

A large, light gray play button icon is positioned on the left side of the slide. It consists of a white right-pointing triangle centered within a series of concentric circles in varying shades of gray.

Creating Spring Boot Applications

1. Creating a console application
2. Creating a web application
3. Defining application properties



1. Creating a Console Application

- Overview
- Creating a Spring Boot project
- Specifying project dependencies
- Understanding the application structure
- Understanding the Maven POM file
- Understanding the application code
- Running the application

Overview

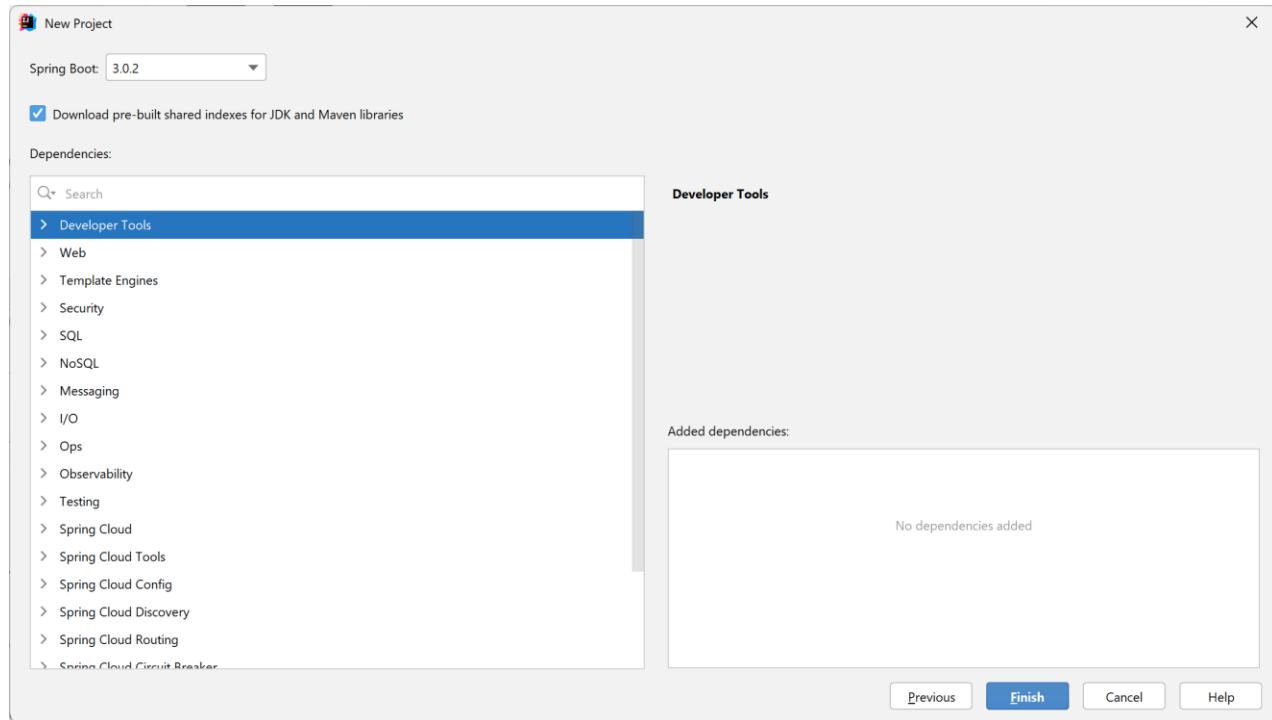
- IntelliJ IDEA Ultimate has excellent support for Spring
 - Spring Boot and Spring Framework
- IntelliJ Java dependencies:
 - JDK (e.g. JDK 17)
 - Set `JAVA_HOME` to the JDK folder
 - Set `PATH` to include the JDK binary folder

Creating a Spring Boot Project

- Start IntelliJ, click New Project, and select Spring Initializr
- Specify project info as follows:
 - Enter a suitable project name and location
 - Language - Java
 - Type - Maven
 - Enter a suitable group ID, artifact ID, and package name
 - Project SDK - Java 17
 - Java version - 17
 - Packaging - Jar
 - Then click Next

