

Final Project

The purpose of this assignment is to produce a comprehensive multi-tiered enterprise application, consisting of a persistence layer, business layer, and presentation layer, also making use of services provided by the container such as security and transactions.

The assignment consists of a new Web Application project, and should build on your prior work using the Glassfish application server:

- Your MP3 domain model as the persistence layer
- Your MP4 EJB or Service components as the business layer
- Your MP4 security structure and configuration

For the final project, you will be adding **functionality** and **presentation** to your prior work using JSF or another MVC approach. You are always free to re-design or re-factor any of your prior work, but if you followed the MP3 and MP4 specifications, you should be able to use them as the **foundation** for your final project. You should expect to enhance and add to your business layer for the Final Project.

Graduate students are expected to complete the entire specification. Undergraduate students are encouraged to complete the entire specification, but can omit the requirement to provide functionality for creating a new user in the system.

Time Management

The Final project has a hard deadline of 12/12/2014. **No** late work can be accepted for the final project. Manage your time carefully. **Do not wait** until the last minute to work on your project. You are being given ample time to produce a comprehensive final project, and we expect your output to reflect that.

IDE and Tools

As mentioned in class, you are free to use whatever IDE, tools and libraries you choose. I do not want to restrict your creativity. As long as you carefully document your project and associated steps in the README, you are free to use whatever tools you chose. Other (other than EclipseLink) JPA providers include:

- Hibernate
- OpenJPA
- DataNucleus
- Batoo

The MVC frameworks we have studied include:

- JSP/Servlet (MP2)
- Spring MVC
- JavaServer Faces (JSF)

For the MVC portion of the Final Project you may use any of these, or another framework of your choosing if you would like to explore a different technology. Please let me know **in advance** if you decide to use something other than JSF, Spring or JSP/Servlet. My examples will be using JSF. JSP/Servlet refers to the approach we studied in MP2.

That being said, to keep things as simple as possible, I would **strongly recommend** using NetBeans and the EclipseLink JPA implementation as demonstrated in class. Shalini and I will be able to support you on NetBeans with the EclipseLink JPA library. If you elect to explore, please feel free, but budget your time accordingly to research any issues that arise. We will always try and help where we are able, but NetBeans and EclipseLink are the reference platform for the course.



If you wish to use Spring for the final project, then you would be considering a `@Service` class instead of a `@Stateless` EJB. Also, you might want to use spring-security instead of Java EE security.

Database Requirements

As demonstrated in class, create a new MySQL database schema named **itmd4515**. We will be using this database for the remaining projects this semester. If you have already created this database for prior projects, you do not need to recreate it.

In all cases, create a MySQL user named **itmd4515** with full rights to the database(s) as demonstrated in the first week. The user must have a password of **itmd4515**. Do not use the root user for persistence units. Use the **itmd4515** user and password for your persistence unit. This user should already exist from your prior MP2 work, but I am noting it again to be clear.

If using NetBeans, create a NetBeans database connection to your new MySQL database. This database connection will be used in the JPA tools within NetBeans.

Application Requirements



Important: Unless your implementation is **radically** different than mine, you should **not** be using my in-class domain model of Author/Publisher/Textbook/Chapter

Create a Web Application project named "UID FP" but use a web context of /uidFP. For example, my NetBeans project will be called "Spyrison FP" with a web context of /spyrisonFP.

Create a JTA Persistence Unit named UID_FP_PU connecting to your itmd4515 database using the itmd4515 user via a JDBC Resource. For example, my Persistence Unit will be named Spyrison_FP_PU. Use the persistence.xml configuration we discussed in class to drop-and-create the tables, and also to generate both create.ddl and drop.ddl scripts.

All MP3 and MP4 requirements apply to the Final Project, as related to the persistence layer, business layer, and security layer.

New requirements for functionality and presentation are as follows:

- Whatever MVC framework you select, make appropriate use of its capabilities for reusability, for example Templating and Composite Components in JSF
- Whatever MVC framework you select, follow the separation of concerns principle and make appropriate use of the various application layers. For example, use Backing Beans as your model and Facelets as your view in JSF.
- Provide functionality to create a new user account in your system. For example, on your login screen, "If you do not have an account, click here to sign-up" (or create an admin role that has access to that functionality)
- Provide functionality for finding and selecting entities based on various criteria. Consider your application roles when designing and implementing this functionality.
- Provide functionality for displaying entities in tabular format where appropriate, or in a single-record form where appropriate. Consider your application roles when designing and implementing this functionality.
- Provide functionality for adding, deleting and modifying entities. Consider your application roles when designing and implementing this functionality.
- Include appropriate navigation between pages, including the ability to return "home." Do not leave dead-end pages with no navigation. Consider usability in all aspects of your design and layout.
- Ensure server-side validation of user input, and display appropriate messages and navigation if user input fails to validate. For JSF, this means appropriate use of Bean Validation and/or JSF Validation.
- Basic CSS is highly encouraged.
- Provide enough sample data to demonstrate your functionality and security model, and to show that your application is usable and testable.
- Make appropriate use of logging and exception handling, such that the user receives messages appropriate for a user, whereas technical messages are logged

Your end result should be a fully functional application for your domain model.

