Spring Boot

What is Spring Boot?

Spring Boot simplifies Java application development by handling complex setup and configuration tasks for you. It provides a streamlined approach that allows developers to start coding immediately without worrying about the usual setup challenges.

Under the hood, Spring Boot builds upon the Spring Framework, which is the underlying technology that provides extensive support for building Java applications. While you can work directly with the Spring Framework, Spring Boot enhances productivity by automating common tasks and providing predefined defaults.

In essence, Spring Boot accelerates Java development by eliminating setup complexities, which come along using Spring only.

Features

- **Instant Setup**: No need to spend hours configuring your project. Spring Boot sets everything up for you automatically.
- **Speedy Development**: Start building your application immediately. It's like hitting the ground running!
- **Selective Focus**: Easily pick and use only the classes (books) you need, even if you have thousands. Spring Boot handles the rest, ensuring smooth performance.

In simple words, Spring Boot saves you time, reduces complexity, and lets you focus on what really matters—building amazing applications quickly and efficiently.

Spring Boot Highlights:

- Embedded Server: Spring Boot includes an embedded server like Tomcat, eliminating the need for setting up an external server. Your application runs independently, making deployment hassle-free.

- **Webapp Folder:** It organizes essential setup files, such as XML configurations, neatly within a webapp folder. This ensures everything is conveniently accessible and ready for deployment.

- **Multiple IDE Support:** You can kickstart your Spring Boot project from the official site https://start.spring.io/ .Here, you configure your project structure, download it, and seamlessly integrate it into your preferred IDE—whether it's Eclipse, IntelliJ, or NetBeans. For IntelliJ Community Edition, you can follow the same method. Eclipse users benefit from Spring plugins that simplify IDE setup and configuration directly.

Documentation and Community:

- A good framework requires comprehensive features, an active community, and excellent documentation.
- The Spring Framework provides detailed documentation to support developers.
- https://docs.spring.io/spring-framework/reference/index.html

Creating the first Spring Boot Project:

So, after visiting the https://start.spring.io/

Step 1: Choose Maven as the project management tool.

Step 2: Choose Java Language

Step 3: Choose Spring Boot 3.2.7 as version (Stable).

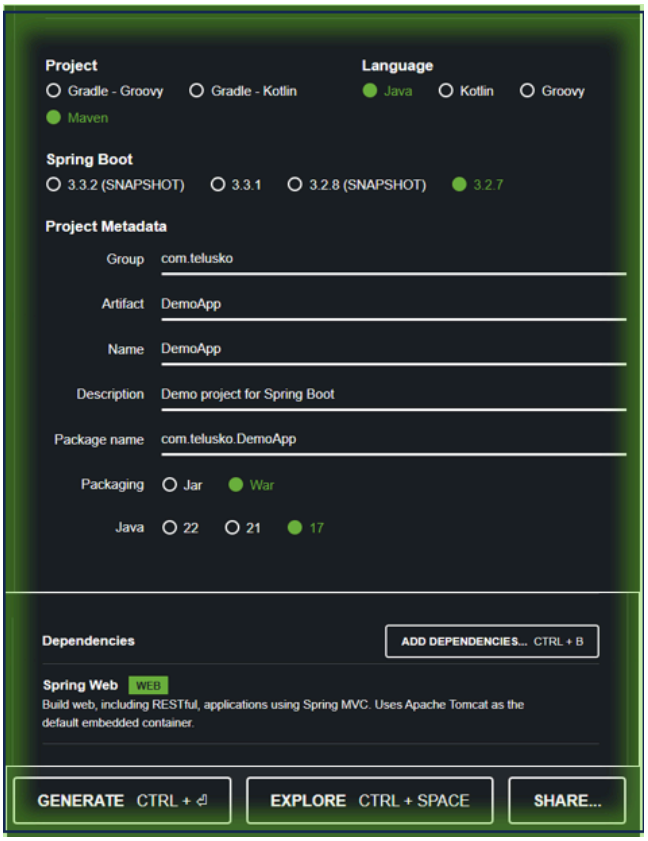
Step 4: Fill the project metadata according to requirements.

Step 5: Packaging Type as a war file

Step 6: Select the Java version as 17 or later.

Step 7: Click Dependencies and select Spring Web

Step 8: Click on Generate.



