Andrew Frieze
Homework Assignment – Architectural Design

Overview

As SoFi expands the development team, problems associated with how to effectively ensure that each developer is brought up to speed become more and more apparent. The SoFi Homework Assignment addresses these issues in an efficient manner. The homework introduces requires that the developer implement an application which utilizes many of the core components in the SoFi technology stack, while at the same time limiting the need to tie up a full time development resource.

For this implementation of the SoFi homework assignment, I have chosen to implement a very basic TaskList application. Users of the TaskList application are empowered to input new tasks and view previously entered tasks.

Requirements Analysis

For ease of reference, the requirement number of requirements that relate to user functionality are prefixed with "F"; those related to non-functional requirements are prefixed with "NF"; and finally, those related to technology requirements are prefixed with "TF".

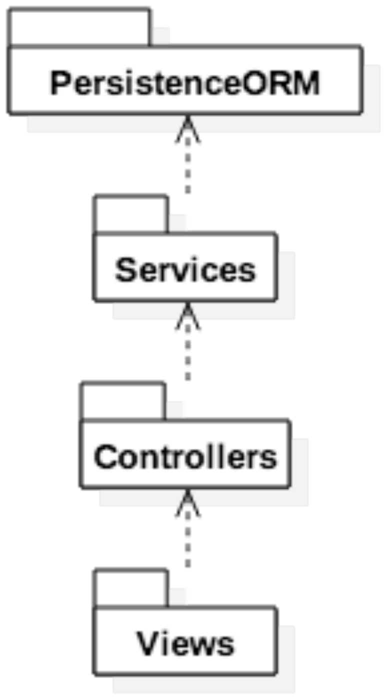
F1. A user must be able to enter a new task.
F2. A user must be able to view previously entered tasks.

TF1. SLF4J Logging must be utilized and demonstrating
TF2. Logging must be configurable through a logging.xml file
TF3. Data Persistence must be implemented, preferably using Hibernate
TF4. Spring must be used to inject Play application controllers.
TF5. Play framework 2.3 must be used
TF6. There must be at least one comprehensive JUnit test verifying one Spring injected object.
TF7. SoFi style guidelines must be followed.

NF1. At no point should any process conducted by the TaskList take more than 3 seconds.

Andrew Frieze
Homework Assignment – Architectural Design

UML Class-Diagrams



Major Architectural Decisions

The architecture style chosen was a Layered Architecture. This architecture serves to provide a a nice separation between the various logical “layers”. It should be noted that this is not a relaxed implementation, and it is expected that the controller will map data from the service package to view models which the views package has access to. This practice is considered correct, but heavy as it can lead to similar classes and boiler plate mapping code. Despite the downsides, mapping of data allows the view to function independently of changes made in the service data objects.

The MVC pattern was chosen primarily as this is the pattern implemented by the Play framework, which was a technical requirement of this project.

Spring’s Dependency Injection is utilized in order to inject Services and Controllers where applicable.

Reflections

Upon completion of this project, it is expected that I shall have a high level understanding of many of the various technologies utilized at SoFi. Implementation of the design will require setup and configuration of the components, and the functional requirements of entering and viewing tasks will ensure that process of “round-tripping” data is understood. Overall, this homework assignment should serve as an excellent introduction to the SoFi technology stack.