Running the Sample Application

1. Download Zip File from :
2. Cd into program4_demo
3. In “src/main/java/hello” you will find a file named Application.java. Open it in a text editor.
4. In ‘Application.java’ find the method dataSource(). Update ds.setUserName() and ds.setPassword() with your oracle ID and password

```
  @Bean
  public DataSource dataSource() {
   DriverManagerDataSource ds = new DriverManagerDataSource();
   ds.setDriverClassName(oracle.jdbc.driver.OracleDriver.class.getName());
   ds.setUrl("jdbc:oracle:thin:@aloe.cs.arizona.edu:1521:oracle");
   ds.setUsername("baileynottingham");
   ds.setPassword("******");
   return ds;
}
```

5. Tunnel into Lectura.
    1. For **mac/*nix** users:
       Tunnel into lectura using by typing the following command in your terminal window: ssh -L 29999:localhost:29999 netid@lectura.cs.arizona.edu.
       Here, 29999 is the port number.

    2. For windows user: Please follow this guide: Port forwarding using PuTTy (created by a previous TA, Zhenge Zhao): use 29999 as the port number for the Source Port Field and localhost:29999 for the Destination field.

       Note: When you create your project everyone will be choosing their own unique port numbers.

6. Copy program4_demo to lectura
7. Cd into program4_demo
8. Execute this command: **mvn install:install-file -Dfile=ojdbc14.jar -DgroupId=com.oracle -DartifactId=ojdbc14 -Dversion=11.2.0 -Dpackaging=jar**

   You should only have to execute this command once. Proceeding startups can start at step #9.

9. Execute this command: **mvn spring-boot:run**

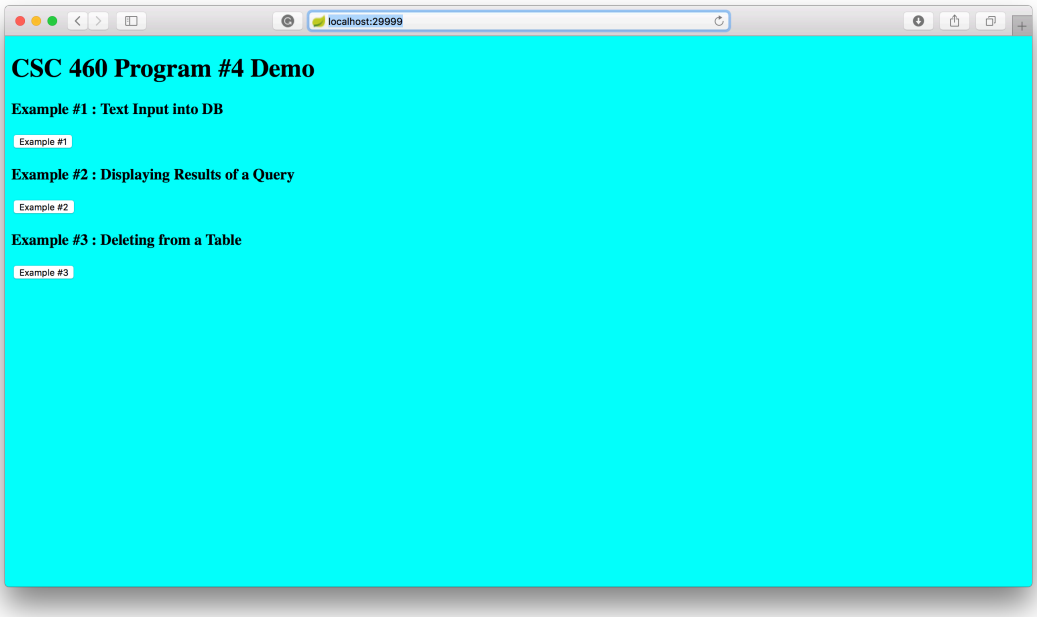
The first time you execute this, maven will download all the dependencies. Further startups will be faster.

