

# 1、课程名称：Restful 基础架构



## 2、具体内容

对于 Rest 基础架构实现处理是 SpringCloud 核心所在，其基本操作形式在 SpringBoot 之中已经有了明确的讲解，那么本次为了清晰可见，创建一套新的微服务架构：部门微服务（Dept）。

如果要想进行 SpringCloud 开发，那么一定要对 SpringBoot 有一定的了解，同时本次也将融合 MyBatis 开发技术实现整体的微服务的创建处理。

### 2.1、搭建项目环境

对于现在的项目创建一个：microcloud 的整体父 pom 项目，那么随后为了方便管理，将创建其三个子模块：

- microcloud-api 模块，作为公共的信息导入配置模块；
- microcloud-provider-dept-8001：作为服务提供者，该提供者负责使用 Mybatis 与数据库交互；
- microcloud-consumer-80：作为微服务调用的客户端使用。

1、 创建一个新的 maven 项目：microcloud； **SpringCloud离不开SpringBoot，所以必须要配置此依赖包**

2、 【microcloud】修改 pom.xml 文件，主要追加 springcloud 与 springboot 两个开发包的依赖关系；

注意：SpringCloud 中针对于依赖包的版本并不是像传统那样采用数字的形式定义的，而是使用了一系列的英国的地铁或陈真名字来定义的。

Spring Cloud	
RELEASE	DOCUMENTATION
Finchley <small>SNAPSHOT</small>	
Edgware <small>SNAPSHOT</small>	
Dalston SR1 <small>CURRENT</small> <a href="#">Reference</a>	
Camden SR7 <small>GA</small>	<a href="#">Reference</a>
Brixton SR7 <small>GA</small>	<a href="#">Reference</a>
Angel SR6 <small>GA</small>	<a href="#">Reference</a>

```
<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>
  <groupId>cn.mldn</groupId>
  <artifactId>microcloud</artifactId>
  <version>0.0.1</version>
  <packaging>pom</packaging>
  <name>microcloud</name>
  <url>http://maven.apache.org</url>
  <properties>
    <jdk.version>1.8</jdk.version>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <dependencyManagement>
    <dependencies>
      <dependency> <!-- 进行SpringCloud依赖包的导入处理 -->
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-dependencies</artifactId>
        <version>Dalston.SR1</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
      <dependency> <!-- SpringCloud离不开SpringBoot，所以必须要配置此依赖包 -->
        <groupId>org.springframework.boot</groupId>
```

```

        <artifactId>spring-boot-dependencies</artifactId>
        <version>1.5.4.RELEASE</version>
        <type>pom</type>
        <scope>import</scope>
    </dependency>
</dependencies>
</dependencyManagement>
<build>
    <finalName>microcloud</finalName>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <configuration>
                <source>${jdk.version}</source><!-- 源代码使用的开发版本 -->
                <target>${jdk.version}</target><!-- 需要生成的目标class文件的编译版本 -->
                <encoding>${project.build.sourceEncoding}</encoding>
            </configuration>
        </plugin>
    </plugins>
</build>
</project>

```

3、【microcloud-api】建立一个 api 的公共模块，该模块的主要功能是提供有公共处理类，本次预计建立一个 Dept 数据表，里面的字段：deptno (Long)、dname (String)、loc (保存的数据库的名字)；

```

package cn.mldn.vo;
import java.io.Serializable;
@SuppressWarnings("serial")
public class Dept implements Serializable {
    private Long deptno ;
    private String dname ;
    private String loc ;
}

```

4、【microcloud-provider-dept-8001】创建一个 Rest 提供者的项目模块，在这个模块里面主要定义要使用的数据库脚本：

```

DROP DATABASE IF EXISTS "mldn8001" ;
CREATE DATABASE mldn8001 CHARACTER SET UTF8 ;
USE mldn8001 ;
CREATE TABLE dept (
    deptno      BIGINT      AUTO_INCREMENT ,
    dname       VARCHAR(50) ,
    loc         VARCHAR(50) ,
    CONSTRAINT pk_deptno PRIMARY KEY(deptno)
) ;
INSERT INTO dept(dname,loc) VALUES ('开发部',database()) ;
INSERT INTO dept(dname,loc) VALUES ('财务部',database()) ;

```

```
INSERT INTO dept(dname,loc) VALUES ('市场部',database()) ;
INSERT INTO dept(dname,loc) VALUES ('后勤部',database()) ;
INSERT INTO dept(dname,loc) VALUES ('公关部',database()) ;
```

由于在整个微服务里面需要进行负载均衡的操作，所以本次在使用的时候加入了数据库的名称信息。

## 2.2、创建 Dept 微服务

所谓的微服务的核心本质就是 JSON 的传输，那么既然现在要求使用 MyBatis 进行数据库操作，所以应该在项目里面配置 Druid 数据库连接池，而后对外进行项目的发布。

1、【microcloud-provider-dept-8001】修改 pom.xml 配置文件，追加相关的依赖程序支持包：

```
<dependencies>
    <dependency>
        <groupId>cn.mldn</groupId>
        <artifactId>microcloud-api</artifactId>
    </dependency>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>
    <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>druid</artifactId>
    </dependency>
    <dependency>
        <groupId>ch.qos.logback</groupId>
        <artifactId>logback-core</artifactId>
    </dependency>
    <dependency>
        <groupId>org.mybatis.spring.boot</groupId>
        <artifactId>mybatis-spring-boot-starter</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-jetty</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
```

```

</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>springloaded</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
</dependency>
</dependencies>

```

2、【microcloud-provider-dept-8001】创建一个 IDeptDAO 数据操作接口，这个接口里面将提供有三个数据操作方法：

```

package cn.mldn.microcloud.dao;
import java.util.List;
import org.apache.ibatis.annotations.Mapper;
import cn.mldn.vo.Dept;
@Mapper
public interface IDeptDAO {
    public boolean doCreate(Dept vo) ;
    public Dept findById(Long id) ;
    public List<Dept> findAll() ;
}

```

3、【microcloud-provider-dept-8001】修改 application.yml 配置文件，追加 mybatis 和服务的相关配置信息：

```

server:
  port: 8001
mybatis:
  config-location: classpath:mybatis/mybatis.cfg.xml    # mybatis配置文件所在路径
  type-aliases-package: cn.mldn.vo                    # 定义所有操作类的别名所在包
  mapper-locations:                                    # 所有的mapper映射文件
    - classpath:mybatis/mapper/**/*.xml
spring:
  datasource:
    type: com.alibaba.druid.pool.DruidDataSource        # 配置当前要使用的数据源的操作类型
    driver-class-name: org.gjt.mm.mysql.Driver          # 配置MySQL的驱动程序类
    url: jdbc:mysql://localhost:3306/mldn8001           # 数据库连接地址
    username: root                                       # 数据库用户名
    password: mysqladmin                                # 数据库连接密码
    dbcp2:                                                # 进行数据库连接池的配置
      min-idle: 5                                       # 数据库连接池的最小维持连接数
      initial-size: 5                                   # 初始化提供的连接数

```

```
max-total: 5 # 最大的连接数
max-wait-millis: 200 # 等待连接获取的最大超时时间
```

4、【microcloud-provider-dept-8001】定义 src/main/resources/mybatis/mybatis.cfg.xml 配置文件:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration> <!-- 进行Mybatis的相应的环境的属性定义 -->
  <settings> <!-- 在本项目之中开启二级缓存 -->
    <setting name="cacheEnabled" value="true"/>
  </settings>
</configuration>
```

5、【microcloud-provider-dept-8001】修改 src/main/resources/mybatis/mapper/cn/mldn/Dept.xml 配置文件:

```
<?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">
<mapper namespace="cn.mldn.microcloud.dao.IDeptDAO">
  <select id="findById" resultType="Dept" parameterType="Long">
    SELECT deptno,dname,loc FROM dept WHERE deptno=#{deptno} ;
  </select>
  <select id="findALL" resultType="Dept">
    SELECT deptno,dname,loc FROM dept ;
  </select>
  <insert id="doCreate" parameterType="Dept">
    INSERT INTO dept(dname,loc) VALUES (#{dname},database()) ;
  </insert>
</mapper>
```

6、【microcloud-provider-dept-8001】建立 IDeptService 接口, 做业务实现:

```
package cn.mldn.microcloud.service;
import java.util.List;
import cn.mldn.vo.Dept;
public interface IDeptService {
  public Dept get(long id) ;
  public boolean add(Dept dept) ;
  public List<Dept> list() ;
}

package cn.mldn.microcloud.service.impl;
import java.util.List;
import javax.annotation.Resource;
import org.springframework.stereotype.Service;
import cn.mldn.microcloud.dao.IDeptDAO;
import cn.mldn.microcloud.service.IDeptService;
import cn.mldn.vo.Dept;
@Service
```

```
public class DeptServiceImpl implements IDeptService {
    @Resource
    private IDeptDAO deptDAO ;
    @Override
    public Dept get(long id) {
        return this.deptDAO.findById(id);
    }
    @Override
    public boolean add(Dept dept) {
        return this.deptDAO.doCreate(dept);
    }
    @Override
    public List<Dept> list() {
        return this.deptDAO.findAll();
    }
}
```

7、【microcloud-provider-dept-8001】定义程序的运行主类

```
package cn.mldn.microcloud;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class Dept_8001_StartSpringCloudApplication {
    public static void main(String[] args) {
        SpringApplication.run(Dept_8001_StartSpringCloudApplication.class, args);
    }
}
```

8、【microcloud-provider-dept-8001】进行业务接口测试编写：

```
package cn.mldn.test;
import javax.annotation.Resource;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import org.springframework.test.context.web.WebAppConfiguration;
import cn.mldn.microcloud.Dept_8001_StartSpringCloudApplication;
import cn.mldn.microcloud.service.IDeptService;
import cn.mldn.vo.Dept;
@SpringBootTest(classes = Dept_8001_StartSpringCloudApplication.class)
@RunWith(SpringJUnit4ClassRunner.class)
@WebAppConfiguration
public class IDeptServiceTest {
    @Resource
    private IDeptService deptService ;
    @Test
```

```
public void testGet() {
    System.out.println(this.deptService.get(1));
}
@Test
public void testAdd() {
    Dept dept = new Dept() ;
    dept.setDname("测试部-" + System.currentTimeMillis());
    System.out.println(this.deptService.add(dept));
}
@Test
public void testList() {
    System.out.println(this.deptService.list());
}
}
```

9、【microcloud-provider-dept-8001】建立 DeptRest 服务类:

```
package cn.mldn.microcloud.rest;
import javax.annotation.Resource;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
import cn.mldn.microcloud.service.IDeptService;
import cn.mldn.vo.Dept;
@RestController
public class DeptRest {
    @Resource
    private IDeptService deptService ;
    @RequestMapping(value="/dept/get/{id}",method=RequestMethod.GET)
    public Object get(@PathVariable("id") long id) {
        return this.deptService.get(id) ;
    }
    @RequestMapping(value="/dept/add",method=RequestMethod.GET)
    public Object add(@RequestBody Dept dept) {
        return this.deptService.add(dept) ;
    }
    @RequestMapping(value="/dept/list",method=RequestMethod.GET)
    public Object list() {
        return this.deptService.list() ;
    }
}
```

10、修改 hosts 配置文件，追加一个映射路径（路径：C:\Windows\System32\drivers\etc\hosts）。

127.0.0.1 dept-8001.com

11、观察 Rest 服务能否正常提供:

- 调用 get 操作: dept-8001.com:8001/dept/get/1;



- 调用 list 操作: <http://dept-8001.com:8001/dept/list>;

## 2.3、客户端调用微服务

- 1、 创建一个 Maven 的新的模块: microcloud-consumer-80;
- 2、 【microcloud-consumer-80】修改 application.yml 配置文件:

```
server:  
  port: 80
```

- 3、 【microcloud-consumer-80】创建一个 Rest 的配置程序类, 主要进行 RestTemplate 类对象创建:

```
package cn.mldn.microcloud.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.web.client.RestTemplate;  
  
@Configuration  
public class RestConfig {  
    @Bean  
    public RestTemplate getRestTemplate() {  
        return new RestTemplate() ;  
    }  
}
```

- 4、 【microcloud-consumer-80】创建一个控制器, 为了简化处理, 本次不再进行页面定义了, 所有服务结果都使用 Rest 返回;

```
package cn.mldn.microcloud.controller;  
  
import java.util.List;  
import javax.annotation.Resource;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.client.RestTemplate;  
import cn.mldn.vo.Dept;  
  
@RestController  
public class ConsumerDeptController {  
    public static final String DEPT_GET_URL = "http://dept-8001.com:8001/dept/get/";  
    public static final String DEPT_LIST_URL = "http://dept-8001.com:8001/dept/list/";  
    public static final String DEPT_ADD_URL = "http://dept-8001.com:8001/dept/add";  
    @Resource  
    private RestTemplate restTemplate;  
    @RequestMapping(value = "/consumer/dept/get")  
    public Object getDept(long id) {  
        Dept dept = this.restTemplate.getForObject(DEPT_GET_URL + id,  
            Dept.class);  
        return dept;  
    }  
    @SuppressWarnings("unchecked")  
    @RequestMapping(value = "/consumer/dept/list")
```

```
public Object listDept() {  
    List<Dept> allDepts = this.restTemplate.getForObject(DEPT_LIST_URL,  
        List.class);  
    return allDepts;  
}  
  
@RequestMapping(value = "/consumer/dept/add")  
public Object addDept(Dept dept) {  
    Boolean flag = this.restTemplate.postForObject(DEPT_ADD_URL, dept,  
        Boolean.class);  
    return flag;  
}  
}
```

5、【microcloud-consumer-80】编写启动程序类

```
package cn.mldn.microcloud;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class Consumer_80_StartSpringCloudApplication {  
    public static void main(String[] args) {  
        SpringApplication.run(Consumer_80_StartSpringCloudApplication.class,  
            args);  
    }  
}
```

6、修改 hosts 配置文件，追加访问控制路径：

```
127.0.0.1 client.com
```

7、进行代码测试调用：

- 调用数据返回：http://client.com/consumer/dept/get?id=1;
- 测试列表数据：http://client.com/consumer/dept/list;
- 测试数据增加：http://client.com/consumer/dept/add?dname=WEB 测试;