第 2-2 课：Spring Boot 项目中使用 JSP

JSP（Java Server Pages）中文名叫 Java 服务器页面，其根本是一个简化的 Servlet 设计，它是由 Sun Microsystems 公司倡导、许多公司参与一起建立的一种动态网页技术标准。JSP 技术类似 ASP 技术，它是在传统的网页 HTML（标准通用标记语言的子集）文件（.html）中插入 Java 程序段（Scriptlet）和 JSP 标记（tag），从而形成 JSP 文件，后缀名为（*.jsp）。用 JSP 开发的 Web 应用是跨平台的，既能在 Linux 下运行，也能在其他操作系统上运行。

JSP 其实就是 Java 为了支持 Web 开发而推出的类前端 Servlet，可以在 JSP 中写 Java 或者 Html 语法等，后端根据 JSP 语法渲染后返回到前端显示，在没有模板引擎之前 JSP 是 Java 程序员开发人员的首选，到现在仍然有很多公司使用 JSP 开发后台管理系统。本课内容将介绍如何在 Spring Boot 项目中使用 JSP。

快速上手

项目结构

首先看一下添加 JSP 支持后的项目结构：

```
spring-boot-jsp
 +-src
    +- main
         +- java
         +- resources
         +- webapp
              +- WEB-INF
                 +- jsp
                    +- welcome.jsp
    +- test
 +-pom.xml
```

对比以前的项目结构 main 目录下多了 webapp 目录，用来存放目录 jsp 文件。

配置文件

需要在配置文件中指定 jsp 的位置和后缀。

```
spring.mvc.view.prefix: /WEB-INF/jsp/
spring.mvc.view.suffix: .jsp
```

- spring.mvc.view.prefix 指明 jsp 文件在 webapp 下的哪个目录
- spring.mvc.view.suffix 指明 jsp 以什么样的后缀结尾

引入依赖包

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
</dependency>
<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
</dependency>
```

spring-boot-starter-web 包依赖了 spring-boot-starter-tomcat 不需要再单独配置。引入 jstl 和内嵌的 tomcat，jstl 是一个 JSP 标签集合，它封装了 JSP 应用的通用核心功能。tomcat-embed-jasper 主要用来支持 JSP 的解析和运行。

编写页面

简单写一个页面：

```
<!DOCTYPE html>
<html lang="en">

<body>
    Time:  ${time}
    <br>
    Message: ${message}
</body>

</html>
```

很简单的一个页面，展示后端传到页面的时间和消息。

后端程序

```
@Controller
public class WelcomeController {

    @GetMapping("/")
    public String welcome(Map<String, Object> model) {
        model.put("time", new Date());
        model.put("message", "hello world");
        return "welcome";
    }

}
```

time 获取当前时间，message 赋值为 hello world。

测试

cmd 进入项目跟路径下：

```
cd ...\spring-boot-jsp
```

执行以下命令启动：

```
mvn clean spring-boot:run
```

启动完成后，在浏览器中访问地址：http://localhost:8080/，返回信息如下：

```
Time: Sat Aug 11 13:26:35 CST 2018
Message: hello world
```

说明项目运行成功。

常用示例

页面中常用的展示后端传值、if 判断、循环等功能，可以使用 jstl 语法处理，也可以直接写 Java 代码来实现这些逻辑，在 jsp 页面中这两种方式都支持。但不建议在 jsp 页面中编写大量的 Java 代码，从而导致前端业务复杂，可读性差等问题。下面通过一些小的示例来学习。

在 WelcomeController 类中定义一个 user() 的方法，设置一些值从后端传递到前端：

```
@GetMapping("/user")
public String user(Map<String, Object> model, HttpServletRequest request) {
    model.put("username", "neo");
    model.put("salary", 666);
    request.getSession().setAttribute("count",6);
    return "user";
}
```

将参数和值以键值对的方式存储在 Map 中，jsp 页面可以直接根据属性名来获取值，也可以通过 request 来传递后端属性和值，返回值会以键值对的方式传递到 user.jsp 页面。

在 user.jsp 文件头部添加两个标签：

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %><html lang="en">
```

