Introduction to building web apps using Spring Boot (Part 1)

Overview

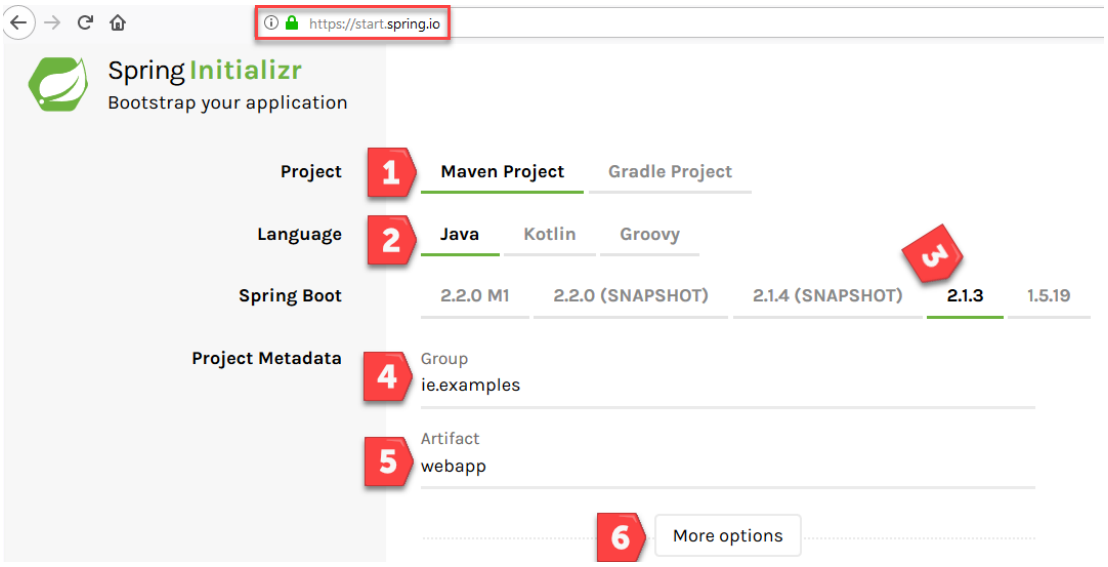
Spring is an application framework for the Java platform which includes a wide range of extensions and tools for building web applications. We will be using Spring Boot, a version of the framework designed for getting applications up and running quickly. See https://spring.io/projects/spring-boot

This tutorial shows how to create and run your first Java Spring web application.

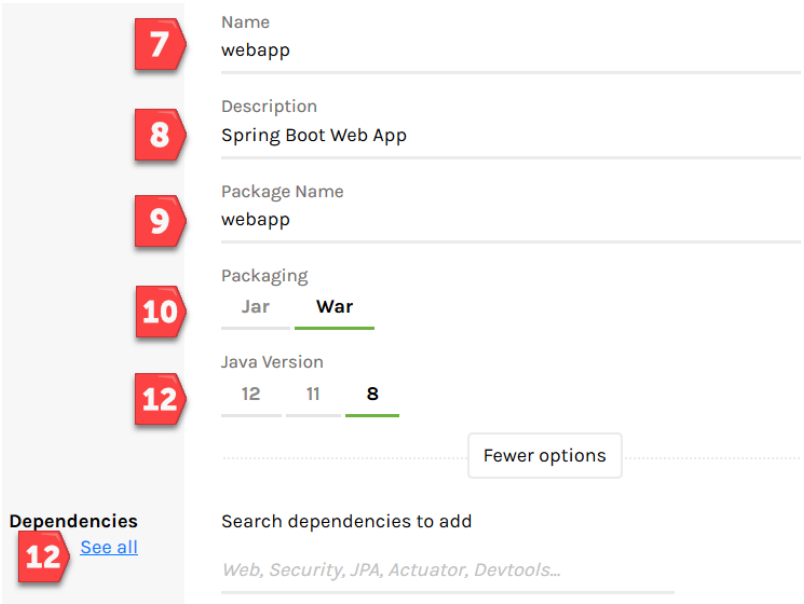
1. Defining and generating a Spring Boot application

Open http://start.spring.io in your web browser. This page allows quick creation of a Java Spring Boot application.

1. Choose Maven Project for Project
2. Language: Java
3. Spring Boot Version: Choose latest stable (currently 2.1.3)
4. Enter ie.examples for Group
5. Artifact: Enter a name for the application, webappp



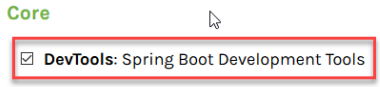
6. Click the More Options button, then continue filling the options:



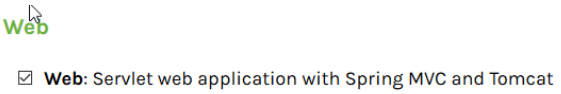
12. Click the See All link under Dependencies for even more options

Tick the following Dependencies:

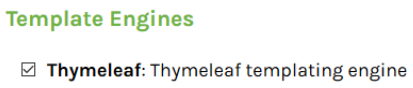
- Core:
  - DevTools enables auto re-compile and re-start when the application changes.



