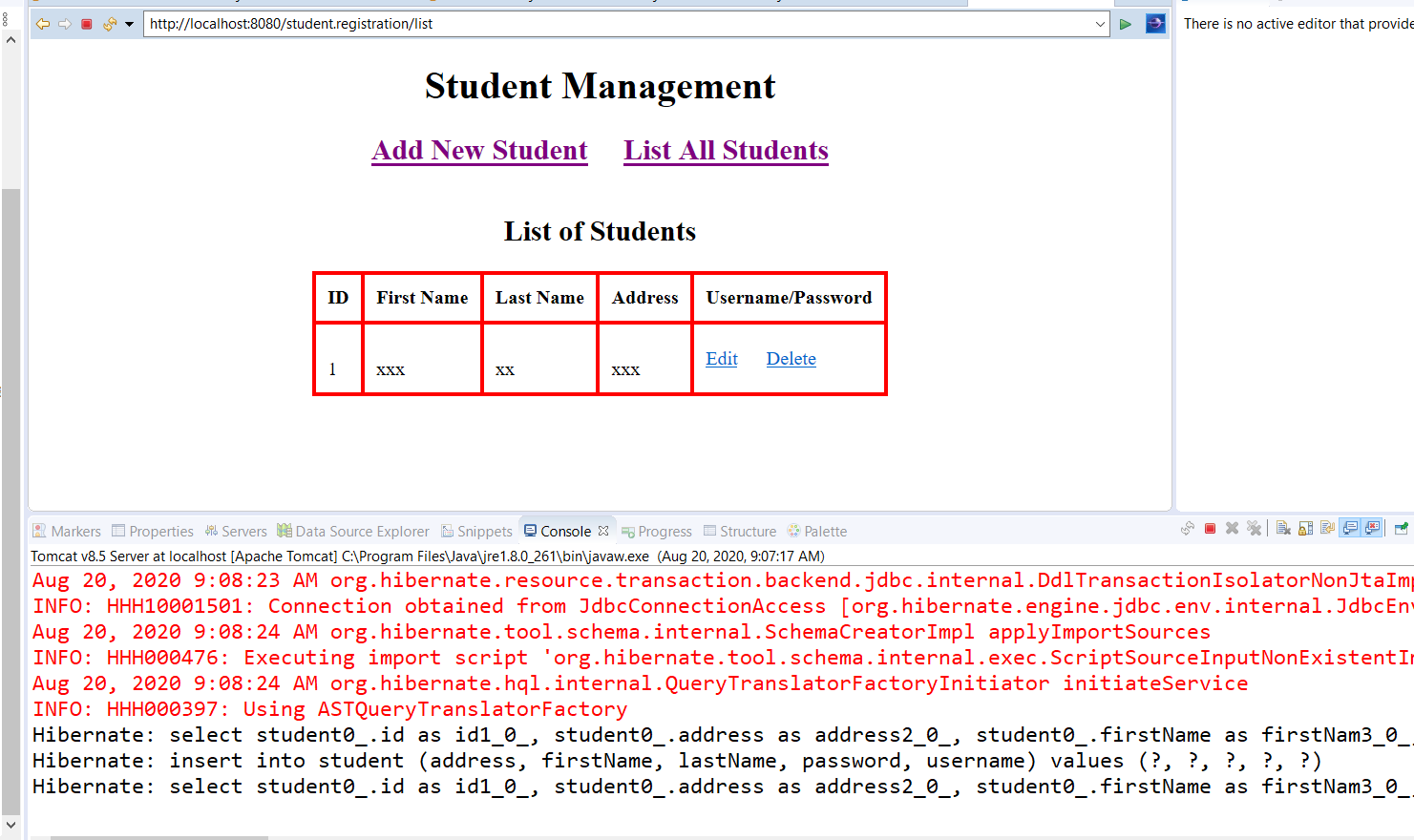


Figure 51:hibernate add new student

Just need to work a few kinks out,like why is it adding new student with a “0” ID?

OMG! It is working correctly! I have the Delete Table then Add New Fields in the Query. So it deleted the existing table of course. **IT’S WORKING! YAHOO!! HEBERNATE AND PERSIST!**

