LightHouseIT

3章 开发框架Spring　開発フレームワーク Spring

- 开发框架　開発フレームワーク
- Springboot环境配置　Springbootの環境設定
- 实践操作及讲解　プログラム実践と解説

LightHouseIT

1 开发框架
[開発フレームワーク]

2 Springboot环境配置
[Springbootの環境設定]

3 实践操作及讲解
[プログラム実践と解説]

4

开发框架

- 框架[framework]一般被定义一种软件，在开发应用程序时预先提供常用的通用功能，作为应用程序的基础。
- 换句话说，框架是一个系统，其中始终使用的部分从一开始就被创建，只有那些因应用程序而改变的部分才被创建，以创建一个单一的应用程序。

使用开发框架的好处和坏处

框架优势

- 提高生产力：通过使用应用程序基础部分的框架，可以省略基础部分的开发。可以将更多时间用于主要部分。
- 同构开发：通过遵循框架使用规则，可以开发出保持一定质量水平的应用程序。
- 缩短测试过程：通过使用框架提供的功能，不需要对该部分进行单元测试。
- 可维护性提高：由于是根据规则创建的，因此更容易掌握应用程序的整体情况，可维护性有望相应提高。

框架的缺点

- 使用成本高：使用的框架功能越多，整体结构和处理就越难掌握。
- 需要学习：需要学习新的框架规则（如何使用方法，函数等）。
- 框架错误：可能包含无法解释的错误。
- 框架选择问题：如果不选择适合正在开发的系统的框架，将无法满足开发要求，或者相反，将因匹配框架规范而产生开发成本。

常用的一些开发框架

1 Spring Framework
与其创建类（或库）工作所需的其他类，不如让 Spring 框架生成它们。生成的类由 Spring Framework 配置文件自动组合。通过这样做，您可以消除类和库的依赖关系，使单元测试更容易，并提高每个类和库的可重用性。

2 Spring boot
Spring boot是 Java 平台的应用程序框架，专门用于 Web 应用程序开发。无需编辑 Spring 框架所需的配置文件，大部分设置都是自动化的，无需指定类路径。

3 Apache Struts
它是自 2001 年以来一直在使用的 Java 中最著名的框架。采用MVC（Model View Controller）模型。它是应用程序设计技术之一，Struts 2 有很多改进。尽管它作为一个框架有历史，但近年来，由于发现漏洞，越来越多的用户正在迁移到别的框架。

4 JSF
它是 2004 年开发的 Java 标准框架，被 Java EE 规范采用。它使用类似于 Apache Struts 的 MVC，但它有一些差异，例如基于组件的框架和用于显示的 XML 样式的 HTML。在 HTML 的情况下，它在浏览器中显示，因此很容易检查设计。

5 Play Framework
它不仅可以在 Java 中使用，还可以在与 Java 具有高度亲和力的语言 Scala 中使用。它深受“Ruby on Rails”和“Django”的影响，并且因为它使用较少的CPU资源和内存，所以它是轻量级和高产的。您还可以在不使用 Servlet 或 JSP 的情况下使用新方法构建应用程序。

总结
除了上面介绍的框架之外，还有其他框架。不能说每个框架中哪个是最好的。各有各的特点，重要的是要根据系统开发的目的来决定选择哪个框架。

LightHouseIT

1 开发框架
[開発フレームワーク]

2 **Springboot环境配置**
[Springbootの環境設定]

3 实践操作及讲解
[プログラム実践と解説]

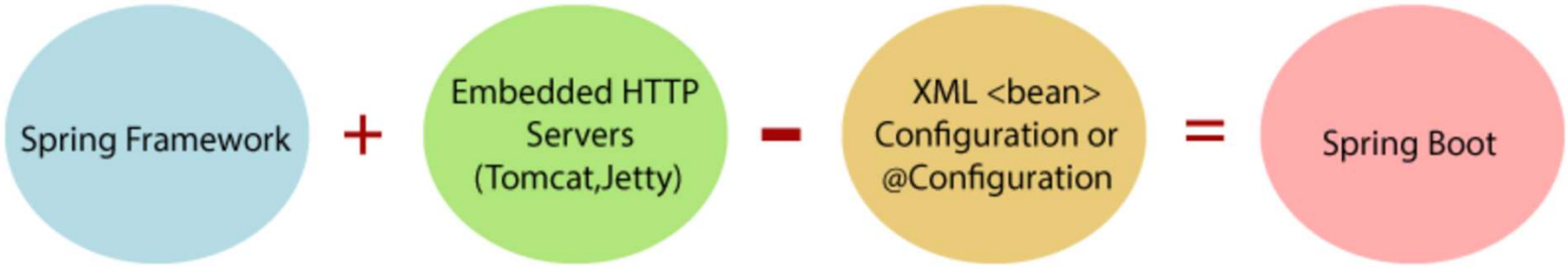
4

Spring框架

- Spring是一个支持快速开发Java EE应用程序的框架。它提供了一系列底层容器和基础设施，并可以和大量常用的开源框架无缝集成。

Spring Boot 概述

- Spring Boot是一个基于Spring的套件，它帮我们预组装了Spring的一系列组件，以便以尽可能少的代码和配置来开发基于Spring的Java应用程序。
- Spring Boot的目标就是提供一个开箱即用的应用程序架构，我们基于Spring Boot的预置结构继续开发，省时省力。



