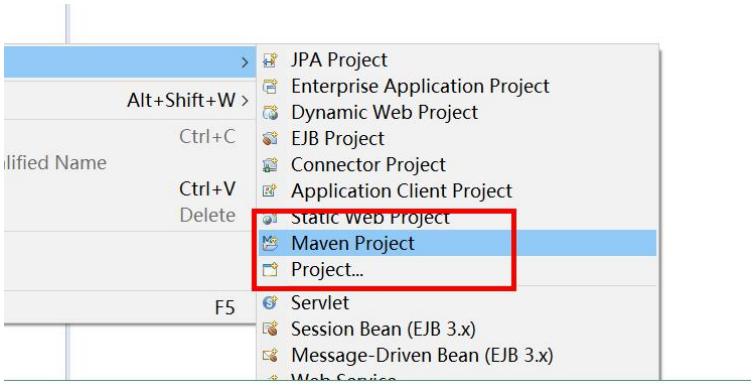
本章学习目标

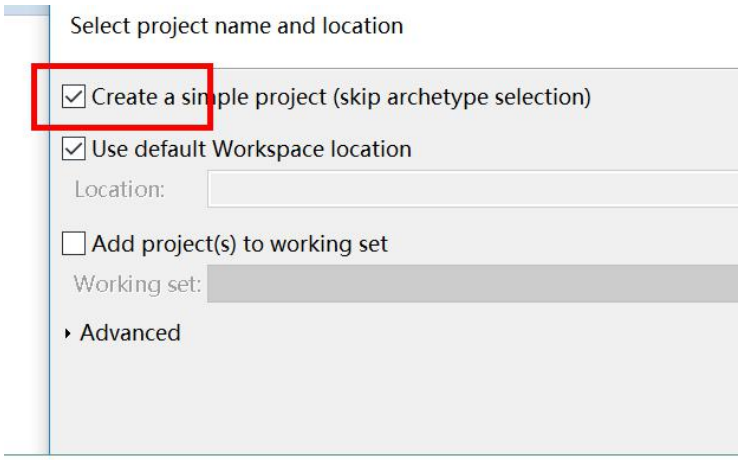
- Spring Boot 简介
- Spring Boot 入门案例
- Spring Boot 整合 Servlet，Filter，Listener
- Spring Boot 访问静态资源
- Spring Boot 实现文件上传
- Spring Boot 整合 Freemarker
- Spring Boot 整合 JSP
- Spring Boot 整合 Thymeleaf
- Spring Boot 整合 MyBatis

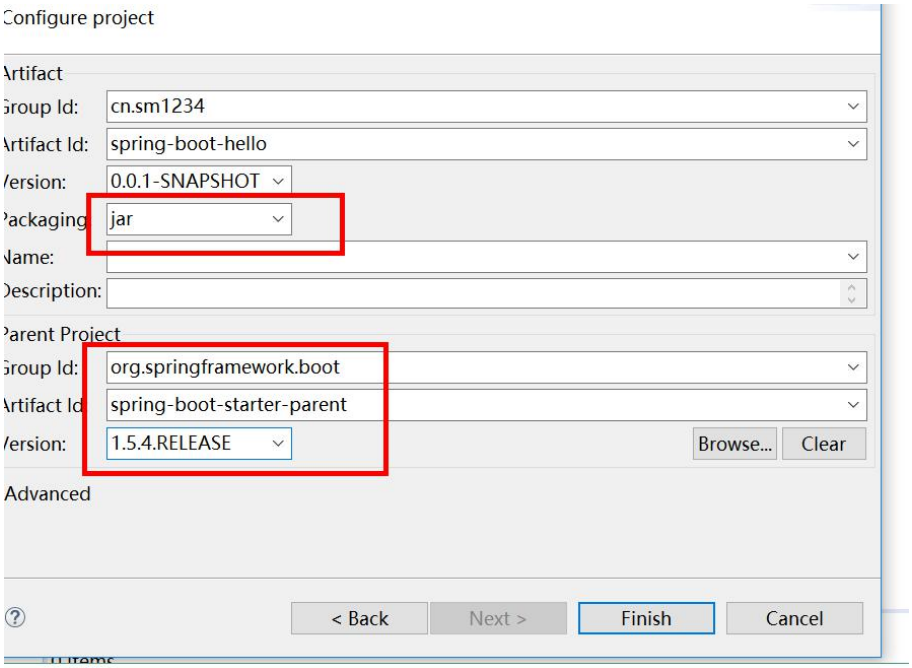
1. Spring Boot 简介

2. Spring Boot 入门案例

2.1. 建立 Maven 项目







2.2. 修改 JDK 的编译版本

```
<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/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.5.4.RELEASE</version>
  </parent>
  <groupId>cn.sm1234</groupId>
  <artifactId>spring-boot-hello</artifactId>
  <version>0.0.1-SNAPSHOT</version>



  <properties>
      <!-- 修改 JDK 版本 -->
      <java.version>1.7</java.version>
  </properties>
</project>
```

