Spring Boot 从入门到实战（微课视频版）

实 验 指 导 书

2020.9

目录

概 述

本课程实验教学的目的在于训练学生的实际能手能力，以期更好地掌握 Spring Boot 相关原理、技术及方法，并为后期毕业设计及以后从事 Java EE 相关开发打下坚实的基础。

实验内容

与教材《Spring Boot 从入门到实战（微课视频版）》对应的实验共有 11 个（共 28 学时），分别如下：

1、Spring 开发环境　2 学时

2、基于注解的依赖注入 2 学时

3、Spring 的事务管理（基于注解的事务管理） 2 学时

4、Controller 接收请求参数（注册与登录系统） 2 学时

5、Spring Boot 的环境构建（第一个 Spring Boot 应用） 2 学时

6、自定义条件与 Starters 2 学时

7、基于 Thymeleaf 模板引擎的 Spring Boot Web 开发 2 学时

8、基于 JSP 引擎的 Spring Boot Web 开发 2 学时

9、基于 Thymeleaf + Spring Boot + Spring JPA 的用户系统 4 学时

10、基于 JSP + Spring Boot + MyBatis 的用户系统 4 学时

11、基于 Vue.js + Spring Boot +Spring JPA 的用户系统 4 学时

实验环境

硬件为个人微机，软件为操作系统 Windows 7 或 10，开发环境为 Eclipse-jee（或 STS 或 IDEA） + JDK8（或以上） + Tomcat9.0 + MySQL 5.5。

实验要求

本课程实验教学的要求包括：

1、根据教材中规定的内容在要求的时间内完成实验内容；

2、成功部署实验中要求实现的内容，并能演示；

3、提交实验中规定的开发内容的核心代码；

4、提交实验报告。

实验一 Spring 开发环境

（实验课时：2　实验性质：设计）

实验名称：

Spring 开发环境

实验目的和要求：

1、掌握基于 Eclipse 平台（或 STS 或 IDEA）的 Spring 集成开发环境的构建。

2、通过在 Spring 开发环境中创建和运行一些实例项目，熟悉 Spring 的基本开发、部署和运行过程，为后续实验打下基础。

3、认真书写实验报告，如实填写各项实验内容。

实验内容和步骤:

1、安装并配置 JDK

按照提示安装完成 JDK 后，需要配置“环境变量”的“系统变量”Java_Home 和 Path。

2、安装 Tomcat

将下载的 apache-tomcat-9.0.2-windows-x64.zip 解压到某个目录下，比如解压到 E:\Java soft，解压缩即完成安装。

3、安装 Eclipse

Eclipse 下载完成后，解压缩到自己设置的路径下，即完成安装。

4、集成 Tomcat

启动 Eclipse，选择“Window”->“Preferences”菜单项，在弹出的对话框中选择“Server”->“Runtime Environments”命令。在弹出的窗口中，单击“Add”按钮，弹出如图 1 所示的“New Server Runtime Environment”界面，在此可以配置各种版本的 Web 服务器。