安装Spring开发工具STS —1

- 官方下载链接：
https://spring.io/tools

Spring Tools 4 for Eclipse

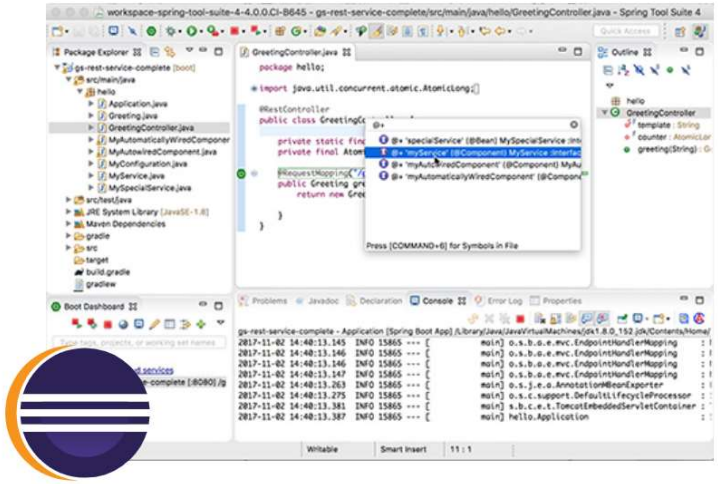
The all-new Spring Tool Suite 4.
Free. Open source.

