# 今日任务

1. 知道常用RPC框架
2. Eureka实现服务发现和远程调用（难点）
3. 商城首页轮播图实现

# 联合测试

开启商品服务、商城项目，查看首页是否能正确加载所有分类数据。

# 思考？？

商品分类数据，我们是通过前端页面ajax异步加载的，那么广告的轮播图为什么没有使用同样的技术来实现呢？？？

还记得我们提到过的商城项目页面的特点吗？？其中有一条就是利于SEO。

SEO（Search Engine Optimization）：汉译为搜索引擎优化。是一种方式：利用搜索引擎的**规则**提高网站在有关搜索引擎内的自然排名。

搜索引擎要爬别人网站，喜欢爬什么样的网站？？喜欢“**内容经常更新的静态网页**”，而且一般都是网站的首页。

.jsp，.aspx, .php, **.html**

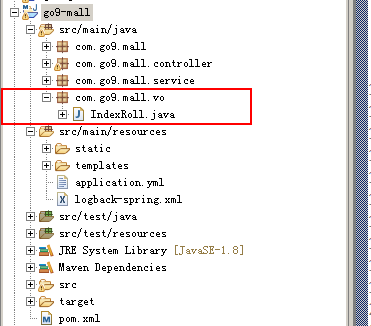
**所以为了便于SEO，我们将经常变化的广告数据，直接放入html页面。这样就要求后端在返回首页的时候，广告数据已经存在了。**

# 代码

Index.html

|  |
| --- |
| <li th:each=*"roll : ${rolls}"*>  <a class=*"box"* th:href=*"@{${roll.link}}"*>  <img th:src=*"@{${roll.image}}"* alt=*""*>  </a>  </li> |

业务层



|  |
| --- |
| **package** com.go9.mall.vo;  **public** **class** IndexRoll {  **private** String title;  **private** String image;  **private** String link;  **public** String getTitle() {  **return** title;  }  **public** **void** setTitle(String title) {  **this**.title = title;  }  **public** String getImage() {  **return** image;  }  **public** **void** setImage(String image) {  **this**.image = image;  }  **public** String getLink() {  **return** link;  }  **public** **void** setLink(String link) {  **this**.link = link;  }    } |

|  |
| --- |
| **package** com.go9.mall.service;  **import** java.util.List;  **import** org.springframework.stereotype.Service;  **import** com.go9.mall.vo.IndexRoll;  @Service  **public** **class** IndexRollServiceImpl **implements** IndexRollService {  @Override  **public** List<IndexRoll> getAll() {  //远程调用商品服务提供的轮播图接口，获取轮播图数据？？？  //其实就是请求http+json的格式的协议接口  **return** **null**;  }  } |

# 远程调用

别忘了，第一次加载首页页面时，分类的根节点数据已经在后台页面中一起返回给浏览器了，这个数据目前是写死的，如何在go8-mall项目中远程调用goods-service中的分类加载服务，动态加载根分类的数据呢？？？？？

为了让大家更清楚的明白问题，还是来总结下吧。如下图：

go8-admin

goods-service

javascript

ajax

http+json

go8-mall

Java

?????

?????

总结：后台管理系统使用的javascript语言通过ajax技术，使用http+json协议，实现了2个完全独立的系统之间的远程调用，同学们，不知不觉中，你们已经完成了一个壮举！！

那么问题来了，go8-mall项目是用java语言写的，如何让一个虚拟机里的java对象方法去调用另一个虚拟机里的java对象的一个方法呢？？？远古的老鸟们是如何解决的呢？？现在的技术是什么呢？？

这个问题呢，如果详细讲，半年都讲不完，下面就简单的带大家回顾下主要的解决方案吧。程序界，把这类问题定义了一个统一的概念：RPC。

# RPC

## 概念

RPC（Remote Procedure Call）：远程过程调用，它是一种通过网络从远程计算机程序上请求服务，而不需要了解底层网络技术的协议。RPC协议假定某些传输协议的存在，如TCP或UDP，为通信程序之间携带信息数据。在OSI网络通信模型中，RPC跨越了传输层和应用层。RPC使得开发包括网络**分布式**多程序在内的应用程序更加容易。