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.0.5.RELEASE)

2018-11-13 08:06:59.487  INFO 18120 --- [           main] hello.Application                        : Starting Application on lectura.cs.arizona.edu with
PID 18120 (/p3/hb/baileynottingham/CSC460_TA_FA18/testingTheDemo/program4_demo/target/classes started by baileynottingham in
/p3/hb/baileynottingham/CSC460_TA_FA18/testingTheDemo/program4_demo)
2018-11-13 08:06:59.493  INFO 18120 --- [           main] hello.Application                        : No active profile set, falling back to default
profiles: default
2018-11-13 08:06:59.578  INFO 18120 --- [           main] ConfigServletWebServerApplicationContext : Refreshing
org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicationContext@1e2dee96: startup date [Tue Nov 13 08:06:59 MST 2018]; root
of context hierarchy
2018-11-13 08:07:01.647  INFO 18120 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 29999 (http)
2018-11-13 08:07:01.678  INFO 18120 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2018-11-13 08:07:01.678  INFO 18120 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet Engine: Apache Tomcat/8.5.34
2018-11-13 08:07:01.697  INFO 18120 --- [ost-startStop-1] o.a.catalina.core.AprLifecycleListener   : The APR based Apache Tomcat Native library which
allows optimal performance in production environments was not found on the java.library.path:
[/usr/java/packages/lib/amd64:/usr/lib64:/lib64:/lib:/usr/lib]
2018-11-13 08:07:02.026  INFO 18120 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2018-11-13 08:07:02.028  INFO 18120 --- [ost-startStop-1] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed
in 2455 ms
2018-11-13 08:07:02.141  INFO 18120 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean  : Servlet dispatcherServlet mapped to [/]
2018-11-13 08:07:02.148  INFO 18120 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'characterEncodingFilter' to: [/*]
2018-11-13 08:07:02.149  INFO 18120 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'hiddenHttpMethodFilter' to: [/*]
2018-11-13 08:07:02.149  INFO 18120 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'httpPutFormContentFilter' to: [/*]
2018-11-13 08:07:02.149  INFO 18120 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'requestContextFilter' to: [/*]
2018-11-13 08:07:02.221  INFO 18120 --- [           main] o.s.j.d.DriverManagerDataSource          : Loaded JDBC driver: oracle.jdbc.driver.OracleDriver
2018-11-13 08:07:02.368  INFO 18120 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**/favicon.ico] onto handler of type
[class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-11-13 08:07:02.732  INFO 18120 --- [           main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice:
org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicationContext@1e2dee96: startup date [Tue Nov 13 08:06:59 MST 2018]; root
of context hierarchy
2018-11-13 08:07:02.867  INFO 18120 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/greeting],methods=[GET]}" onto public
java.lang.String hello.GreetingController.greetingForm(org.springframework.ui.Model)
2018-11-13 08:07:02.868  INFO 18120 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/greeting],methods=[POST]}" onto public
java.lang.String hello.GreetingController.greetingSubmit(hello.Greeting)
2018-11-13 08:07:02.874  INFO 18120 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/addPerson],methods=[GET]}" onto public
java.lang.String hello.PersonController.personForm(org.springframework.ui.Model)
2018-11-13 08:07:02.875  INFO 18120 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/addPerson],methods=[POST]}" onto public
java.lang.String hello.PersonController.personSubmit(hello.Person)
2018-11-13 08:07:02.875  INFO 18120 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/deletePerson],methods=[GET]}" onto public
java.lang.String hello.PersonController.personFormDelete(org.springframework.ui.Model)