4.14.1 - LINUX X86_64

4.14.1 - MACOS X86_64

4.14.1 - MACOS ARM_64

4.14.1 - WINDOWS X86_64

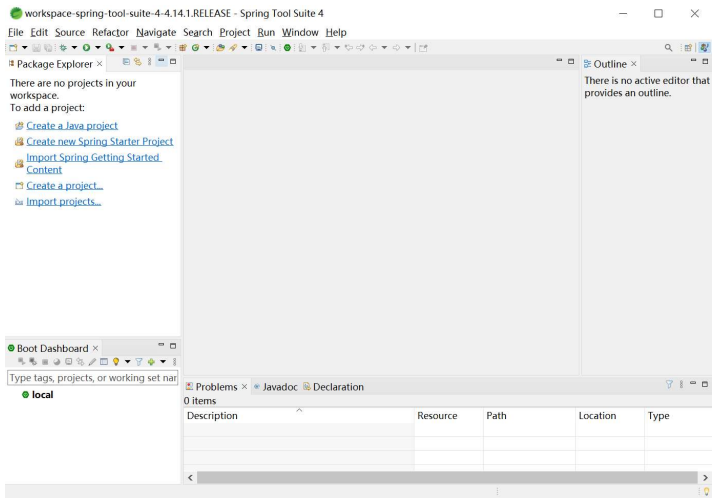


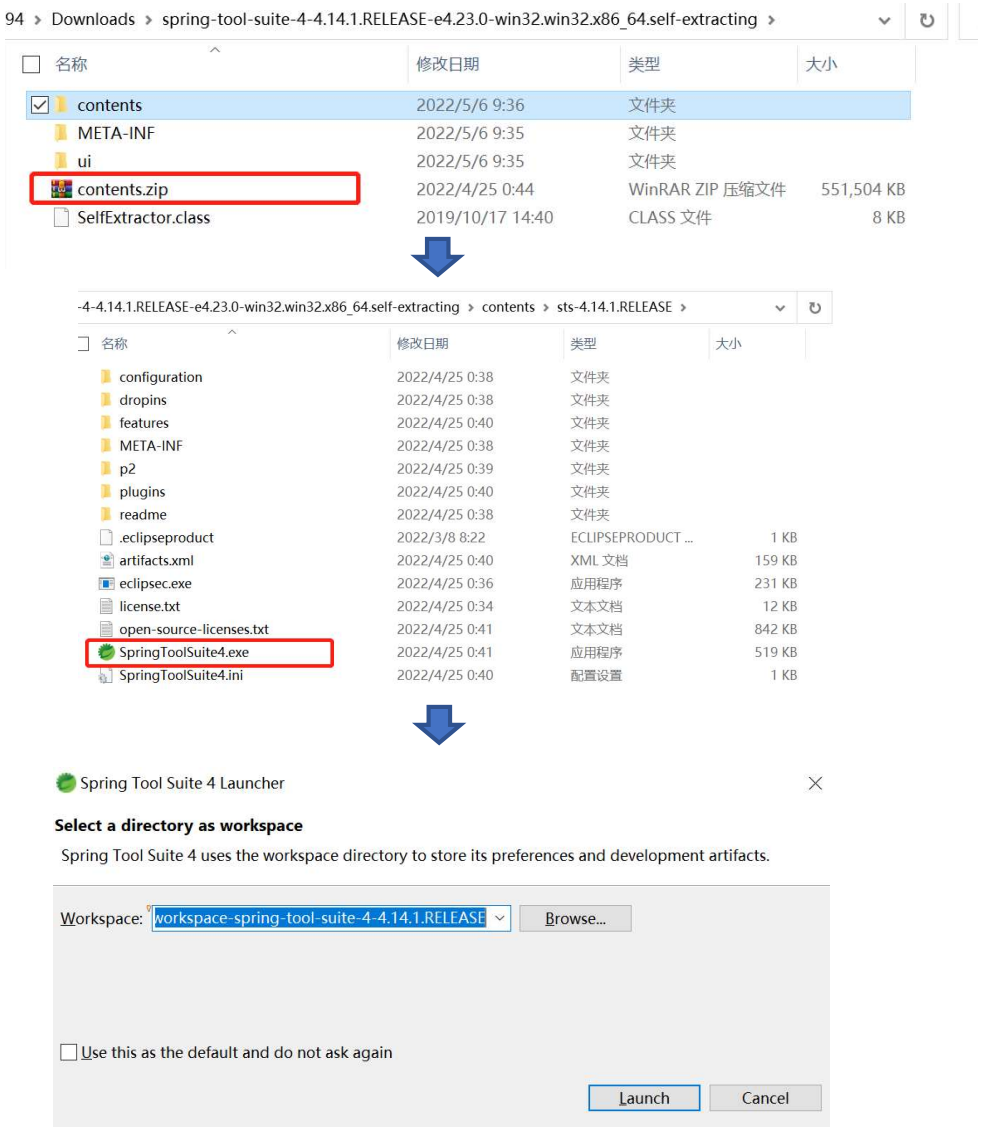
安装Spring开发工具STS —2

- 双击下载的“spring-tool-suite-4-4.7.0.RELEASE-e4.16.0-win32.win32.x86_64.self-extracting.jar”*解压。
  * 即使文件名的版本略有不同也没有问题。

- 将解压后的文件夹“sts-4.7.0.RELEASE”连同C盘正下方的文件夹一起移动。
  移动后的路径：C:\sts-4.7.0.RELEASE
  * 即使文件夹名称的版本稍有不同也没有问题。
  * 确切地说，放在任何文件夹中都没有问题，但这次将如上文所述。

- 双击“sts-4.7.0.RELEASE”文件夹中的“SpringToolSuite4.exe”，确认STS的启动。





LightHouseIT

1 开发框架
[開発フレームワーク]

2 Springboot环境配置
[Springbootの環境設定]

3 **实践操作及讲解**
[プログラム実践と解説]

