# 今日任务

1. 熟悉商城项目特点、创建商城项目
2. 理解首页分类加载原理及功能实现（重点）
3. 知道常用RPC框架
4. Eureka实现服务发现和远程调用（难点）
5. **Thymeleaf**入门

# 商城项目

商城项目是整个项目的核心，所有后台的付出，最后都要靠这个商城来展示给用户。像天猫商城，京东商城。那么面向大流量用户群体的商城项目自然和普通的后台支撑系统相比，有明显的不同。

1. 界面设计要更专业，更漂亮
2. 利于搜索、需要进行SEO的优化
3. 页面伪静态化
4. 使用模板真静态化
5. 响应时间有严格要求
6. 经得起大流量的高并发访问
7. 其他。。。。。。。

总结起来一点就是，对技术要求更高了，尤其是搞活动时，面对超高的瞬时并发，依然能保证服务的可用性，是衡量一家公司技术水准的关键指标。阿里的双十一，一次又一次的突破了极限，刷新了我们对技术的认知。也就是只要你技术够牛逼，PC级的服务器，你也能玩出**超级计算机**的感觉。Spring之父Rod Johnson当年在写J2EE指南的时候，是极力推荐轻量级单体结构的。Spring框架当初就是以救世主的身份出现，拯救了所有深陷权威中，使用分布式EJB的java程序员的。唉！！时代真是变了，现在你出去找工作，不吹个分布式你都不好意思讲话，C:\Users\ADMINI~1\AppData\Local\Temp\SGPicFaceTpBq\1336\02F2DFE3.pngC:\Users\ADMINI~1\AppData\Local\Temp\SGPicFaceTpBq\1336\02F2F057.png。

# go8-mall

## 架构图

客户端（应用中心），展示层

商品服务

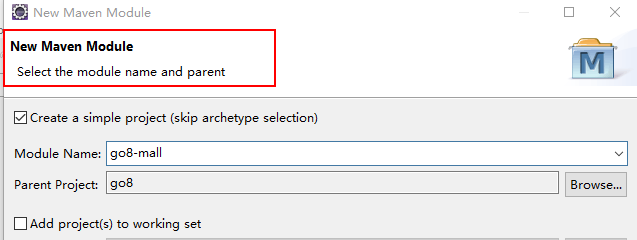
Pc-mall

App-mall

…………….

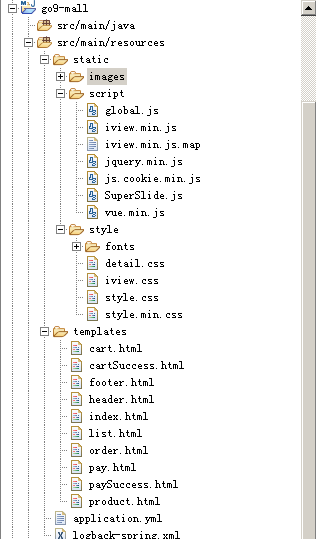
Rpc(调用)

## 创建工程



## 静态资源

见resources文件夹



## 配置项目

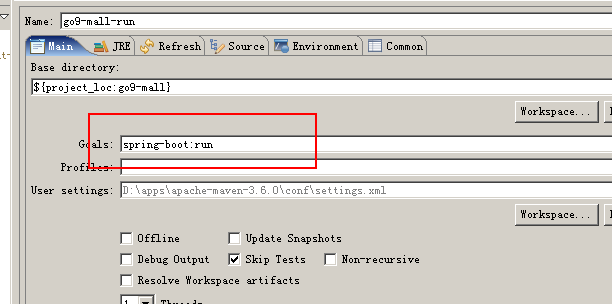
|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <parent>  <groupId>com.go9</groupId>  <artifactId>go9</artifactId>  <version>1.0.0-SNAPSHOT</version>  </parent>  <artifactId>go9-mall</artifactId>  <dependencies>  <dependency>  <groupId>com.go9</groupId>  <artifactId>go9-common</artifactId>  <version>1.0.0-SNAPSHOT</version>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-thymeleaf -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-thymeleaf</artifactId>  </dependency>  <!-- 热部署模块 -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  <optional>true</optional> <!-- 这个需要为 true 热部署才有效 -->  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </project> |

|  |
| --- |
| server:  port: 8888  #eureka:  # client:  # serviceUrl:  # defaultZone: http://localhost:666/eureka/  spring:  application:  name: go9-mall  thymeleaf:  cache: **false**  logging:  level:  root: info  #custom properties  #LOGIN\_URL: http://localhost:3030/login.html |