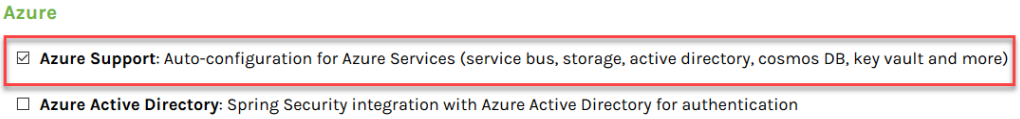
- Web:



- Template Engines:

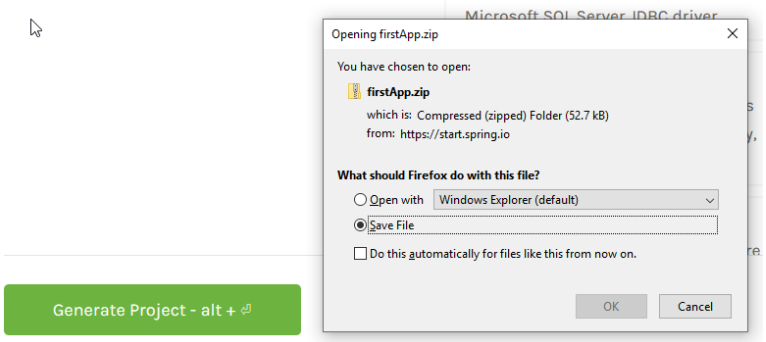


- Azure:



Finally click update dependencies

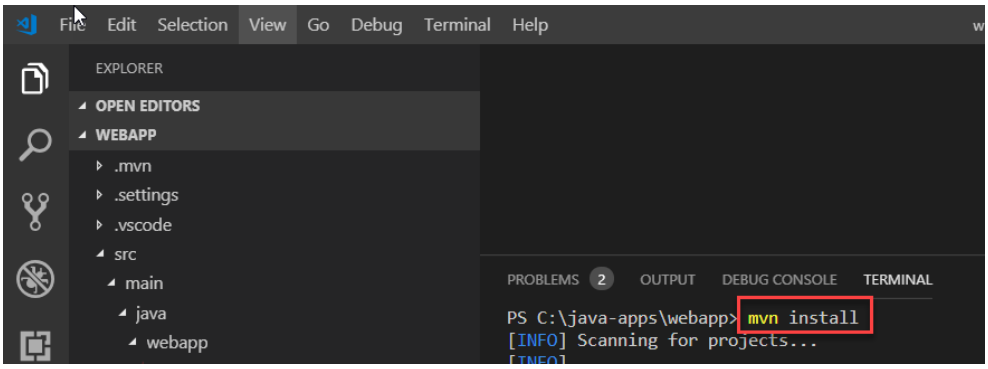
…and then Generate Project and download the zip file



Open the web app in VS Code

First copy the zip file you downloaded to your java-apps folder and unzip. Then open the folder in VS Code.

In VS Code, open a terminal window and run `mvn install` to build the application

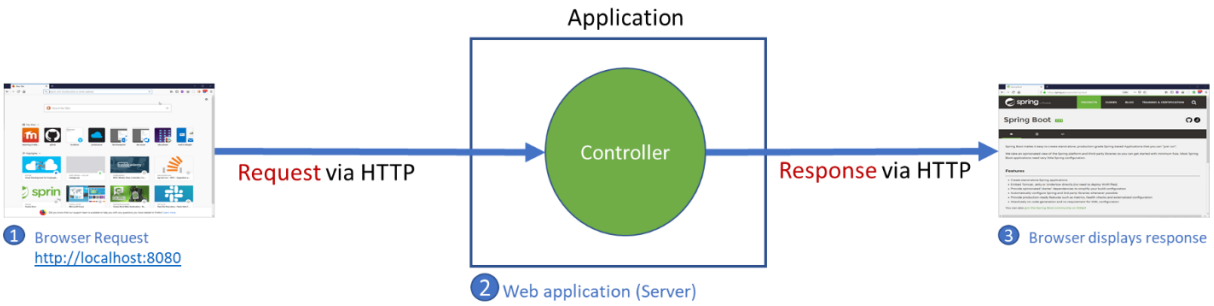


At this point the application doesn't do anything. The folders and content you see is the framework which will support the application.

2. Create a simple web application

Now we will build a simple web application.