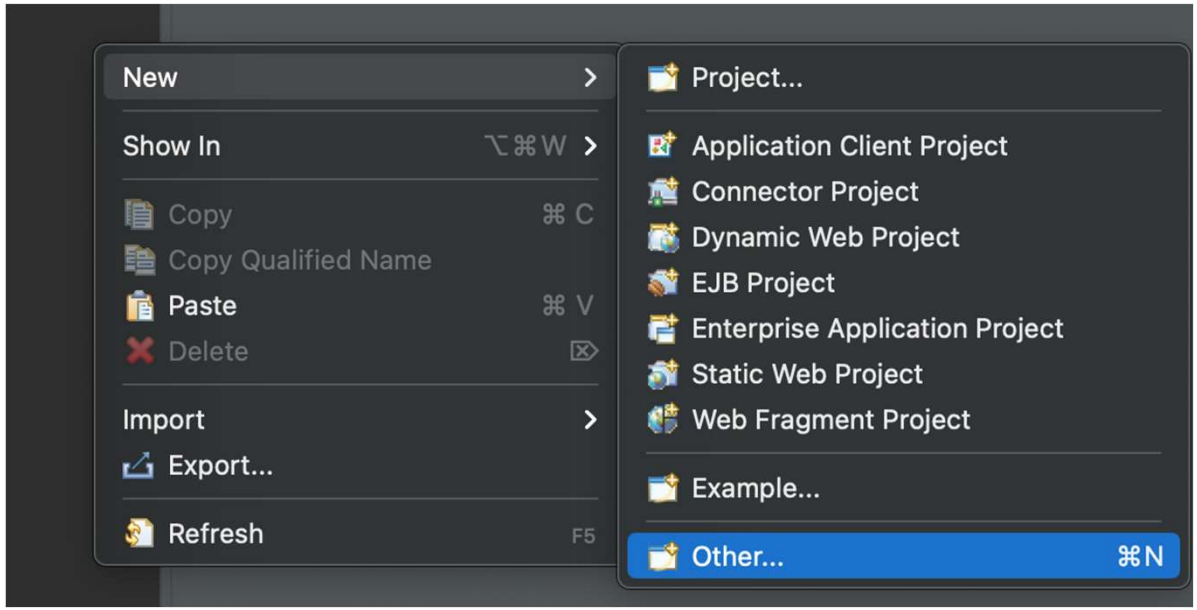
4

本节课目标—简单登录系统

- 本节课带大家创建没有数据库交互的简单登录系统

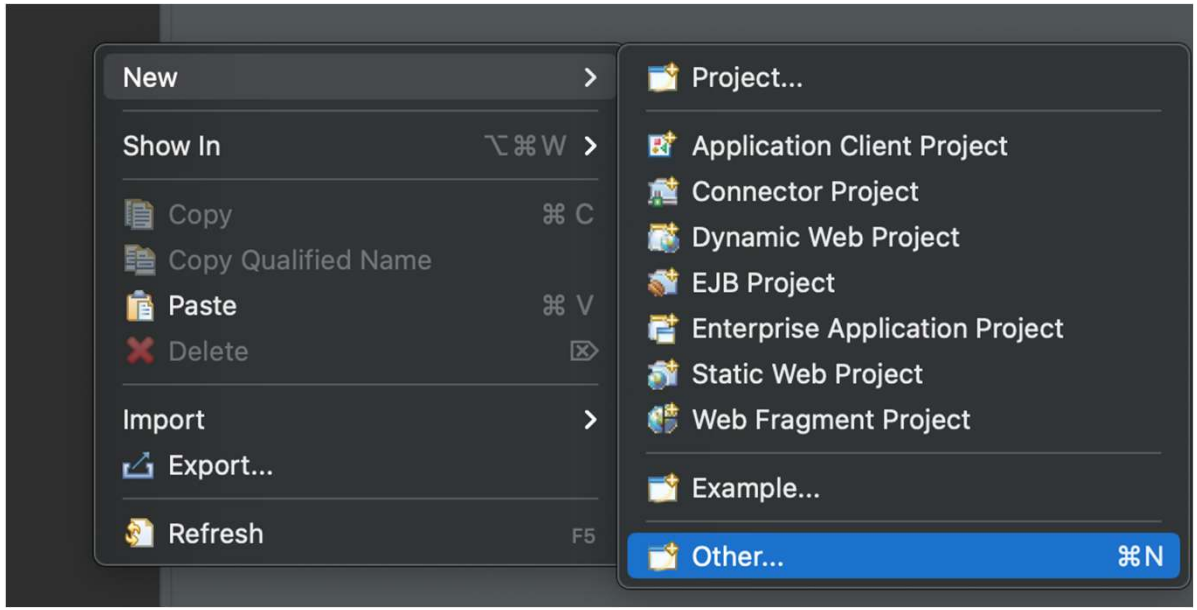
创建Spring Boot的Maven项目 一 1

- 在Eclipse左侧右键（或者点击左上角File）选择New→Other



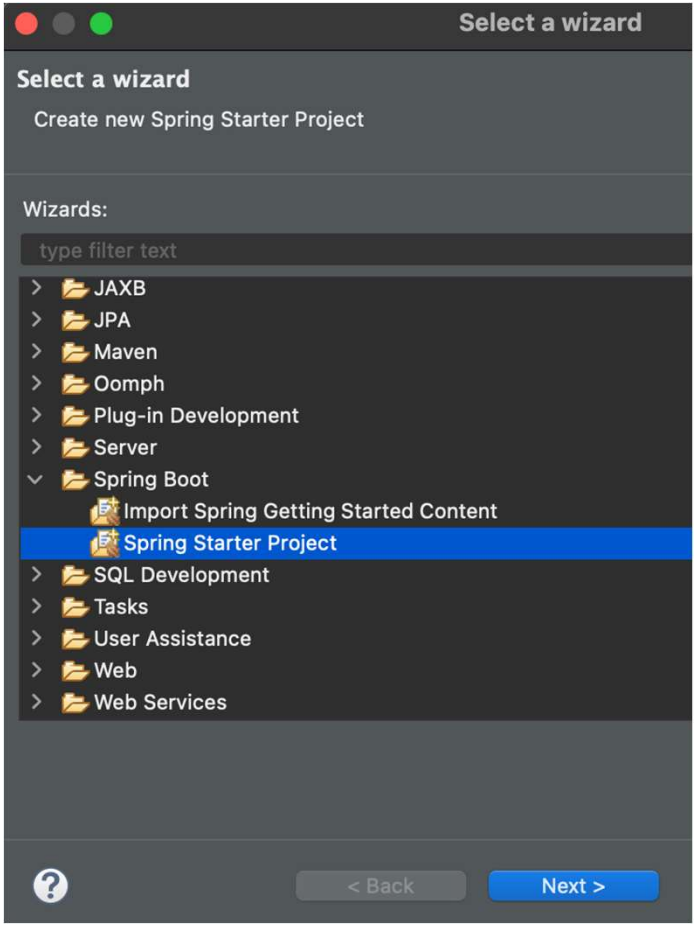
创建Spring Boot的Maven项目 一 2

- 在Eclipse左侧右键（或者点击左上角File）选择New→Other



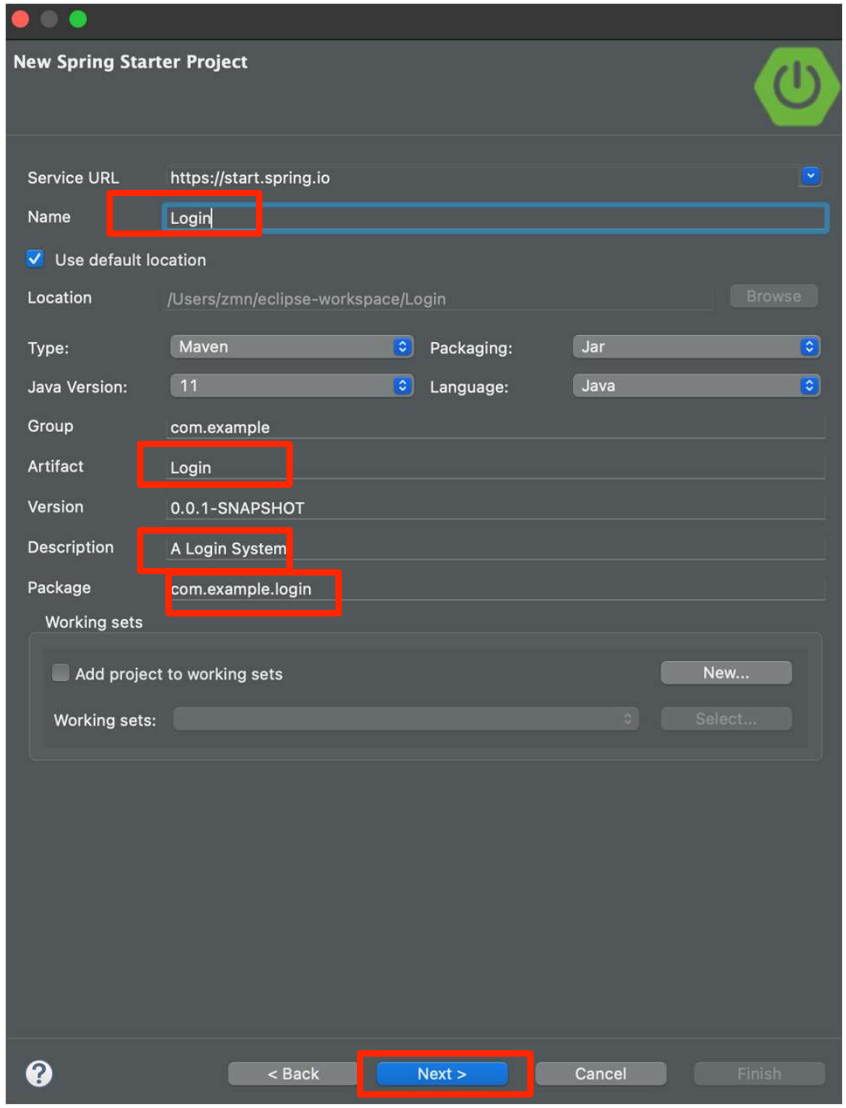
创建Spring Boot的Maven项目 — 3

- 找到Spring Boot > Spring Starter Project，点击Next



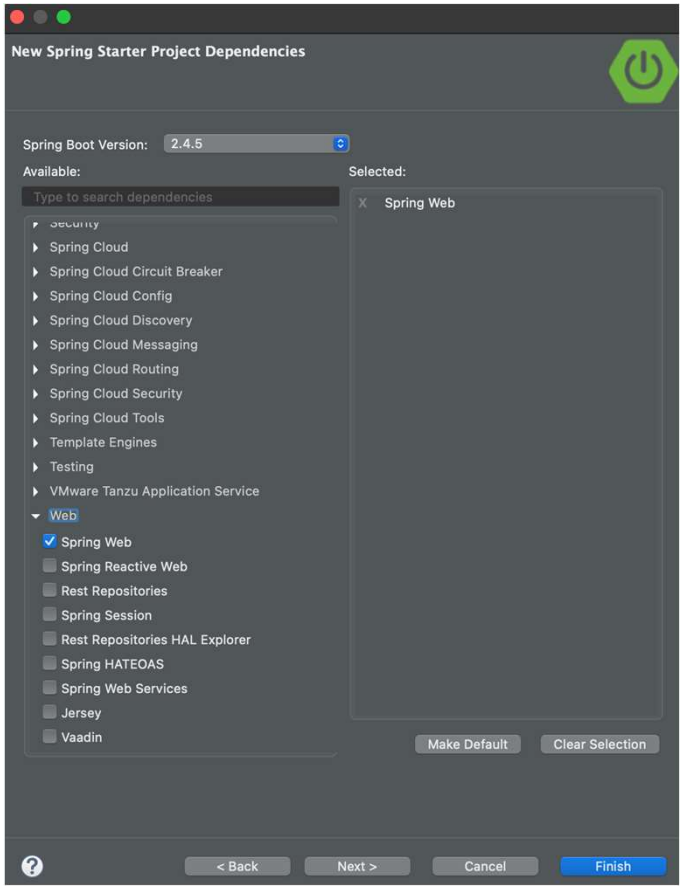
创建Spring Boot的Maven项目 一4

- 修改为如图所示配置

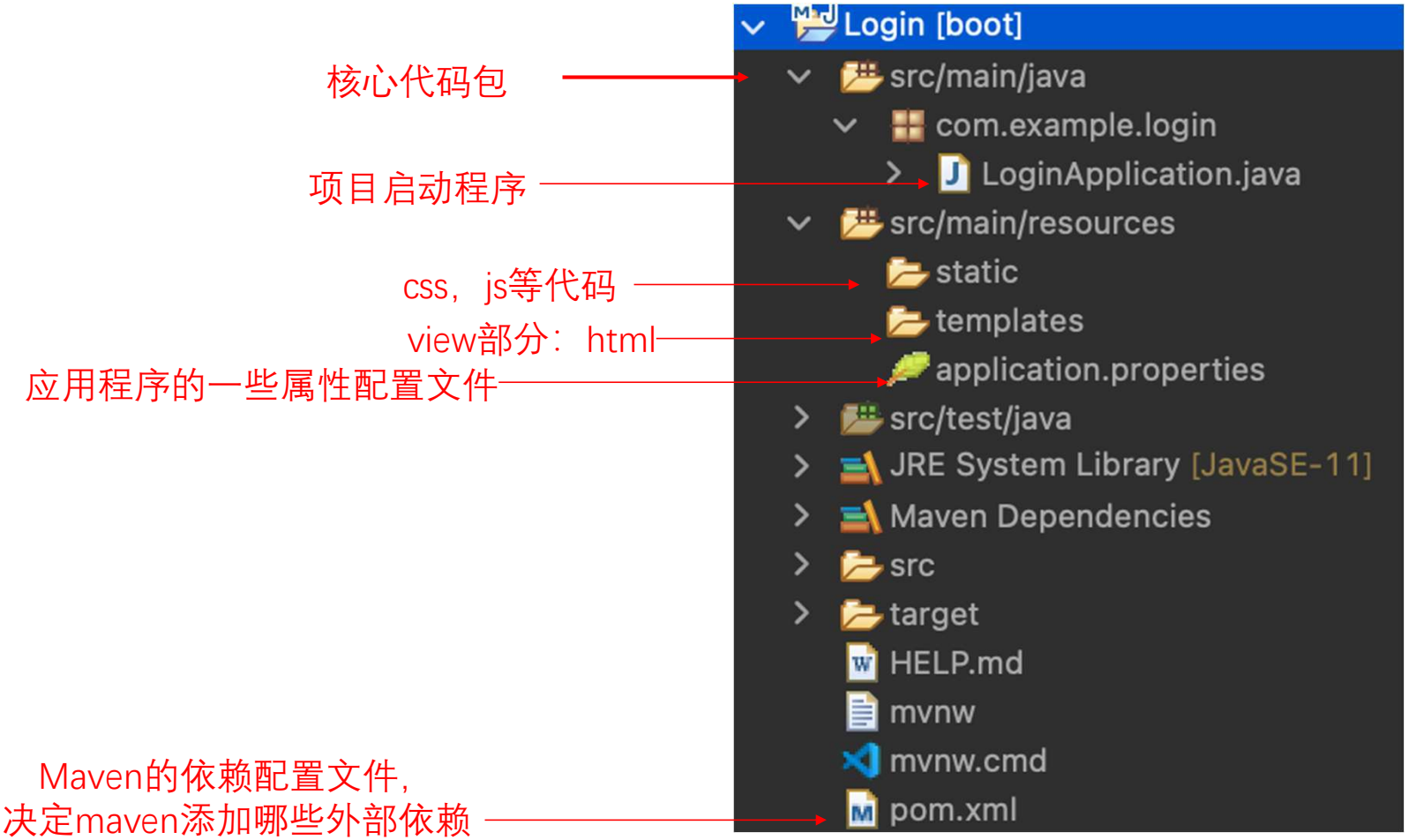


创建Spring Boot的Maven项目 —5

- 搜索并选择依赖Spring Web和Thymeleaf
- 点击Finish



项目结构



查看pom.xml

- pom.xml是用来告诉maven这个项目都需要用到哪些外部依赖，让maven去远程仓库下载需要的依赖包到本地
- 这些是这个项目会用到的依赖

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

项目启动程序

- STS 创建的 Spring starter 项目从一开始就带有一个 Java 源代码文件。从 STS 包资源管理器中，展开“src/main/java”文件夹。