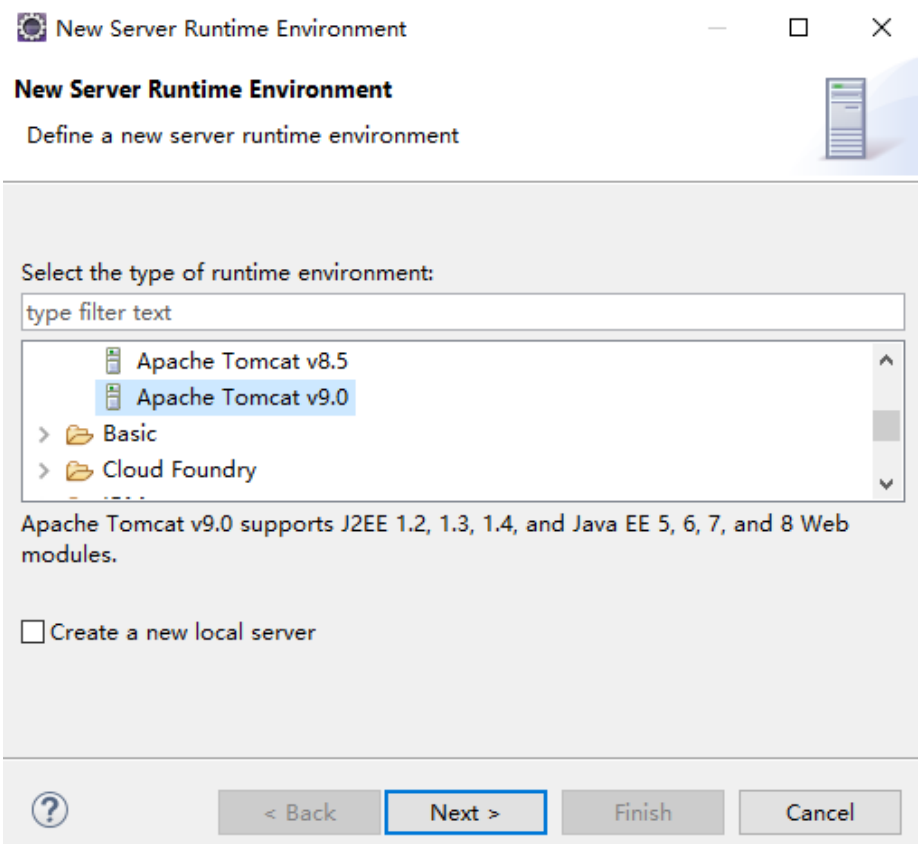

图 1 Tomcat 配置界面

在图 1 中选择“Apache Tomcat v9.0”服务器版本，单击“Next”按钮，进入如图 2 所示界面。

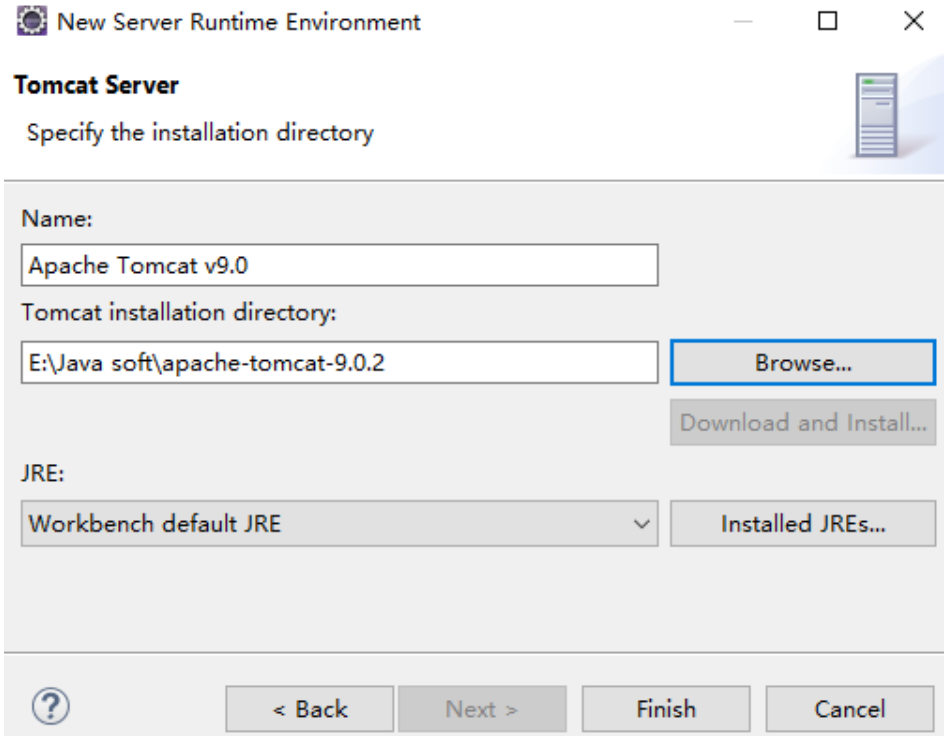

图 2　选择 Tomcat 目录

在图 2 中单击“Browse…”按钮，选择 Tomcat 的安装目录，单击“Finish”即可完成 Tomcat

配置。

至此，可以使用 Eclipse 创建 Dynamic Web project，并在 Tomcat 下运行。

5、下载 Spring

使用 Spring 框架开发应用程序时，除了引用 Spring 自身的 JAR 包外，还需要引用 commons.logging 的 JAR 包。

Spring 官方网站升级后，建议通过 Maven 和 Gradle 下载。对于不使用 Maven 和 Gradle 下载的开发者，本实验给出一个 Spring Framework jar 官方直接下载路径：https://repo.spring.io/release/org/springframework/spring/。本实验采用的 spring-framework-5.1.4.RELEASE-dist.zip。将下载到的 ZIP 文件解压缩，解压缩后的 libs 目录包含开发 Spring 应用所需要的 JAR 包（spring-XXX-5.1.4.RELEASE.jar）。