2.3. 引入 Web 启动器

```
<!-- 引入 Web 支持的坐标 ： SpringMV，Servlet，Filter.Listener 等 -->
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
    </dependencies>
```

2.4. 编写 Controller 类

```
package cn.sm1234.controller;


import java.util.HashMap;
```

```
import java.util.Map;


import org.springframework.stereotype.Controller;

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

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


@Controller

//@RequestMapping("/hello")

public class HelloController {


    private Map<String,Object> result = new HashMap<String,Object>();


    @RequestMapping("/hello")

    @ResponseBody  // 转换 json 注解

    public Map<String,Object> hello(){

        result.put("name", "eric");

        result.put("gender", "男");

        return result;

    }

}
```

2.5. 编写 SpringBoot 启动类

```
package cn.sm1234;


import org.springframework.boot.SpringApplication;

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

```
/**
 * SpringBoot 的启动器
 * @author lenovo
 *
 */
@SpringBootApplication
public class Application {

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

3. Spring Boot 整合 Servlet， Filter， Listener

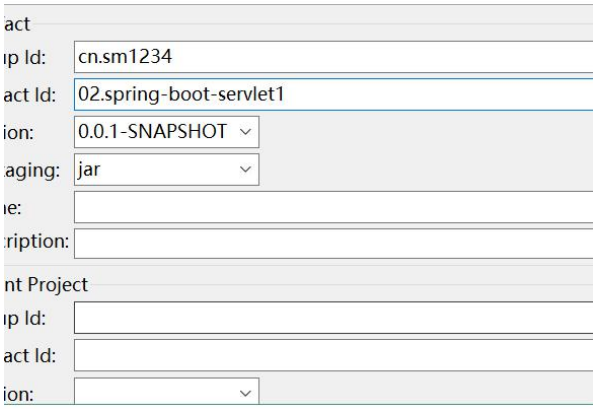
Spring Boot 使用 Servlet 的 API 有两种方法：

1）使用@ServletComponentScan 注解

2）使用@Bean 注解

3.1. 使用@ServletComponentScan 注解

3.1.1. 建立 maven 项目



3.1.2. 编写 pom.xml

```
<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/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>



    <parent>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-parent</artifactId>

        <version>1.5.4.RELEASE</version>

    </parent>



    <groupId>cn.sm1234</groupId>

    <artifactId>02.spring-boot-servlet1</artifactId>

    <version>0.0.1-SNAPSHOT</version>
```

```
    <dependencies>
        <!-- 支持 SpringMVC，Servlet，Filter，Listener -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
    </dependencies>

    <properties>
        <java.version>1.7</java.version>
    </properties>
</project>
```

3.1.3. 使用注解编写 Servlet 程序

```
package cn.sm1234.servlet;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(name="helloServlet",urlPatterns="/helloServlet")  // @WebServlet:声明该类
为 Servlet 程序
/**
 * 等同于 web.xml 配置
```

```
 *     <servlet>
 *         <servlet-name>helloServlet</servlet-name>
 *         <servlet-class>cn.sm1234.servlet.HelloServlet</servlet-class>
 *     </servlet>
 *     <servlet-mapping>
 *          <servlet-name>helloServlet</servlet-name>
 *         <url-pattern>/helloServlet</url-pattern>
 *     </servlet-mapping>
 * @author lenovo
 *
 */
public class HelloServlet extends HttpServlet{


    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        System.out.println("执行了 HelloServlet 的 doGet 方法....");
    }


}
```

3.1.4. 使用注解编写 Filter

```
package cn.sm1234.servlet;


import java.io.IOException;


import javax.servlet.Filter;
```

```
import javax.servlet.FilterChain;

import javax.servlet.FilterConfig;

import javax.servlet.ServletException;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;

import javax.servlet.annotation.WebFilter;


@WebFilter(filterName="helloFilter",urlPatterns="/helloServlet")

public class HelloFilter implements Filter{


    @Override

    public void destroy() {


    }


    @Override

    public void doFilter(ServletRequest arg0, ServletResponse arg1, FilterChain arg2)

            throws IOException, ServletException {

        System.out.println("执行了前面代码");


        //放行执行目标资源：HelloServlet

        arg2.doFilter(arg0, arg1);


        System.out.println("执行了后面代码");

    }


    @Override

    public void init(FilterConfig arg0) throws ServletException {
```

```
    }


}
```

