****

**绥 化 学 院**

# 《**软件开发实践》 实验报告**

学 号

姓 名

年级班级

院系专业

授课教师 刘鹏

2024-2025学年 冬 季学期

**信息工程学院**

**实验一：**Spring Boot**入门案例**

**一、实验目的：**

1. 了解JavaWeb发展历程
2. 了解Spring Boot的常见功能
3. 使用Spring Boot进行日常开发

**二、实验内容及实现要求：**

**请按照以下步骤完成实验，将实验截图放在每个步骤之后**

**示例一：创建Spring Boot项目**

1. 在IDEA中创建一个 Module，选择类型为Spring Initializr快速构建
2. 设置 GAV坐标及pom配置信息
3. 选择Spring Boot版本及依赖
4. 设置模块名称、Content Root路径及模块文件的目录
5. 初始化pom文件
6. 初始化项目结构，运行测试案例

**示例二：使用Spring Boot创建Web项目**

1. 创建一个新的 Module，选择类型为 Spring Initializr
2. 设置 GAV坐标及pom配置信息
3. 选择Spring Boot版本及依赖
4. 设置模块名称、Content Root路径及模块文件的目录
5. 初始化pom文件
6. 初始化项目结构
7. 在controller包中创建一个SpringBootController.java文件
8. 在IDEA中运行Application类中的main方法
9. 在浏览器中输入url访问，访问SpringBootController.java的接口
10. 输出测试信息

**注：Spring Boot的核心配置文件用于配置Spring Boot程序，名字必须以application开始**

**示例三：.properties文件**

1. 在项目中查看.properties文件（默认采用该文件）
2. 编辑application.properties配置文件
3. 修改默认tomcat端口号及项目上下文
4. 运行项目，观察控制台启动参数
5. 输出页面显示结果

**示例四：.yml文件**

1. 在项目中创建.yml文件
2. 注释掉原有的application.properties文件，新建application.yml配置文件
3. 修改默认tomcat端口号及项目上下文
4. 运行项目，观察控制台启动参数
5. 输出页面显示结果

**示例五：多环境切换**

1. 为每个环境创建一个配置文件，命名必须以application-环境标识.properties或yml
2. SpringBoot总配置文件：application.yml
3. 开发环境配置文件：application-dev.yml
4. 测试环境配置文件：application-test.yml
5. 生产环境配置文件：application-product.yml
6. 选择开发环境的配置信息，运行项目显示结果

**示例六：Spring Boot自定义配置**

**在Spring Boot的核心配置文件中，除了使用内置的配置项之外，还可以在自定义配置，然后采用如下注解去读取配置的属性值**

1. 使用@Value注解，用于逐个读取 application.properties 中的配置

2、在核心配置文件applicatin.properties中，添加两个自定义配置项的属性school.name和website

3、在核心配置文件applicatin.properties中，添加两个自定义配置项的属性 school.name和website

4、或者在application.yml格式下添加两个自定义配置项school.name和school.website

5、在SpringBootController中定义属性，并使用@Value注解或者自定义配置值，并对其方法进行测试

6、运行 Application，在浏览器中进行测试

**使用注解@ConfigurationProperties，将整个文件映射成一个对象，用于自定义配置项比较多的情况**

1. 创建包结构，在config包下创建ConfigInfo类，并为该类加上Component和ConfigurationProperties注解，并在ConfigurationProperties注解中添加属性 prefix，作用可以区分同名配置
2. 在核心配置文件applicatin.properties中，添加两个自定义配置项 school.name和school.website
3. 或者在application.yml格式下添加两个自定义配置项school.name和school.website
4. 在 SpringBootController中使用@Autowired注解注入ConfigInfo配置类
5. 在SpringBootController类添加测试方法
6. 运行 Application，在浏览器中进行测试