Spring 框架依赖于 Apache Commons Logging 组件，该组件的 JAR 包可以通过网址“http://commons.apache.org/proper/commons-logging/download_logging.cgi”下载，本实验下载的是“commons-logging-1.2-bin.zip”，解压缩后，即可找到 “commons-logging-1.2.jar”。

对于 Spring 框架的初学者，开发 Spring 应用时，只需要将 Spring 的四个基础包（spring-core-5.1.4.RELEASE.jar、spring-beans-5.1.4.RELEASE.jar、spring-context-5.1.4.RELEASE.jar 和 spring-expression-5.1.4.RELEASE.jar）和 commons-logging-1.2.jar 复制到 Web 应用的 WEB-INF/lib 目录下即可。

6、开发一个简单的 Spring 程序

参考教材 1.2 节，使用 Eclipse 再开发一个简单的 Spring 程序，并测试运行。

实验二 基于注解的依赖注入

（实验课时：2　实验性质：设计）

实验名称：

基于注解的依赖注入

实验目的和要求：

1、掌握基于注解的依赖注入。

2、掌握 Spring 框架定义的一系列常用注解的使用方法，包括@Component、@Repository、@Service、@Controller 和@Autowired 等注解。

3、认真书写实验报告，如实填写各项实验内容。

实验内容和步骤：

1、使用@Repository 注解声明 DAO 层

参考 1.3.3 节，创建 Web 应用 shiyan2，并导入相关 JAR 包，在 shiyan2 应用的 src 中，创建 dao 包，该包下创建 MyDao 接口和 MyDaoImpl 实现类，并将实现类 MyDaoImpl 使用@Repository 注解标注为数据访问层。

2、使用@Service 注解声明 Service 层

在 shiyan2 应用的 src 中，创建 service 包，该包下创建 MyService 接口和 MySeviceImpl 实现类，并将实现类 MySeviceImpl 使用@Service 注解标注为业务逻辑层。

在 MySeviceImpl 类中使用@Autowired 注解装配 DAO 层声明的 Bean。

3、使用@Controller 注解声明控制器层

在 shiyan2 应用的 src 中，创建 controller 包，该包下创建 TestController 类，并将 MyController 类使用@Controller 注解标注为控制器层。

在 MyController 类中使用@Autowired 注解装配 Service 层声明的 Bean。

4、配置注解

在 shiyan2 应用的 src 中，创建包 annotation，并在该包中，创建名为 ConfigAnnotation 的配置类。

5、创建测试类

在 shiyan2 应用的 src 中，创建 test 包，并在该包中创建与运行测试类 MyTest。

实验三 Spring 的事务管理（基于注解的事务管理）

（实验课时：2　实验性质：设计）

实验名称：

Spring 的数据库编程（基于注解的事务处理）

实验目的和要求：

1、了解 Spring JDBC 的配置。

2、了解 Spring JdbcTemplate 的常用方法。

3、掌握基于@Transactional 注解的声明式事务管理。

4、认真书写实验报告，如实填写各项实验内容。

实验内容和步骤:

1、创建 Web 应用并导入 JAR 包

创建一个名为 shiyan3 的 Web 应用，将图 3 所示的 JAR 包复制到应用的/ WEB-INF/lib 目录。

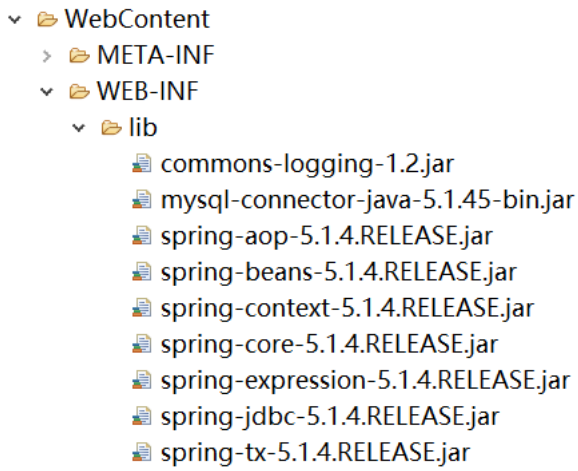

图 3 所需 JAR 包

2、创建属性文件与配置类

在 Web 应用 shiyan3 的 src 目录下，创建数据库配置的属性文件 jdbc.properties，具体内容如下：

```
jdbc.driverClassName=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/springtest?characterEncoding=utf8
jdbc.username=root
jdbc.password=root
```