Step 9: Extract the zip file. Now open a suitable IDE and then go to ***File->New->Project from existing sources->DemoApp*** and select pom.xml. Click on import changes on prompt and wait for the project to sync.

After complete syncing of the pom file, check the dependencies it has added, like the Jackson module ,tomcat (embeded)se rver ,spring core & web dependency.

Now, lets dive into writing some lines of project code

```java
package com.telusko.demo;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

}

```

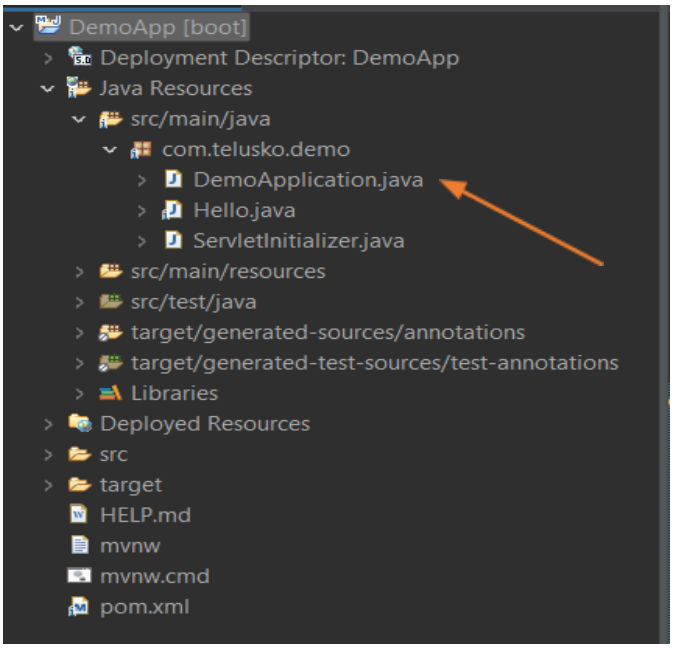
First, create a Java class named Hello.java under the package com.telusko.DemoApp

```java
package com.telusko.demo;

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

@RestController
public class Hello {

    @RequestMapping("/")
    public String greet() {
        return "Hello Aliens, Welcome to Telusko";


    }

}
```

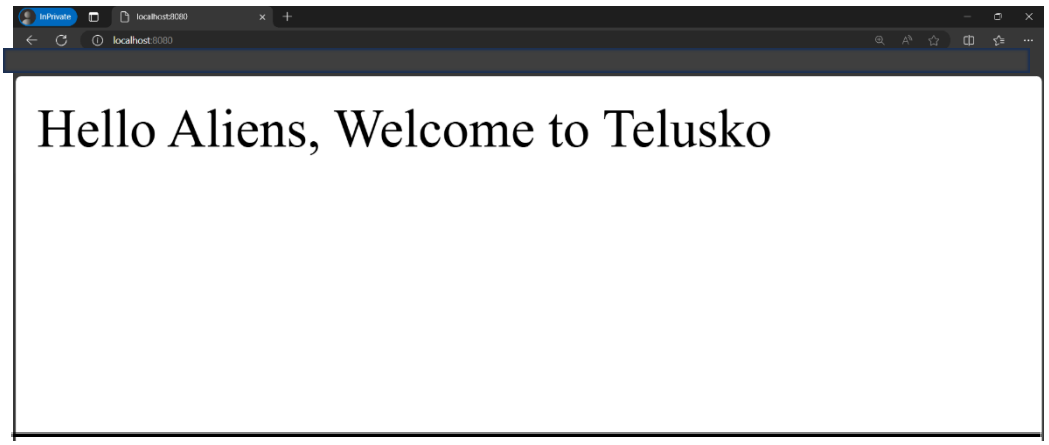
- After that, create a method named greet which returns a greeting message.

//image needs to be reconsidered

- Now, add ***@RestController*** at the top of the class definition and ***@RequestMapping("/")*** at the top of the method declaration.
- Get back to the main file i.e. DemoApp and right click the file & select run the application.
- The application will run on the Embedded Tomcat server with the default port 8080, but this can be changed.
- Finally, open your web browser, type localhost:8080 in the address bar, and hit enter. You will see the output on the screen



- **Output**


InPrivate
localhost:8080
localhost:8080
Hello Aliens, Welcome to Telusko