引入第一个标签是为了让页面支持中文展示，第二个标签引入表示页面使用 jstl 语法来处理页面逻辑。

Java 代码

可以直接在 jsp 页面中使用 Java 代码，如果是一行 Java 代码使用 `<%= %>` 的语法，如果是多行 Java 代码则使用 `<% %>` 的语法，示例如下：

```
<h3>一行 Java 代码</h3>
<p>
    今天的日期是：<%=(new java.util.Date())%>
</p>
<h3>多行 Java 代码</h3>
<p>
    你的 IP 地址是：
<%
    out.println("Your IP address is " + request.getRemoteAddr()+"</br>");
    out.println("一段代码 ");
%>
</p>
```

For 循环是页面最常用的功能之一，一般用在循环展示表格、列表等。

```
<h3>For 循环实例</h3>
<%
    int count = (int)session.getAttribute("count");
    for ( int fontSize = 1; fontSize <=count; fontSize++){
 %>
          纯洁的微笑
<br />
<%}%>
```

根据后端传递的 count 值来选择前端页面循环次数。

jstl 语法

页面常常会使用一些逻辑判断，使用 jstl 语法很容易实现这些功能。

```
<h3>标签 c:if</h3>
<c:if test="${username !=null}">
<p>用户名为：username<p>
</c:if>
```

使用 jstl 标签的 `<c:if>` 来判断传递过来的 username 是否为空，如果不为空将 username 展示到页面。当有多条件判断时可以使用 `<c:choose>` 更方便。

```
<h3>标签 c:choose</h3>
<c:choose>
    <c:when test="${salary <= 0}">
    太惨了。
    </c:when>
    <c:when test="${salary > 1000}">
    不错的薪水，还能生活。
    </c:when>
    <c:otherwise>
    什么都没有。
    </c:otherwise>
</c:choose>
```

上述代码根据 salary 值的大小来输出不同的内容。

jstl 语法可以支持常见的功能，这里只是列举最常用的两个作为示例。

页面布局

一般网站的每个页面都拥有相同的页眉和页脚，这两部分内容一般很少发生变动，这部分内容特别适合将它提取出来，让每个页面来重复利用。

JSP 可以通过 include 指令来实现此效果，include 指令用于在编译阶段包括一个文件，这个指令告诉容器在

编译阶段，把其他外部文件的内容合并到当前 JSP 文件中，可在 JSP 页面的任何位置使用 include 指令进行编码。

include 有两种用法：`<%@ include file="relative url"%>` 和 `<jsp:include page="relative url" flush="true"/>` 。前者是在翻译阶段执行，后者是在请求处理阶段执行；前者叫作静态包含，后者叫作动态包含，会在执行时检查包含内容变化。两者使用语法没有太大区别，下面举一个简单示例。

新建一个 footer.jsp 内容如下：

```
<!DOCTYPE html>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
    <body>
    我是页尾
    </body>
</html>
```

