Gmall

电商简介&Dubbo

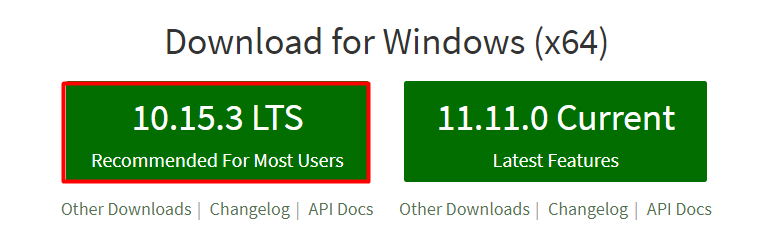
# 一、电商演示项目部署

## 1）、后台管理系统web端（JS+Vue等技术栈）：

进入gmall/project；mall-admin-web.zip 就是前端项目，我们来安装它；

### 1、安装node环境并配置好cnpm

1）、下载node <https://nodejs.org/en/> ；安装并使用node –v检查



2）、配置cnpm <http://npm.taobao.org/> ；并使用cnpm –v 检查

|  |
| --- |
| npm install -g cnpm --registry=https://registry.npm.taobao.org |

### 2、安装并启动后台管理web端

1）、解压；用vscode打开，方便我们查看源码。

并执行以下命令

请先安装cnpm

|  |
| --- |
| npm install -g cnpm --registry=https://registry.npm.taobao.org |

|  |
| --- |
| cnpm install |
| 上一步做完，一般会出现错误，关于node-sass的问题，我们用cnpm解决    cnpm install node-sass  重新 cnpm install 整个项目 |
| npm run dev |