## 类型

|  |  |  |  |
| --- | --- | --- | --- |
| 协议 | 描述 | 优点 | 缺点 |
| RMI | JAVA 远程方法调用、使用原生二进制方式进行序列化 | 简单易用、SDK支持，提高开发效率 | 不支持跨语言 |
| Web Service | 比较早系统调用解决方案 ，跨语言, 其基于WSDL 生成 SOAP 进行消息的传递。 | SDK支持、跨语言 | 实现较重，发布繁琐 |
| Http | 采用http +json 实现 | 简单、轻量、跨语言，夸系统，平台 | 不支持SDK |

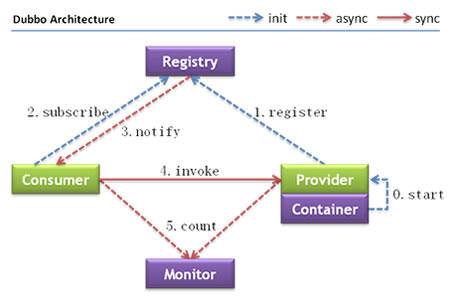
提示：java界大名鼎鼎的**EJB**底层远程调用使用的就是RMI

## 框架

目前企业使用较多的2个框架：

1. **Dubbo**

阿里开源的服务**治理**框架，在国内的互联网公司中享有盛名，不过它就是一个服务治理框架而已，也仅此而已。



1. **Spring cloud**

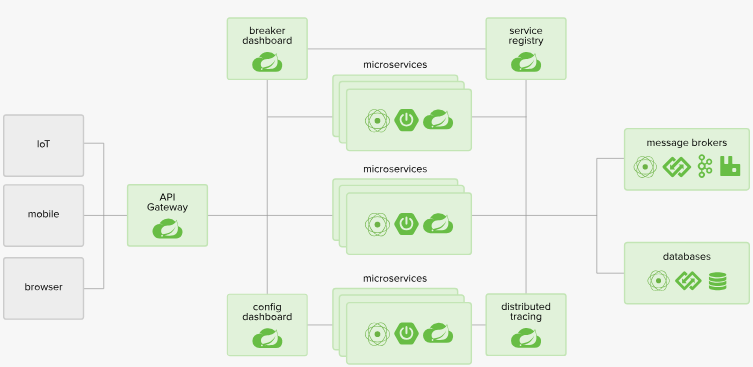
近两年在国内特别火的微服务**全家桶**框架，是spring官方重点推出的开源项目，spring作为java开发的霸主地位更加的牢靠了。



1. 抉择

又到了选择环节，痛苦啊，不过这一次倒是不难选，前几年没得选，因为只有dubbo，虽然阿里14年之后就停止了对开源dubbo的更新（阿里内部使用的dubbo，是另一个样子），但是因为只有一个dubbo，互联网公司还是忍痛在玩，其中玩的比较好的就是由**当当网**工程师自己扩展的**dubbox**。终于spring出手了，spring cloud的出现，让国内互联网公司纷纷从dubbo向cloud迁移，这就是现状。虽然阿里于17年7月宣布恢复对开源dubbo的更新，并在18年2月成为apache基金会孵化项目，但是依然改变不了一个事实，**spring cloud**成为微服务开发的主流技术。

# Eureka



由上图看出，spring cloud是一个生态，一个完整的微服务开发、治理、追踪及维护的全家桶技术。我们本节课先来学习其中的服务治理框架：**Eureka，**用来解决前面的服务之间远程调用的问题。

## 结构图

应用客户端

go8-mall

Eureka 服务端

go8-service-register

Eureka 客户端

goods-service

注册服务

查询服务

调用服务

## 角色

1. 注册中心

在服务治理框架中，通常都会构建一个注册中心，每个服务单元向注册中心登记自己提供的服务，包括服务的主机与端口号、服务版本号、通讯协议等一些附加信息。注册中心按照服务名分类组织服务清单，同时还需要以心跳检测的方式去监测清单中的服务是否可用，若不可用需要从服务清单中剔除，以达到排除故障服务的效果。即上图中的**Eureka 服务端**

