Creating a Spring Boot Web App

1. Creating a web application
2. Additional techniques

1. Creating a Web Application

- Getting started with IntelliJ
- Creating a web app in IntelliJ
- Application structure
- Maven POM file
- Code artifacts
- Adding an HTML home page
- Running the application
- Pinging the application

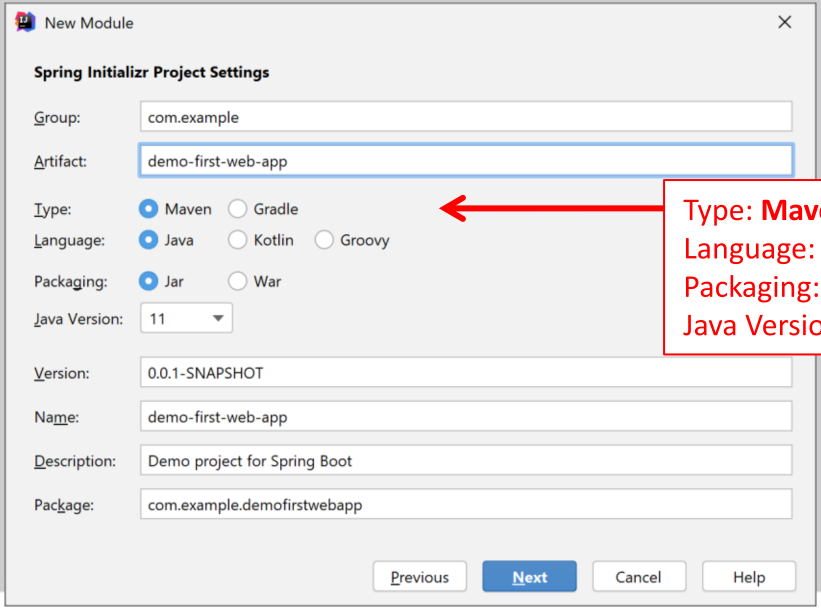
Getting Started with IntelliJ

- IntelliJ has excellent support for Spring
  - E.g. create Spring Boot apps (via Spring Initializr)
  - E.g. create Spring Framework apps

- We've provided an IntelliJ project with full demos
  - Start IntelliJ
  - In the dialog box, click Open Project
  - Select the **FullStackDev** folder

Creating a Web App in IntelliJ (1 of 3)

- To create a new module, click File | New | Module
  - Select Spring Initializr, click Next, and enter these details:



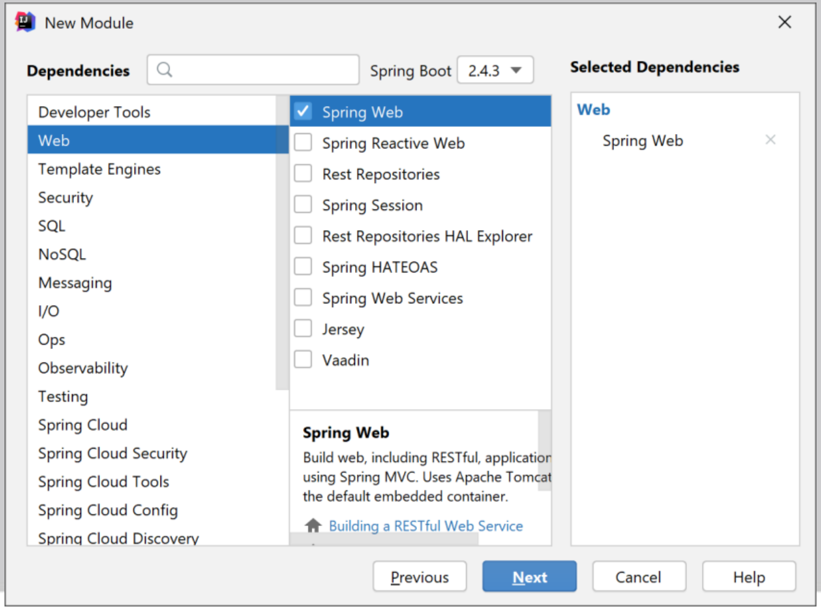
Spring Initializr Project Settings

Group:	com.example
Artifact:	demo-first-web-app
Type:	[✓ Maven] Gradle
Language:	[✓ Java] Kotlin Groovy
Packaging:	[✓ Jar] War
Java Version:	11
Version:	0.0.1-SNAPSHOT
Name:	demo-first-web-app
Description:	Demo project for Spring Boot
Package:	com.example.demofirstwebapp