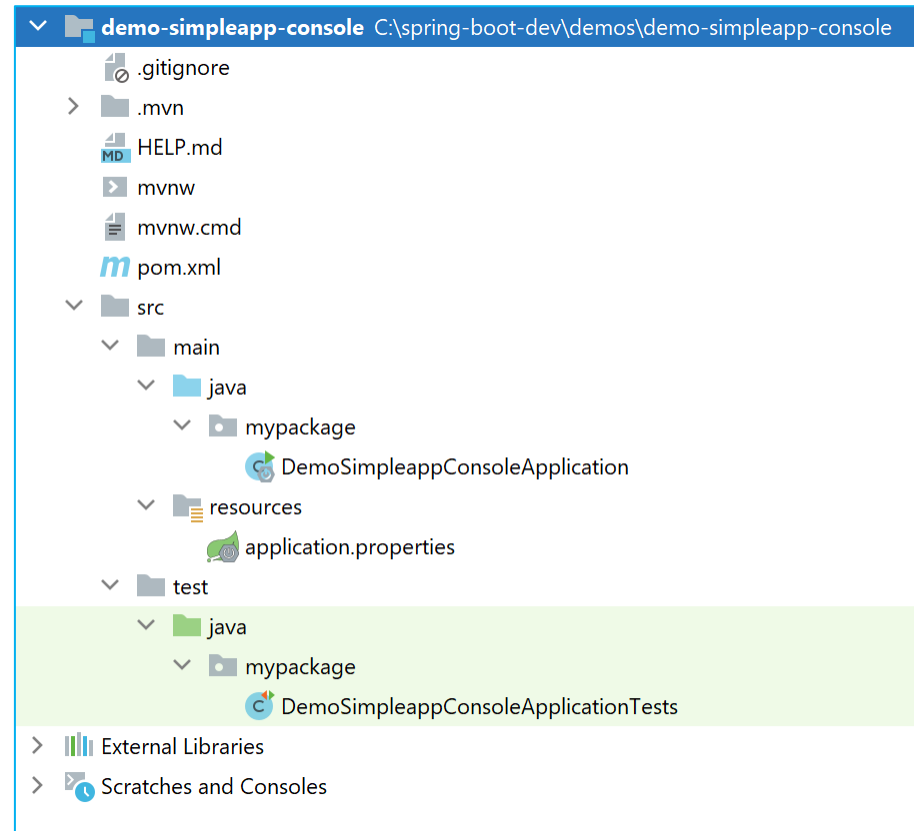
Specifying Project Dependencies

- In the next window you can add dependencies to your project
 - We don't need any dependencies yet, so just click Finish



Understanding the Application Structure

- The generated application is a regular Maven project



Understanding the Maven POM File

- Here are the relevant sections in the Maven POM file:

```
<project ... >
  ...
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.0.2</version>
    <relativePath/>
  </parent>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
  ...
```

Parent POM

Spring Boot dependency

Spring Boot test dependency

pom.xml

Understanding the Application Code

- Here's the generated application code:

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

@SpringBootApplication
public class DemoSimpleappConsoleApplication {

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

DemoSimpleappConsoleApplication.java

- @SpringBootApplication is equivalent to:
 - @Configuration
 - @EnableAutoConfiguration
 - @ComponentScan

Running the Application (1 of 2)

- You can build/run the app via the `mvn` command in the project root directory as follows:

```
mvn spring-boot:run
```

- If you don't have Maven installed separately on your machine, you can run the following command instead:

```
mvnw spring-boot:run
```

- It's also possible to run the application directly in IntelliJ
 - Right-click in the main class, and click Run

Running the Application (2 of 2)

- This is the application output
 - Displays a "Spring Boot" banner
 - The app terminates immediately, because it's so simple 😊

[illegible]

