Learning Spring Boot - Basics

By: Eileen Yee

CS 406 - Independent Project

Term: Fall 2023

GitHub Repository: <https://github.com/Yee152/CS-406-Learning-Spring-Boot>

This one-credit-hour project focuses on learning Spring Boot. Spring boot is a popular framework used by many companies in their tech department. During my internship, my company used spring boot as part of their tech stack. The goal of this project is to get familiar with spring boot and learn about building a spring boot project.

Spring boot helps users begin a project. In addition, Spring boot helps users easily configure projects and start coding, without the need to worry about issues that would arise from configuring projects. In addition, it is known to be easy to use because it provides the user with dependencies needed for their projects. Spring boot was designed to help users have production ready projects for a more streamlined project building process. Many people use Spring Boot to build REST microservices. Some actions you can use Spring Boot for are GET, POST, PUT, and DELETE. With my personal project I will mainly be focusing on GET and POST using Postman. Using Spring Boot, a programmer can easily start an application and connect to databases, configure a project and get started almost immediately. Spring Boot is used to build web and back end applications. In general, spring boot will have a client request, it will go through each of the following layers: presentation layer, business layer, persistence layer and the database layer and then send a response back to the client in the form of JSON, status, code, payload or image.

Steps to Setup Project:

1. Go to start.spring.io and generate a spring boot project.
2. Build project using Maven, Java 3.1.3.
3. Add dependencies to the project.
4. Started project up in IntelliJ
5. Open the generated spring folder in Intellij
6. Open the Pom.xml as a project.
7. Hit the play button to run spring boot on the demo folder after using spring initializer.

Databases for Spring Boot

Spring boot can connect multiple databases for any project. Spring boot provides a datasource that can connect to the database of your choosing. The rest controller can use the datasource to make connections and get the data.

Dependencies for Spring Boot

Spring boot is built upon Spring framework and one of the key features of Spring framework is injecting dependencies. The “@” sign will help inject dependencies into Spring Boot using annotations. Some of the key ideas are that an object is created by the spring container, the developer will then get this object in our application. This object is known as a bean, which is a dependency injection. Otherwise, a bean is simply one of many objects in your application. Even if we do not want the object the spring framework will give it to us. In addition, Spring annotation is a component of component annotation. This will help instantiate components and inject them into Spring. Where Instantiating things create new instances of objects to be used. An external service will inject the dependencies, they are then responsible for adding the dependencies. There are many other annotations that can be used depending on the type of project desired to be created.

Applications typically need extra objects,typically we also need to initialize objects. Spring will give you the object instead. In Java if you want to use an object, you have to call the method. You are not responsible for creating the object with spring framework.

Spring Framework

Behind the scenes we have a bean factory, where typically we do not want to instantiate objects ourselves either. We can also search for dependencies from Maven and paste them into the POM.xml file. Every bean will have an id, so now you can say when you have a class of “something” then you will give the object of “something.

Annotations used in my program

@Qualifier:

* Eliminate which bean needs to be injected

@Autowired:

* Making a dependency injection disambiguous
* Injects the actual service into the constructor

@PostMapping:

* Telling spring it is a post request

@Component:

* Allows for custom bean detection

Creating Controllers

Applications will have many services, if you have certain requests a server will handle, the controller will handle those things.

According to the Spring Boot Docs:  
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. Spring implements a controller in a very abstract way, which enables you to create a wide variety of controllers.

Restful Applications

Controllers mainly control how data is displayed, it seems like

In conclusion it seems like with Dependency injections we can split things into layers.

@Controller

Spring Boot will take care of the configuration, but if you want more specific configurations you have to go and code that information.

Rest Controllers

Spring boot uses two main controllers, either @Controller or @RestController. The main difference between the two is that @RestController is a combination of @Controller and @ResponseBody. According to BaelDung, @ResponseBody primarily returns the information that the object is already in a JSON format and passed back into HttpResponse.

@Restcontroller

* Returns the JSON response
* So that we can have a method and expose some endpoints that clients can consume, clients being React, android etc…

Spring Framework and Spring MVC

The idea of using Spring Boot is to make things easier on developers, before Spring Boot we would have to do a lot of configurations with Spring Framework. By using Spring Boot, developers would not have to do as much configuration by using Spring Framework. Spring Boot is an open source framework based on Spring framework.One module of Spring framework, is the Spring Model-View-Controller. This allows the separation of business logic, presentation logic and navigation logic. Meaning it allows for the separation of tasks. Spring Boot user to configure and map certain methods by using annotations. From my previous writeup I talked about annotations, where Spring annotation is a component of component annotation. This will help instantiate components and inject them into Spring. Where Instantiating things create new instances of objects to be used. An external service will inject the dependencies, they are then responsible for adding the dependencies. There are many other annotations that can be used depending on the type of project desired to be created. Developers can use spring.io to find relevant documentation and information, it is an open source platform.

Spring MVC

We have a client that sends a request to the server and the server sends a response back to the client, client needs data, but wants the data in a well formatted way, Servlet on the server side will create the layout and data. Where a servlet is a Java class that extends the capabilities of servers. Servlets are single action controllers. So someone needs to give a task, design and something needs to accept the view and model. Basically in the server, a controller will accept the request and send a response. In an instance we have multiple controllers as we would need a website, user, account, laptop, this would need multiple servers, so you would need to make a servlet for each request, one servlet can only handle one request. Using Spring MVC, we can get rid of the front controller, where the old way of using this was sending a request to a front controller that would send it to the correct controller. In general, users will send a general request, then the request goes to a dispatcher servlet, and then that goes to the user controller, then the dispatcher servlet knows where to send it because a configuration file is made. Spring Boot takes care of this configuration.

Spring Boot

Much of the time, Spring Boot allows us to focus on the logic and it will focus on the convention. The properties file will take care of configuration.

Advantages:

Spring Boot has many advantages, including built in servers, can be built as a standalone application, has some default settings for a speedier development environment, integrates easily with the Spring framework, can build microservices, and is easy to test among other things.

Disadvantages:

Some disadvantages are limited personalization, most of the things you use spring boot are, are already made. In addition some dependencies are bundled in already, and increased memory consumption among other things.

API Endpoints

API stands for application programming interface, where it allows for communication between applications. According to the Postman blog post, API endpoints is a URL that acts as a point of contact between an API client and an API server. API clients send requests to API endpoints in order to access the API’s functionality and data. Basically API endpoints connect API clients and API servers.

Postman

In this case, we use Postman to design, develop and consume API endpoints in regards to our Spring Boot project.

Requests to API

According to the documentation it must include a method, which indicates the operation to be performed.

Spring Boot Endpoints

According to the documentation for Spring Boot, it has a number of built in endpoints and lets you add your own. Endpoints can be enabled or disabled.

Application Properties

Define connection details to the database you have made.