Type: **Maven**
Language: **Java**
Packaging: **Jar**
Java Version: **11** (for example)

Creating a Web App in IntelliJ (2 of 3)

- Select the project dependencies you want
  - E.g. select the **Spring Web** dependency



New Module

Dependencies Spring Boot 2.4.3 **Selected Dependencies**

Developer Tools
Web
Template Engines
Security
SQL
NoSQL
Messaging
I/O
Ops
Observability
Testing
Spring Cloud
Spring Cloud Security
Spring Cloud Tools
Spring Cloud Config
Spring Cloud Discovery

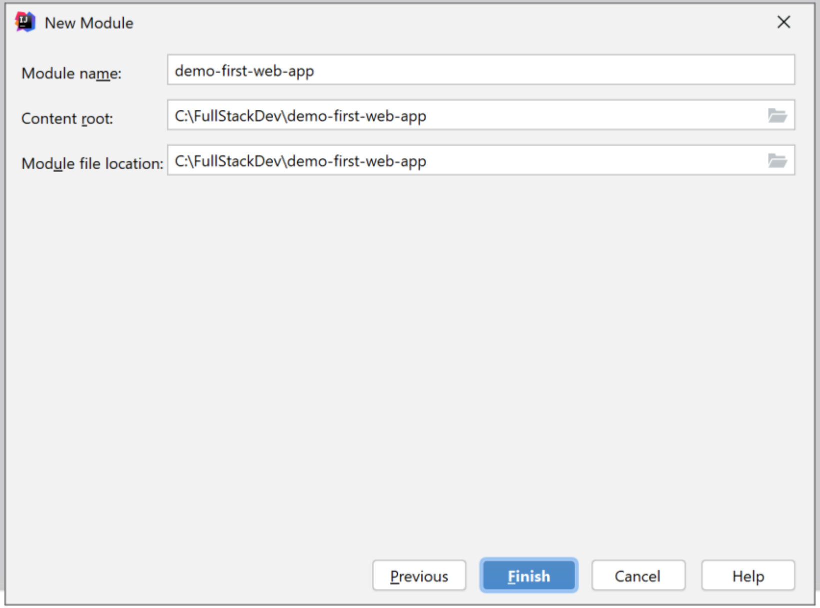
- [✓] Spring Web
- [ ] Spring Reactive Web
- [ ] Rest Repositories
- [ ] Spring Session
- [ ] Rest Repositories HAL Explorer
- [ ] Spring HATEOAS
- [ ] Spring Web Services
- [ ] Jersey
- [ ] Vaadin

Spring Web
Build web, including RESTful, application
using Spring MVC. Uses Apache Tomcat
the default embedded container.
Building a RESTful Web Service

Web
Spring Web

Creating a Web App in IntelliJ (3 of 3)