在 Web 应用 shiyan3 的 src 目录下，创建 config 包，并在该包中创建配置类 SpringJDBCConfig。在该配置类中使用@PropertySource 注解读取属性文件 jdbc.properties，

并配置数据源和 JdbcTemplate，具体代码如下：

```
package config;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.datasource.DriverManagerDataSource;
@Configuration //通过该注解来表明该类是一个 Spring 的配置，相当于一个 xml 文件
@ComponentScan(basePackages = {"dao","service"}) //配置扫描包
@PropertySource(value={"classpath:jdbc.properties"},ignoreResourceNotFound=true)
//配置多个配置文件   value={"classpath:jdbc.properties","xx","xxx"}
@EnableTransactionManagement//开启声明式事务的支持
public class SpringJDBCConfig {
    @Value("${jdbc.url}")//注入属性文件 jdbc.properties 中的 jdbc.url
    private String jdbcUrl;
    @Value("${jdbc.driverClassName}")
    private String jdbcDriverClassName;
    @Value("${jdbc.username}")
    private String jdbcUsername;
    @Value("${jdbc.password}")
    private String jdbcPassword;
    /**
     * 配置数据源
     */
    @Bean
    public DriverManagerDataSource dataSource() {
     DriverManagerDataSource myDataSource    = new DriverManagerDataSource();
        // 数据库驱动
     myDataSource.setDriverClassName(jdbcDriverClassName);;
        // 相应驱动的 jdbcUrl
     myDataSource.setUrl(jdbcUrl);
        // 数据库的用户名
     myDataSource.setUsername(jdbcUsername);
        // 数据库的密码
     myDataSource.setPassword(jdbcUsername);
        return myDataSource;
    }
    /**
     * 配置 JdbcTemplate
     */
    @Bean(value="jdbcTemplate")
    public JdbcTemplate getJdbcTemplate() {
```

```
        return new JdbcTemplate(dataSource());
    }
    /**
     * 为数据源添加事务管理器
     */
    @Bean
    public DataSourceTransactionManager transactionManager() {
     DataSourceTransactionManager dt = new DataSourceTransactionManager();
     dt.setDataSource(dataSource());
     return dt;
    }
}
```

3、创建数据表与实体类

在 Web 应用 shiyan3 的 src 目录下，创建包 com.entity，在该包中创建实体类 MyUser。该类的属性与数据表 user 的字段一致。

```
package com.entity;
public class MyUser {
    private Integer uid;
    private String uname;
    private String usex;
    //此处省略 setter 和 getter 方法
    public String toString() {
        return "myUser [uid=" + uid +", uname=" + uname + ", usex=" + usex + "]";
    }
}
```

在名为 springtest 的数据库中创建数据表 user，其结构如图 4 所示。

	名	类型	长度	小数点	允许空值 (	
▶	uid	int	10	0	☐	🔑 1
	uname	varchar	20	0	☑	
	usex	varchar	10	0	☑	

图 4　user 表的结构

4、创建数据访问层 Dao

在 Web 应用 shiyan3 的 src 目录下，创建名为 com.dao 的包，并在该包中，创建 UserDao 接口和 UserDaoImpl 实现类。在实现类 UserDaoImpl 中使用 JDBC 模块 JdbcTemplate 访问数据库（添加与查询用户），并将该类使用@Repository 注解。

UserDao 接口的代码如下：

```
package com.dao;
public interface UserDao{
```

```
    public int save(String sql, Object param[]);
    public int delete(String sql, Object param[]);
}
```

UserDaoImpl 实现类的代码如下：

```
package com.dao;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;
@Repository
public class UserDaoImpl implements UserDao {
    @Autowired
    private JdbcTemplate jdbcTemplate;
    @Override
    public int save(String sql, Object[] param) {
        return jdbcTemplate.update(sql,param);
    }
    @Override
    public int delete(String sql, Object[] param) {
        return jdbcTemplate.update(sql,param);
    }
}
```

5、创建 Service 层

在 Web 应用 shiyan3 的 src 目录下，创建名为 com.service 的包，并在该包中创建 UserService 接口和 UserServiceImpl 实现类。在 Service 层依赖注入数据访问层，并添加 @Transactional 注解进行事务管理。具体代码如下：

```
package com.service;
public interface UserService {
    public void test();
}
```

```
package com.service;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import com.dao.UserDao;
@Service
@Transactional
//加上注解@Transactional,就可以指定这个类需要受 Spring 的事务管理
//注意@Transactional 只能针对 public 属性范围内的方法添加
public class UserServiceImpl implements UserService{
    @Autowired
    private UserDao userDao;
```

```
    @Override
    public void test() {
        String deleteSql ="delete from user";
        String saveSql = "insert into user values(?,?,?)";
        Object param[] = {100,"chenheng","男"};
        userDao.delete(deleteSql, null);
        userDao.save(saveSql, param);
        //插入两条主键重复的数据
        userDao.save(saveSql, param);
    }
}
```

