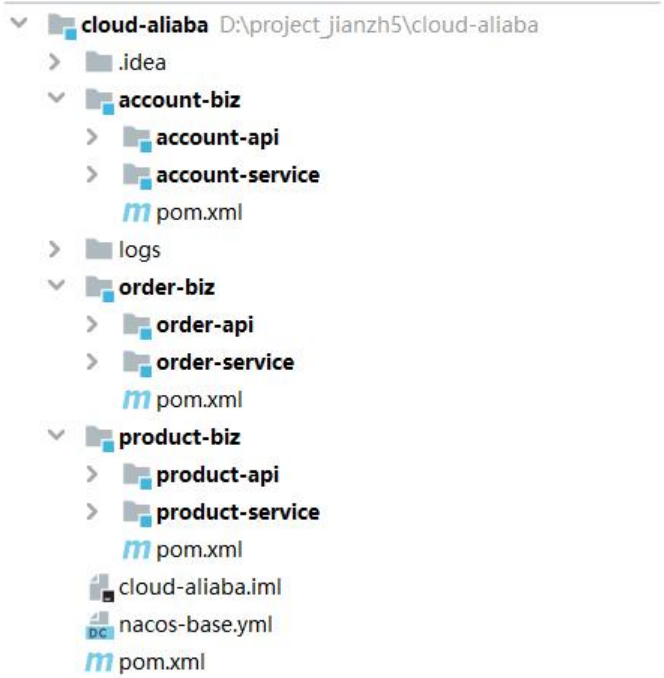
SpringCloud Alibaba 微服务实战二 - 服务注册

作者：jianzh5
原文链接：https://ld246.com/article/1575360228181
来源网站：链滴

导读：在之前一篇文章中我们准备好了基于SpringCloud Alibaba的基础组件，本期主要内容是将所的服务注册进Nacos，并让account-service和product-service能对外提供基础的增删改查能力。

基础框架搭建

在你的IDEA中建立一个多模块的项目（过程略...），项目整体截图如下：



- 在主pom中定义基础组件版本，使用 dependencyManagement引入版本依赖。

```
<properties>
    <java.version>1.8</java.version>
    <spring-boot.version>2.1.9.RELEASE</spring-boot.version>
```

```
    <springcloud-alibaba.version>0.9.0.RELEASE</springcloud-alibaba.version>
    <mybatis-plus.version>3.1.1</mybatis-plus.version>
    <mysql.version>5.1.47</mysql.version>
    <encoding>UTF-8</encoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
</properties>

<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-dependencies</artifactId>
            <version>${spring-boot.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>

        <!--database-->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>${mysql.version}</version>
        </dependency>

        <dependency>
            <groupId>com.baomidou</groupId>
            <artifactId>mybatis-plus-boot-starter</artifactId>
            <version>${mybatis-plus.version}</version>
        </dependency>

        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-alibaba-dependencies</artifactId>
            <version>${springcloud-alibaba.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>

    </dependencies>
</dependencyManagement>
```

- 在Service模块中引入具体依赖，个人习惯使用log4j2作为日志组件，根据你们习惯自行选择

```
<dependencies>
    <!--Spring Boot-->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
        <exclusions>
            <exclusion>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-logging</artifactId>
            </exclusion>
```

```
        </exclusions>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-log4j2</artifactId>
    </dependency>

    <!--Spring Cloud Alibaba-->
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
    </dependency>

    <!--database-->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>

    <dependency>
        <groupId>com.baomidou</groupId>
        <artifactId>mybatis-plus-boot-starter</artifactId>
    </dependency>

    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>

</dependencies>
```

集成Nacos注册中心

- 引入 spring-cloud-starter-alibaba-nacos-discovery，上一步骤已完成；
- 修改配置文件 application.yml，配置nacos的服务地址（注意修改服务端口）；