- Click Finish
  - IntelliJ generates your module



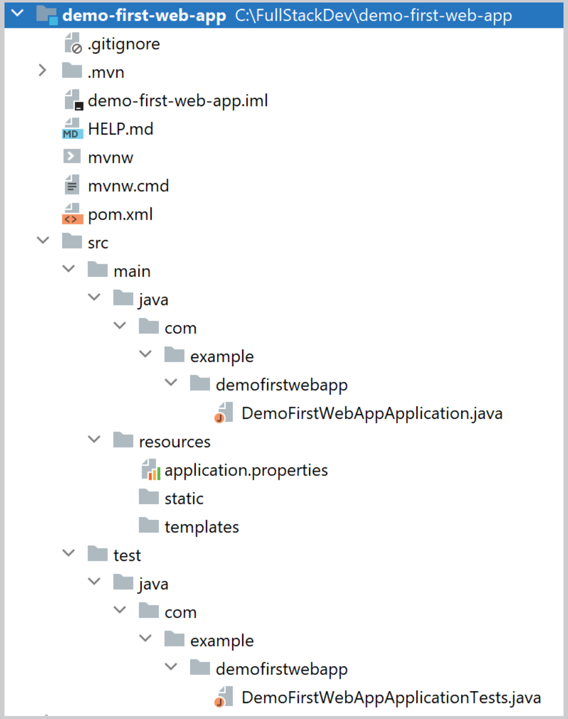
New Module

Module name: demo-first-web-app

Content root: C:\FullStackDev\demo-first-web-app

Module file location: C:\FullStackDev\demo-first-web-app

Application Structure



```
demo-first-web-app  C:\FullStackDev\demo-first-web-app
    .gitignore
  > .mvn
    demo-first-web-app.iml
    HELP.md
    mvnw
    mvnw.cmd
    pom.xml
  v src
    v main
      v java
        v com
          v example
            v demofirstwebapp
                DemoFirstWebAppApplication.java
      v resources
          application.properties
          static
          templates
    v test
      v java
        v com
          v example
            v demofirstwebapp
                DemoFirstWebAppApplicationTests.java
```

- The generated app is a regular Maven web app project
- To ensure IntelliJ can build it:
  - Right-click **pom.xml**
  - Click Add as Maven Project

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>2.4.3.RELEASE</version>
        <relativePath/>
    </parent>

    <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>
    …
```

pom.xml

Parent POM

Spring Boot web dependency

Spring Boot test dependency

Code Artifacts

- Here's the generated application code

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

@SpringBootApplication
public class DemoFirstWebAppApplication {

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

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

Adding an HTML Home Page

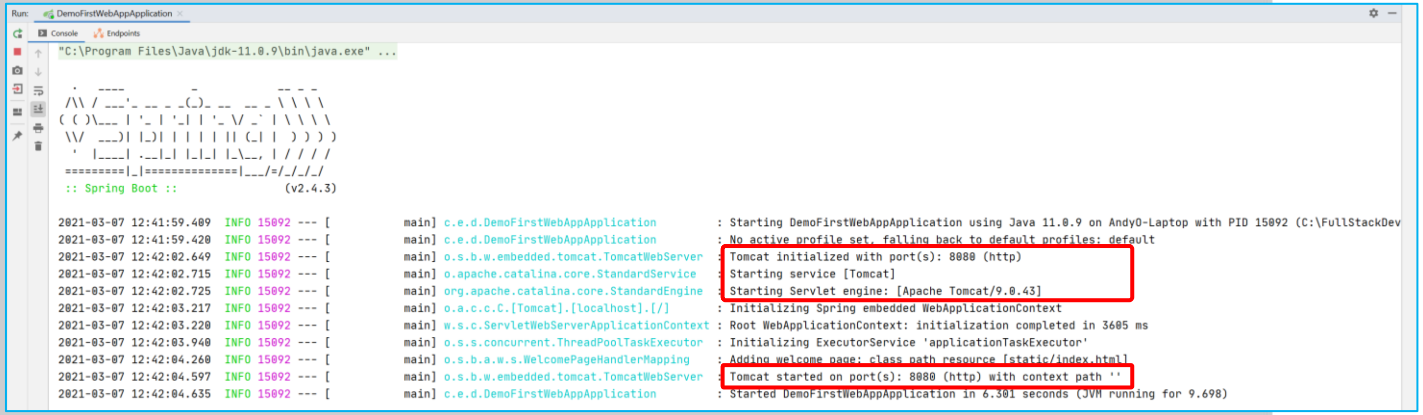
- We can just add static content directly
  - In the `src\main\resources\static` folder
  - E.g. let's add a simple HTML page