## 代码

|  |
| --- |
| **package** com.go9.mall.controller;  **import** org.springframework.stereotype.Controller;  **import** org.springframework.web.bind.annotation.RequestMapping;  @Controller  **public** **class** IndexController {  @RequestMapping("/")  **public** String index() {  **return** "index";  }  } |

## 启动项目



访问首页，不考虑数据，如果成功，算是各种资源整合成功。

思考问题：首页的分类数据如何获取呢？

# 分类加载

演示淘宝或者京东商城首页分类的加载过程。得到如下的加载流程：

首页显示默认顶级节点名称

加载所有分类数据

鼠标上移事件

事件

是否是第一次上移

是

是

不是

将已经加载的数据

根据顶级类，

解析相应的数据显示

根据当前鼠标指向的分类显示相应数据

***需要使用ajax跨域请求，需要服务器配合***

***不需要服务器配合，由前端js完成***

***不需要服务器配合，由前端js完成***

# 分类服务

在商品服务项目中，新发布一个供商城调用的加载分类数据的接口，这个接口要一次性完成所有分类数据查询的功能。注意下递归查询就行。

# 接口约定

|  |  |  |  |
| --- | --- | --- | --- |
| **业务类型** | **method** | **url** | **return body** |
| 加载所有分类 | get | catalogs/all?pid=0 | 返回一个统一的json  对象，  {  “code”: xxx,  “message”:”xxxx”,  “data”:{} or []  } |
| xxxxx | xxxxx | xxxxxx |

# 代码

暂无

# 测试

暂无

# 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。

# 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htpp +json 实现 | 简单、轻量、跨语言 | 不支持SDK |

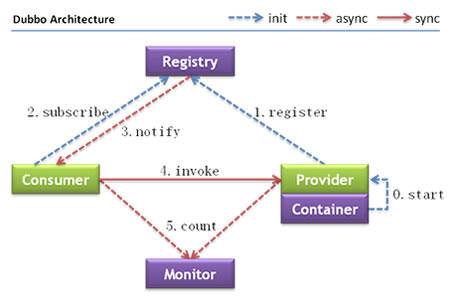
提示：java界大名鼎鼎的EJB底层远程调用使用的就是RMI

## 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

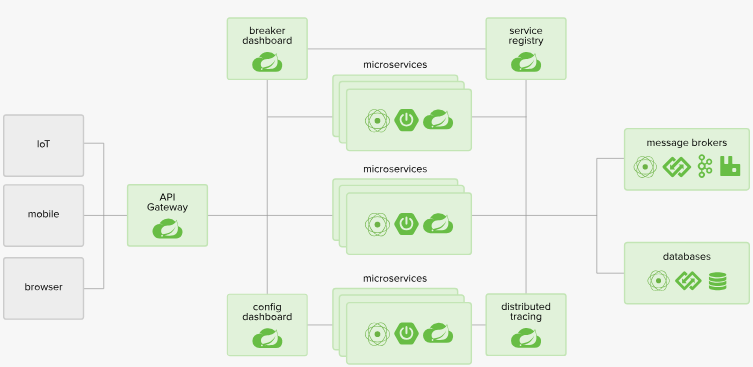
近两年在国内特别火的微服务全家桶框架，是spring官方重点推出的开源项目，spring作为java开发的霸主地位更加的牢靠了。



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

创建项目，暂无。

# 改造goods-service

略

# 改造go8-mall

略

# Thymeleaf

Jsp这种页面模板，已经彻底退出了历史舞台，无论是前后端的彻底分离，还是未分离，都无条件的向“静态化”靠拢，这就是技术的时代性。

简单入门。配合前端代码实现循环输出“顶级类目”到页面的功能。

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。