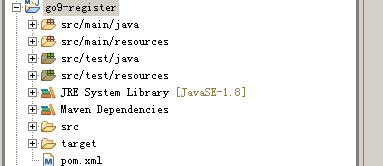
1. Eureka客户端（服务提供者）

Eureka客户端向服务注册中心注册自身提供的服务，并周期性的发送心跳来更新它的服务租约。上图中的 **goods-service**

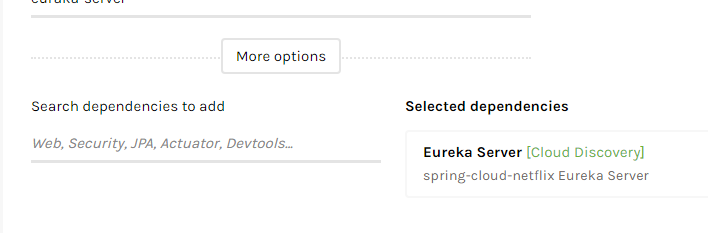
1. Eureka客户端（服务消费者）

从服务端查询当前注册的服务信息把它们缓存到本地，并周期性的刷新服务状态。上图中的 **go8-mall**

# 注册中心



可以使用<https://start.spring.io/>



配置

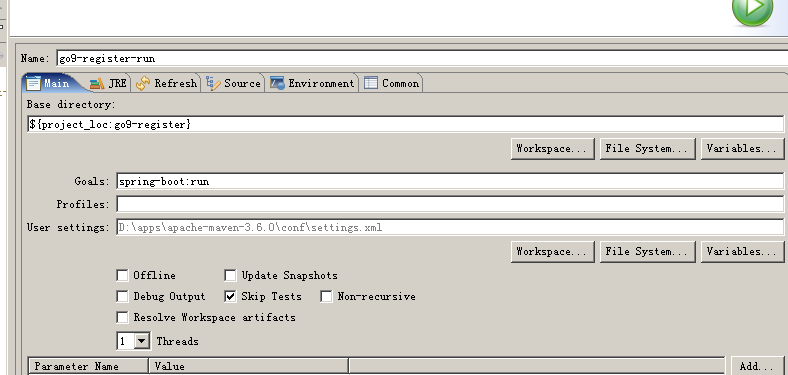
|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.1.3.RELEASE</version>  <relativePath /> <!-- lookup parent from repository -->  </parent>  <groupId>com.go9</groupId>  <artifactId>go9</artifactId>  <version>1.0.0-SNAPSHOT</version>  <packaging>pom</packaging>  <description>java 01 02 班级</description>  <!-- 公用的配置 -->  <modules>  <module>go9-goods-service</module>  <module>go9-common</module>  <module>go9-mall</module>  <module>go9-register</module>  </modules>  <properties>  <java.version>1.8</java.version>  <spring-cloud.version>Greenwich.SR1</spring-cloud.version>  </properties>  <dependencies>  <!-- <dependency> <groupId>org.slf4j</groupId> <artifactId>slf4j-api</artifactId>  <version>1.7.26</version> </dependency> -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  </dependencies>  <dependencyManagement>  <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-dependencies</artifactId>  <version>${spring-cloud.version}</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies>  </dependencyManagement>  <build>  <resources>  <resource>  <directory>src\main\resources</directory>  </resource>  <resource>  <directory>src\main\java</directory>  <includes>  <include>\*\*/\*.xml</include>  </includes>  </resource>  </resources>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </project> |

|  |
| --- |
| **package** com.go9.register;  **import** org.springframework.boot.SpringApplication;  **import** org.springframework.boot.autoconfigure.SpringBootApplication;  **import** org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;  @SpringBootApplication  @EnableEurekaServer  **public** **class** RegisterApplication {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(RegisterApplication.**class**, args);  }  } |

配置文件：

|  |
| --- |
| server:  port: 666  spring:  application:  name: go8-eureka-register  eureka:  server:  enableSelfPreservation: false  instance:  hostname: localhost  client:  registerWithEureka: false  fetchRegistry: false  serviceUrl:  defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/  logging:  level:  root: info  com.netflix.eureka.registry.AbstractInstanceRegistry: error  com.netflix.eureka.cluster.PeerEurekaNodes: error  com.netflix.eureka.registry.PeerAwareInstanceRegistryImpl: error |

启动



# 改造goods-service

略

# 改造go8-mall

略

代码暂无。

# 联合测试

启动服务注册中心，商品服务，商城项目，访问页面数据是否能正确展示。