```
<!DOCTYPE html>
<html>
    <head>
        <title>Home</title>
    </head>
    <body>
        Prynhawn da pawb o Gymru!
    </body>
</html>
```

`index.html`

Running the Application

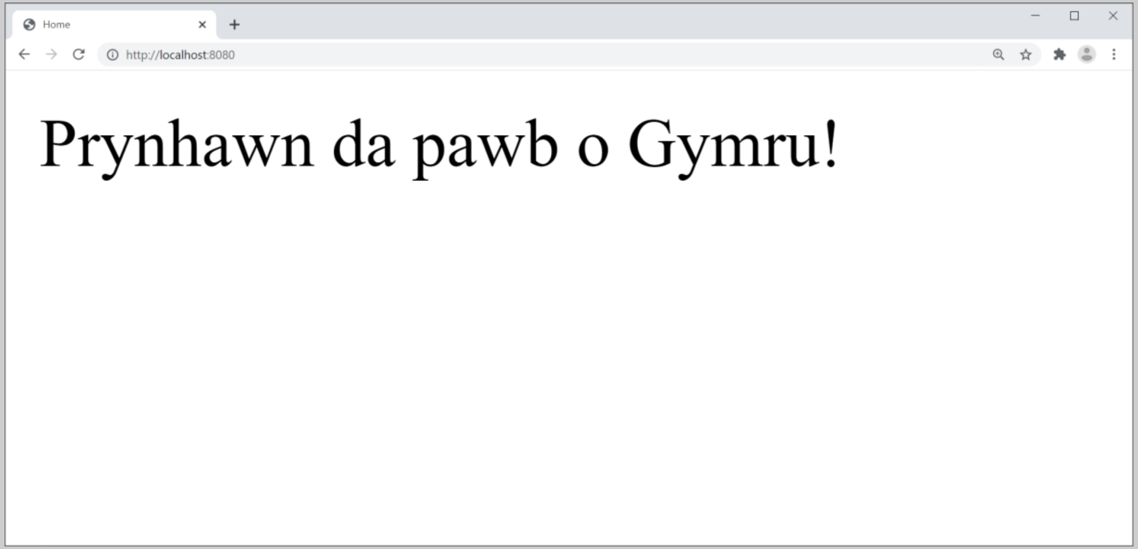
- To run the application:
  - Right-click the Java app file, then click Run
  - Compiles code, bundles into a JAR, then runs the JAR
  - The application has an embedded Tomcat web server



```
"C:\Program Files\Java\jdk-11.0.9\bin\java.exe" ...

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.4.3)

