Spring Boot 入門 ハンズオン

Java Doでしょう #09

講師役：haruki-ueno

環境セットアップ

- 教科書のURL
- ハンズオン[環境のセットアップ]の手順URL
- 下記の手順ぐらいまで順次進めてください

1. https://github.com/java-do/20170422_seminar　にブラウザでアクセスしよう

SpringBoot

Mavenでプロジェクトを作成

- 以下のコマンドを実行する

- $ mvn -B archetype:generate
  -DgroupId=jp.javado.springboot
  -DartifactId=springboot-handson1
  -Dversion=1.0.0-SNAPSHOT
  -DarchetypeArtifactId=
  maven-archetype-quickstart

- 入力する際は改行なしの一行（表示の都合で改行されています）

Mavenのプロジェクト確認

- springboot-handson1のフォルダができていることを確認
- フォルダ内に、下記のフォルダ/ファイルが存在
  - src
  - pom.xml

作成したプロジェクトを統合開発環境にインポート

- IntelliJの場合
  - “Project from Existing Sources”
    > “Import project from external model”
    > “Maven”
  - “Import Maven projects automatically”をチェック
  - あとはデフォルト設定で作成
- Eclipseの場合

pom.xmlを編集（赤字を追加）

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
～省略～
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.5.2.RELEASE</version>
  </parent>
  <dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>junit</groupId>
～省略～
   </dependency>
  </dependencies>
  <build>
    <plugins>
     <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
  <properties>
    <java.version>1.8</java.version>
  </properties>
</project>
```

赤字は
SpringBootの設定

SpringBootの起動クラス

jp.javado.springbootパッケージに、
JavaDoSpringApplicationクラスを作成してみましょう

```
package jp.javado.springboot;

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

@SpringBootApplication
public class JavaDoSpringApplication {

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

起動確認

- javaDoSpringApplicationクラスを実行してみましょう
  - コマンドラインの場合
    - $ mvn spring-boot:run
  - IntelliJの場合
  - Eclipseの場合

起動確認

・ 正常に起動すると以下が出ます

```
dhcp7:springboot-handson1 ueno$ mvn spring-boot:run
[INFO] Scanning for projects...
〜〜中略〜〜
[INFO] --- spring-boot-maven-plugin:1.5.2.RELEASE:run (default-cli) @ springboot-handson1 ---


  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.5.2.RELEASE)

2017-04-17 13:38:16.428  INFO 9666 --- [main] j.j.springboot.JavaDoSpringApplication   : Starting
JavaDoSpringApplication on dhcp7.kk.pilot.chitose.ac.jp with PID 9666 (/Users/ueno/javado/springboot-
handson1/target/classes started by ueno in /Users/ueno/javado/springboot-handson1)
〜〜中略〜〜
2017-04-17 13:38:19.008  INFO 9666 --- [main] o.s.j.e.a.AnnotationMBeanExporter        : Registering
beans for JMX exposure on startup
2017-04-17 13:38:19.086  INFO 9666 --- [main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat
started on port(s): 8080 (http)
2017-04-17 13:38:19.094  INFO 9666 --- [main] j.j.springboot.JavaDoSpringApplication   : Started
JavaDoSpringApplication in 3.043 seconds (JVM running for 7.424)
```

ここまでの流れ

画面を作って見ましょう

- pom.xmlに赤字の箇所を追加

```
<dependencies>
  <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
  </dependency>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Web画面用Controller

- 以下のIndexControllerクラスを
  jp.javado.springbootパッケージに作成しましょう

```
package jp.javado.springboot;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class IndexController {

    @RequestMapping("/web/index")
    public String index() {
        return "index";
    }
}
```

templatesフォルダ作成

- springboot-handson1/src/main/resourcesフォルダ配下にtemplatesフォルダを作成

- もし、springboot-handson1/src/main以下にresourcesフォルダがない場合は作成しましょう

htmlの作成

- src/main/resources/templates配下に
  index.htmlを作成しましょう

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8"/>
    <title>Index</title>
</head>
<body>
<h1>Hello Web page</h1>
<p>This is Thymeleaf page on Spring Boot.</p>
</body>
</html>
```

Controller側で設定した値を表示

- では、Controller側とhtmlでの値の受け渡しを作って見ましょう

- IndexControllerを編集

- index.htmlを編集

Controllerの編集

```
package jp.javado.springboot;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;



@Controller
public class IndexController {

    @RequestMapping("/web/index")
    public ModelAndView index(ModelAndView modelAndView) {
        modelAndView.setViewName("index");
        modelAndView.addObject("message1", "Controllerで設定した文字列");

        return modelAndView;
    }
}
```

修正：赤字の箇所

ModelAndViewクラスを使うことで、
ControllerからHtmlに値を渡すことができる

SpringでDI

- SpringでDIについて

ServiceクラスをDI

- IndexControllerにServiceクラスをDIしてみましょう

ISampleServiceの作成

- ServiceクラスのInterfaceを作成

```
package jp.javado.springboot;


public interface ISampleService {

    public String getSample(String id);

}
```

SampleServiceを作成

- ISampleServiceをimplementsしたクラス

```
package jp.javado.springboot;

import org.springframework.stereotype.Service;

import java.util.HashMap;
import java.util.Map;


@Service
public class SampleService implements ISampleService {

    @Override
    public String getSample(String id) {
        Map<String, String> map = new HashMap<>();
        map.put("1", "spring");
        map.put("2", "boot");
        map.put("3", "di");
        return map.get(id);
    }
}
```

IndexControllerを編集

- 作成したSampleServiceをDIする

```
package jp.javado.springboot;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;


@Controller
public class IndexController {

    @Autowired
    private ISampleService sampleService;

    @RequestMapping("/web/index")
    public ModelAndView index(ModelAndView modelAndView) {
        modelAndView.setViewName("index");
        modelAndView.addObject("message1", "Controllerで設定した文字列");
        modelAndView.addObject("dimessage", sampleService.getSample("2"));
        return modelAndView;
    }
}
```

ページのリンク

- @{ アドレス }でページのリンクを作ることができます

<a th:href=“@{/web/list}”>List page Link

ここでは、一覧ページへのリンクを想定して
作ってみましょう

リンクを作成

- index.htmlにリンクを追加
- リンクページを作成
    - list.htmlを作成
    - ListControllerを作成

index.htmlを編集

```
<!DOCTYPE html>
<html lang="ja" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8"/>
    <title>Index</title>
</head>
<body>
<h1>Hello Web page</h1>
<p>This is Thymeleaf page on Spring Boot.</p>
<p th:text="${message1}">replace message</p>
<p><a th:href="@{/web/list}">List page Link</a></p>
</body>
</html>
```

赤字箇所を追記

aタグのth:href要素に @{/web/list} を書くことで、
http://localhost:8080/web/listへのリンクとなる

ListControllerの作成

- IndexControllerをコピーして以下となるようにしましょう

```
package jp.javado.springboot;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;


@Controller
public class ListController {

    @RequestMapping("/web/list")
    public ModelAndView list(ModelAndView modelAndView) {
        modelAndView.setViewName("list");
        modelAndView.addObject("message1", "Controllerで設定した文字列");
        return modelAndView;
    }
}
```

list.htmlの作成

- index.htmlをコピーして以下となるようにlist.html を作成しましょう

```
<!DOCTYPE html>
<html lang="ja" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8"/>
    <title>List</title>
</head>
<body>
<h1>List page</h1>
<p>This is Thymeleaf page on Spring Boot.</p>
<p th:text="${message1}">replace message</p>
</body>
</html>
```