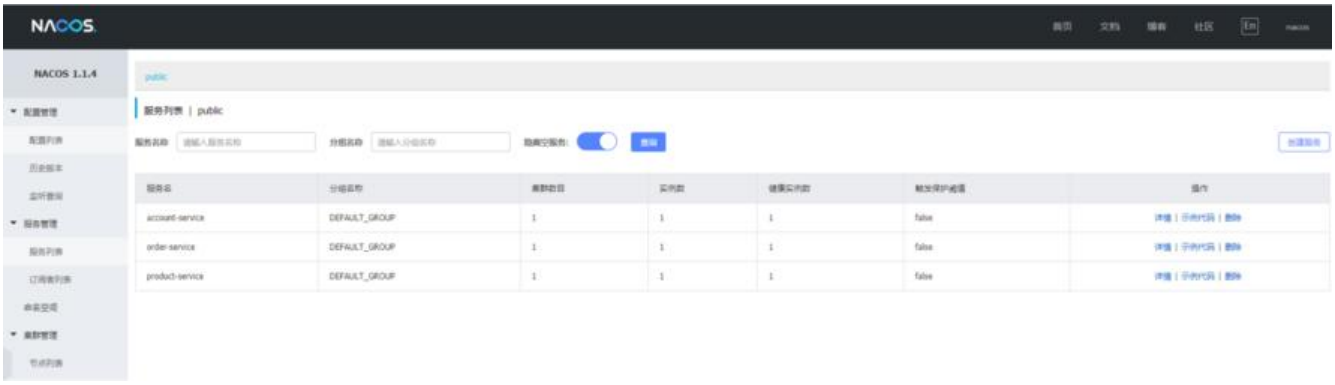
```
server:
  port: 8010
spring:
  application:
    name: account-service
  cloud:
    nacos:
      discovery:
        server-addr: 10.0.10.48:8848/
```

- 在项目启动类上添加 @EnableDiscoveryClient注解

```
@SpringBootApplication
@RestController
@EnableDiscoveryClient
```

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

简单三步就将服务注册进了nacos，分别启动account-service，product-service，order-service，动完成后访问nacos服务端地址http://10.0.10.48:8848/nacos可以发现服务正常注册。



提供基础增删改查能力

数据准备

本系列文章都是基于account-service，product-service，order-service，所以我们先准备好这三张础表结构。

```
SET FOREIGN_KEY_CHECKS=0;

-- ----------------------------
-- Table structure for t_order
-- ----------------------------
DROP TABLE IF EXISTS `t_order`;
CREATE TABLE `order` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `ORDER_NO` varchar(255) DEFAULT NULL,
  `ACCOUNT_CODE` varchar(255) DEFAULT NULL,
  `PRODUCT_CODE` varchar(255) DEFAULT NULL,
  `COUNT` int(11) DEFAULT '0',
  `AMOUNT` decimal(10,2) DEFAULT '0.00',
  PRIMARY KEY (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


DROP TABLE IF EXISTS `product`;
CREATE TABLE `product` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `PRODUCT_CODE` varchar(255) DEFAULT NULL COMMENT '编码',
  `PRODUCT_NAME` varchar(255) DEFAULT NULL COMMENT '名称',
  `COUNT` int(11) DEFAULT '0' COMMENT '库存数量',
  `PRICE` decimal(10,2) DEFAULT '0.00' COMMENT '单价',
  PRIMARY KEY (`ID`),
  UNIQUE KEY `UK_PRODUCT_CODE` (`PRODUCT_CODE`)
```

```
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4;

-- ----------------------------
-- Records of product
-- ----------------------------
INSERT INTO `product` VALUES ('1', 'P001', '笔记本', '10', '3000.00');
INSERT INTO `product` VALUES ('2', 'P002', '手表', '5', '250.00');
INSERT INTO `product` VALUES ('3', 'P003', '键盘', '50', '100.00');
INSERT INTO `product` VALUES ('4', 'P004', '辣条', '1000', '0.50');

DROP TABLE IF EXISTS `account`;
CREATE TABLE `account` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `ACCOUNT_CODE` varchar(255) DEFAULT NULL,
  `ACCOUNT_NAME` varchar(255) DEFAULT NULL,
  `AMOUNT` decimal(10,2) DEFAULT '0.00',
  PRIMARY KEY (`ID`),
  UNIQUE KEY `UK_ACCOUNT_CODE` (`ACCOUNT_CODE`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4;

-- ----------------------------
-- Records of account
-- ----------------------------
INSERT INTO `account` VALUES ('1', 'javadaily', 'JAVA日知录', '10000.00');
```

