# Testudo Bank Spring Documentation

*Owner: Ananya Ramkumar (ananyaramkumar06@gmail.com)*

# Spring Framework Overview

Spring is one of the most popular open source framework for developing enterprise applications. It provides comprehensive infrastructure support for developing Java based applications.

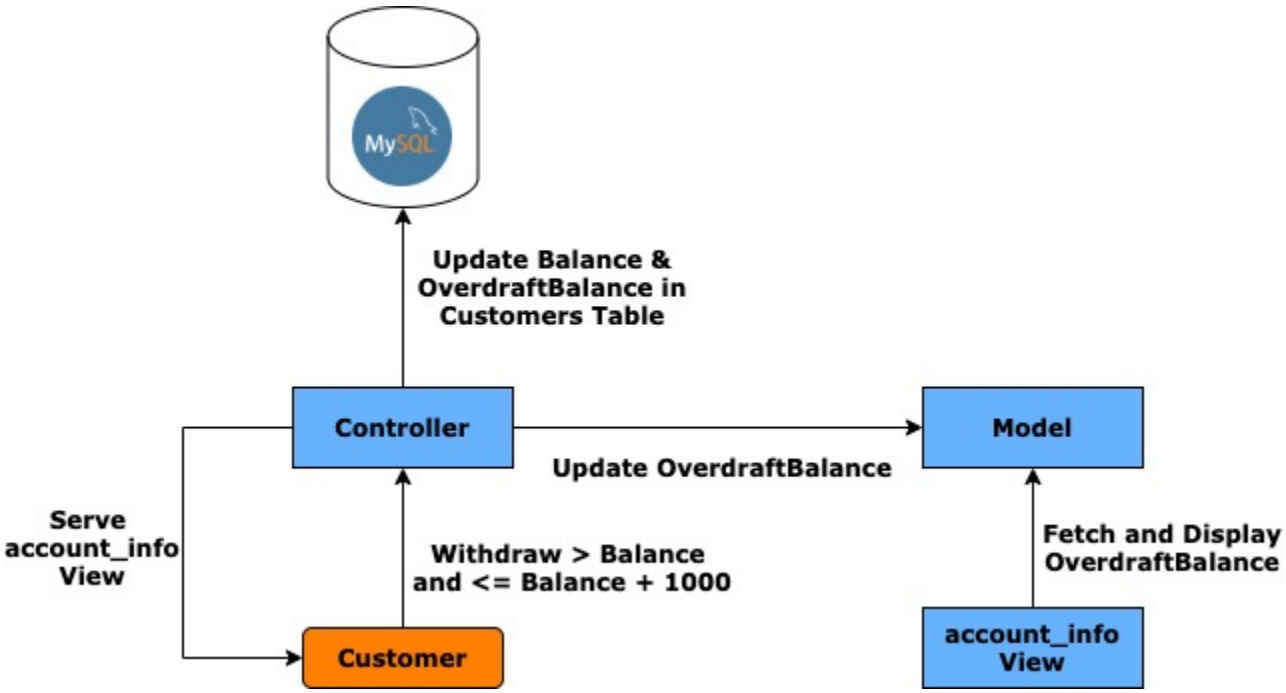
Spring also enables the developer to create high performing, reusable, easily testable and loose coupling enterprise Java application. Spring handles the infrastructure so for us so that we can focus on the contents of the application.

Examples of how you, as an application developer, can use the Spring platform advantage:

- Make a Java method execute in a database transaction without having to deal with transaction APIs.
- Make a local Java method a remote procedure without having to deal with remote APIs.

## Diagram of Spring interaction for Overdraft Feature

In Assignment #2, you will be tasked with building out a new feature for TestudoBank: the Overdraft feature. In the diagram below, you can see how Spring components play a role in this feature.




Let's break down this diagram and try to understand each part:

- **SpringBootApplication**
  - The `@SpringApplication` class is used to bootstrap and launch a Spring application from a Java main method. We see this annotation in our TestudoBankApplication class.
- **Controller**
  - Controllers provide access to the application behavior that you typically define through a service interface. Controllers interpret user input and transform it into a model that is represented to the user by the view.

```
@Controller
public class HelloWorldController {

    @RequestMapping("/helloWorld")
    public ModelAndView helloWorld() {
        ModelAndView mac = new ModelAndView();
        mav.setViewName("helloWorld");
        mav.addObject("message", "Hello World!");
        return mav;
    }
}
```

  - In this example, `@Controller`, and `@RequestMapping` form the basis for the Spring MVC implementation. In our case, the @Controller annotation is used in the MvcController class.
  - In our MvcController class, you will see that most of the methods have an annotation similar to @RequestMapping in the example above.
    - **@GetMapping** is a specialized version of `@RequestMapping` (https://docs.spring.io/spring/docs/current/javadoc-

api/org/springframework/web/bind/annotation/RequestMapping.html) annotation that acts as a shortcut for `@RequestMapping(method = RequestMethod.GET)`.
        - `@GetMapping` (https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/bind/annotation/GetMapping.html) annotated methods handle the HTTP GET requests.
    - Similarly, **@PostMapping** is a specialized version of `@RequestMapping` annotation that acts as a shortcut for `@RequestMapping(method = RequestMethod.POST)`.
- `@PostMapping` (https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/bind/annotation/PostMapping.html) annotated methods handle the HTTP POST requests.
- **Model**
    - Model is the part of MVC which used to handle the customer/user data passed between the database and the user interface.
- **Customer**
    - The customer is represented by our User class, which stores customer data including username, password, balance, etc.
- We also have 5 UI `views` for each of the pages that a customer may navigate to when visiting the TestudoBank web application. These include: 'account_info', 'deposit_form', 'login_form', 'welcome', and 'withdraw_form' which are under the webapp/WEB-INF/views directory.
    - These views are what the customer sees when interacting with our TestudoBank application. You won't need to be tweaking these files that much.
    - The account_info view is what customer sees when they log into their account. This view displays the user's first and last name as well as their balance.

## Further information and learning

Check out this official documentation for the Spring Framework to explore all of the different features it has: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/

- We aren't expecting or requiring you to read this, but it could be useful if you are trying to do some cooler or more technical stuff for your final project.

If you're a visual learner, there are tons of Spring MVC tutorials on YouTube that will take you through the process of creating a Spring web app. Here are a couple of ones that track relatively close to our current implementation of the TestudoBank application:

- https://www.youtube.com/watch?v=v9-3u5Hd8Qg&ab_channel=LearnCodeWithDurgesh.
- https://www.youtube.com/watch?v=dvL7Xp01HYc&ab_channel=CodeJava