3.1.5. 使用注解编写 Listener

```
package cn.sm1234.servlet;


import javax.servlet.ServletContextEvent;

import javax.servlet.ServletContextListener;

import javax.servlet.annotation.WebListener;


@WebListener

public class HelloListener implements ServletContextListener{


    @Override

    public void contextDestroyed(ServletContextEvent arg0) {

        System.out.println("ServletContext 对象消耗了");

    }


    @Override

    public void contextInitialized(ServletContextEvent arg0) {

        System.out.println("ServletContext 对象创建了");

    }


}
```

3.1.6. 编写启动类，加上@ServletComponentScan 注解

```
package cn.sm1234;


import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.web.servlet.ServletComponentScan;


@SpringBootApplication

@ServletComponentScan  //@ServletComponentScan:作用让 SpringBoot 扫描@WebServlet 等注解

public class Application {


    public static void main(String[] args) {

        SpringApplication.run(Application.class, args);

    }

}
```

3.2. 使用@Bean 注解

```
package cn.sm1234;


import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.web.servlet.FilterRegistrationBean;

import org.springframework.boot.web.servlet.ServletListenerRegistrationBean;

import org.springframework.boot.web.servlet.ServletRegistrationBean;

import org.springframework.context.annotation.Bean;
```

```
import cn.sm1234.servlet.HelloFilter;

import cn.sm1234.servlet.HelloListener;

import cn.sm1234.servlet.HelloServlet;


@SpringBootApplication

public class Application {


    public static void main(String[] args) {

        SpringApplication.run(Application.class, args);

    }


    //注册 Servlet 程序

    @Bean

    public ServletRegistrationBean getServletRegistrationBean(){

        ServletRegistrationBean bean = new ServletRegistrationBean(new

HelloServlet());

        //设置访问路径

        bean.addUrlMappings("/helloServlet");

        return bean;

    }


    //注册 Filter

    @Bean

    public FilterRegistrationBean getFilterRegistrationBean(){

        FilterRegistrationBean bean = new FilterRegistrationBean(new HelloFilter());

        //过滤器拦截路径

        bean.addUrlPatterns("/helloServlet");

        return bean;

    }
```

```
    //注册 Listener
    @Bean
    public ServletListenerRegistrationBean<HelloListener>
getServletListenerRegistrationBean(){
        ServletListenerRegistrationBean<HelloListener> bean = new
ServletListenerRegistrationBean<HelloListener>(new HelloListener());
        return bean;
    }
}
```

4. Spring Boot 访问静态资源

直接在 src/main/resources/下创建 static 目录把资源放在该目录即可！

5. Spring Boot 实现文件上传

5.1. 编写上传页面

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>文件上传页面</title>
</head>
<body>
文件上传页面
```

```
<hr/>
<form action="upload" method="post" enctype="multipart/form-data">
    请选择文件：<input type="file" name="attach"/><br/>
    <input type="submit" value="开始上传"/>
</form>
</body>
</html>
```

5.2. 编写 Controller 接收文件

```
package cn.sm1234.controller;

import java.io.File;
import java.util.HashMap;
import java.util.Map;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.multipart.MultipartFile;

@RestController
public class UploadController {

    Map<String,Object> result = new HashMap<String,Object>();
    /*
     * 接收文件
     */
    @RequestMapping("/uploadAttach")
```

```
    public Map<String,Object> upload(@RequestParam("attach")MultipartFile file)
throws Exception{
        //处理文件
        System.out.println("文件原名称："+file.getOriginalFilename());
        System.out.println("文件类型："+file.getContentType());

        //保存到硬盘
        file.transferTo(new File("e:/"+file.getOriginalFilename()));

        result.put("success", true);
        return result;
    }
}
```

这时发现 SpringBoot 上传文件限制不超过 10M，但是可以修改限制

在 src/main/resources 目录下建立 application.properties 文件：

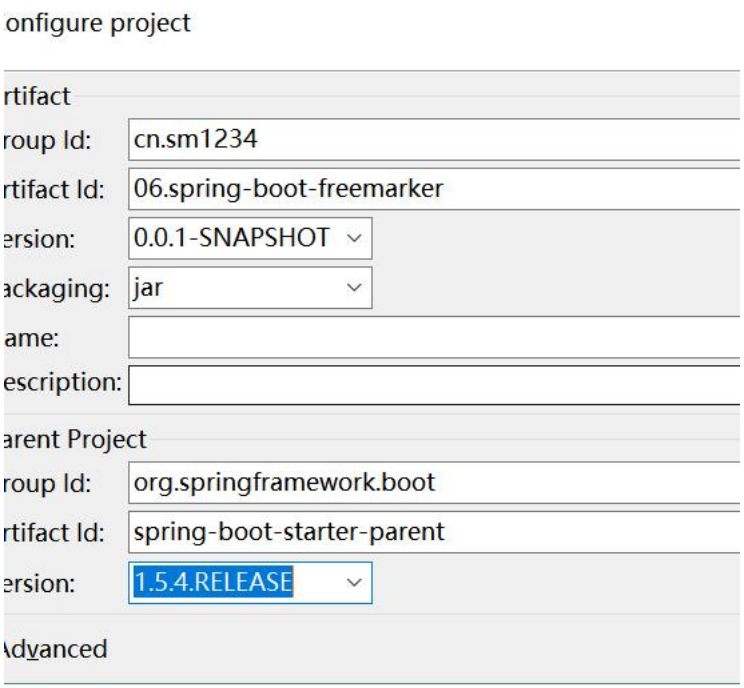
```
spring.http.multipart.maxFileSize=100MB
spring.http.multipart.maxRequestSize=200MB
```

spring.http.multipart.maxFileSize： 修改单个文件的大小限制

spring.http.multipart.maxRequestSize： 修改一个请求（包括多个文件）的大小限制

6. Spring Boot 整合 Freemarker

6.1. 建立 maven 项目



6.2. 导入坐标，配置 pom 文件