2）、访问指定地址；

http://localhost:8090

## 2）、后台管理系统server端（Java+SpringBoot等技术栈）：

1）、本机导入gmall.sql数据库文件，执行数据库初始化

2）、解压，修改里面的application-dev.properties的内容中的数据库为你自己的数据库访问信息；

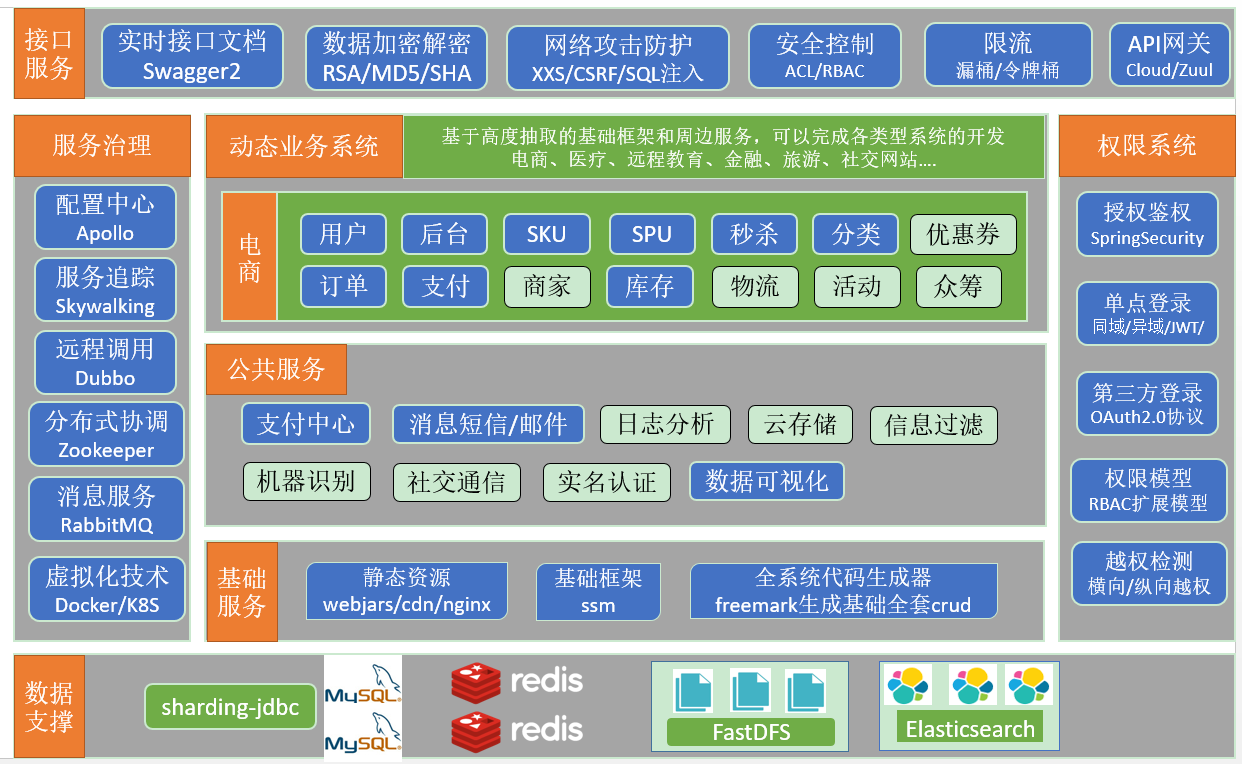
3）、进入cmd，java –jar mall-admin.jar运行

4）、访问前端web 8090项目，测试项目。

5）、也可以访问 localhost:8080/swagger-ui.html 查看接口文档。

# 二、电商的架构

## 1、架构



## 2、整体业务



|  |  |
| --- | --- |
| 首页 | 静态页面，包含了商品分类，搜索栏，商品广告位。 |
| 全文搜索 | 通过搜索栏填入的关键字进行搜索，并列表展示 |
| 分类查询 | 根据首页的商品类目进行查询 |
| 商品详情 | 商品的详细信息展示 |
| 购物车 | 将有购买意向的商品临时存放的地方 |
| 单点登录 | 用户统一登录的管理 |
| 结算 | 将购物车中勾选的商品初始化成要填写的订单 |
| 下单 | 填好的订单提交 |
| 支付服务 | 下单后，用户点击支付，负责对接第三方支付系统。 |
| 订单服务 | 负责确认订单是否付款成功，并对接仓储物流系统。 |
| 仓储物流 | 独立的管理系统，负责商品的库存。（只对接接口不的单独讲解） |
| 后台管理 | 主要维护类目、商品、库存单元、广告位等信息。 |

# 三、电商的难点

分布式&集群，高可用&高并发，DevOps，电商核心业务

# 四、分布式基础概念（复习）

## 1、什么是分布式？

《分布式系统原理与范型》定义：

“分布式系统是若干独立计算机的集合，这些计算机对于用户来说就像单个相关系统”

分布式系统（distributed system）是建立在网络之上的软件系统。

## 2、分布式与集群的关系?

集群指的是将几台服务器集中在一起，实现同一业务。

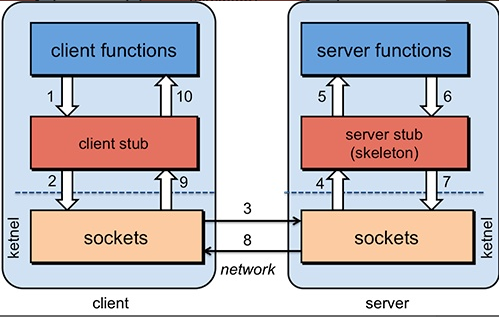
分布式中的每一个节点，都可以做集群。而集群并不一定就是分布式的。

## 3、RPC是什么？

RPC【Remote Procedure Call】是指远程过程调用，是一种**进程间通信方式**，他是一种技术的思想，而不是规范。它允许程序调用另一个地址空间（通常是共享网络的另一台机器上）的过程或函数，而不用程序员显式编码这个远程调用的细节。即程序员无论是调用本地的还是远程的函数，本质上编写的调用代码基本相同。