6、创建 Controller 层

在 Web 应用 shiyan3 的 src 目录下，创建名为 com.controller 的包，并在该包中创建 UserController 控制器类。在控制层依赖注入 Service 层，代码如下：

```
package com.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import com.service.UserService;
@Controller
public class UserController{
    @Autowired
    private UserService userService;
    public void test() {
        userService.test();
    }
}
```

7、创建测试类

在 Web 应用 shiyan3 的 src 目录下，创建名为 com.test 的包，并在该包中创建测试类 UserTest。在测试类中通过访问 Controller，测试基于注解的声明式事务管理。

测试类 UserTest 的代码如下：

```
package com.test;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import com.controller.UserController;
import config.SpringJDBCConfig;
public class UserTest {
    public static void main(String[] args) {
        //初始化 Spring 容器 ApplicationContext
        AnnotationConfigApplicationContext appCon =
            new AnnotationConfigApplicationContext(SpringJDBCConfig.class);
        UserController uc = appCon.getBean(UserController.class);
        uc.test();
        appCon.close();
```

```
    }
}
```

8、运行测试类，测试事务管理

运行测试类，查看数据库插入两条 ID 相同的数据，事务管理是否好用。

实验四 Controller 接收请求参数（注册与登录系统）

（实验课时：2　实验性质：设计）

实验名称：

Controller 接收请求参数（注册与登录系统）

实验目的和要求：

1、掌握 Controller 接收请求参数的方式。

2、掌握 Spring MVC 的重定向和转发的实现方法。

3、掌握 RequestMapping 注解的用法。

实验内容和步骤：

1、创建 Web 应用并导入相关的 JAR 包

创建 Web 应用 shiyan4，并导入如图 5 所示的相关 JAR 包。

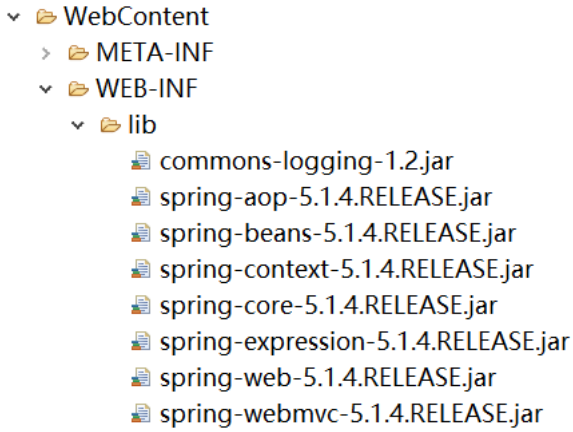

图 5 JAR 包

2、创建 Web 应用的页面

该应用共涉及 4 个 JSP 页面，分别为 index.jsp、login.jsp、register.jsp 以及 main.jsp。单击 index.jsp 中的“去注册”超链接打开 register.jsp，单击 index.jsp 中的“去登录”超链接打开 login.jsp。注册成功(用户名为 chenheng，密码为 123456)跳转到 login.jsp，登录成功(用户名为 chenheng，密码为 123456)跳转到 main.jsp。

在应用 shiyan4 的 WebContent 目录下创建 index.jsp 页面，运行效果如图 6 所示。

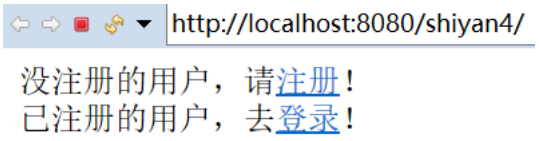

图 6 index.jsp

在 WEB-INF 目录下，创建 pages 目录，并在该目录下创建 login.jsp、register.jsp 以及

main.jsp。

login.jsp 的运行效果如图 7 所示。

图 7 login.jsp

register.jsp 的运行效果如图 8 所示。

图 8 register.jsp

main.jsp 的运行效果如图 9 所示。

图 9 main.jsp

3、实体类

创建名为 com.entity 的包，并在该包中创建实体类 UserEntity。代码略。

4、创建控制器层

创建名为 com.controller 的包，并在该包下创建两个控制器类。一个处理首页的超链接请求 IndexController.java，一个处理注册与登录请求 UserController.java。