```
<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/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.5.4.RELEASE</version>
    </parent>
    <groupId>cn.sm1234</groupId>
```

```
    <artifactId>06.spring-boot-freemarker</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <dependencies>
        <!-- web 支持，SpringMVC， Servlet 支持等 -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <!-- freemarker -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-freemarker</artifactId>
        </dependency>
    </dependencies>

    <properties>
        <java.version>1.7</java.version>
    </properties>
</project>
```

6.3. 编写 Controller 查询数据

```
package cn.sm1234.controller;

import java.util.ArrayList;
import java.util.List;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
```

```
import org.springframework.web.bind.annotation.RequestMapping;


import cn.sm1234.domain.User;


@Controller
public class UserController {


    /**
     * 用户列表展示
     */
    @RequestMapping("/list")
    public String list(Model model){
        //模拟用户数据
        List<User> list = new ArrayList<User>();
        list.add(new User(1,"小张",18));
        list.add(new User(2,"小徐",20));
        list.add(new User(3,"小陈",22));


        //把数据存入 model
        model.addAttribute("list", list);


        //跳转到 freemarker 页面: list.ftl
        return "list";
    }
}
```

6.4. 建立 list.ftl 模板页面

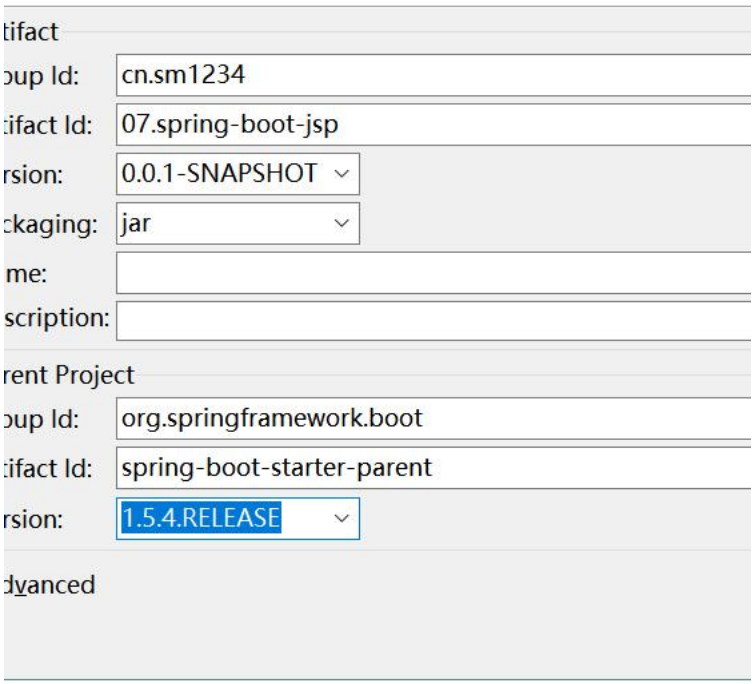
注意：首先需要在 src/main/resources 目录下新建 templates 目录。

建立 list.ftl 文件：

```
<html>
    <title>用户列表展示</title>
    <meta charset="utf-8"/>
    <body>
        <h3>用户列表展示</h3>
        <table>
            <tr>
                <th>编号</th>
                <th>姓名</th>
                <th>年龄</th>
            </tr>
            <#list list as user>
            <tr>
                <td>${user.id}</td>
                <td>${user.name}</td>
                <td>${user.age}</td>
            </tr>
            </#list>
        </table>
    </body>
</html>
```

7. Spring Boot 使用 JSP 页面

7.1. 建立 maven 项目



7.2. 导入坐标，配置 pom 文件

```
<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/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.5.4.RELEASE</version>
    </parent>
    <groupId>cn.sm1234</groupId>
```

