(../..)

# Requirements

Lab Requirements (../requirements)

# Purpose of this lab

- Create a simple Spring web application.
- Create a simple Spring Boot application.
- Compare and contrast each.

# Create a Spring Web Application

1. Create a new directory for your app and navigate to it.

```
mkdir ~/workspace/spring-web-mvc
cd ~/workspace/spring-web-mvc
```

2. Create gradle wrapper and a `build.gradle` file.

```
gradle wrapper
touch build.gradle
```

3. Open your project in IntelliJ. If you have not installed the IntelliJ command line launcher, do it now: `Tools > Command-line Launcher...`.

```
idea .
```

4. Fill `build.gradle` with the following content:

    - Spring Web MVC dependency
    - Spring Context Support dependency
    - Freemarker template dependency
    - Javax Servlet API dependency
    - Gradle War plugin
    - Gradle Java plugin
    - Gretty plugin (http://akhikhl.github.io/gretty-doc/Getting-started.html) (to run the warfile with tomcat)
    - Gretty configuration to run server at `localhost:8080/`

    Take a look at our solution for hints.

    Show build.gradle

5. Create a `WebAppInitializer` class in the `io.pivotal.workshop` package. Implement the `WebApplicationInitializer` interface and set up the Spring Context and Servlet. Traditionally this configuration is done in the `web.xml`, but we prefer not to write XML configuration. Again, take a look at our solution for hints.

    Show WebAppInitializer.java

6. Create a `WebMvcConfig` class in the `io.pivotal.workshop` package. Add a `FreeMarkerViewResolver` bean that looks for `.ftl` templates. Add a `FreeMarkerConfigurer` bean that looks for the templates in the `/WEB-INF/templates/` directory.

    Show WebMvcConfig.java

7. Create an `index.ftl` file in `/src/main/webapp/WEB-INF/templates/` with the following content:

    Hide index.ftl

    src/main/webapp/WEB-INF/templates/index.ftl (https://github.com/platform-acceleration-lab/apps-spring-mvc-code/blob/master/src/main/webapp/WEB-INF/templates/index.ftl)

    view on **GitHub** (https://github.com/platform-acceleration-lab/apps-spring-mvc-code/blob/master/src/main/webapp/WEB-INF/templates/index.ftl)

```
<!DOCTYPE html>

<html>
<head>
    <meta charset="utf-8">
    <title>Spring MVC App</title>
</head>
<body>
<a href="/message">Show the message</a>
</body>
</html>
```

8. Create an `IndexController` with a method that maps the root path to our `index.ftl` file.

   Show IndexController.java

9. Run your application, then navigate to localhost:8080 (http://localhost:8080) to make sure you are on the right track.

```
./gradlew tomcatRunWar
```

   Note that the link on the index page is broken. We will hook it up in the next step.

10. Create a `message.ftl` file in `/src/main/webapp/WEB-INF/templates/` with the following content:

   Hide message.ftl

   src/main/webapp/WEB-INF/templates/message.ftl (https://github.com/platform-acceleration-lab/apps-spring-mvc-code/blob/master/src/main/webapp/WEB-INF/templates/message.ftl)

   view on **GitHub** (https://github.com/platform-acceleration-lab/apps-spring-mvc-code/blob/master/src/main/webapp/WEB-INF/templates/message.ftl)

```
<!DOCTYPE html>

<html>
<head>
    <meta charset="utf-8">
    <title>Spring MVC App</title>
</head>
<body>
<h2>${message}</h2>
</body>
</html>
```

11. Create a `MessageController` with a `message()` method that maps the `/message` path to our `message.ftl` file. Within this method create a model with a message attribute.

    Show MessageController.java

12. Run your application, then navigate to localhost:8080 (http://localhost:8080). Click on *Show the message* to view your message.

You now have a working Spring Web MVC application. Next, we will create a Spring Boot app, which will require a bit less configuration.

# Create a Spring Boot Web Application

## From scratch

1. Create a new directory for your app and navigate to it.

```
mkdir ~/workspace/spring-boot
cd ~/workspace/spring-boot
```

2. Create gradle wrapper and a `build.gradle` file.

```
gradle wrapper
touch build.gradle
```

3. Open your project in IntelliJ.

```
idea .
```

4. Fill `build.gradle` with the following content:
   - `spring-boot-gradle-plugin` buildscript dependency
   - `spring-boot-starter-freemarker` dependency
   - `java` plugin

   Take a look at our solution for hints.

   Show build.gradle

5. Create a Spring Boot `Application` class in the `io.pivotal.workshop` package.

   Show Application.java

6. Create an `index.ftl` in `/src/main/resources/templates` with the following content:

   Hide index.ftl

   src/main/resources/templates/index.ftl (https://github.com/platform-acceleration-lab/apps-spring-boot-code/blob/master/src/main/resources/templates/index.ftl)

   view on **GitHub** (https://github.com/platform-acceleration-lab/apps-spring-boot-code/blob/master/src/main/resources/templates/index.ftl)

```
<!DOCTYPE html>

<html>
<head>
    <meta charset="utf-8">
    <title>Spring Boot App</title>
</head>
<body>
<a href="/message">Show the message</a>
</body>
</html>
```

7. Create an `IndexController` with a method that maps the root path to our `index.ftl` file.

   Show IndexController.java

8. Create a `message.ftl` file in `/src/main/resources/templates` with the following content:

   Hide message.ftl

   src/main/resources/templates/message.ftl (https://github.com/platform-acceleration-lab/apps-spring-boot-

code/blob/master/src/main/resources/templates/message.ftl)

view on **GitHub** (https://github.com/platform-acceleration-lab/apps-spring-boot-code/blob/master/src/main/resources/templates/message.ftl)

```
<!DOCTYPE html>

<html>
<head>
    <meta charset="utf-8">
    <title>Spring Boot App</title>
</head>
<body>
<h2>${message}</h2>
</body>
</html>
```

9. Create a `MessageController` with a `message()` method that maps the `/message` path to our `message.ftl` file. Within this method create a model with a message attribute.

   Show MessageController.java

10. Run your application with `./gradlew bootRun`, then navigate to localhost:8080 (http://localhost:8080). Click on *Show the message* to view your message.

Compare the amount of configuration required for a Spring Boot application to the configuration required for a Spring Web MVC application. What trade-offs are we making here?

## Using IntelliJ

This method will only work using IntelliJ ultimate edition. If you are using the Community Edition, use start.spring.io (http://start.spring.io/) to generate your application.

1. Open IntelliJ and choose `File > New > Project...` from the menu.
2. Select `Spring Initializr` from the right-hand pane and click `Next`.
3. Select type: `Gradle`, and click `Next`
4. You may explore other options that are available to create a new Spring Boot project.

We can use this feature to save time when creating Spring Boot applications in the upcoming labs.