How to build web services using SpringBoot

Introduction

The guideline below will demonstrate how to rapidly build a an web service using SpringBoot.

The tutorial will create an SprintBoot application.

It will also show how to call multiple upstream APIs using SprintBoot RestTemplate and aggregate the results.

Tooling

Java SE Development Kit (JDK): This is the standard Java platform for software development. The version of Java used for this article is 1.8.

Visual Studio Code: Visual Studio Code is a free platform independent source code editor for software development. It includes support for debugging, Git integration, syntax highlighting, code completion, refactoring in a rich array of languages.

Spring: A Java based Application and Inversion of Control Framework for building web applications. The version used for this article is 5.

SpringBoot: Spring Boot is an open source Spring-based framework used to create production grade web services and REST APIs in quick time.

Apache Maven: Maven is a build automation and dependency management tool primarily used for Java based projects. The major version used here is 3.

Prerequisite

Basic knowledge around Java and Maven.

Basic Knowledge around MVC architecture

Project Setup

We will use a VS code extension called **Spring Initializr** to create a springboot application quickly. To do this please follow the steps below:

To install, launch VS Code and from the Extensions view (Ctrl+Shift+X), search for vscode-spring-initializr.

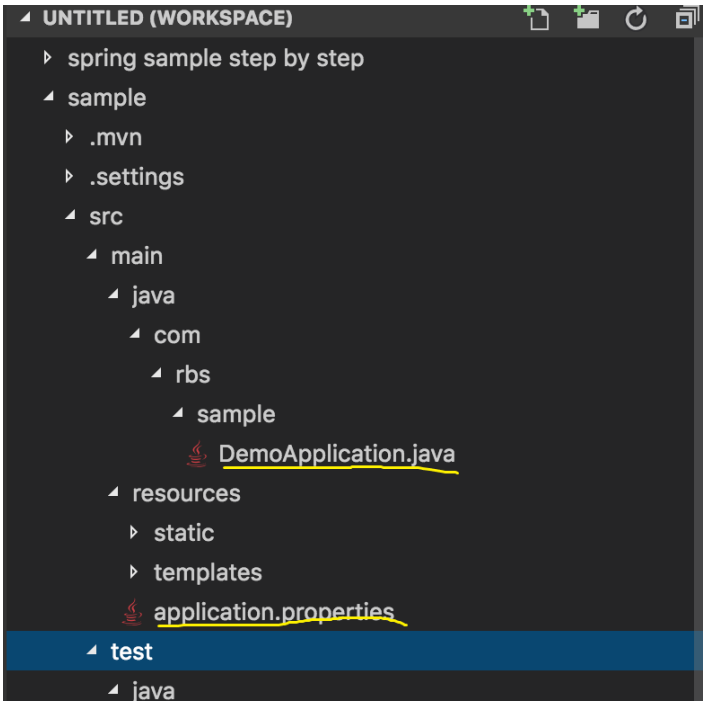
Once you have the extension installed, open the Command Palette (Ctrl+Shift+P) and type Spring Initializr to start generating the application.

Follow the wizard to select a Java and Maven based Spring application.

{screenshot here}

Key Files

Once the setup is complete, you will see a project structure with some key files has already been created for you.



pom.xml :

This is where all the dependencies are listed and will be resolved when you build and run your project.

```
<groupId>com.rbs</groupId>
<artifactId>sample</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>

<name>demo</name>
<description>Demo project for Spring Boot</description>

<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.1.1.RELEASE</version>
    <relativePath/> <!-- lookup parent from repository -->
</parent>
```

You will find the the dependencies required for the SprintBoot are already added.

application.properties

Under src/resources, you will find the file where all your project configuration would go.

main

Under src/main you will find the main class with it's static main, which will be the application entry point.

```java
package com.rbs.sample;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoApplication {

    ▶ Run | Debug
    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
```

Controllers

At this point, you are ready to create our first controller. In an MVC pattern, a controller handles the routes and data flow when a request comes in.

In this example, we will create a simple REST controller, which will return a text.

Firstly, we create a new folder under 'sample' called controllers, and create a new java file named HelloController.java.

Import necessary Spring libraries to create a REST endoint.

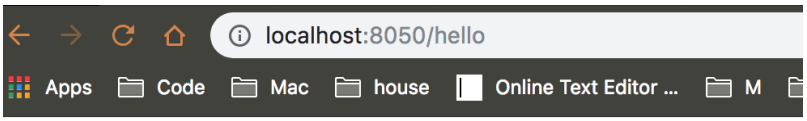
```
package com.rbs.sample.Controllers;


import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController {
@GetMapping("/hello")
public String index(){
                return "Hello from Sample RBS Spring App. Your first
Spring Boot Web app is here";
        }
}
```

The application will run on an inbuilt tomcat server on localhost:8080. Use application.properties file to change the settings.

```
server.port = 8095
```

Press F5 to start debugging the application. The debug console will load the libraries and call the application entry point. At this juncture, your application is ready to receive it's first request.

Go to your browser or postman, and fire http://localhost:8095/hello. This will output the following.



Hello from Sample RBS Spring App. Your first Spring Boot Web app is here

Your first SpringBoot application is now up and running.

Next Chapter: Advanced Route mapping and Returning Objects as Response