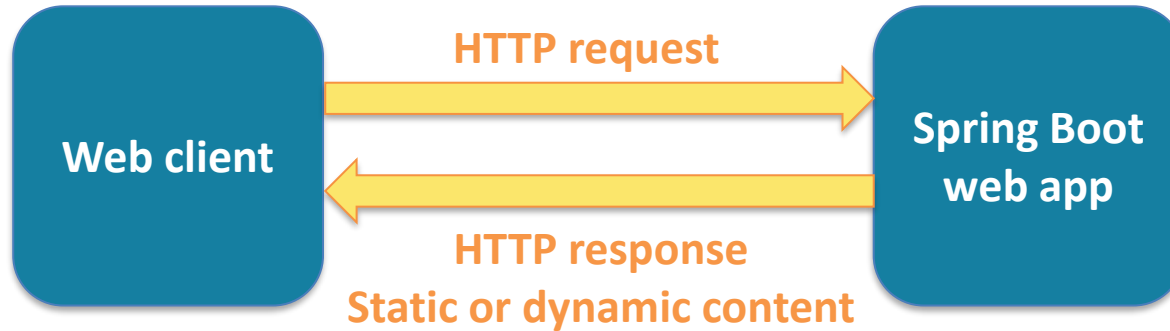
```
2023-02-03T11:08:16.461Z INFO 23036 --- [main] m.DemoSimpleappConsoleApplication : Starting DemoSimpleappConsoleApplication using Java 17.0.1 with PID 23036 (C:\spring\
2023-02-03T11:08:16.465Z INFO 23036 --- [main] m.DemoSimpleappConsoleApplication : No active profile set, falling back to 1 default profile: "default"
2023-02-03T11:08:17.615Z INFO 23036 --- [main] m.DemoSimpleappConsoleApplication : Started DemoSimpleappConsoleApplication in 1.8 seconds (process running for 1.945)
```

2. Creating a Web Application

- Overview
- Creating a Spring Boot web project
- Specifying project dependencies
- Understanding the web application structure
- Understanding the Maven POM file
- Adding web content
- Running the application
- Pinging the application

Overview

- Spring Boot applications are typically "web apps"
 - Listen for HTTP requests from web client (e.g. a browser)
 - Return static or dynamic content



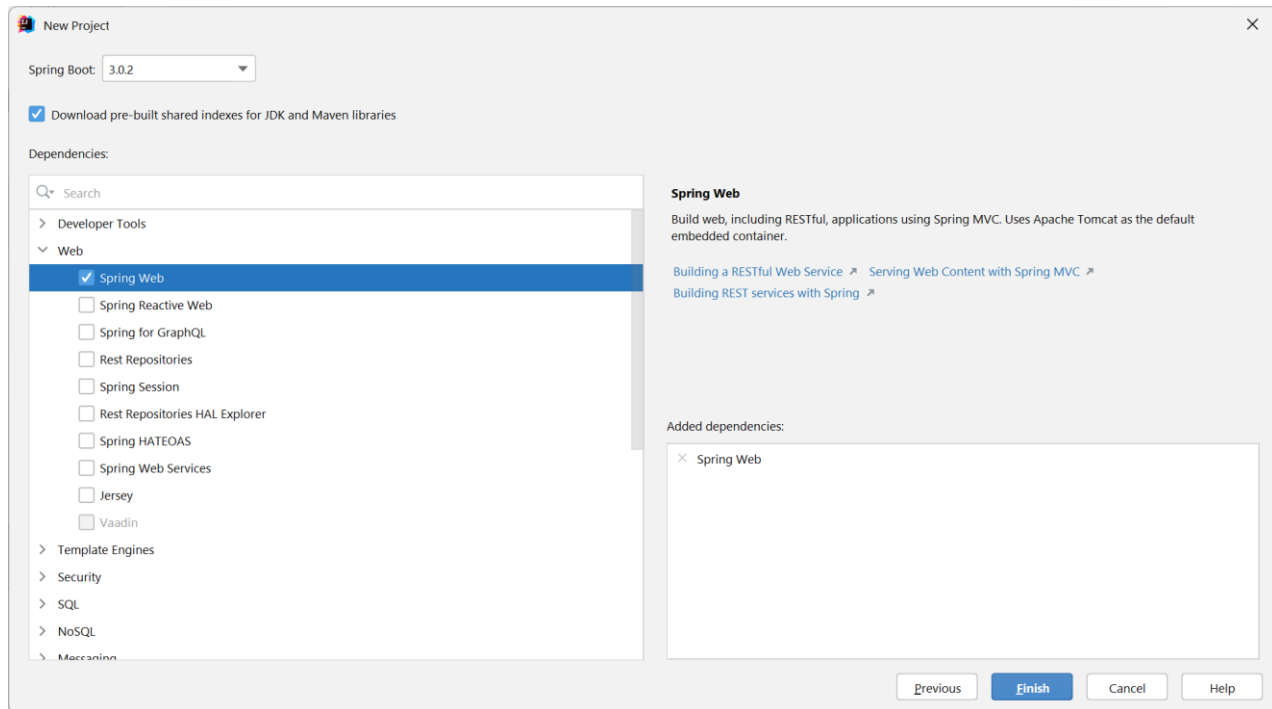
- We'll see how to return **static content** for now
 - Later we'll see how to return dynamic content, via REST services