2018-11-13 08:07:02.876  INFO 18120 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/deletePerson],methods=[POST]}" onto public
java.lang.String hello.PersonController.personDelete(hello.Person)
2018-11-13 08:07:02.876  INFO 18120 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/queryResults],methods=[GET]}" onto public
java.lang.String hello.PersonController.queryResults(org.springframework.ui.Model)
2018-11-13 08:07:02.885  INFO 18120 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto public
org.springframework.web.servlet.ModelAndView
org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.errorHtml(javax.servlet.http.HttpServletRequest,javax.servlet.http.HttpServle
tResponse)
2018-11-13 08:07:02.885  INFO 18120 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public
org.springframework.http.ResponseEntity<java.util.Map<java.lang.String, java.lang.Object>>
org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.error(javax.servlet.http.HttpServletRequest)
2018-11-13 08:07:02.938  INFO 18120 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/webjars/**] onto handler of type
[class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-11-13 08:07:02.939  INFO 18120 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**] onto handler of type [class
org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-11-13 08:07:03.153  INFO 18120 --- [           main] o.s.b.a.w.s.WelcomePageHandlerMapping    : Adding welcome page: class path resource
[static/index.html]
2018-11-13 08:07:03.452  INFO 18120 --- [           main] o.s.j.e.a.AnnotationMBeanExporter        : Registering beans for JMX exposure on startup
2018-11-13 08:07:03.493  INFO 18120 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 29999 (http) with context
path ''
2018-11-13 08:07:03.498  INFO 18120 --- [           main] hello.Application                        : Started Application in 4.713 seconds (JVM running for
23.746)
```

You should see ‘Started Application’ as highlighted above.

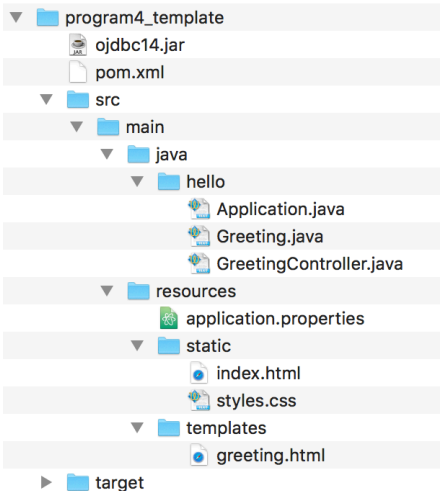
10. In the browser of your choice go to:
    localhost:29999



11. You can now play around with the demo. Example 1 will let you add text input into a database table. After you add a person to the table, you can verify it is there by querying baileynottingham.people in our oracle DB. Example 2 will also verify it by displaying all the names in baileynottingham.people. Example 3 will let you enter the first and last names of a person you would like to remove from the table.

Starting your own Application

1. After unzipping the zip file you should have received two folders: program4_demo and program4_template. Program4_template contains everything you need to create your own application. The directory structure is as followed:



Do not make any changes to the target directory. This is where generated class files and files generated during the build are placed.

The file pom.xml contains all the dependencies needed to make this application work. Do not delete anything from this file. You may only add additional dependencies if you feel you need external libraries to complete your project.

Understanding Project Object Module XML file

Here is the breakdown of pom.xml, as explained by Spring.io

1. <modelVersion>. POM model version (always 4.0.0).
2. <groupId>. Group or organization that the project belongs to. Often expressed as an inverted domain name
3. <artifactId>. Name to be given to the project's library artifact (for example, the name of its JAR or WAR file).
4. <version>. Version of the project that is being built.
5. <packaging> - How the project should be packaged. Defaults to "jar" for JAR file packaging. Use "war" for WAR file packaging.
6. The dependencies can be added under the dependency section. Let's say someone wants to use test their classes. Just add Junit as a dependency like this:

```
<dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
</dependency>
```

7. We'll be using Spring Boot maven plugin to run the application. The Spring Boot Maven Plugin provides Spring Boot support in Maven, allowing you to package executable jar or war archives and run an application "in-place".

The Spring Boot Plugin has the following goals.

- spring-boot:run runs your Spring Boot application.
- spring-boot:repackage repackages your jar/war to be executable.
- spring-boot:start and spring-boot:stop to manage the lifecycle of your Spring Boot application (i.e. for integration tests).
- spring-boot:build-info generates build information that can be used by the Actuator.

The Spring Boot Maven plugin provides many convenient features like:

- It searches for the public static void main() method to flag as a runnable class.

- It provides a built-in dependency resolver that sets the version number to match Spring Boot dependencies. You can override any version you wish, but it will default to Boot's chosen set of versions.

2. **Understanding the Web Controller and Views**

I have added an additional file to help you create your own controller. The controller file 'GreetingController.java' is found in 'src/main/java/hello' and its associated view 'greeting.html' is found in 'src/main/resources/templates'

In Spring's approach to building web sites, HTTP requests are handled by a controller. These components are easily identified by the @Controller annotation. The GreetingController below handles GET requests for /greeting by returning the name of a View, in this case, "greeting". A View is responsible for rendering the HTML content:

`src/main/java/hello/GreetingController.java`