2021-03-07 12:41:59.409  INFO 15092 --- [           main] c.e.d.DemoFirstWebAppApplication         : Starting DemoFirstWebAppApplication using Java 11.0.9 on AndyO-Laptop with PID 15092 (C:\FullStackDev
2021-03-07 12:41:59.420  INFO 15092 --- [           main] c.e.d.DemoFirstWebAppApplication         : No active profile set, falling back to default profiles: default
2021-03-07 12:42:02.649  INFO 15092 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2021-03-07 12:42:02.715  INFO 15092 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2021-03-07 12:42:02.725  INFO 15092 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.43]
2021-03-07 12:42:03.217  INFO 15092 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2021-03-07 12:42:03.220  INFO 15092 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 3605 ms
2021-03-07 12:42:03.940  INFO 15092 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2021-03-07 12:42:04.260  INFO 15092 --- [           main] o.s.b.a.w.s.WelcomePageHandlerMapping    : Adding welcome page: class path resource [static/index.html]
2021-03-07 12:42:04.597  INFO 15092 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2021-03-07 12:42:04.635  INFO 15092 --- [           main] c.e.d.DemoFirstWebAppApplication         : Started DemoFirstWebAppApplication in 6.301 seconds (JVM running for 9.698)
```

Pinging the Application

- Open a browser and navigate to http://localhost:8080
  - Renders index.html, because this is a default filename



Prynhawn da pawb o Gymru!

2. Additional Techniques

- Overview of application properties
- Editing application properties
- Restarting the application
- Implementing a simple REST service
- Pinging the REST service

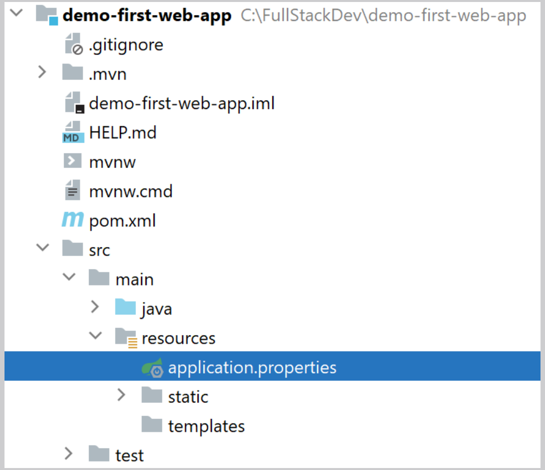
Overview of Application Properties

- Spring Boot applications have a standard text file named `application.properties`
  - The recommended place to set application properties
  - i.e. name=value pairs

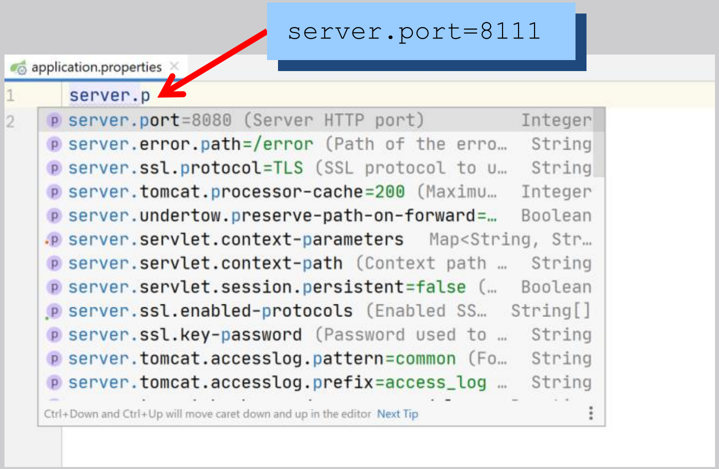
- You can also use YAML if you like
  - YAML = "YAML Ain't Markup Language"

Editing Application Properties

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



```
demo-first-web-app  C:\FullStackDev\demo-first-web-app
  .gitignore
  .mvn
  demo-first-web-app.iml
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
  src
    main
      java
      resources
        application.properties
        static
        templates
    test
```

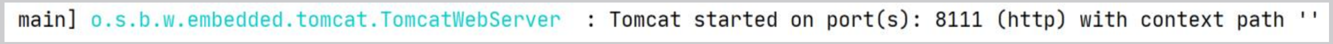