**Browser check is good:**

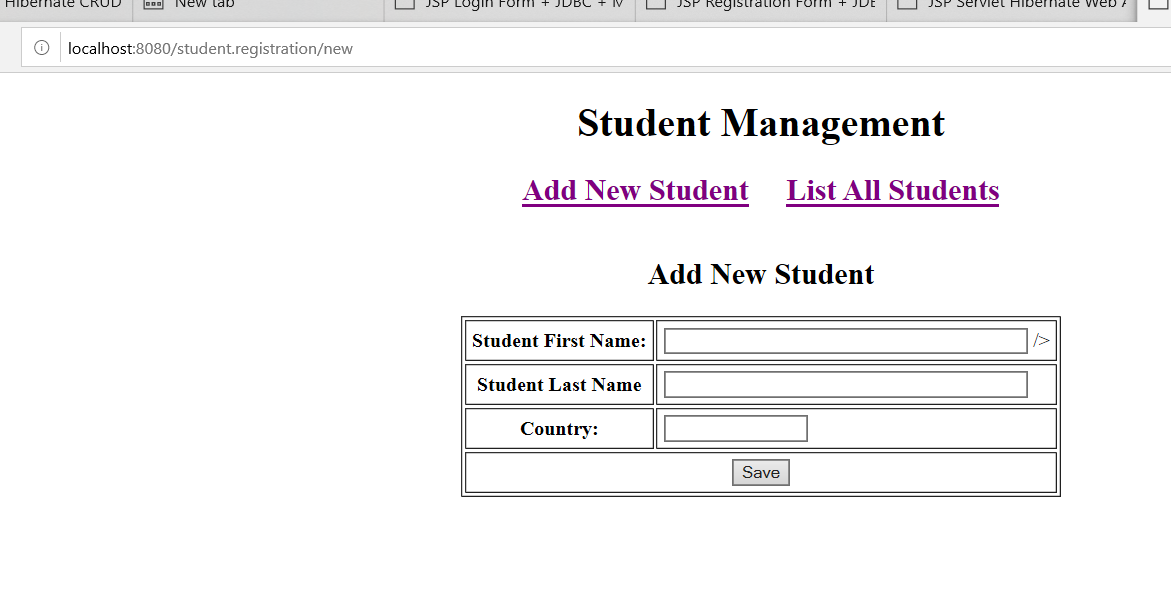
****

Figure 52: Add new student check

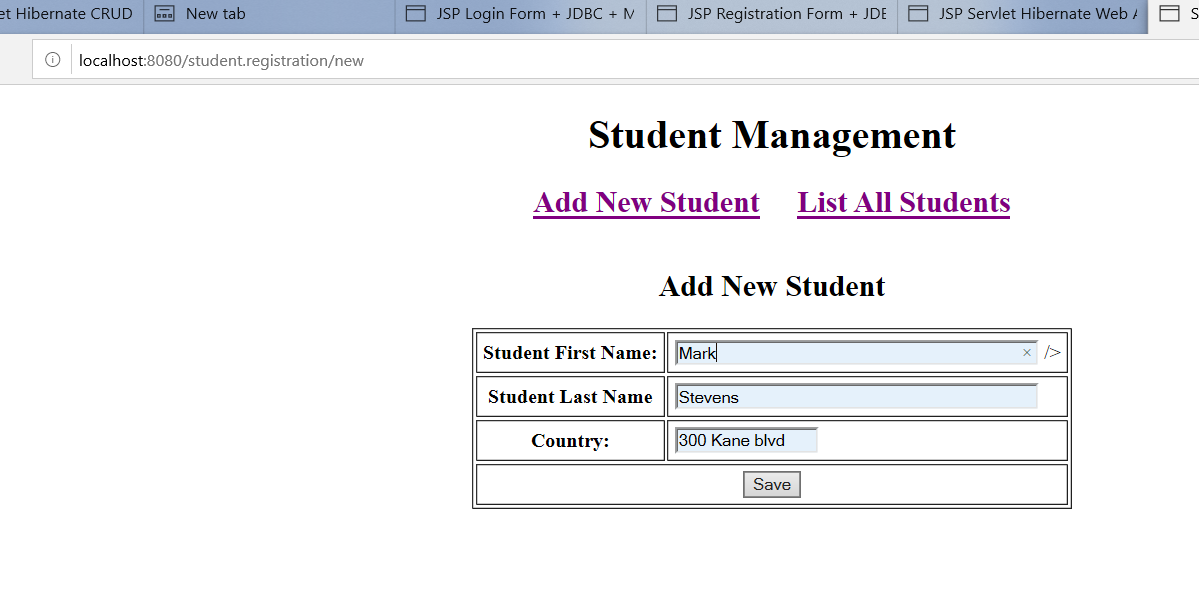


Figure 53: adding new student test

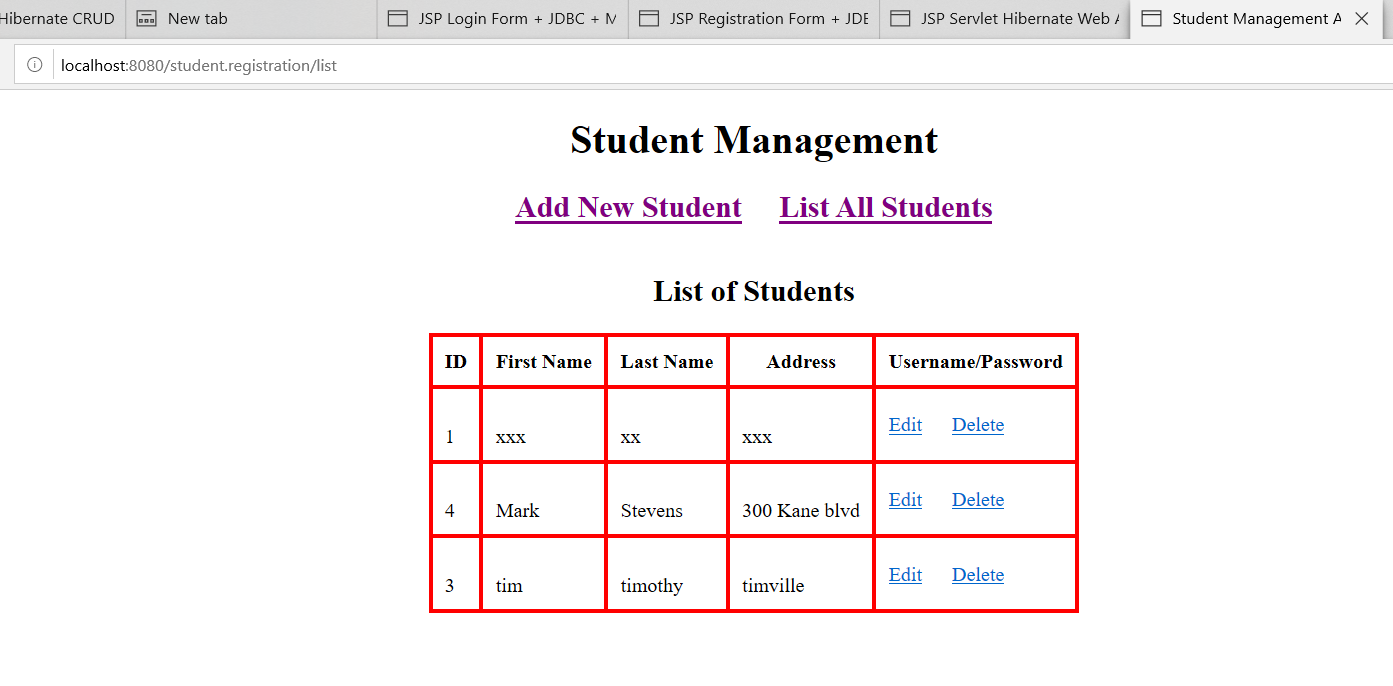


Figure 54: Add verification