```
package hello;


import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

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

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


@Controller

public class GreetingController {
```

```
    @GetMapping("/greeting")
    public String greetingForm(Model model) {
        model.addAttribute("greeting", new Greeting());
        return "greeting";
    }

    @PostMapping("/greeting")
    public String greetingSubmit(@ModelAttribute Greeting greeting) {
        return "result";
    }

}
```

This controller is concise and simple, but a lot is going on. Let's analyze it step by step.

The mapping annotations allows you to map HTTP requests to specific controller methods. The two methods in this controller are both mapped to /greeting. You can use @RequestMapping which by default maps all HTTP operations, such as GET, POST, and so forth. But in this case the greetingForm() method is specifically mapped to GET using @GetMapping, while greetingSubmit() is mapped to POST with @PostMapping. This mapping allows the controller to differentiate the requests to the /greeting endpoint.

The greetingForm() method uses a Model object to expose a new Greeting to the view template. The Greeting object in the following code contains fields such as id and content that correspond to the form fields in the greeting view, and will be used to capture the information from the form.

src/main/java/hello/Greeting.java

```
package hello;

public class Greeting {

    private long id;
    private String content;

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getContent() {
        return content;
    }

    public void setContent(String content) {
```

```
        this.content = content;

    }



}
```

The implementation of the method body relies on a view technology to perform server-side rendering of the HTML by converting the view name "greeting" into a template to render. In this case we are using Thymeleaf, which parses the greeting.html template below and evaluates the various template expressions to render the form.

src/main/resources/templates/greeting.html

```
<!DOCTYPE HTML>

<html xmlns:th="http://www.thymeleaf.org">

<head>

    <title>Getting Started: Handling Form Submission</title>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

</head>

<body>

        <h1>Form</h1>

    <form action="#" th:action="@{/greeting}" th:object="${greeting}"
method="post">

        <p>Id: <input type="text" th:field="*{id}" /></p>

         <p>Message: <input type="text" th:field="*{content}" /></p>

         <p><input type="submit" value="Submit" /> <input type="reset"
value="Reset" /></p>
```

```
    </form>

</body>

</html>
```

The th:action="@{/greeting}" expression directs the form to POST to the /greetingendpoint, while the th:object="${greeting}" expression declares the model object to use for collecting the form data. The two form fields, expressed with th:field="**{id}" and th:field="**{content}", correspond to the fields in the Greeting object above.

That covers the controller, model, and view for presenting the form. Now let's review the process of submitting the form. As noted above, the form submits to the /greeting endpoint using a POST. The greetingSubmit() method receives the Greeting object that was populated by the form. The Greeting is a @ModelAttribute so it is bound to the incoming form content, and also the submitted data can be rendered in the result view by referring to it by name (the name of the method parameter by default, so "greeting" in this case). The idis rendered in the <p th:text="'id: ' + ${greeting.id}" /> expression. Likewise the content is rendered in the <p th:text="'content: ' + ${greeting.content}" />expression.

`src/main/resources/templates/result.html`