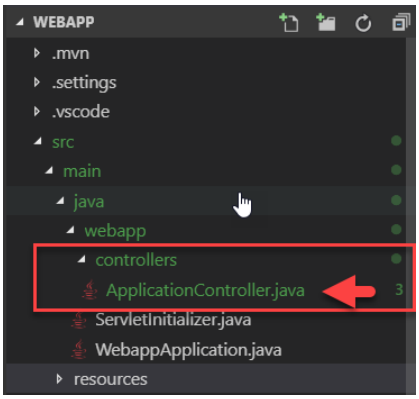
- It will accept a request – sent via a web browser (client)
- The request will be sent via HTTP to the server, Tomcat, which hosts the application
- The application will process the request.
- Finally, a response will be sent back to the browser



2.1 Add a Controller

In an MVC web application, a controller is used to process incoming requests and send an appropriate response.

Before Adding a controller, first create a new folder in the `webapp` folder and name it `controllers`



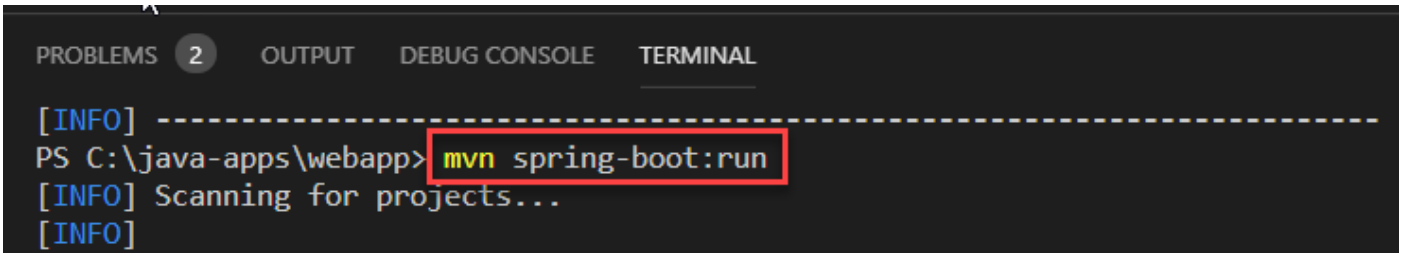
Then add a Java file, named `ApplicationController.java` inside the `controllers` folder.

Edit `ApplicationController.java`, add the following code, then save. Read the comment for more details.

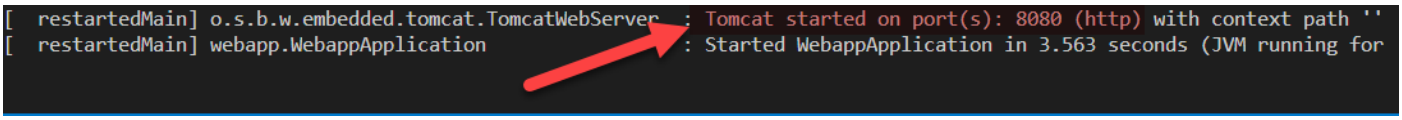
```
package webapp.controllers;

// Import framework dependencies - required to handle the HTTP request and send a response
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.WebApplicationInitializer;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestParam;

// The @ annotation identifies this as a Controller class
@Controller
public class ApplicationController {

    // This method, index(), serves as the site index - the default page
    // Requests for the root '/' will be handled by this method
    @RequestMapping(value = "/", method = RequestMethod.GET)
    @ResponseBody // Send a direct response without a view template
    public String index() {
        // Return some text (this will be the response back to the browser)
        return "Hello World! This is the Home page";
    }
}
```

2.2 Run the application

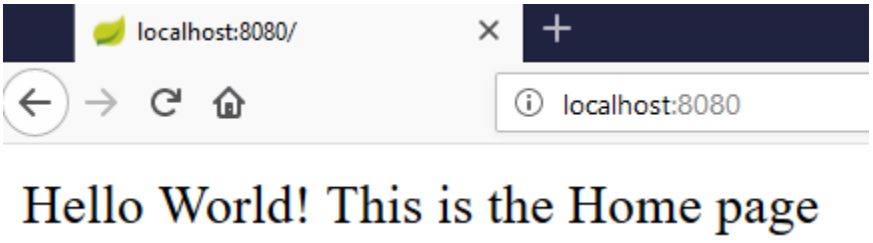
Run the command `mvn spring-boot:run` in the VS Code terminal



If all goes well, the HTTP (WWW) server should start on port 8080



Open http://localhost:8080 in a web browser and you should see a page like this (note the message returned by the controller):



2.3 A closer look

Open the developer tools in your browser – usually by pressing F12 or right-click the page and choose inspect element. Click on the network tab and reload the page.

- This will reveal the `HTTP GET` request made by the browser
- Also note the response from the app server.
    - status code `200 (OK)` indicates that the request was handled successfully.
- For more about HTTP see https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview