面向服务的系统设计课程

微服务架构基础

实验设计

说明：本实验设计要求每两人一组，以小组为单位进行。每次实验结束后，各小组提交一次实验报告。实验报告**仅需**记录（包括文字记录、截图记录等）实验的过程，如实验题目中提出的问题、运行代码的结果截图等。

实验一 初识 Spring Boot

Spring Boot 是由 Spring 团队提供的一个全新框架，其设计目的是为了简化 Spring 应用的初始搭建过程和开发过程。请同学们根据课程内容，分组进行 Spring Boot 项目的快速构建。本实验预期目标分为：

1. 请简述 Spring Boot 的执行流程；
2. 通过代码生成器 Spring Initializer 构建名字为 GroupX（X 为小组编号）的 Spring Boot 项目；
3. 编写 Spring Boot 项目控制层订单控制类 OrderController，并在其中输出“Hello Order”提示；
4. 启动 Spring Boot 项目，并对启动结果进行截图。

实验二 Spring Boot 与 MyBatis 的集成

MyBatis 是一款优秀的持久层框架，它支持定制化 SQL、存储过程以及高级映射。MyBatis 避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。请同学们根据课程内容，分组进行 Spring Boot 与 MyBatis 的集成。本实验预期目标分为：

1. 创建名为 microservice 的 MySQL 数据库，并同时创建 tb_user 用户表；
2. 准备用户表相关数据，类型为（‘用户编号’，‘用户名’，‘住址’），请同学们自行准备至少 3 条相关数据；
3. 为实验一中的项目(pom.xml)添加 MyBatis 启动器及 MySQL 驱动；

```
<!-- MyBatis启动器 -->
<dependency>
    <groupId>org.mybatis.spring.boot</groupId>
    <artifactId>mybatis-spring-boot-starter</artifactId>
    <version>1.1.1</version>
</dependency>
<!-- MySQL驱动 -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
</dependency>
```

4. 修改项目 Properties，并添加相关数据库配置；

```
#DB Configuration
spring.datasource.driverClassName=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/microservice
spring.datasource.username=root
spring.datasource.password=root
#logging
logging.level.com.itheima.springboot=debug
```

5. 创建用户实体类 User。用户实体类的具体内容如下：

```
public class User{
/*请同学们自行补充属性 id，username，address 以及相应的 get 和 set 方法*/
}
```

6. 编写 Mapper。@Mapper 是 MyBatis 框架用于声明 Mapper 接口的注解，@Select 是用于映射查询 SQL 语句的注解，@Delete 是用于映射删除 SQL 语句的注解。示例如下：

```
public interface UserMapper{
    @Select("select * from tb_user")
    List<User> getAllUsers();
    @Delete("DELETE FROM tb_user WHERE id = #{id}")
    void delete(Integer id);
}
```

7. 编写 Service。此处分别分为 Service 接口以及实现类。

Service 接口

```
public interface UserService{
    List<User> getAllUsers();
    void deleteUser(Integer id);
}
```

Service 实现类

```
@Service
@Transactional
public class UserServiceImpl implements UserService{
    //注入用户 Mapper
    @Autowired
    private UserMapper userMapper;
    public List<User> getAllUsers(){
        return this.userMapper.getAllUsers();
    }
    public void deleteUser(Integer id){
        System.out.println("删除了 id 为"+id+"的用户");
        this.userMapper.delete(id);
    }
}
```

8. 编写 Controller。

```
@RestController
@RequestMapping("/user")
```

```
public class UserController{
    //注入用户 Service
    @Autowired
    private UserService userService;
    @RequestMapping("/userList")
    public List<User> getAllUsers(){
        List<User> list = this.userService.getAllUsers();
        return list;
    }
    @GetMapping("/delete")
    public void delete(@RequestParam Integer id){
        this.userService.deleteUser(id);
    }
}
```

请同学们依照以上示例代码进行项目构建并自行构思如何对以上代码进行测试。如有可能，请基于以上 Spring Boot 及 MyBatis 的实现，编写前端 HTML 及 JS 代码进行数据库信息的显示及删除。

注意，删除用户的时候，浏览器中地址后缀名为/delete?id=2 (此处 2 为要删除的用户)。如仍遇到困难，可参考 https://blog.csdn.net/Lushengshi/article/details/130003813

实验三 Spring Cloud 服务发现

Spring Cloud 是在 Spring Boot 基础上构建的，用于简化分布式系统构建的工具集。该工具集为微服务架构中所涉及的配置管理、服务发现、智能路由、断路器、微代理、控制总线等操作提供了一种简单的开发方式。在微服务架构中，服务发现是最为核心和基础的模块，该模块主要用于实现各个微服务实例的自动化注册与发现。请同学们根据课程内容，完成 Spring Cloud 服务发现的基础搭建。本实验预期目标分为：

1. 简述 Spring Cloud 服务发现工具 Eureka 两大组件的功能（Eureka Server 以及 Eureka Client）；
2. 搭建 Maven 父项目 microservice-eureka-server，引入 Spring Cloud 工具集依赖（可参照 https://blog.csdn.net/qq_45926015/article/details/118611850）；

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-dependencies</artifactId>
            <version>Dalston.SR3</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

3. 引入注册中心 Eureka Server 依赖；

```
<dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-eureka-server</artifactId>
        </dependency>
</dependencies>
```

4. 编写配置文件，设置 Eureka Server 服务地址；

```
server:
 port: 8761
eureka:
 instance:
  hostname: localhost
 client:
  register-with-eureka: false
  fetch-registry: false
  service-url:
   defaultZone:
        http://${eureka.instance.hostname}:${server.port}/eureka/
```

5. 在项目的引导类上添加注解@EnableEurekaServer，并测试启动效

果；

```
@SpringBootApplication
@EnableEurekaServer
public class EurekaApplication {
        public static void main(String[] args) {
                SpringApplication.run(EurekaApplication.class, args);
        }
}
```

6. 同理参照以上步骤及课程 4.2.2，创建项目 microservice-eureka-user 并进行服务注册的测试；

实验四 Docker 的使用

使用 Docker 可以让开发者封装他们的应用以及依赖包到一个可移植的容器中，然后发布到任意的 Linux 机器上，也可以实现虚拟化。请同学们依照课程 6.2 的步骤，首先安装并运行 Docker。在此基础上，本实验预期目标为：

1. 简述 Docker 的工作机制；
2. 编写 Docker 入门程序并进行测试（参照课程 7.1 编写 Dockerfile，requirements.txt，app.py）；
3. 请同学们自行对 Docker 各类基本命令进行测试，并提供相应的运行截图。

实验五 微服务项目整合

本实验旨在基于商城管理系统对微服务框架进行使用测试，请同学们依据指导教师提供的商品管理系统代码开展测试（具体操作步骤请参照课程 9.1 所述）。请同学们对所提供的软件系统中每个 Java 文件的功能进行描述，并在成功运行系统后对不同功能进行测试并截图。