- 可以看到里面有一个“jp.co.f1.spring”包，里面准备了一个叫“LoginApplication.java”的源代码文件。
- 在 Spring Boot 中，不用准备配置文件等，只需编写注解，程序中使用的所有组件都会自动加载。

```
package com.example.login;

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

//注解“@SpringBootApplication”表示这个类是一个 Spring Boot 应用类。
@SpringBootApplication
public class LoginApplication {
	public static void main(String[] args) {
		//只调用了一个run方法，执行“SpringApplication”类的“run”方法可以理解为启动Spring项目。
		SpringApplication.run(LoginApplication.class, args);
	}
}
```

创建简单的网页

- 在com.example.login下创建一个HelloController.java
- 将课程代码中的HelloController.java的内容复制上来

```java
package com.example.login;

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

@RestController
public class HelloController {
    @GetMapping("/")
    public String index() {
        return "Hello World";
    }
}
```

- 打开LoginApplication.java，点击运行（Spring boot app）
- 服务器启动后，查看网址 http://localhost:8080

HelloController代码解读

```java
package com.example.login;

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

@RestController
public class HelloController {
    @GetMapping("/")
    public String index() {
        return "Hello World";
    }
}
```

- @RestController注释的目的是告诉Spring，下面这个类描述了一个网站页面，该页面要在整个项目中都有效，并返回文本
- @GetMapping("/") 注释的目的是告诉Spring使用index()方法来回应用户访问http://localhost:8080/这个网址的请求动作
- GetMapping("/")中的/部分表示用户访问的网址域名的后缀部分，Mapping注释的作用是将用户可以访问到的网址和处理该网址请求的方法匹配到一起。Get表示HTTP请求的GET方法

HelloController代码解读

- 如果将/改为/hello，则需要访问http://localhost:8080/hello才能看到刚才的页面
- 尝试以下代码（修改Controller之后需要重新运行项目）：

```
@RestController
public class HelloController {
    @GetMapping("/hello")
    public String index() {
        return "Hello World";
    }
    @GetMapping("/GoodBye")
    public String goodbye() {
        return "GoodBye World";
    }
}
```