5、创建 Spring MVC 的配置类及 Web 配置类

创建名为 config 的包，在该包中创建 Spring MVC 的 Java 配置类 SpringMVCConfig。在该配置类中使用@Configuration 注解声明该类为 Java 配置类；使用@EnableWebMvc 注解开启默认配置，如 ViewResolver；使用@ComponentScan 注解扫描注解的类；使用@Bean 注解配置视图解析器；该类需要实现 WebMvcConfigurer 接口来配置 Spring MVC。具体代码如下：

```
package config;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
```

```
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.view.InternalResourceViewResolver;
@Configuration
@EnableWebMvc
@ComponentScan("com")
public class SpringMVCConfig implements WebMvcConfigurer {
    /**
     * 配置视图解析器
     */
    @Bean
    public InternalResourceViewResolver getViewResolver() {
        InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
        viewResolver.setPrefix("/WEB-INF/pages/");
        viewResolver.setSuffix(".jsp");
        return viewResolver;
    }
    /**
     * 配置静态资源
     */
    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler("/images/**").addResourceLocations("/images/");
    }
}
```

在 config 包中，创建 Web 的 Java 类 WebConfig。该类需要实现 WebApplicationInitializer 接口替代 web.xml 文件的配置。实现该接口将会自动启动 Servlet 容器。在 WebConfig 类中需要使用 AnnotationConfigWebApplicationContext 注册 Spring MVC 的 Java 配置类 SpringMVCConfig，并和当前 ServletContext 关联。最后，在该类中需要注册 Spring MVC 的 DispatcherServlet。具体代码如下：

```
package config;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRegistration.Dynamic;
import org.springframework.web.WebApplicationInitializer;
import org.springframework.web.context.support.AnnotationConfigWebApplicationContext;
import org.springframework.web.servlet.DispatcherServlet;
public class WebConfig implements WebApplicationInitializer{
    @Override
    public void onStartup(ServletContext arg0) throws ServletException {
        AnnotationConfigWebApplicationContext ctx
```

```
            = new AnnotationConfigWebApplicationContext();
            ctx.register(SpringMVCConfig.class);//注册 Spring MVC 的 Java 配置类 SpringMVCConfig
            ctx.setServletContext(arg0);//和当前 ServletContext 关联
            /**
             * 注册 Spring MVC 的 DispatcherServlet
             */
            Dynamic servlet = arg0.addServlet("dispatcher", new DispatcherServlet(ctx));
            servlet.addMapping("/");
            servlet.setLoadOnStartup(1);
        }
}
```

7、测试运行

运行主页 index.jsp，进行注册与登录功能的测试。

实验五 Spring Boot 的环境构建（第一个 Spring Boot 应用）

（实验课时：2　实验性质：设计）

实验名称：

Spring Boot 的环境构建（第一个 Spring Boot 应用）

实验目的和要求：

1、掌握基于 Eclipse 平台（或 STS 或 IDEA）的 Spring Boot 开发环境的构建。

2、通过在 Spring Boot 开发环境中创建和运行第一个 Spring Boot 应用，熟悉 Spring Boot 的基本开发、部署和运行过程，为后续学习打下基础。

3、认真书写实验报告，如实填写各项实验内容。

实验内容和步骤:

1、使用 Eclipse 开发一个简单的 Spring Boot 应用

参考教材 3.2 节，使用 Eclipse 手工构建一个 Spring Boot 应用，并测试运行。

2、快速构建一个简单的 Spring Boot 应用

参考教材 3.3 节，使用 STS 快速构建一个简单的 Spring Boot 应用，并测试运行。

实验六 自定义条件与 Starters

（实验课时：2 实验性质：设计）

实验名称：

自定义条件与 Starters

实验目的和要求：

1、理解条件注解的原理，掌握条件的自定义与配置。

2、理解 Spring Boot 的自动配置原理，掌握 Starters 的自定义与使用。

3、认真书写实验报告，如实填写各项实验内容。

实验内容和步骤:

1、参考例 4-7，编写 Spring Boot 应用程序 shiyan6_1。要求如下：以不同的操作系统作为条件，若在 Windows 系统下运行程序，则输出列表命令为 dir；若在 Linux 操作系统下运行程序，则输出列表命令为 ls。