```
server.port=8111
```

```
application.properties
server.p
server.port=8080 (Server HTTP port)                          Integer
server.error.path=/error (Path of the erro…                  String
server.ssl.protocol=TLS (SSL protocol to u…                  String
server.tomcat.processor-cache=200 (Maximu…                   Integer
server.undertow.preserve-path-on-forward=…                   Boolean
server.servlet.context-parameters                  Map<String, Str…
server.servlet.context-path (Context path …                  String
server.servlet.session.persistent=false (…                   Boolean
server.ssl.enabled-protocols (Enabled SS…                    String[]
server.ssl.key-password (Password used to …                  String
server.tomcat.accesslog.pattern=common (Fo…                  String
server.tomcat.accesslog.prefix=access_log …                  String
Ctrl+Down and Ctrl+Up will move caret down and up in the editor  Next Tip
```

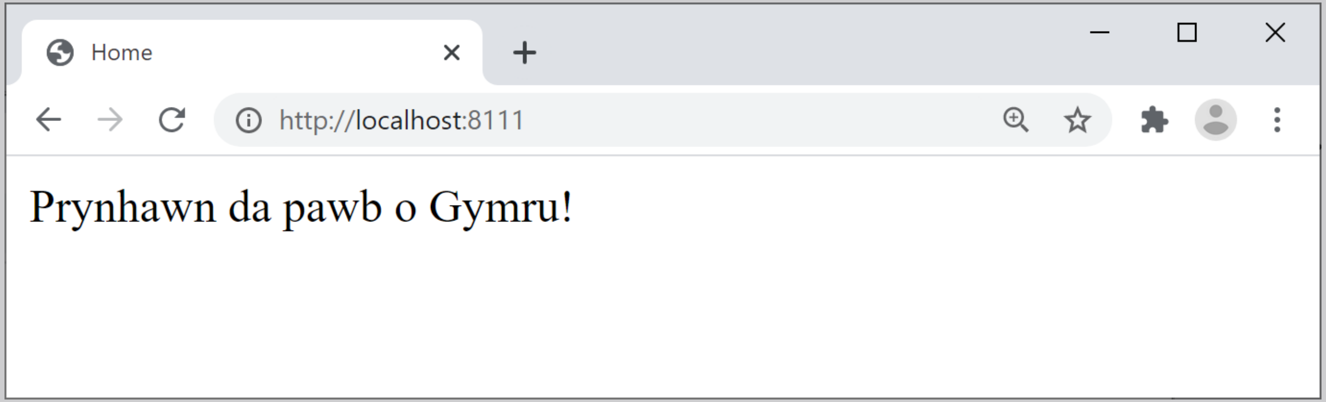
Restarting the Application

- 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



Prynhawn da pawb o Gymru!

Implementing a Simple REST Service

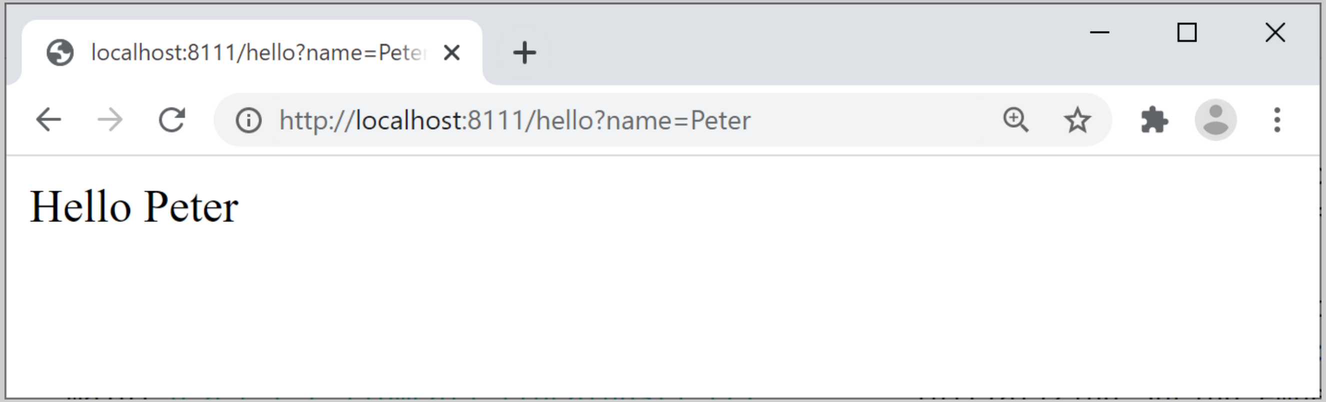
- It's easy to implement a REST service
  - Add a class and annotate with `@RestController`
  - Add methods and annotate with `@RequestMapping`

```java
package com.example.demofirstwebapp;

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

@RestController
public class HelloController {

    @RequestMapping("/hello")
    public String hello(@RequestParam String name) {
        return "Hello " + name;
    }
}
```

HelloController.java

Pinging the REST Service

- Restart the application and browse to a URL such as:
  - http://localhost:8111/hello?name=Peter



localhost:8111/hello?name=Peter
http://localhost:8111/hello?name=Peter

Hello Peter

Summary

- Creating a web application
- Additional techniques

Pearson