**CST-361 Design Patterns in Java**

**Student** **Name**: Michael Weaver

**Project** **Title**: Activity 9: Research Enterprise Roles

**Date**: 02/17/2019

**GitHub URL**: <https://github.com/battousairurik/CST361_Design_Patterns_In_Java>

Application Retrospective

* *How did the final design and implementation of your application conform to an N-Layer application architecture?*

The final design contains a number of layers each communicating with the next to ensure proper application execution. The views communication with the business classes via EJB. These EJB pass between the top presentation later to the back business layer, and are intercepted by the logging layer. The EJB are passed through the business application layer to the database application layer, which handles all database transactions., Each layer is abstracted to pass a variety of parameters between each other but also encapsulated so that no other layer has any knowledge of the execution methods of any other layer.

* *Where could the design of the application be improved?*

With the music app, the most underdeveloped portion would be the presentation layer. Right now, there is only basic functions with little schemes or patterns. Implementing additional presentation patterns would give the app more interaction flexibility.

* *Are there other design patterns that could be used to improve the design?*

To review all the patterns listed in the book:

1. *Façade Pattern*: Currently being implemented by the business classes utilizing interfaces.
2. *Singleton Pattern*: Currently being utilized by the logger class.
3. *CDI Pattern*: Currently being utilized by the entire application
4. *Factory Pattern*: Would be useful in the creation of EJB.
5. *Decorator Pattern*: Would be difficult to implement effectively as the application does not need to decorate the beans.
6. *Aspect Oriented Programming*: Would be difficult to implement because the program is already arranged with the MVC pattern.
7. *Asynchronous*: Currently implemented by the RESTful web services.
8. *Timer Service*: Currently not implemented, might be useful for automatically logging users out or freeing up services when an error occurs.
9. *Observer Pattern*: Sort of implemented by the interceptor and logger classes.
10. *Data Access Pattern*: Currently implemented by the data access classes.
11. *Restful Web Services*: Currently implemented.

Industry Retrospective

* *How could the design and implementation of an N-Layer application architecture decrease the number of skills and resources roles required to build a complex enterprise application in Enterprise Java?*

Typically, a programmer that specializes in, say data processing, would have all the skills necessary to implement all the data processing options necessary for the project. With general programmers, you might need three different people whose skills match the workload because each one does not have the required knowledge to complete the layer in its entirety. With N-Layer you also have very specific requirements for each layer, which can be detailed out beforehand, allowing development for only what is needed. It is also easier to scale out each individual tier as the application grows in its resource needs.

* *How could the design and implementation of an N-Layer application architecture increase the number skills and resources roles required to build a complex enterprise application in Enterprise Java?*

If there are too many layers or each layer is too large, then there will be an equivalent increase in the work that needs to be done. Perfecting a multi tier application can be time consuming and require a large investment of talent. From personal experience, developing the 5 tiers of the music application required a lot of time and research, and that was only for very basic executions.

* *Given the task of building a large enterprise application (e.g., a customer management system, banking/financial application, insurance application) that also integrates with many internal and external systems what various architect and developers roles would be applicable to build such a system?*

Roles for the development and management of an N-Tier system vary based upon the scope of the project. Typically you could have a project manager for each tier, accompanied by numerous grunt workers. Other roles include database operator tasked with handling that specific tiers DB needs, and connectivity manager tasked with handling interactions with other tiers. Additional roles can appear based on the complexity of the project, which can be of a wide variety for N-Tier architecture.