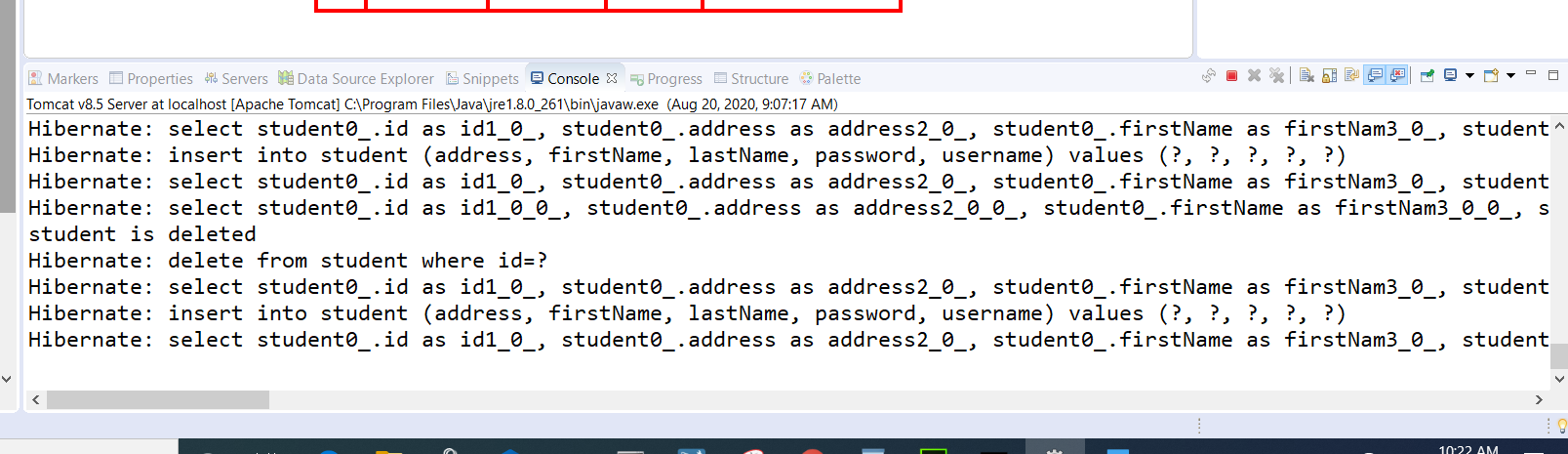


Figure 55:SQL statements sent to server

Everything looks great! I saved the project to my local Git Repository as CourseManagementRev3. Now the next step, do I try the Registration/Login form from W3Schools? Yes I am! LOL back to my old HTML/CSS days.