```
    <artifactId>07.spring-boot-jsp</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <dependencies>
        <!-- web 支持，SpringMVC， Servlet 支持等 -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <!-- jsp 依赖 -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
        </dependency>
        <dependency>
            <groupId>org.apache.tomcat.embed</groupId>
            <artifactId>tomcat-embed-jasper</artifactId>
            <scope>provided</scope>
        </dependency>
    </dependencies>

    <properties>
        <java.version>1.7</java.version>
    </properties>
</project>
```

7.3. 在 application.properties 配置视图

```
spring.mvc.view.prefix=/WEB-INF/jsp/

spring.mvc.view.suffix=.jsp
```

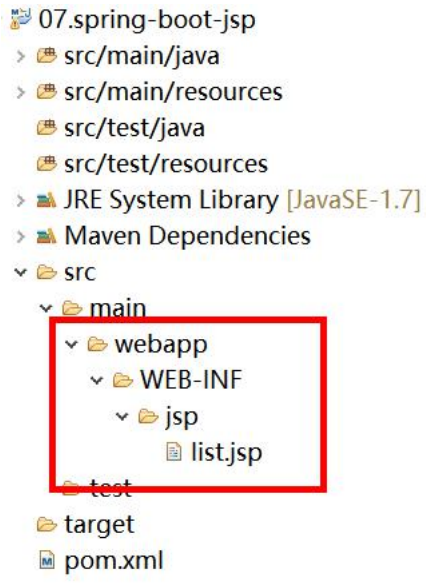
7.4. 编写 Controller 查询数据

```
package cn.sm1234.controller;


import java.util.ArrayList;
import java.util.List;


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


import cn.sm1234.domain.User;


@Controller
public class UserController {


    /**
     * 用户列表展示
     */
    @RequestMapping("/list")
    public String list(Model model){
        //模拟用户数据
        List<User> list = new ArrayList<User>();
        list.add(new User(1,"小张",18));
```

```
        list.add(new User(2,"小徐",20));
        list.add(new User(3,"小陈",22));

        //把数据存入 model
        model.addAttribute("list", list);

        //跳转到 jsp 页面: list.jsp
        return "list";
    }
}
```

7.5. 建立 jsp 页面展示列表

在 src/main 目录下创建 webapp/WEB-INF/jsp 目录，在该目录下建立 list.jsp




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

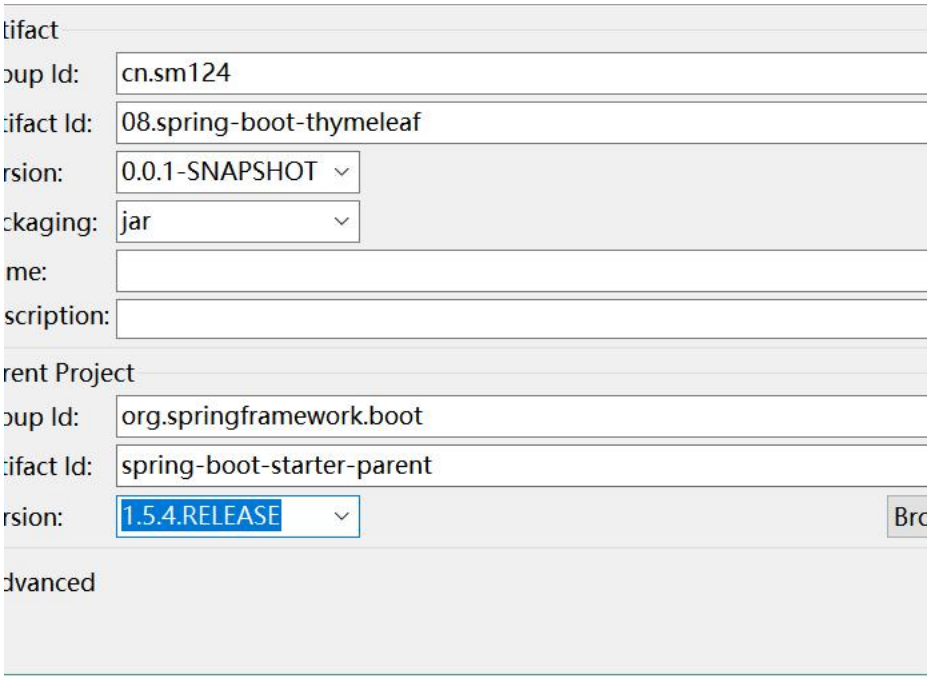
```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>用户列表展示</title>
</head>
<body>

<h3>用户列表展示</h3>
        <table border="1">
            <tr>
                <th>编号</th>
                <th>姓名</th>
                <th>年龄</th>
            </tr>
            <c:forEach items="${list}" var="user">
            <tr>
                <td>${user.id}</td>
                <td>${user.name}</td>
                <td>${user.age}</td>
            </tr>
            </c:forEach>
        </table>
</body>
</html>
```

8. Spring Boot 整合 Thymeleaf（*）

Spring Boot 推荐使用 Thymeleaf 作为页面模块

8.1. Thymeleaf 入门开发

8.1.1. 建立 maven 项目



8.1.2. 导入坐标，配置 pom 文件