在 user.jsp 页面中引入 footer.jsp：

```
<h3>布局</h3>
<%@include file="footer.jsp"%>
```

这样在访问 user.jsp 时，会将 footer.jsp 内容展示到 user.jsp 页面引入的位置。

上述代码都完成后，整体看一下页面的效果，启动项目在浏览器中输入网址：http://localhost:8080/user，页面展示效果如下：

一行 Java 代码

今天的日期是: Wed Sep 26 10:48:51 CST 2018

多行 Java 代码

你的 IP 地址是： Your IP address is 0:0:0:0:0:0:0:1
一段代码

标签 c:if

用户名为：username

标签 c:choose

什么都没有。

For 循环实例

纯洁的微笑
纯洁的微笑
纯洁的微笑
纯洁的微笑
纯洁的微笑
纯洁的微笑

布局

我是页尾

调试和部署

在 IDEA 中运行

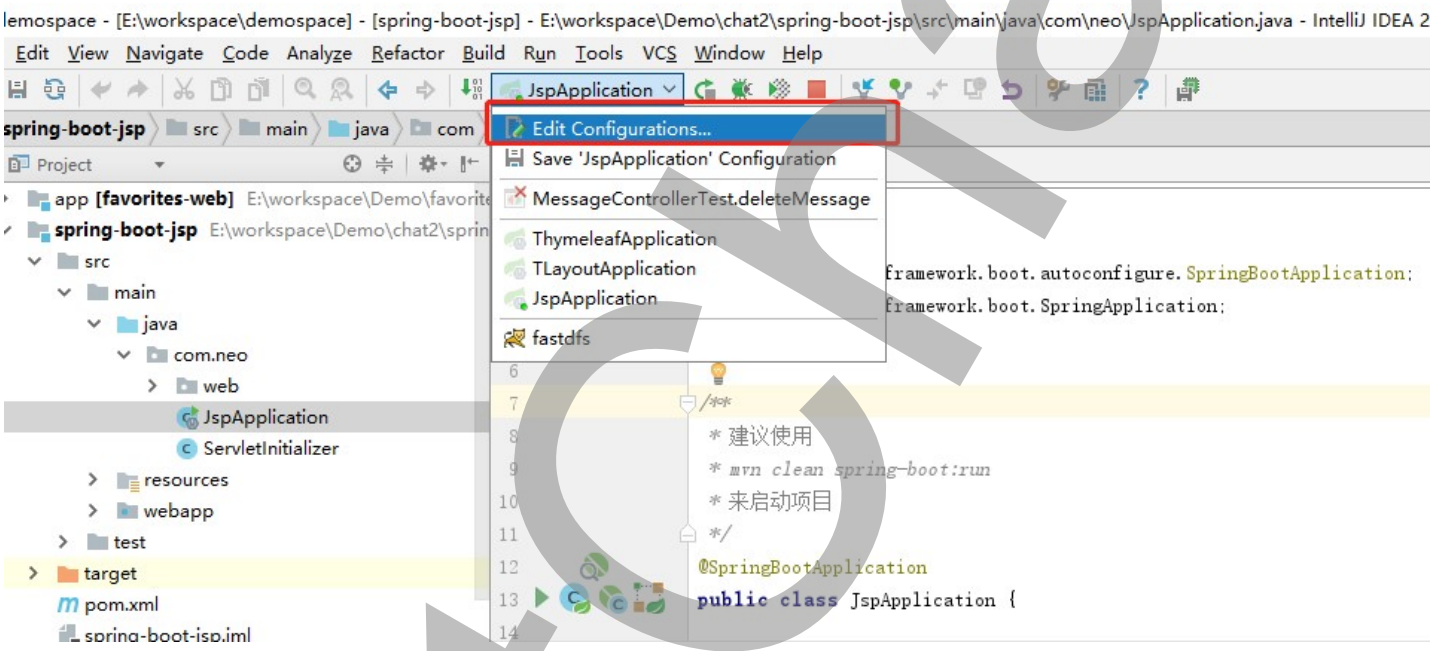
如果像其他项目一样，直接在 IDEA 中通过 main 方法来启动项目，在访问测试的时候会出现 404 not found。