添加devtools依赖

- 如你所见，修改代码后需要重启项目才能在网页中更新变化，这样很麻烦，不利于开发
- 解决办法：在pom.xml文件中的dependencies里面添加devtools依赖，就可以在不重启的状态下更新页面：

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

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
    </dependency>
</dependencies>
```

HTTP请求 GET方法

- 功能：从指定的资源请求数据。
- GET方法是默认的HTTP请求方法，例如当我们通过在浏览器的地址栏中直接输入网址的方式去访问网页的时候，浏览器采用的就是 GET 方法向服务器获取资源。
- Spring中的@GetMapping是专门处理GET方法请求的注释

HTTP请求 POST方法

- 功能：GET方法的一个替代方法，它主要是向Web服务器提交表单数据，尤其是大批量的数据。

```
<form action="login" method="post">
```

- Spring中的@PostMapping是专门处理POST方法请求的注释，通常处理表单提交

问题

- 登录页面的点击Login按钮的请求应该用GetMapping还是PostMapping进行处理?

如何让Controller向用户返回页面呢？—1

- 到目前为止Controller向用户浏览器返回的只是一句话（"Hello World"/"Goodbye World"）
- 我们希望可以返回一个html页面
- 步骤：
  - 将RestController注释改为Controller（可以理解为RestController返回文本，Controller返回HTML文件）
  - 在src/main/resources/templates里创建hello.html，内容如下：

```
<!DOCTYPE html>
<html lang="en" xmlns:th="https://thymeleaf.org">
    <head>
        <title>Hello</title>
    </head>
    <body>
        <h1 align="center">Hello World!</h1>
    </body>
</html>
```

之后写thymeleaf时会用到

如何让Controller向用户返回页面呢？—2

- 步骤（接上页）：
  - 将HelloController中的返回值改为如下所示：

```java
package com.example.login;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@Controller
public class HelloController {
    @GetMapping("/hello")
    public String index() {
        return "hello"; // 自动去templates里找到hello.html并返回给用户
    }
}
```

访问网址http://localhost:8080/hello

Hello World!

创建一个登录界面—1

- 步骤：
  - 在templates中创建login.html （课件代码中有）
  - 在com.example.login中创建一个LoginController.java，代码如下：

```java
package com.example.login;

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

@Controller
public class LoginController {
    @GetMapping("/login")
    public String login() {
        return "login"; // 自动去templates里找到login.html并返回给用户
    }
}
```

  - 访问网页http://localhost:8080/login

创建一个登录界面—2

如何处理登录表单？—1

- 目前的登录界面没有实际的登录功能，我们希望点击login按钮后，根据我们的登录信息，页面跳转到另一个界面。
- 解决办法：在LoginController中添加一个处理表单提交的PostMapping。
- 步骤：
  - 在LoginController中添加处理表单信息的方法，代码如下：

```
@Controller
public class LoginController {
    @GetMapping("/login")
    public String login() {
        return "login"; // 自动去templates里找到login.html并返回给用户
    }

    @PostMapping("/home") // 匹配html中action为“/home”的form标签
    public String validate(@RequestParam("username") String userName, @RequestParam("password") String password) {
        if(userName.equals("admin") && password.equals("admin")) {
            return "hello";
        }
        return "login";

    }
}
```

如何处理登录表单？—2

- 步骤（接上页）：
  - 将login.html中form的action属性改为“/home”，这样才可以和PostMapping（“/home”）匹配到

```
<form action="/home" method="post">
```

  - 如果出现Request method ‘POST’ not supported错误，有可能是post方法指定的链接不对，可以尝试以下设置