```
<!DOCTYPE HTML>

<html xmlns:th="http://www.thymeleaf.org">

<head>

    <title>Getting Started: Handling Form Submission</title>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

</head>

<body>

       <h1>Result</h1>
```

```
    <p th:text="'id: ' + ${greeting.id}" />

    <p th:text="'content: ' + ${greeting.content}" />

    <a href="/greeting">Submit another message</a>

</body>

</html>
```

For clarity, this example uses two separate view templates for rendering the form and displaying the submitted data; however, you can also use a single view for both purposes.

3. **Understanding the Application Executable**

‘Application.java’ in ‘src/main/java/hello’ shown below creates a standalone application. Along the way, you use Spring’s support for embedding the Tomcat servlet container as the HTTP runtime, instead of deploying to an external instance.

`src/main/java/hello/Application.java`

```
package hello;



import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;



@SpringBootApplication

public class Application {



    public static void main(String[] args) {

        SpringApplication.run(Application.class, args);
```

```
    }


}
```

`@SpringBootApplication` is a convenience annotation that adds all of the following:

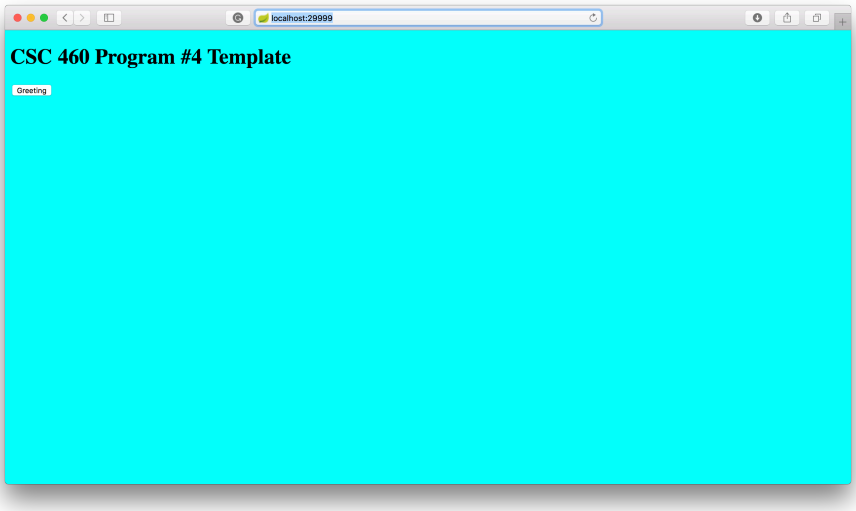
- `@Configuration` tags the class as a source of bean definitions for the application context.
- `@EnableAutoConfiguration` tells Spring Boot to start adding beans based on classpath settings, other beans, and various property settings.
- Normally you would add `@EnableWebMvc` for a Spring MVC app, but Spring Boot adds it automatically when it sees **spring-webmvc** on the classpath. This flags the application as a web application and activates key behaviors such as setting up a `DispatcherServlet`.
- `@ComponentScan` tells Spring to look for other components, configurations, and services in the `hello` package, allowing it to find the controllers.

The `main()` method uses Spring Boot's `SpringApplication.run()` method to launch an application.

4. **Homepage**

   Static resources, like HTML or JavaScript or CSS, can easily be served from your Spring Boot application just be dropping them into the right place in the source code. By default Spring Boot serves static content from resources in the classpath at "/static" (or "/public"). The index.html resource is special because it is used as a "welcome page" if it exists, which means it will be served up as the root resource.

   'src/main/resources/static/index.html'



5. **Understanding Connection to Oracle**
   Connect to oracle using data source. Java's *javax.sql.DataSource* interface provides a standard method of working with database connections. Traditionally, a 'DataSource' uses a URL along with some credentials to establish a database connection."

   In 'src/main/java/hello/Application.java'