2、参考例 4-8 与例 4-9，自定义一个 Starter（spring_boot_addstarters）和 Spring Boot 的 Web 应用 shiyan6_2。在 shiyan6_2 中，使用 spring_boot_addstarters 计算两个整数的和，通过访问 http://localhost:8080/testAddStarters 返回两个整数的和。在 spring_boot_addstarters 中，首先创建属性配置类 AddProperties（有 Integer 类型的 number1 与 number2 两个属性），在该属性配置类中使用@ConfigurationProperties(prefix="add")注解设置属性前缀为 add；其次，创建判断依据类 AddService（有 Integer 类型的 number1 与 number2 两个属性），在 AddService 类中提供 add 方法（计算 number1 与 number2 的和）；再次，创建自动配置类 AddAutoConfiguration，当类路径中存在 AddService 类时，自动配置该类的 Bean，并可以将相应 Bean 的属性在 application.properties 中配置；最后，注册自动配置类 AddAutoConfiguration。

实验七 基于 Thymeleaf 模板引擎的 Spring Boot Web 开发

（实验课时：2　实验性质：设计）

实验名称：

基于 Thymeleaf 模板引擎的 Spring Boot Web 开发

实验目的和要求：

1、掌握 Thymeleaf 视图模板引擎技术。

2、认真书写实验报告，如实填写各项实验内容。

实验内容和步骤：

参考【例 5-5】创建基于 Thymeleaf 模板引擎的 Spring Boot Web 应用 shiyan7。在应用 shiyan7 中创建两个视图页面 addGoods.html 和 goodsList.html。addGoods.html 页面的显示效果如图 10 所示，goodsList.html 页面的显示效果如图 11 所示。

图 10　addGoods.html 页面

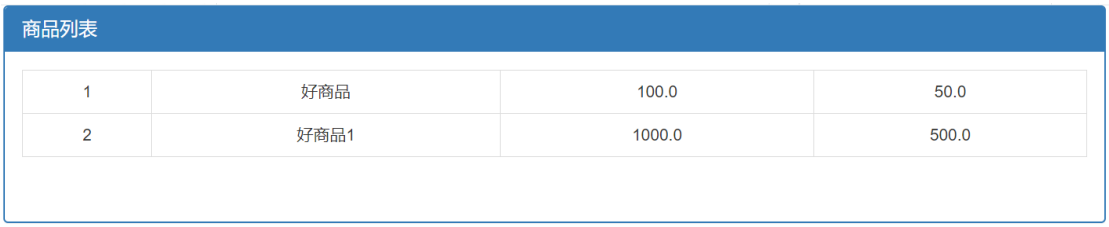

图 11　goodsList.html 页面

具体要求：

1、控制器类 GoodsController 中共有两个方法：toAdd 和 addGoods。

2、使用 Goods 模型类封装请求参数。

3、使用 Service 层，在 Service 的实现类中，使用静态集合变量模拟数据库存储商品信息，在控制器类中使用@Autowired 注解 Service。

4．通过地址 http://localhost:8080/shiyan7/toAdd 访问 addGoods.html 页面。

实验八 基于 JSP 引擎的 Spring Boot Web 开发

（实验课时：2　实验性质：设计）

实验名称：

基于 JSP 引擎的 Spring Boot Web 开发

实验目的和要求：

1、掌握 Spring Boot 对 JSP 的支持。

2、认真书写实验报告，如实填写各项实验内容。

实验内容和步骤：

参考【例 5-11】创建基于 JSP 引擎的 Spring Boot Web 应用 shiyan8。在应用 shiyan8 中创建两个视图页面 addGoods.jsp 和 goodsList.jsp。addGoods.jsp 页面的显示效果如图 12 所示，goodsList.jsp 页面的显示效果如图 13 所示。

图 12　addGoods.jsp 页面

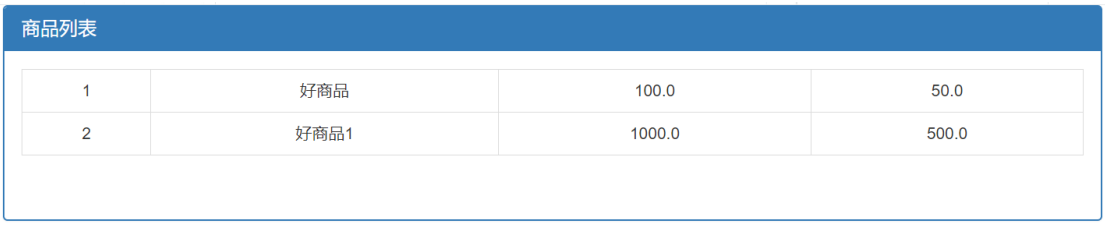

图 13　goodsList.jsp 页面

具体要求：

1、控制器类 GoodsController 中共有两个方法：toAdd 和 addGoods。

2、使用 Goods 模型类封装请求参数。

3、使用 Service 层，在 Service 的实现类中，使用静态集合变量模拟数据库存储商品信息，在控制器类中使用@Autowired 注解 Service。