代码实现

由于目前只是对外提供简单CRUD的能力，所以这块代码我就不贴出来了，只提供一下我们对外的接Controller，Service层和Dao层大家可以根据技术栈自行实现。这里向大家推荐下mybatis-plus插件可以大大提高CRUD的实现效率，有兴趣的可以去官网查看文档。

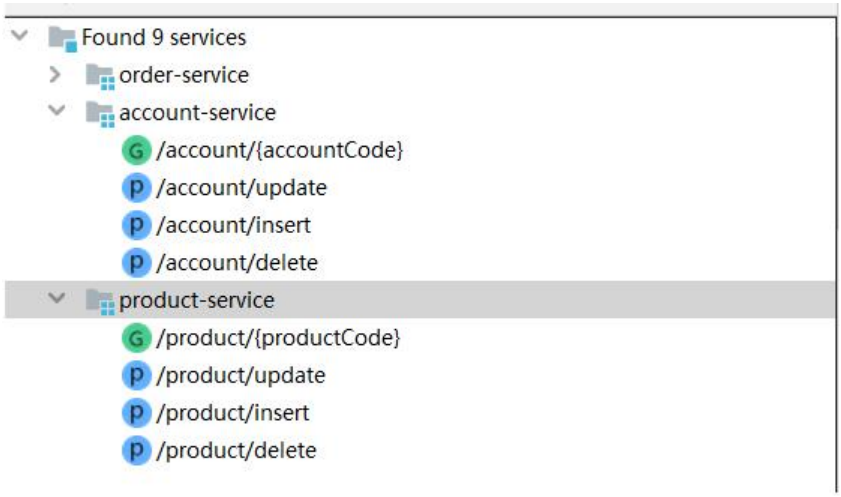
- 接口Controller层

```
@RestController
@Log4j2
public class AccountController {
    @Autowired
    private AccountService accountService;

    @GetMapping("/account/{accountCode}")
    public AccountVO getByCode(@PathVariable String accountCode){
        log.info("get account detail,accountCode is :{}",accountCode);
        return accountService.selectByCode(accountCode);
    }

    @PostMapping("/account/update")
    public AccountVO update(AccountVO accountVO){
        log.info("update account:{}",accountVO);
        return accountService.updateAccount(accountVO);
    }

    @PostMapping("/account/insert")
    public AccountVO insert(AccountVO accountVO){
        log.info("insert account:{}",accountVO);
```

```
        return accountService.insertAccount(accountVO);
    }

    @PostMapping("/account/delete")
    public int delete(@RequestParam String accountCode){
        log.info("delete account,accountCode is {}",accountCode);
        return accountService.deleteAccount(accountCode);
    }
}
```

controller层的代码很丑陋有木有，不过没关系，我们后面会对其进行改造，毕竟写代码是个持续的程，总之一句话，持续关注就对了。

- 接口列表



这里是使用Idea的插件RestfulToolkit进行展示，而且还可以很方便的进行接口测试，也推荐给大家。

至此服务都注册进了注册中心Nacos并且都能对外提供基本的增删改查能力，那么本期的“SpringCloud Alibaba微服务实战 - 服务注册”篇也就该结束啦，咱们下期有缘再见，期待你的关注！