```
@Bean
public DataSource dataSource() {
  DriverManagerDataSource ds = new DriverManagerDataSource();
  ds.setDriverClassName(oracle.jdbc.driver.OracleDriver.class.getName());
  ds.setUrl("jdbc:oracle:thin:@aloe.cs.arizona.edu:1521:oracle");
  ds.setUsername("");
  ds.setPassword("");
  return ds;
}
```

   In controller classes:

```
@Autowired
  private DataSource dataSource;
  private JdbcTemplate jdbcTemplate;

  @PostConstruct
```

```
private void postConstruct() {
   jdbcTemplate = new JdbcTemplate(dataSource);
}
```

1. **Autowired: "**Autowiring feature of spring framework enables you to inject the object dependency implicitly. It internally uses setter or constructor injection. Autowiring can't be used to inject primitive and string values. It works with reference only."
2. **PostConstruct: "**Javax's @PostConstruct annotation can be used for annotating a method that should be run once immediately after the bean's initialization. Keep in mind that the annotated method will be executed by Spring even if there is nothing to inject."

Now that we our JdbcTemplate in our controller, we can now use it to perform queries:

```
jdbcTemplate.update(
      "INSERT INTO EMPLOYEE VALUES (?, ?, ?, ?)", 5, "Bill", "Gates", "USA");
```

Section 3 of this site provides common and useful jdbcTemplate methods:
https://www.baeldung.com/spring-jdbc-jdbctemplate

6. **Running your application**

1. In "src/main/java/hello" you will find a file named Application.java. Open it in a text editor.
2. In 'Application.java' find the method dataSource(). Update ds.setUserName() and ds.setPassword() with your oracle ID and password

```
@Bean
 public DataSource dataSource() {
        DriverManagerDataSource ds = new DriverManagerDataSource();
        ds.setDriverClassName(oracle.jdbc.driver.OracleDriver.class.getName());
        ds.setUrl("jdbc:oracle:thin:@aloe.cs.arizona.edu:1521:oracle");
        ds.setUsername("baileynottingham");
        ds.setPassword("******");
        return ds;
 }
```

Getting your own port number

There's a chance that the port number you're trying to use might already be in use by someone else.

Here are the instructions to use your own port number:
In file **application.properties** in .../**src/main/resources/**
specify your port number using

server.port=**yourPortNumber**

I'd highly recommend that everyone use their own port numbers.
Port number has to be chosen carefully. If a port is in use by another process then the server will not be able to start. I suggest **adding the last 4 digits of your Class Identifier to 20000 to obtain your port number.**
Example: if you class identifier is 123456, then your port number would be 20000+3456 = 23456

3. Tunnel into Lectura:

   For **mac/*nix** users:
   Tunnel into lectura using by typing the following command in your terminal window: ssh -L **portnumber**:localhost:**portnumber** netid@lectura.cs.arizona.edu.

   For windows user: Please follow this guide: Port forwarding using PuTTy (created by a previous TA, Zhenge Zhao): use **portnumber** as the port number for the Source Port Field and **localhost:portnumber** for the Destination field.

   portnumber should be the number obtained in **Getting your own port number** section

4. Copy program4_template to lectura
5. Cd into program4_template
6. Execute this command: **mvn install:install-file -Dfile=ojdbc14.jar -DgroupId=com.oracle -DartifactId=ojdbc14 -Dversion=11.2.0 -Dpackaging=jar**

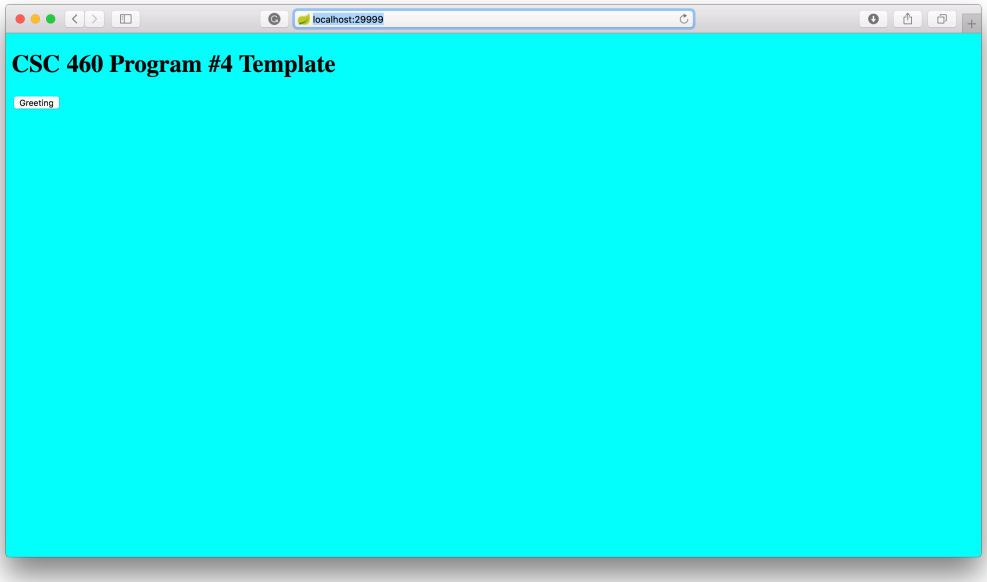
   You should only have to execute this command once. Proceeding startups can start at step #7.

7. Execute this command: **mvn spring-boot:run**
   The first time you execute this, maven will download all the dependencies. Further startups will be faster.