**他来解决分布式系统的各个服务之间互相交互问题；**

## 4、RPC基本原理？



**RPC两个核心模块：通讯，序列化。**

## 5、CAP？BASE？

### 1）、CAP定理

分布式系统的三个指标：

|  |
| --- |
| ***C****onsistency（一致性）：*  在分布式系统中的所有数据备份，在同一时刻是否同样的值。（写操作之后的读操作，必须返回该值。更新操作执行成功后所有的用户都应该读到最新的值，要求所有的备份数据保持一致）  ***A****vailability（可用性）：*  在集群中一部分节点故障后，集群整体是否还能响应客户端的读写请求。（要求数据需要备份）  ***P****artition tolerance（分区容错）：*  大多数分布式系统都分布在多个子网络。每个子网络就叫做一个区（partition）。分区容错的意思是，区间通信可能失败。 |

CAP理论就是说在分布式存储系统中，最多只能实现上面的两点。而由于当前的网络硬件肯定会出现延迟丢包等问题，所以分区容忍性是我们必须需要实现的。所以我们只能在一致性和可用性之间进行权衡，没有NoSQL系统能同时保证这三点。

CP、AP

### 2）、Consistency 和 Availability 的矛盾

一致性和可用性，为什么不可能同时成立？答案很简单，因为可能通信失败（即出现分区容错）。

Zookeeper如何在分布式系统中实现一致性的raft算法；

<http://thesecretlivesofdata.com/raft/>

### 3）、BASE

BASE：

Basically Available（基本可用）、

Soft state（软状态）

Eventually consistent（最终一致性）

三个短语的简写，BASE是对CAP中一致性和可用性权衡的结果，其来源于对大规模互联网系统分布式实践的结论，是基于CAP定理逐步演化而来的，其核心思想是即使无法做到强一致性（Strong consistency），但每个应用都可以根据自身的业务特点，采用适当的方式来使系统达到最终一致性（Eventual consistency）。接下来我们着重对BASE中的三要素进行详细讲解。

# 五、Dubbo

## 1、简介

Apache Dubbo (incubating) |ˈdʌbəʊ| 是一款高性能、轻量级的开源Java RPC框架，它提供了三大核心能力：面向接口的远程方法调用，智能容错和负载均衡，以及服务自动注册和发现。

官网：

<http://dubbo.apache.org/>

## 2、基本概念



调用关系：

**服务提供者（Provider）**：暴露服务的服务提供方，服务提供者在启动时，向注册中心注册自己提供的服务。

**服务消费者（Consumer）**: 调用远程服务的服务消费方，服务消费者在启动时，向注册中心订阅自己所需的服务，服务消费者，从提供者地址列表中，基于软负载均衡算法，选一台提供者进行调用，如果调用失败，再选另一台调用。

**注册中心（Registry）**：注册中心返回服务提供者地址列表给消费者，如果有变更，注册中心将基于长连接推送变更数据给消费者

**监控中心（Monitor）**：服务消费者和提供者，在内存中累计调用次数和调用时间，定时每分钟发送一次统计数据到监控中心

* 调用关系说明
* 服务容器负责启动，加载，运行服务提供者。
* 服务提供者在启动时，向注册中心注册自己提供的服务。
* 服务消费者在启动时，向注册中心订阅自己所需的服务。
* 注册中心返回服务提供者地址列表给消费者，如果有变更，注册中心将基于长连接推送变更数据给消费者。
* 服务消费者，从提供者地址列表中，基于软负载均衡算法，选一台提供者进行调用，如果调用失败，再选另一台调用。
* 服务消费者和提供者，在内存中累计调用次数和调用时间，定时每分钟发送一次统计数据到监控中心。

## 3、环境搭建

### 1）、Docker安装Zookeeper

### 2）、安装Dubbo的监控管理后台

<https://github.com/apache/incubator-dubbo-admin>

|  |
| --- |
| Production Setup   * Clone source code on develop branch git clone https://github.com/apache/incubator-dubbo-admin.git * **Specify registry address** in   dubbo-admin-server/src/main/resources/application-production.properties   * Build   mvn clean package   * Start   cd dubbo-admin-distribution/target; java -jar dubbo-admin-0.1.jar   * Visit http://localhost:8080 |