Creating a Spring Boot Web Project

- Start IntelliJ, click New Project, and select Spring Initializr
- Specify project info as follows:
 - Enter a suitable project name and location
 - Language - Java
 - Type - Maven
 - Enter a suitable group ID, artifact ID, and package name
 - Project SDK - Java 17
 - Java version - 17
 - Packaging - Jar
 - Then click Next

Specifying Project Dependencies

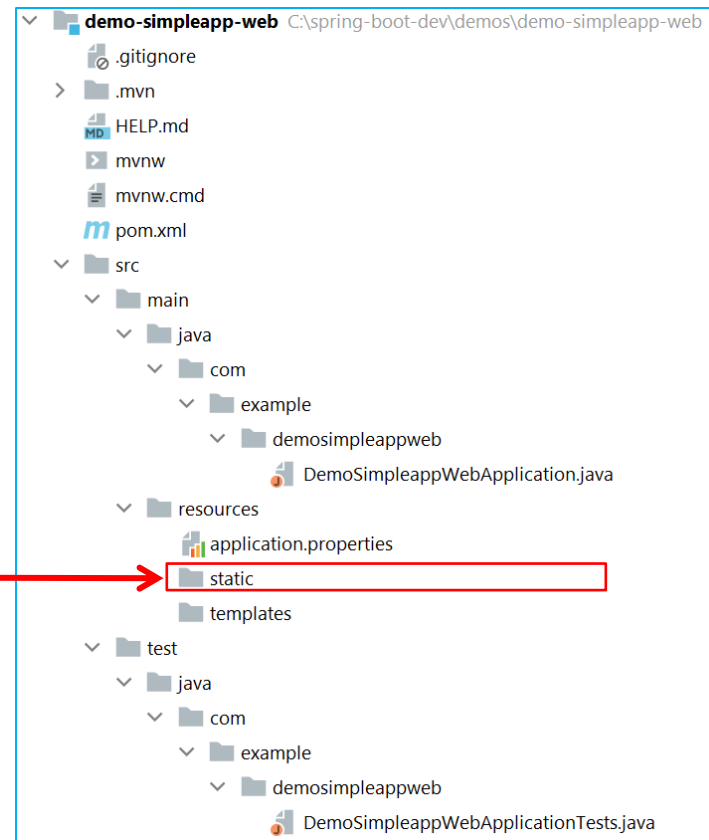
- In the next window you can add dependencies to your project
 - Expand **Web** and select **Spring Web**



Understanding the Web Application Structure

- The generated application is a regular Maven **web** project:

Put static web content here



Understanding the Maven POM File

- Here's the relevant section in the Maven POM file:

```
<project ... >
...

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
...
```

Spring Boot web dependency

pom.xml

Adding Web Content

- You can add static web content in the following directory:
 - `src\main\resources\static`
- For example, add an `index.html` file:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Home</title>
</head>
<body>
  Hello world!
</body>
</html>
```

`index.html`

Running the Application

- To run the application:
 - Right-click the Java app file, then click Run
 - Compiles the code, bundles into a JAR, then runs the JAR
 - The application has an embedded Tomcat web server