Whitelabel Error Page

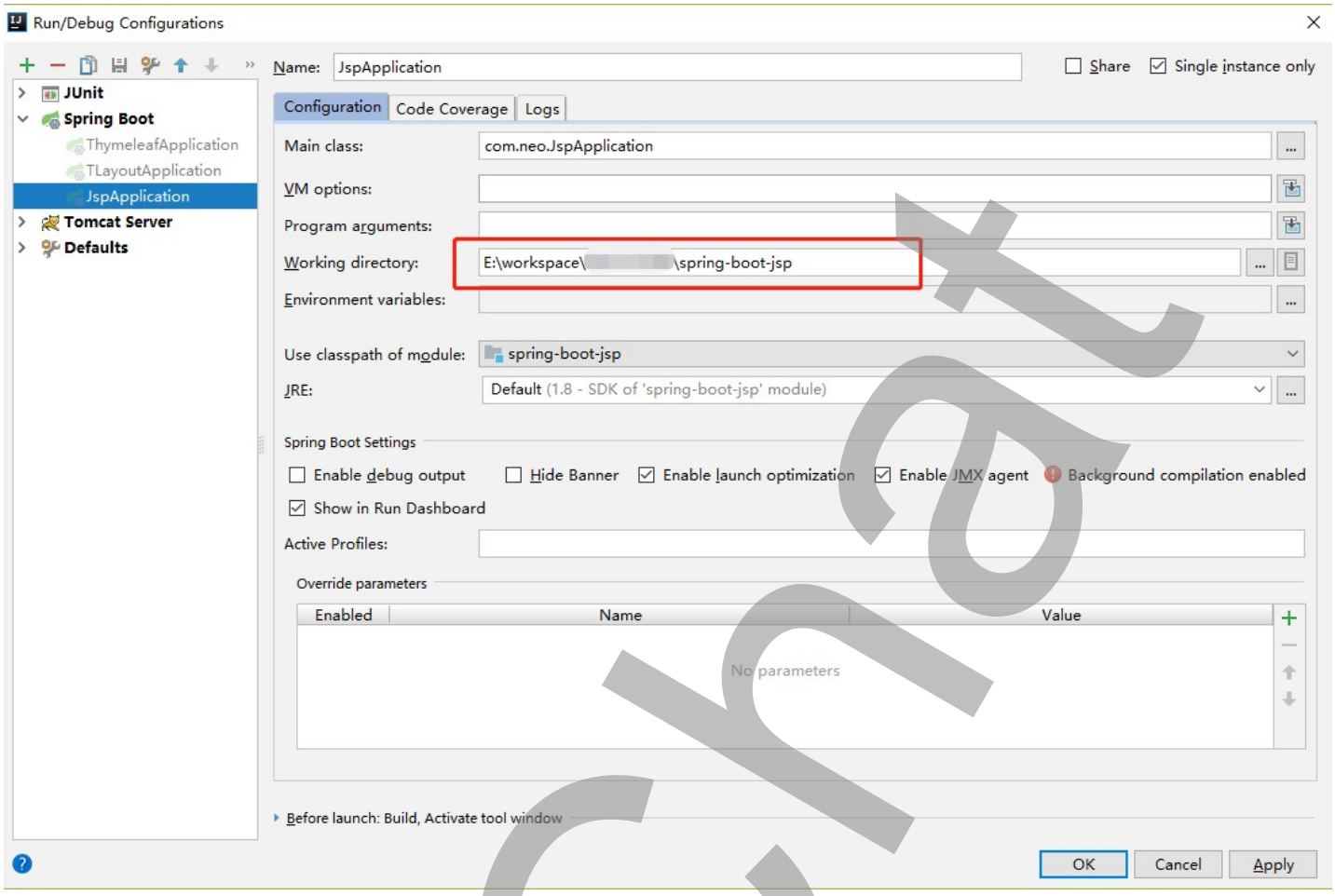
This application has no explicit mapping for /error, so you are seeing this as a fallback.

Sat Aug 11 13:32:15 CST 2018
There was an unexpected error (type=Not Found, status=404).
/WEB-INF/jsp/welcome.jsp

这是因为 Spring Boot JSP 项目需要额外进行一个设置：选择 Edit Configurations 选项，打开 Run/Debug Configurations：



设置 Working directory 的路径为项目根路径：



然后重启项目就可以正常的访问到页面内了。

在单独的 Tomcat 中运行

（1）在 pom.xml 里设置打包格式为 war。

```
<packaging>war</packaging>
```

（2）排除内嵌的 Tomcat 依赖

打包时排除掉内嵌的 Tomcat 依赖，避免 jar 包冲突。

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <!-- 排除内置容器，排除内置容器导出成 war 包可以让外部容器运行spring-boot项目-->
    <exclusions>
        <exclusion>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

（3）Servlet 的支持

Spring Boot 项目必须实现 SpringBootServletInitializer 接口的 configure() 方法才能让外部容器运行 Spring Boot 项目，启动类同目录下创建 ServletInitializer 类：

```
public class ServletInitializer extends SpringBootServletInitializer {
    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder applicat
ion) {
        return application.sources(JspApplication.class);
    }
}
```

（4）打包发布，在项目根目录执行 maven 命令：

```
mvn clean package
```

（5）将 war 包发布到 Tomcat 即可。

总结

通过本课的学习我们掌握了如何在 Spring Boot 项目中集成 JSP，Spring Boot 支持使用内嵌的 Tomcat 来运行 JSP，也支持将项目打包成 War 包部署到独立的 Tomcat 中。实际项目中推荐使用单独的 Tomcat 来部署使用 JSP 的项目，内嵌的 Tomcat 还不是很稳定，偶尔会出现访问迟缓的现象。

> 点击这里下载源码。