Spring Framework – 3 months training plan

The purpose of this document is to describe the schedule for acquiring core Software Engineering knowledge and skills, by solving a pragmatic business problem and providing an implementation on the very well-known Spring framework.

This three-month-plan focuses on the growth of pragmatic Software Engineering skills, provides an example which is very close to the real world, and is designed so that the person following it will be able to get familiar with many concepts which have to be known and practiced by a competent Software Engineer.

Let's begin!

What is Spring, and why?

Why are we discussing about frameworks and why Spring, to begin with? Well, there are a few wise reasons for which we are doing both.

However, to understand those reasons, we first have to explain what a framework is and why developers are using them, in order to lead a better life.

Frameworks are foundations from which a developer can start building a project. They are typically associated with a specific programming language, as Spring is for Java / Kotlin, or React Native is for JavaScript / TypeScript. What they basically do is allow the developer to focus on the actual business problem and implementation, rather than some low-level functionality that could have already be taken care of in an effective manner.

A few benefits of Software Development frameworks are:

- Establishment of better programming practices and fitting use of design patterns.
- More secure code.
- Avoidance of redundant and duplicate code.
- Consistent code development and lessening of risks / errors.
- Faster and more focused development.
- Easier testing and debugging.

Spring, specifically, is a framework which provides a comprehensive programming and configuration model for modern Java-based enterprise applications. As Spring's developers say, it focuses on the "plumbing" of enterprise applications so that teams can focus on application-level business logic, without unnecessary ties to specific deployment environments.

We have selected Spring since it is the most popular application development framework for enterprise Java. It is widely used in enterprises and can be utilized to develop many kinds of applications, such as web applications, or web services. It abstracts a lot of the "boring" stuff that developers typically do, and it creates a very pleasant experience for the person who is producing code, something which is often omitted, but shall be given special attention to.

The Exercise

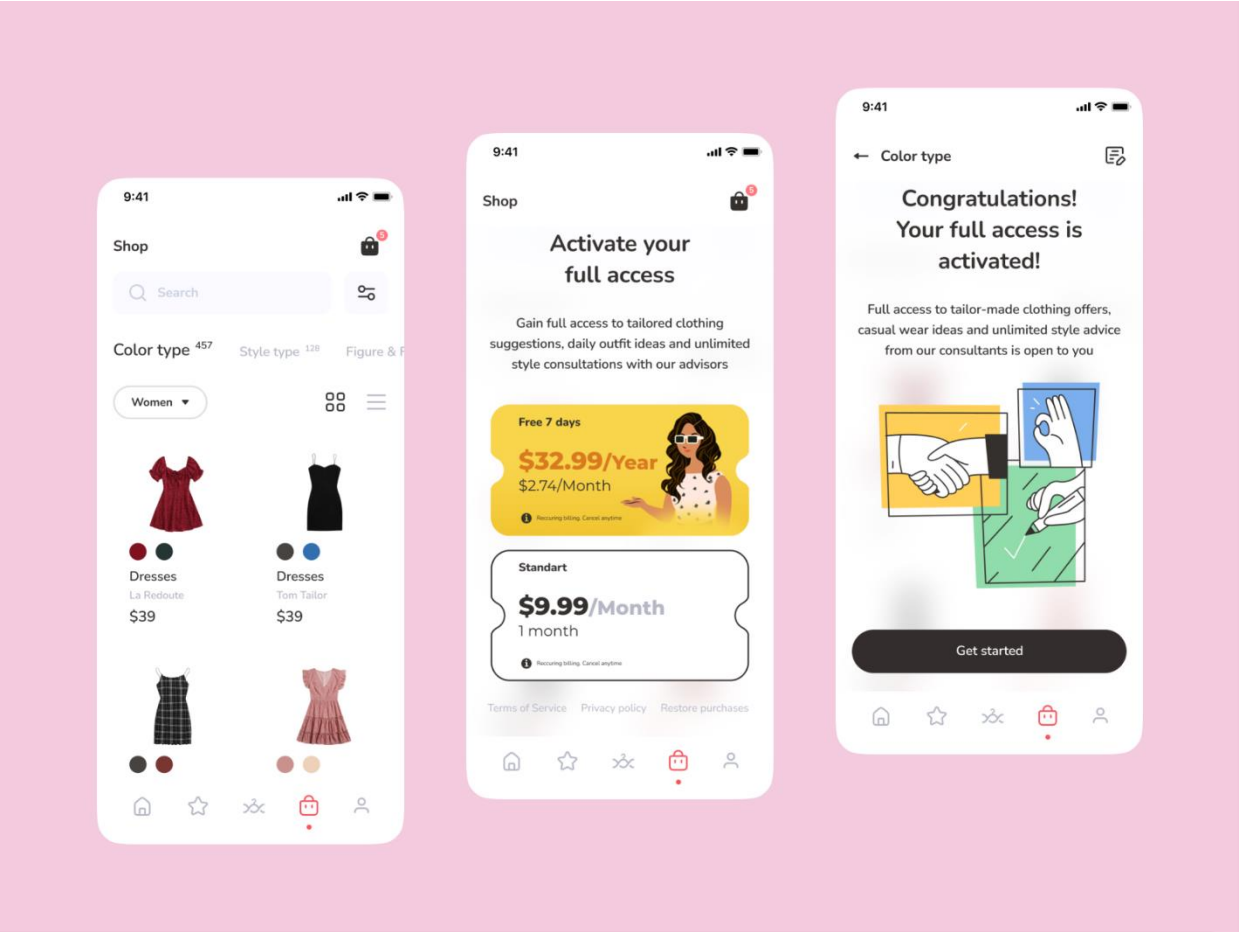
With that covered, what is the problem that we will be trying to solve?