```

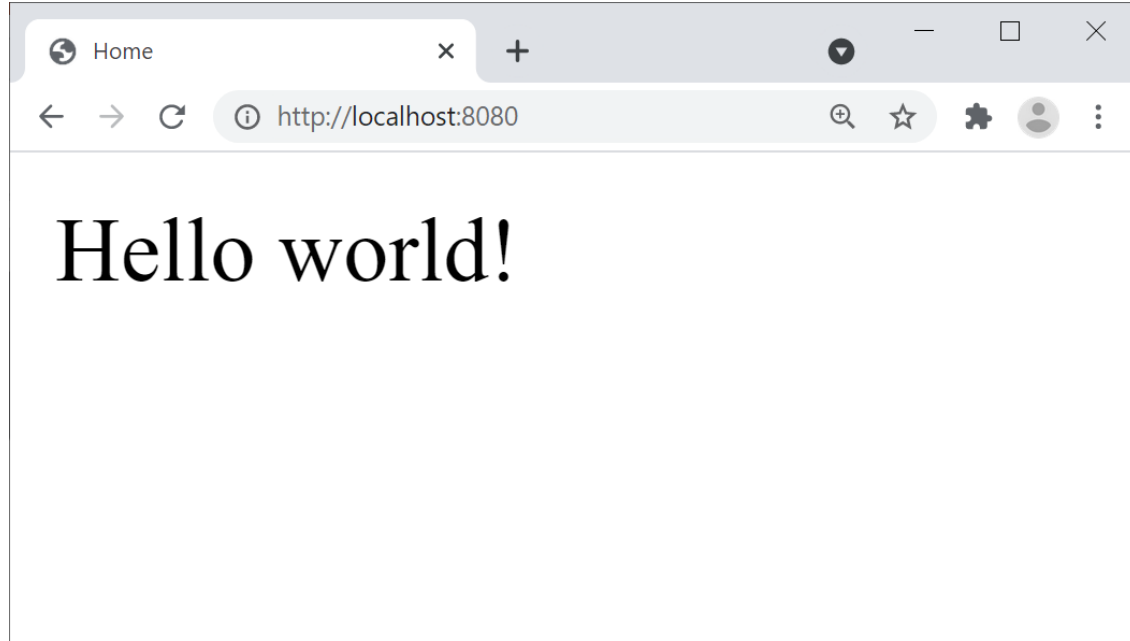
      /\ /-----\ C)          /\ \ \ \ \
     (\ )\----| ' |' |' |' V \ \ \ \ \
      W ----| | | | | | | | ( | | ) ) )
       _ |___| | | | | | | | / / / / /
      =====|_|=====|_____/___/\_/_/_/
:: Spring Boot ::                (v3.0.2)