4．通过地址 http://localhost:8080/shiyan8/toAdd 访问 addGoods.jsp 页面。

实验九 基于 Thymeleaf + Spring Boot + Spring JPA 的用户系统

（实验课时：4 实验性质：设计）

实验名称：

基于 Thymeleaf + Spring Boot + Spring JPA 的用户系统

实验目的和要求：

1、掌握基于 Spring Boot + Spring Data JPA 的应用程序的开发流程、方法以及技术。

2、认真书写实验报告，如实填写各项实验内容。

实验内容和步骤：

用户系统功能如下：从首页面进入“注册”或“登录”页面，注册成功后，进入登录页面，登录成功后进入主页面，主页面分页显示所有用户。

首页面如图 14 所示。

图 14 首页面

注册页面如图 15 所示。

图 15 注册页面

登录页面如图 16 所示。

图 16　登录页面

主页面如图 17 所示。

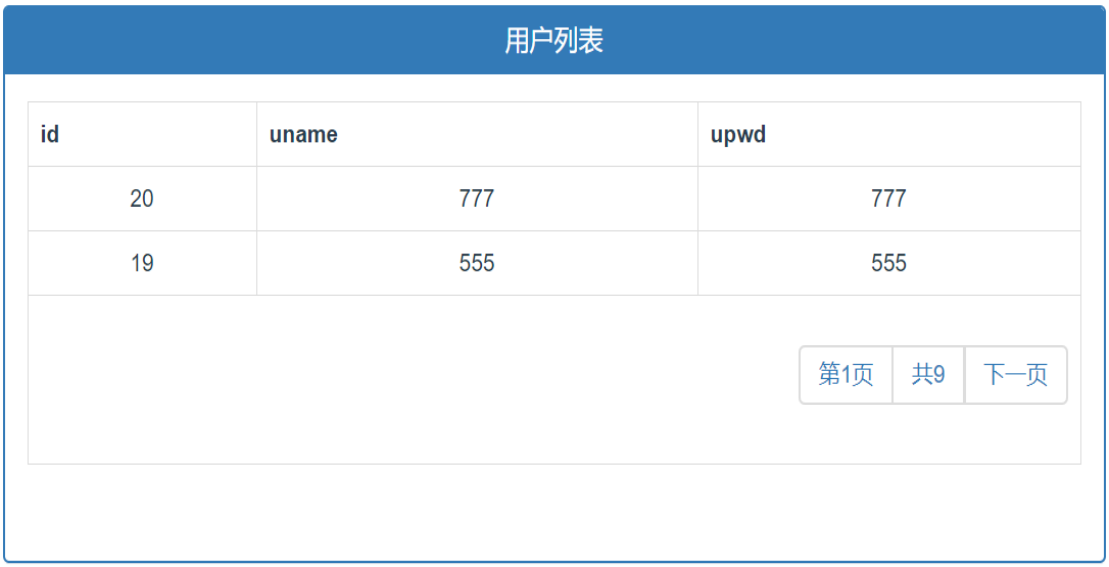

图 17　主页面

实验十 基于 JSP + Spring Boot + MyBatis 的用户系统

（实验课时：4　实验性质：设计）

实验名称：

基于 JSP + Spring Boot + MyBatis 的用户系统

实验目的和要求：

1、掌握基于 Spring Boot + MyBatis 的应用程序的开发流程、方法以及技术。

2、认真书写实验报告，如实填写各项实验内容。

实验内容和步骤:

内容与实验九一致，不再赘述。

实验十一 基于 Vue.js + Spring Boot +Spring JPA 的用户系统

（实验课时：4　实验性质：设计）

实验名称：

基于 Vue.js + Spring Boot +Spring JPA 的用户系统

实验目的和要求：

1、掌握基于 Vue.js+Spring Data JPA 的前后端分离的应用程序的开发流程、方法以及技术。

2、认真书写实验报告，如实填写各项实验内容。

实验内容和步骤：

内容与实验九一致，不再赘述。