```
<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/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
```

```
        <version>1.5.4.RELEASE</version>
    </parent>
    <groupId>cn.sm124</groupId>
    <artifactId>08.spring-boot-thymeleaf</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <dependencies>
        <!-- web 支持，SpringMVC， Servlet 支持等 -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

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

    </dependencies>


    <properties>
        <java.version>1.7</java.version>
    </properties>
</project>
```

8.1.3. 建立 Controller 传递数据

```
package cn.sm1234.controller;


import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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


@Controller

public class UserController {


    @RequestMapping("/demo1")

    public String demo1(Model model){

        model.addAttribute("message", "你好，Thymeleaf");

        //跳转到 templates/demo1.html

        return "demo1";

    }

}
```

8.1.4. 建立 demo1.html 页面

在 src/main/resources 目录建立 templates 目录（和 Freemarker 做法类似），在该目录下建立 demo1.html（Thymeleaf 文件后缀名就是 html）

```
<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>一个 Thymeleaf 入门案例</title>
```

```
</head>

<body>


<span th:text="${message}"></span>


</body>

</html>
```

8.1.5. 入门程序出现的问题

启动程序后访问会出现以下错误：

```
.505 ERROR 17264 --- [nio-8080-exec-1] org.thymeleaf.TemplateEngine
.507 ERROR 17264 --- [nio-8080-exec-1] o.a.c.c.C.[.[.[/].[dispatcherServlet]

eException: 元素类型 "meta" 必须由匹配的结束标记 "</meta>" 终止。
org.apache.xerces.internal.util.ErrorHandlerWrapper.createSAXParseException(Un
org.apache.xerces.internal.util.ErrorHandlerWrapper.fatalError(Unknown Source)
org.apache.xerces.internal.impl.XMLErrorReporter.reportError(Unknown Source) ~
org.apache.xerces.internal.impl.XMLErrorReporter.reportError(Unknown Source) ~
org.apache.xerces.internal.impl.XMLScanner.reportFatalError(Unknown Source) ~[
org.apache.xerces.internal.impl.XMLDocumentFragmentScannerImpl.scanEndElement(
org.apache.xerces.internal.impl.XMLDocumentFragmentScannerImpl$FragmentContent
org.apache.xerces.internal.impl.XMLDocumentScannerImpl.next(Unknown Source) ~[
org.apache.xerces.internal.impl.XMLDocumentFragmentScannerImpl.scanDocument(Un
org.apache.xerces.internal.parsers.XML11Configuration.parse(Unknown Source) ~[
```

原因：Thymeleaf3.0 以下的版本就会严格要求 html 页面上所有标签都要结束。

解决办法：把 thymeleaf 的版本升级到 3.0 以上的版本！

修改 pom.xml:

```
<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/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>
```

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.5.4.RELEASE</version>
</parent>
<groupId>cn.sm124</groupId>
<artifactId>08.spring-boot-thymeleaf</artifactId>
<version>0.0.1-SNAPSHOT</version>

<dependencies>
    <!-- web 支持，SpringMVC， Servlet 支持等 -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

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

</dependencies>

<properties>
    <java.version>1.7</java.version>
    <!-- 修改 thymeleaf 的版本 -->
    <thymeleaf.version>3.0.2.RELEASE</thymeleaf.version>
```

```
        <thymeleaf-layout-dialect.version>2.0.4</thymeleaf-layout-dialect.version>
    </properties>
</project>
```

8.2. Thymeleaf 的语法

8.2.1. 变量输出

```
    //变量输出
    @RequestMapping("/demo2")
    public String demo2(Model model){
        model.addAttribute("name", "张三");
        return "demo2";
    }
```

```
<h3>变量输出</h3>
<h4 th:text="${name}"></h4>
<h4 th:text="李四"></h4>
```

8.2.2. 条件判断

th:if 和 th:switch

```
    //条件判断
    @RequestMapping("/demo3")
    public String demo3(Model model){
        model.addAttribute("gender", "女");

        model.addAttribute("grade",3);
```

```
        return "demo2";
    }
```

```
<h3>条件判断</h3>
<div th:if="${gender} == '男'">
    这是一位男性朋友
</div>
<div th:if="${gender} == '女'">
    这是一位女性朋友
</div>
<br/>
<div th:switch="${grade}">
    <span th:case="1">这是 1 的情况</span>
    <span th:case="2">这是 2 的情况</span>
    <span th:case="3">这是 3 的情况</span>
</div>
```

