Spring Boot Documentation

Icon

Description automatically generated

Spring boot is a framework for the Java programming language. It allows you to create web applications with minimal fuss. It’s supposed to be easy to setup with minimal configuration.

I’m going to use it to create an API for my frontend application. The spring boot app is going to be serving data that my frontend will fetch.

# Getting started

To get started you don’t need much, you don’t even have to install anything. You do need an IDE ofcourse, so if you haven’t already, download and install an IDE of choice. The most used ones for Java programming are Eclipse and Jetbrains’ IntelliJ Idea. I’ll be using IntelliJ Idea.

If your IDE is setup and ready, go ahead and go to <https://start.spring.io>. This application is what you’ll use to generate your project and will do most of the setup for you. When you open the application you’ll a whole setup screen, which gives you quite some options.

First is the type of project. You can choose between Maven and Gradle. These are just tools which help you create software. They have different focuses so choose what’s best for you. Language is the next step, it lets you choose between Java, Kotlin and Groovy. If you’re experience with one of these go for that one, I’ll be choosing good ol’ Java.

Next up is Spring Boot version, choose something different if you know what it means, if not (like me) it’s better to just go with the version already selected, in this case 2.7.4. Project Metadata let’s you give your project a name, group, description, stuff like that. The thing I said about Spring Boot version also goes for Java version. I recommend only changing the version if you know what it means.

After all that you can add dependencies. Obviously this is going to change based on what you need, but since I’m going to be building an API, we most definitely need the “Spring Web” dependency. Chances are high you probably do too. I’ll be using a MySql database so I’ll be adding the “MySql Driver” dependency.

Click on “Generate” and Spring Initializr will generate your project and you can download it as a zip file.

# The Application

When you open the project, you’ll see that it’s REALLY minimal. Like, there’s almost no files. We’re going to be building an API, so we better start creating a simple folder structure for the different types of files we’re going to need. So navigate to “src/main/java/[GROUPNAME + PROJECTNAME]” .

Inside of that “package”, as it’s called in Spring (why?), you’ll see a file called [PROJECTNAME]Application. If you open it you’ll see that it’s just a class with one main function in it. In IntelliJ Idea, to the left of the method, in the bar where the line numbering are located, a green “play” button will appear. Clicking that button will run your application. You can also run your app by running commands. The following command works if your project is built using Maven:



The following command works if your project is built using Gradle:



# Building the API

## Controller

For the actual API itself, we’re going to need some folders/packages. Create them in the same [GROUPNAME + PROJECTNAME] package that you’re already in from the previous chapter. Create a “Controllers” package. Go ahead and create a class with the name of the controller you need, mine is called “TaskController”. Whenever a frontend project wants to make an API call for the “Task” entity it will be using controller-methods from this class.

First of all Spring Boot needs to be told that this is a REST API controller. That can be done by putting the “@RestController” annotation directly above the class.



Next is tell Spring Boot which path should be associated with this controller. Preferably you want to give the name of your controller to your path. For example, my controller is called “TaskController”, so I give the following as path “/tasks”. You can specify this using the “@RequestMapping” annotation. This is what that will look like:



If another app is going to call this API, chances are high that the request will be blocked by CORS (Cross Origin Resource Sharing). That means that it will block other applications from trying to access your API. That’s fine for applications that are not yours, but not for our React app. We’re going to have to specify our React app as an origin that shouldn’t be blocked, by using, you guessed it, an annotation! Which will look like the following:



Next is to create a controller method. Your controller will contain multiple methods that will executed varying tasks. I called my main method “index”. To test this method you can return a hardcoded list or some plain text. If you would like this method to run on specific url, you can add the same “@RequestMapping” annotation to the controller method. I specified just a “/”. So the full url needed to run my “index” method inside of the “TaskController” is [APPURL]/tasks/.

That’s pretty much controllers in Spring Boot.

## Model

A model is a class that represents a single record of a database table. Let’s say you have a “users” table in your database. In that case you would create a model called “User”. So the name goes from plural (users) to singular (user) and you capitalize the first letter. The columns that the table contains should be translated over to class properties inside your Spring Boot app.

If your user table has, let’s say, a username column, that becomes a “String username” property inside your model class.

That model can than be used to represent your database table inside your code. You can see in the example below that each field has “@Getter” and “@Setter” annotations. Those are provided by Lombok if you’re interested. Those annotations lessen the amount of boilerplate code.