**实验二：**Spring Boot**框架**Web**开发**

**一、实验目的：**

1. 了解JavaWeb发展历程
2. 了解Spring Boot的常见功能
3. 使用Spring Boot进行Web开发

**二、实验内容及实现要求：**

**请按照以下步骤完成实验，将实验截图放在每个步骤之后**

**示例一：**Spring Boot**集成**MyBatis

**通过**Spring Boot +MyBatis**实现对数据库学生表的查询操作**

1. 创建一个新的Spring Boot的Module
2. 指定GAV坐标
3. 选择Spring Boot版本以及web依赖
4. 修改 Content root以及Mudule file location
5. 初始化pom文件，在pom.xml中添加相关jar依赖
6. 使用IDEA连接数据库
7. 创建springboot数据库，并在springboot数据库中创建student数据表（student数据表应至少包含id、name、age等属性）
8. 为student数据表添加测试数据
9. 在Springboot的核心配置文件application.properties中配置数据源
10. 构建项目包结构
11. 在pom.xml文件中添加mybatis-generator插件的配置信息
12. 在项目中新建GeneratorMapper.xml配置文件，配置逆向工程的基本信息
13. 使用Mybatis反向工程生成接口、映射文件以及实体bean
14. 在controller包下创建StudentController.java并编写代码
15. 在service包下创建StudentService.java接口并编写代码
16. 在service.impl包下创建StudentService.java的实现类StudentServiceImpl并编写代码
17. 在Mybatis反向工程生成的StudentMapper接口上加一个@Mapper注解（@Mapper作用：mybatis自动扫描数据持久层的映射文件及DAO接口的关系）
18. 启动 Application 应用，浏览器访问测试运行

**示例二：**Spring Boot**集成**MyBatis**的其他方式（请在示例一的基础上进行操作）**

1. 在运行的主类上添加包扫描注解@MapperScan("需要扫描的路径")，通过这种方法可以将StudentMapper接口上的@Mapper注解注释掉
2. 将接口与接口映射的xml文件分开管理，在resources目录下新建目录mapper存放映射文件，将StudentMapper.xml文件移到resources/mapper目录下
3. 在application.properties配置文件中指定映射文件的位置，这个配置只有接口和映射文件不在同一个包的情况下，才需要指定
4. 启动 Application 应用，浏览器访问测试运行

**示例三：**Spring Boot**的事务（请在示例一、二的基础上进行操作）**

1. 在StudentController.java文件中添加更新学生的方法
2. 在StudentService.java接口中添加更新学生方法
3. 在StudentServiceImpl.java接口实现类中对更新学生方法进行实现，并构建一个异常，同时在该方法上加@Transactional注解
4. 在Application类上加@EnableTransactionManagement开启事务支持，注意@EnableTransactionManagement可选，但是业务方法上必须添加@Transactional 事务才生效
5. 启动 Application，通过浏览器访问进行测试
6. 检查数据库中记录变化的情况，检查事务是否生效
7. 注释掉 StudentServiceImpl 上的@Transactional 测试
8. 启动 Application，通过浏览器访问进行测试
9. 检查数据库中记录变化的情况，检查事务是否失效

**示例四：**Spring Boot**下的**SpringMVC

明确常见注解的主要功能和应用场景：

@Controller：SpringMVC的注解，处理http请求

@RestController：Spring 4后新增注解，是@Controller与@ResponseBody 的组合注解如果一个Controller类添加了@RestController，那么该 Controller类下的所有方法都相当于添加了@ResponseBody注解

@ResponseBody：用于返回字符串或 json数据

@RequestMapping（常用）：支持 Get 请求，也支持 Post 请求

@GetMapping：RequestMapping和Get请求方法的组合，只支持Get请求，主要用于查询操作

@PostMapping：RequestMapping和Post请求方法的组合只支持 Post 请求，主要用户新增数据

@PutMapping：RequestMapping和Put请求方法的组合，只支持Put请求，通常用于修改数据

@DeleteMapping：RequestMapping 和Delete请求方法的组合，只支持Delete请求通常用于删除数据

1. 创建一个MVCController，里面使用上面介绍的各种注解接收不同的请求
2. 启动应用，在浏览器中输入不同的请求进行测试
3. 使用Postman对其它请求类型做个测试