8.2.3. 迭代遍历

```
    //迭代遍历
    @RequestMapping("/demo4")
    public String demo4(Model model){
        List<User> list = new ArrayList<User>();
        list.add(new User(1,"eric",20));
        list.add(new User(2,"jack",22));
        list.add(new User(3,"rose",24));

        model.addAttribute("list", list);

        return "demo2";
```

```
    }
```

```
<table border="1">
    <tr>
        <td>编号</td>
        <td>姓名</td>
        <td>年龄</td>
    </tr>
    <tr th:each="user : ${list}">
        <td th:text="${user.id}"></td>
        <td th:text="${user.name}"></td>
        <td th:text="${user.age}"></td>
    </tr>
</table>
```

8.2.4. 域对象的使用

```
    //域对象的获取
    @RequestMapping("/demo5")
    public String demo5(HttpServletRequest request,Model model){

        //request
        request.setAttribute("request", "request's data");

        //session
        request.getSession().setAttribute("session", "session's data");

        //application
        request.getSession().getServletContext().setAttribute("application",
```

```
"application's data");


        return "demo2";
    }
```

```
<h3>域对象数据的获取</h3>
request: <span th:text="${#httpServletRequest.getAttribute('request')}"></span><br/>
session: <span th:text="${session.session}"></span><br/>
application: <span th:text="${application.application}"></span><br/>
```

8.2.5. 链接语法

```
<h3>超链接的语法</h3>
<a th:href="@{~/demo1}">访问 demo1</a><br/>


<a th:href="@{~/demo1(id=1,name=eric)}">访问 demo1,传递参数</a>
```

9. Spring Boot 整合 MyBatis （Thymeleaf）

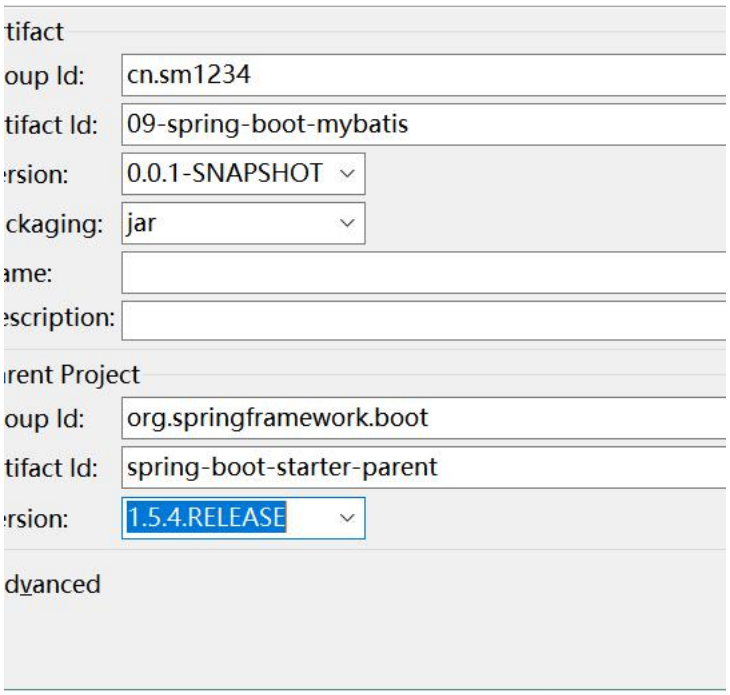
需求：用户的 CRUD

9.1. 建立客户表

```
CREATE  TABLE  `t_customer`  (
      `id`  int(11)  NOT  NULL  AUTO_INCREMENT,
      `name`  varchar(20)  DEFAULT  NULL,
      `gender`  char(1)  DEFAULT  NULL,
      `telephone`  varchar(20)  DEFAULT  NULL,
      `address`  varchar(50)  DEFAULT  NULL,
      PRIMARY  KEY  (`id`)
```

```
) ENGINE=InnoDB AUTO_INCREMENT=19 DEFAULT CHARSET=utf8
```

9.2. 建立 maven 项目



9.3. 导入坐标，配置 pom 文件

```
<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/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.5.4.RELEASE</version>
    </parent>
```

```
    <groupId>cn.sm1234</groupId>
    <artifactId>09-spring-boot-mybatis</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <dependencies>
        <!-- web 支持，SpringMVC， Servlet 支持等 -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <!-- thymeleaf -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>

        <!-- mybatis 相关的坐标 -->
        <!-- mysql -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
        </dependency>
        <!-- druid 连接池 -->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid</artifactId>
            <version>1.0.9</version>
        </dependency>
        <!-- SpringBoot 的 Mybatis 启动器 -->
```

```
        <dependency>
            <groupId>org.mybatis.spring.boot</groupId>
            <artifactId>mybatis-spring-boot-starter</artifactId>
            <version>1.1.1</version>
        </dependency>
    </dependencies>

    <properties>
        <java.version>1.7</java.version>
        <thymeleaf.version>3.0.2.RELEASE</thymeleaf.version>
        <thymeleaf-layout-dialect.version>2.0.4</thymeleaf-layout-dialect.version>
    </properties>
</project>
```