Let’s say that there is a company in the apparel industry. We will name this company “Dresso”.

So, “Dresso” has hired an agency to develop a mobile application from which users can browse and order the clothes they make. This agency will not be responsible for any of the back-end services which are necessary for the operation of the mobile application, so “Dresso” is looking for someone else who can take that on.

That’s us! We will be responsible for the design, development, and maintenance of the database and all the back-end services which will be needed for the “Dresso” mobile application to be able to perform effectively and efficiently.

As a visualization of the mobile application, we can use the following:



The requirements that are given to us from “Dresso” are the following:

- They want their customers to be able to create an account with their first name, last name, postal code, address, city, country, a username, an email, and a password. For all the above, attention should be given so that they are legitimate values before they are accepted by the service.
- They want their customers to be able to sign in by using their username and password.
- They want each customer to start with 15$ in credits, so that they attract people on their mobile application.
- They want the user to be able to browse through the company’s clothes and see in which category they belong, a brief description, the price, the available colors, and a list of images.
- They want the user to be able to filter the clothes list by their name, or their category.

- They want the user to be able have a cart on which they can add / edit / remove items from.
- They want the user to be able to checkout their cart if they have enough balance.
- They want users to be able to “star” a piece of apparel, which is basically like putting it in a favorites list.
- Finally, they want the user to be able to view, edit and delete their profile.

Now, what this application would need to come to life is basically a RDBMS (a database which stores all application-related data), and a REST API which will allow the mobile application to easily perform operations with / on that data, without directly communicating with the database.

This is a very typical architecture for modern applications, where what you basically have is a persistence layer of data, and a few services (or microservices) with which clients can communicate via HTTP(S).

To implement this with Spring, we need to be familiar with:

- The Spring Initializr which will allow us to create the skeleton of our Spring-based project.
- General Spring-related knowledge, such as IoC, Dependency Injection, the various modules that Spring consists of, and Spring Boot.
- Entity-Relationship diagrams and domain driven design, for creating a valid and scalable database schema.
- RDBMS / SQL database engines to store our data. For this example, we will be using MySQL.
- Hibernate, which is an open-source Object Relational Mapping (ORM) tool for mapping an object-oriented domain model to a relational database. What it basically does is making extremely simple for us to work with the entities of our Persistence Layer, on a very abstract level.
- Spring REST Controllers. These will be used for creating the HTTP entry points of our web services.
- Spring Services. This will be the layer on which business functionality, validations and rules will exist.

The steps that are suggested to be followed are:

1. Understand what Spring and Spring Boot are, their core functionalities and how a project is created by using the Spring Initialzr website.
2. Read the requirements carefully and create the ER diagram for this specific domain.
3. Bring up a MySQL Database, based on the ER diagram you have created.
4. Create the Hibernate / JPA Entities on Spring and validate that they are working by creating tests.
5. Design your REST API.
6. Create the input / output Data Transfer Objects (DTOs) as classes.
7. Create your REST Controllers.
8. Create the services which will contain the business logic, validation, and rules, and connect them to your Controllers.
9. Verify that everything works as expected by using a client like Postman.
10. Develop Unit and Integration tests for a better maintenance experience of your application.
11. Aaaand, you are done! You have successfully provided the best services for your first customer.