```
<form action="http://localhost:8080/home" method="post">
```

  - 访问网页http://localhost:8080/login，输入admin/admin，会跳转到hello界面。

validate方法代码解读

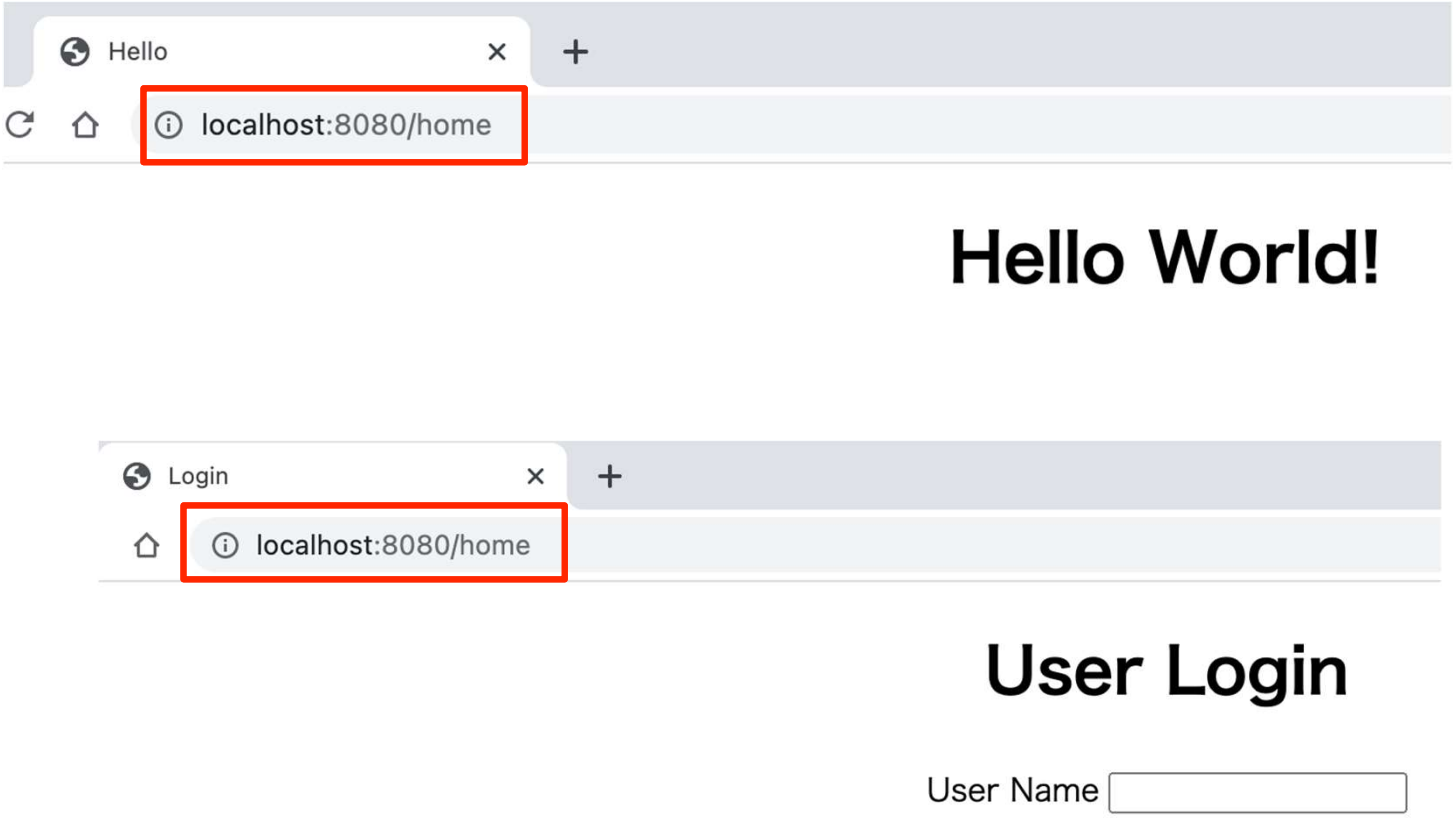
```
@PostMapping("/home") // 匹配html中action为“/home”的form标签
public String validate(@RequestParam("username") String userName, @RequestParam("password") String password) {
    if(userName.equals("admin") && password.equals("admin")) {
        return "hello"; // 用户名和密码为admin，则跳转到hello.html
    }
    return "login"; // 否则重新加载login.html

}
```

- 用户使用post方法发送HTTP请求，意味着向validate方法发送了用户名和密码等信息
- @RequestParam注释就是用来接收表单信息的注释，“username”和 “password”对应的是html中form中input的name属性，String userName和String password是将接收的信息作为这两个Java的变量作为方法的参数。

网址重定向 —1

- 细心的同学会发现login界面无论跳转到那个页面，浏览器上面的网址都是http://localhost:8080/home



网址重定向 —2

- 因为网址后缀会默认变成form的action的内容，但我们希望无论从哪个页面跳转过来，hello页面的网址依然是 http://localhost:8080/hello，login页面的网址依然是 http://localhost:8080/login
- 改成如下代码，就可以了（原理请看注释内容）：

```
@PostMapping("/home") // 匹配html中action为“/home”的form标签
public String validate(@RequestParam("username") String userName, @RequestParam("password") String password) {
    if(userName.equals("admin") && password.equals("admin")) {
        return "redirect:/hello"; // 用户名和密码为admin，则重新定向到GetMapping("/hello")
    }
    return "redirect:/login"; // 否则重新定向到GetMapping("/login")

}
```

Application.properties配置参数

- Spring Boot使用了一个全局的配置文件application.properties，放在src/main/resources目录下或者类路径的/config下。Sping Boot的全局配置文件的作用是对一些默认配置的配置值进行修改。

- 在application.properties文件中输入下面的代码、保存，可以修改应用的名字和服务端口。形式如下：【key=value】

```
spring.application.name=not-hello-world
server.port=8080
```

application.properties还可以配置程序要用到的具体参数，如数据库用户名、密码：

```
datasource.master.username=root
datasource.master.password=1234
```

Yaml文件与properties

- yaml文件是SpringBoot使用一个全局的配置文件， 配置文件名称是固定的，是用于修改SpringBoot自动配置的默认值，因为SpringBoot在底层都给我们自动配置好了。作用与properties文件基本一致，但是比properties更方便，更强大。例如更改项目的端口号。

- properties文件

```
server.port=8080
```

- Yaml文件

```yaml
server:
  port: 8080
```

预告

- 目前的登录界面只有admin能通过验证，如何做到可以登录多个用户呢？
- →之后将会学到如何连接数据库，实现真正的登录系统

本节课资料

- Spring Boot官方API：https://docs.spring.io/spring-boot/docs/current/api/
- Spring Boot 中文参考手册：https://www.springcloud.cc/spring-boot.html