9.4. 在 application.properties 配置连接参数（*）

```
spring.datasource.driverClassName=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/ssm
spring.datasource.username=root
spring.datasource.password=root

spring.datasource.type=com.alibaba.druid.pool.DruidDataSource

mybatis.type-aliases-package=cn.sm1234.domain
```

9.5. 编写 Customer 实体

```
package cn.sm1234.domain;
```

```
public class Customer {
    private Integer id;
    private String name;
    private String gender;
    private String telephone;
    private String address;
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getGender() {
        return gender;
    }
    public void setGender(String gender) {
        this.gender = gender;
    }
    public String getTelephone() {
        return telephone;
    }
    public void setTelephone(String telephone) {
```

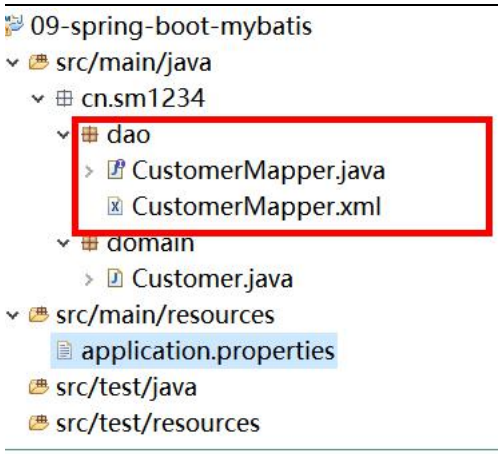
```
        this.telephone = telephone;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }

}
```

9.6. 编写 Mapper 接口

```
package cn.sm1234.dao;

import cn.sm1234.domain.Customer;

public interface CustomerMapper {
    public void save(Customer customer);
}
```

9.7. 编写 sql 映射文件

在 Mapper 接口同目录下建立和 Mapper 接口同名的 xml 文件：




```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE mapper

PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<!-- 该文件存放 CRUD 的 sql 语句 -->

<mapper namespace="cn.sm1234.dao.CustomerMapper">



    <insert id="save" parameterType="customer">

        INSERT INTO ssm.t_customer

            (

            NAME,

            gender,

            telephone,

            address

            )

            VALUES

            (

            #{name},

            #{gender},

            #{telephone},

            #{address}

            )
```

```
    </insert>


</mapper>
```

9.8. 编写 Service 接口和实现

接口：

```
package cn.sm1234.service;


import cn.sm1234.domain.Customer;


public interface CustomerService {

    public void save(Customer customer);

}
```

实现：

```
package cn.sm1234.service.impl;


import javax.annotation.Resource;


import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;


import cn.sm1234.dao.CustomerMapper;

import cn.sm1234.domain.Customer;

import cn.sm1234.service.CustomerService;


@Service

@Transactional

public class CustomerServiceImpl implements CustomerService {
```

```
    //注入 mapper 接口的对象
    @Resource
    private CustomerMapper customerMapper;


    @Override
    public void save(Customer customer) {
        customerMapper.save(customer);
    }

}
```

9.9. 编写 Controller

```
package cn.sm1234.controller;


import javax.annotation.Resource;


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


import cn.sm1234.domain.Customer;
import cn.sm1234.service.CustomerService;


@Controller
@RequestMapping("/customer")
public class CustomerController {
    @Resource
    private CustomerService customerService;
```

```
    /**
     * 保存方法
     */
    @RequestMapping("/save")
    public String save(Customer customer){
        customerService.save(customer);
        return "succ";
    }
}
```

9.10. 编写页面

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>录入客户信息</title>
</head>
<body>
<form th:action="@{~/customer/save}" method="post">
    客户姓名：<input type="text" name="name"/><br/>
    客户性别：<input type="text" name="gender"/><br/>
    客户手机：<input type="text" name="telephone"/><br/>
    客户住址：<input type="text" name="address"/><br/>
    <input type="submit" value="保存"/>
</form>

</body>
```

```
</html>
```

在 CustomerController 补充 input 方法，用于跳转到 input.html 页面：

```
    //跳转到 input.html 页面
    @RequestMapping("/input")
    public String input(){
        return "input";
    }
```

9.11. 编写 SpringBoot 启动类

```
package cn.sm1234;


import org.mybatis.spring.annotation.MapperScan;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.thymeleaf.spring4.processor.SpringActionTagProcessor;


@SpringBootApplication
@MapperScan("cn.sm1234.dao") // @MapperScan：作用是用于扫描 MyBatis 的 mapper 接口的包
public class Application {


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