I will provide a link to my in-class Example project(s) in Blackboard. You are welcome to follow the design and patterns I use in my examples, but if you find yourself doing so please make sure you provide a reference in your javadoc comments and/or README explaining that you based your project on "Instructor Example" or "Example from Web." Always provide references and citations when you use ideas from other sources.

If you find it beneficial to use additional libraries or frameworks in your project, you are free to do so provided you follow the general project requirements and fully document their use in your README. I encourage you to explore and be creative.



Make sure you are using standard Java naming conventions as enforced by our IDEs. These conventions used to be documented by [Sun Microsystems](#) but the pages are no longer maintained by Oracle in anything but a legacy format.

If you use code from a tutorial, hint or example, you must appropriately reference the source and give credit to the author. You may use your Javadoc comments to do so.

Extra Credit (5 points each unless noted)

We have covered some of these in class, and will cover others over the next several weeks. When you are assigned projects in the workplace, you will often need to research and apply new technologies independently, and it is always important for you to try new things. My hope is that the opportunity for extra points will encourage you to try these advanced Java EE concepts.

- Implement an **admin user and role** that can manage Users, Groups, and whatever other functionality makes sense for your application. It all depends on your design choices. Be creative!
- Incorporate the **EJB Timer Service** in your application to send an email using a **JavaMail** Resource for your application. This should make sense for your domain model. For example, send an email reminder based on an assignment deadline or an email reminder for an upcoming appointment. Be creative! This incorporates two things, so this is worth **10 points**.
- Use a **Stateful EJB** for some functional part of your application, such as a shopping cart. Be creative!
- Implement a **Web Service layer** that uses your existing Business layer to expose either a SOAP or REST API. Demonstrate this with either a test case, Java client, shell script, or batch script. This is a significant amount of work, so this is worth **10 points**.
- Implement a **JSF Custom Validator** OR a **Bean Validation Custom Validator**
- Implement a **JSF Custom Converter**
- Implement the **Comparable interface** on your Entity classes. This enables the use of many default sorting algorithms within the Collections framework. (<http://docs.oracle.com/javase/tutorial/collections/interfaces/order.html>)

- Incorporate a **front-end framework** of your choosing. If doing this, you may want to decide early so you can incorporate into your layout and user interface. Examples are Bootstrap, jQuery, etc.
- Incorporate PrimeFaces, RichFaces, IceFaces or another JSF component library into your application to explore the use of additional **JSF component libraries** beyond what basic JSF provides.
- Incorporate the **WebSocket** API into your application in a manner of your choosing
- Have your own idea for an extra enhancement? Email it to me!

Documentation


Comment your code appropriately. Use standard javadoc format. Generate javadocs and move to your web root. Link to your javadocs from your welcome/index page.

We will use Confluence for our documentation. In your Confluence personal space, create a page named **Final Project Readme**. Include the following sections:

- Project Summary
 - Use this section to describe the project in your own words.
 - How did you fulfill each requirement?
- Design
 - Use this section to describe the design of your final project's functionality.
 - What functionality did you implement?
 - How does navigation flow from one functional area to another?
 - Also, use this section to list any extra credit you have implemented, and how the additional features were incorporated into your design, including your insights.
- Development Insights
 - Use this section to tell me anything you want about the project, and your design/development experience during the project.
 - What did you learn?
 - Was there something you would like to explore further?
 - What did you like, or not like?
- Requirements (Installation, Compile, Runtime, etc)
 - Use this section to explain how I should install, build and run your project.
 - Write it step-by-step as if I do not have any knowledge of how to do so.
 - What are the versions of the tools, libraries and API's used in your project?
- Screen Captures
 - Include enough screen captures to illustrate your working project.
- Expected Results/Known Issues
 - Use this section to describe any known issues with your project. Nothing is ever perfect, and it is better to document issues than ignore them.
 - Also, provide me with a known working test script to follow when I run your project. For example:
 - Login as user1 with password foo
 - Enter a customer name of "Fred" in the search box
 - etc

Submission

Submit a zip file of both projects called uidFP.zip and submit to the Blackboard assignment link. For example, mine would be spyrisonFP.zip.

 Please perform an IDE **clean** operation on your project before zipping.

Late Final Project submissions can not be accepted. The assignment is due 12/12/2014 at 23:59.