```
2018-11-13 08:06:59.487  INFO 18120 --- [           main] hello.Application                        : Starting Application on lectura.cs.arizona.edu with
PID 18120 (/p3/hb/baileynottingham/CSC460_TA_FA18/testingTheDemo/program4_demo/target/classes started by baileynottingham in
/p3/hb/baileynottingham/CSC460_TA_FA18/testingTheDemo/program4_demo)
2018-11-13 08:06:59.493  INFO 18120 --- [           main] hello.Application                        : No active profile set, falling back to default
profiles: default
2018-11-13 08:06:59.578  INFO 18120 --- [           main] ConfigServletWebServerApplicationContext : Refreshing
org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicationContext@1e2dee96: startup date [Tue Nov 13 08:06:59 MST 2018]; root
of context hierarchy
2018-11-13 08:07:01.647  INFO 18120 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 29999 (http)
2018-11-13 08:07:01.678  INFO 18120 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2018-11-13 08:07:01.678  INFO 18120 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet Engine: Apache Tomcat/8.5.34
2018-11-13 08:07:01.697  INFO 18120 --- [ost-startStop-1] o.a.catalina.core.AprLifecycleListener   : The APR based Apache Tomcat Native library which
allows optimal performance in production environments was not found on the java.library.path:
[/usr/java/packages/lib/amd64:/usr/lib64:/lib64:/lib:/usr/lib]
2018-11-13 08:07:02.026  INFO 18120 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2018-11-13 08:07:02.028  INFO 18120 --- [ost-startStop-1] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed
in 2455 ms
2018-11-13 08:07:02.141  INFO 18120 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean  : Servlet dispatcherServlet mapped to [/]
2018-11-13 08:07:02.148  INFO 18120 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'characterEncodingFilter' to: [/*]
2018-11-13 08:07:02.149  INFO 18120 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'hiddenHttpMethodFilter' to: [/*]
2018-11-13 08:07:02.149  INFO 18120 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'httpPutFormContentFilter' to: [/*]
2018-11-13 08:07:02.149  INFO 18120 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'requestContextFilter' to: [/*]
2018-11-13 08:07:02.221  INFO 18120 --- [           main] o.s.j.d.DriverManagerDataSource          : Loaded JDBC driver: oracle.jdbc.driver.OracleDriver
2018-11-13 08:07:02.368  INFO 18120 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**/favicon.ico] onto handler of type
[class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-11-13 08:07:02.732  INFO 18120 --- [           main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice:
org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicationContext@1e2dee96: startup date [Tue Nov 13 08:06:59 MST 2018]; root
of context hierarchy
2018-11-13 08:07:02.867  INFO 18120 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/greeting],methods=[GET]}" onto public
java.lang.String hello.GreetingController.greetingForm(org.springframework.ui.Model)
2018-11-13 08:07:02.868  INFO 18120 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/greeting],methods=[POST]}" onto public
java.lang.String hello.GreetingController.greetingSubmit(hello.Greeting)
2018-11-13 08:07:02.874  INFO 18120 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/addPerson],methods=[GET]}" onto public
java.lang.String hello.PersonController.personForm(org.springframework.ui.Model)
2018-11-13 08:07:02.875  INFO 18120 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/addPerson],methods=[POST]}" onto public
java.lang.String hello.PersonController.personSubmit(hello.Person)
2018-11-13 08:07:02.875  INFO 18120 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/deletePerson],methods=[GET]}" onto public
java.lang.String hello.PersonController.personFormDelete(org.springframework.ui.Model)
2018-11-13 08:07:02.876  INFO 18120 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/deletePerson],methods=[POST]}" onto public
java.lang.String hello.PersonController.personDelete(hello.Person)
2018-11-13 08:07:02.876  INFO 18120 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/queryResults],methods=[GET]}" onto public
java.lang.String hello.PersonController.queryResults(org.springframework.ui.Model)
2018-11-13 08:07:02.885  INFO 18120 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto public
org.springframework.web.servlet.ModelAndView
org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.errorHtml(javax.servlet.http.HttpServletRequest,javax.servlet.http.HttpServle
tResponse)
2018-11-13 08:07:02.885  INFO 18120 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public
org.springframework.http.ResponseEntity<java.util.Map<java.lang.String, java.lang.Object>>
org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.error(javax.servlet.http.HttpServletRequest)
2018-11-13 08:07:02.938  INFO 18120 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/webjars/**] onto handler of type
[class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-11-13 08:07:02.939  INFO 18120 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**] onto handler of type [class
org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-11-13 08:07:03.153  INFO 18120 --- [           main] o.s.b.a.w.s.WelcomePageHandlerMapping    : Adding welcome page: class path resource
[static/index.html]
2018-11-13 08:07:03.452  INFO 18120 --- [           main] o.s.j.e.a.AnnotationMBeanExporter        : Registering beans for JMX exposure on startup
2018-11-13 08:07:03.493  INFO 18120 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 29999 (http) with context
path ''
2018-11-13 08:07:03.498  INFO 18120 --- [           main] hello.Application                        : Started Application in 4.713 seconds (JVM running for
23.746)
```

You should see ‘Started Application’ as highlighted above.

In the browser of your choice go to:
localhost:portnumber
to go directly to your home page.
localhost:portnumber/greeting
will take you directly to greeting.html



Sources and further reading:

Spring MVC tutorials

https://spring.io/guides/gs/serving-web-content/
https://spring.io/guides/gs/maven/
https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-sql.html
https://spring.io/guides/gs/handling-form-submission/
https://www.baeldung.com/spring-jdbc-jdbctemplate

JavaPoint

https://www.javatpoint.com/autowiring-in-spring

TutorialsPoint

https://www.tutorialspoint.com/spring/spring_bean_definition.htm
https://www.tutorialspoint.com/spring/spring_bean_definition.htm

logicBig tutorial

https://www.logicbig.com/tutorials/spring-framework/spring-data-access-with-jdbc/connect-oracle.html

Maven

https://maven.apache.org/what-is-maven.html