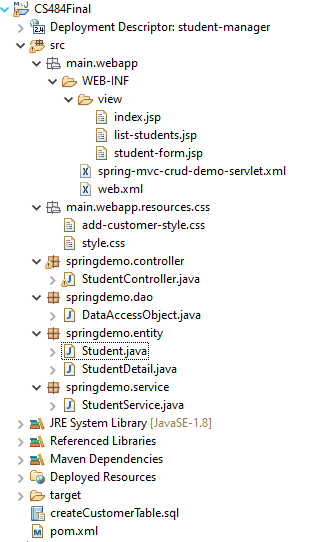
Donnie Beck

CS 484 Final

My project uses Maven to grab various dependencies, like Spring and Hibernate, from the web. The file structure is as follows:



Notice that I separated the project java files into four different packages, each signifying a different layer of the project.

The entity package which houses any classes that map to the entities within the SQL database. The two in this project are Student.java and StudentDetail.java, which map to the tables, “student” and “student\_detail,” respectively. Within the student class, the following annotations map student one-to-one with to its respective student detail. Notice that the CascadeType is set to all, this will make sure that if a student tuple is deleted, then the related student detail will also be deleted.



I have the DAO package, which houses my data access object class. This is the class that performs queries with the database. The @Repository annotation marks this class as the persistence layer for Spring. The SessionFactory object autowired in this class a hibernate provided object that does the actual querying of the database. Notice the existence of .saveOrUpdate() within the addStudent method which says to insert a new student tuple if one with the desired ID does not exist, or if it does exist, then update that tuple.

The service package serves to add a layer of abstraction between the DAO and the controller. It also utilizes the @Transactional annotation so that transactions are started and ended automatically before the method begins and after the method ends.

The controller is where the logic of the project is performed. Each method is called by the servlet when a specific mapping (URL) is accessed with the @RequestMapping annotation. Those methods then perform some action, then return a view (a jsp file). The controller makes extensive use of the Spring-MVC model to pass data between pages. When I use @ModelAttribute <type> <name>, I’m asking the model to fetch me an object from itself, that can be added through model’s addAttribute(<some object>) method.

As for my Java Server Pages, they’re all stored in another folder, src/main/webapp/web-inf/view. They’re stored here so that they’re within the scope of a maven web project (unlike how we stored them outside the src folder in our non-Maven projects). The same goes for my resources files. They’re stored in src/main/webapp/resources.