**示例五：**Spring Boot**实现**RESTful（**使用**RESTful**风格模拟实现对学生的增删改查操作**）

Spring boot开发RESTFul主要是几个注解实现

@PathVariable：获取url中的数据，该注解是实现RESTFul最主要的一个注解

@PostMapping：接收和处理Post方式的请求

@DeleteMapping：接收delete方式的请求，可以使用GetMapping代替

@PutMapping：接收 put 方式的请求，可以用PostMapping代替

@GetMapping：接收 get 方式的请求

1. 创建一个新的Spring Boot的Module
2. 指定GAV坐标
3. 选择Spring Boot版本以及web依赖
4. 修改 Content root以及Mudule file location
5. 初始化pom文件，在pom.xml中添加相关jar依赖
6. 使用IDEA连接数据库
7. 创建springboot数据库，并在springboot数据库中创建student数据表（student数据表应至少包含id、name、age等属性）
8. 为student数据表添加测试数据
9. 在Springboot的核心配置文件application.properties中配置数据源
10. 构建项目包结构
11. 在pom.xml文件中添加mybatis-generator插件的配置信息
12. 在项目中新建GeneratorMapper.xml配置文件，配置逆向工程的基本信息
13. 使用Mybatis反向工程生成接口、映射文件以及实体bean
14. 在controller包下创建 RESTfulController，并编写代码
15. 在service包下创建StudentService.java接口并编写代码
16. 在service.impl包下创建StudentService.java的实现类StudentServiceImpl并编写代码
17. 在Mybatis反向工程生成的StudentMapper接口上加一个@Mapper注解（@Mapper作用：mybatis自动扫描数据持久层的映射文件及DAO接口的关系）
18. 使用 Postman 模拟发送请求，进行测试

**实验三：**Redis与Spring Boot的集成实验报告

**一、实验目的：**

1. 掌握Redis的基本概念及数据类型
2. 学习Redis与Spring Boot的集成方法
3. 了解Redis的缓存应用功能

**二、实验内容及实现要求：**

**请按照以下步骤完成实验，将实验截图放在每个步骤之后**

**示例一：**Spring Boot**集成**Redis

1. 创建一个Spring Boot项目，添加Redis依赖
2. 在 application.yml 配置文件中设置 Redis 连接信息
3. 使用 RedisTemplate 进行基本操作，如字符串存储与读取
4. 验证 Redis 是否连接成功

**示例二：**Redis的基本操作**（请在示例一的基础上进行操作）**

1. 实现对 Redis 的五种基本数据类型操作：
   1. 字符串（String）：存储和读取简单键值对。
   2. 哈希（Hash）：使用 hset 和 hget 操作用户信息。
   3. 列表（List）：添加和获取消息队列数据。
   4. 集合（Set）：存储唯一值集合。
   5. 有序集合（Sorted Set）：根据分数排序存储。
2. 通过代码截图展示各数据类型的操作及结果。

**示例三：**Redis缓存应用**（请在示例一、二的基础上进行操作）**

1. 在项目中集成 Redis 作为缓存，避免重复查询数据库
2. 在 Service 层实现：
   1. 首先从 Redis 缓存中查找数据；
   2. 如果未找到，再从数据库中获取并缓存。
3. 配置 Redis 缓存过期时间。

**实验四：D**ubbo与Spring Boot的集成实验报告

**一、实验目的：**

1. 理解分布式服务框架Dubbo的架构与原理
2. 学习Dubbo与SpringBoot的集成
3. 掌握Dubbo的API模块设计以及Provider和Consumer模块的实现

**二、实验内容及实现要求：**

**请按照以下步骤完成实验，将实验截图放在每个步骤之后**

**示例一：**Spring Boot**集成**Dubbo

1. 创建一个Spring Boot项目，创建三个模块：API、Provider、Consumer
2. 在pom.xml文件中添加Dubbo的相关依赖
3. 在Provider、Consumer项目中引入API模块依赖
4. 配置application.yml

**示例二：**实现基本功能

1. API模块：定义服务接口
2. Provider模块：实现服务接口并注解：
3. Consumer模块：通过Dubbo引入服务接口：

**示例三：**服务调用测试

1. 启动Provider和Consumer服务
2. 在浏览器或Postman中调用Consumer提供的接口，验证Dubbo调用效果