2023-02-03T12:59:48.604Z INFO 26168 --- [main] c.e.d.DemoSimpleappWebApplication : Starting DemoSimpleappWebApplication using Java 17.0.1 with PID 26168 (C:\spring-
2023-02-03T12:59:48.614Z INFO 26168 --- [main] c.e.d.DemoSimpleappWebApplication : No active profile set, falling back to 1 default profile: "default"
2023-02-03T12:59:58.954Z INFO 26168 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2023-02-03T12:59:58.979Z INFO 26168 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-02-03T12:59:58.979Z INFO 26168 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.5]
2023-02-03T12:59:51.348Z INFO 26168 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2023-02-03T12:59:51.349Z INFO 26168 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2626 ms
2023-02-03T12:59:51.860Z INFO 26168 --- [main] o.s.b.a.w.s.WelcomePageHandlerMapping : Adding welcome page: class path resource [static/index.html]
2023-02-03T12:59:52.146Z INFO 26168 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2023-02-03T12:59:52.163Z INFO 26168 --- [main] c.e.d.DemoSimpleappWebApplication : Started DemoSimpleappWebApplication in 4.457 seconds (process running for 6.141

```

Pinging the Application

- Open a browser and go to `http://localhost:8080`
 - Renders `index.html` (a standard welcome page in Java web)



A large, light gray play button icon is positioned on the left side of the slide. It consists of a white right-pointing triangle centered within a series of concentric circles in shades of gray.

3. Defining Application Properties

- Overview of application properties
- Editing application properties
- Restarting the application

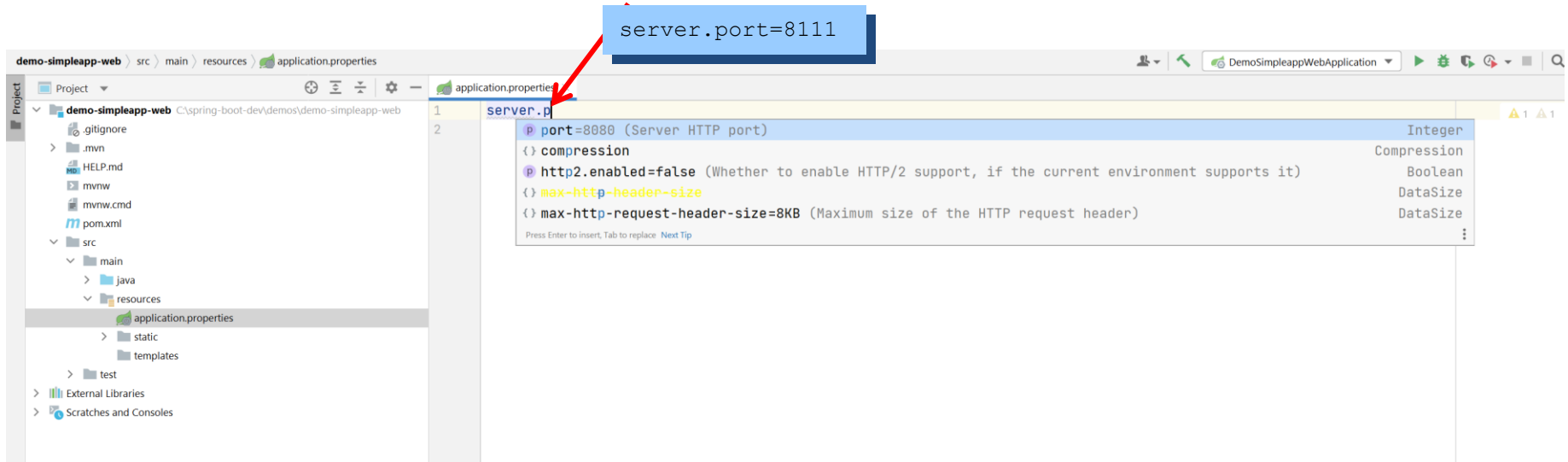


Overview of Application Properties

- Spring Boot applications have a standard text file named `application.properties`
 - Very important!
 - The recommended place to set application properties
 - i.e. name=value pairs
- You can also use YAML if you like
 - YAML = "YAML Ain't Markup Language"
 - More on YAML files later...

Editing Application Properties

- To help you edit `application.properties`, IntelliJ provides a Spring Properties Editor tool
 - Provides nice content assistance and error checking

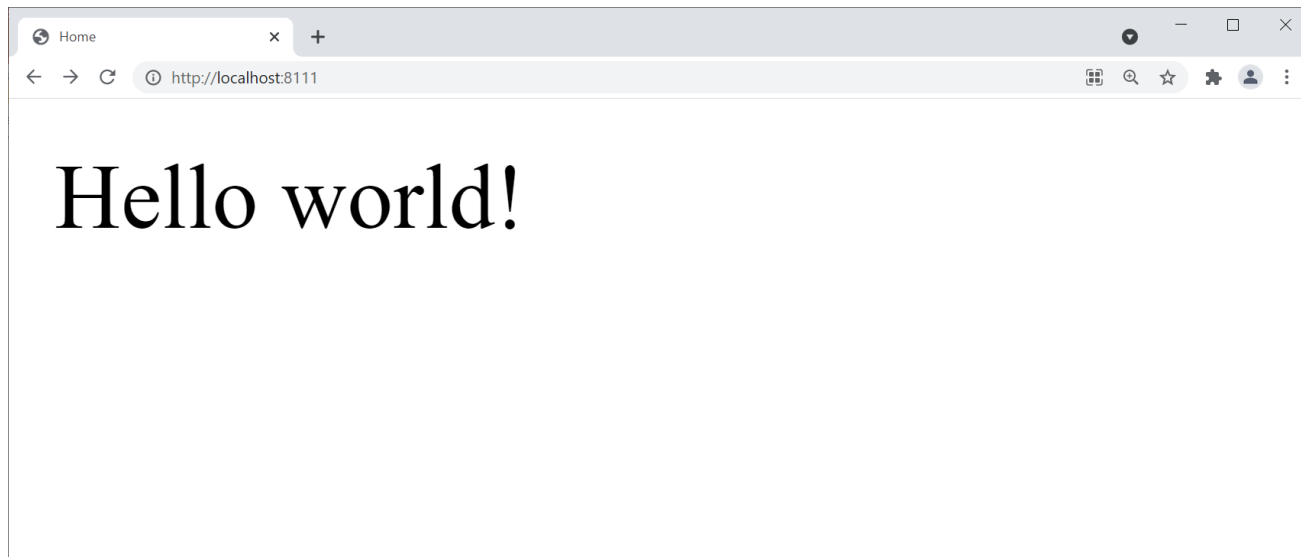


Restarting the Application

- Restart the application, and verify Tomcat starts on the new port number, 8111

```
main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8111 (http) with context path ''
```

- Ping the Web server using the new port number, 8111



A large, stylized play button icon consisting of a white triangle pointing right, centered within a series of concentric circles in shades of gray.

Summary

- Creating a console application
- Creating a web application
- Defining application properties