## 4、Dubbo-helloworld

### 1）、参照dubbo分包原则。创建api层

<http://dubbo.apache.org/zh-cn/docs/user/best-practice.html>

分包

***建议将服务接口、服务模型、服务异常等均放在 API 包中***，因为服务模型和异常也是 API 的一部分，这样做也符合分包原则：重用发布等价原则(REP)，共同重用原则(CRP)。

### 2）、创建服务提供者（movie）

### 3）、创建服务消费者（user）

### 4）、编写Dubbo的消费者与提供者配置，使他们能远程通信

|  |
| --- |
| 1、pom引入  引入dubbo以及操作Zookeeper的框架  <!-- 引入dubbo -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>dubbo</artifactId>  <version>2.6.2</version>  </dependency>  <!-- 由于我们使用zookeeper作为注册中心，所以需要操作zookeeper  dubbo 2.6以前的版本引入zkclient操作zookeeper  dubbo 2.6及以后的版本引入curator操作zookeeper  下面两个zk客户端根据dubbo版本2选1即可  -->  <dependency>  <groupId>com.101tec</groupId>  <artifactId>zkclient</artifactId>  <version>0.10</version>  </dependency>  <!-- curator-framework -->  <dependency>  <groupId>org.apache.curator</groupId>  <artifactId>curator-framework</artifactId>  <version>2.12.0</version>  </dependency> |
| 2、编写提供者配置文件  <!--当前应用的名字 -->  <dubbo:application name=*"gmall-movie"*></dubbo:application>  <!--指定注册中心的地址 -->  <dubbo:registry address=*"zookeeper://118.24.44.169:2181"* />  <!--使用dubbo协议，将服务暴露在20880端口 -->  <dubbo:protocol name=*"dubbo"* port=*"20880"* />  <!-- 指定需要暴露的服务 -->  <dubbo:service interface=*"com.atguigu.gmall.service.MovieService"* ref=*"movieServiceImpl"* />  启动服务提供者  **public** **static** **void** main(String[] args) **throws** IOException {  ClassPathXmlApplicationContext context =  **new** ClassPathXmlApplicationContext("classpath:spring-beans.xml");  System.***in***.read();  } |
| 3、编写消费者配置文件  <!-- 应用名 -->  <dubbo:application name=*"gmall-user "*></dubbo:application>  <!-- 指定注册中心地址 -->  <dubbo:registry address=*"zookeeper://118.24.44.169:2181"* />  <!-- 生成远程服务代理，可以和本地bean一样使用demoService -->  <dubbo:reference id=*"movieService"*  interface=*"com.atguigu.gmall.service.MovieService"*>  </dubbo:reference> |

### 5）、测试使用Dubbo进行RPC

## 5、SpringBoot整合

<https://github.com/apache/incubator-dubbo-spring-boot-project/tree/0.2.1>

### 1、引入spring-boot-starter（包含dubbo和curator的依赖）

<dependency>

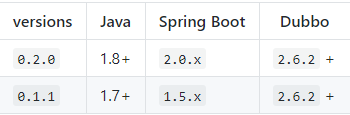
<groupId>com.alibaba.boot</groupId>

<artifactId>dubbo-spring-boot-starter</artifactId>

<version>0.2.0</version>

</dependency>

注意starter版本适配：



### 2、配置application.properties

#### 提供者配置：

**dubbo.application.name**=**gmall-user  
dubbo.registry.protocol**=**zookeeper  
dubbo.registry.address**=**192.168.67.159:2181  
dubbo.scan.base-package**=**com.atguigu.gmall  
dubbo.protocol.name**=**dubbo**

application.name就是服务名，不能跟别的dubbo提供端重复

registry.protocol 是指定注册中心协议

registry.address 是注册中心的地址加端口号

protocol.name 是分布式固定是dubbo,不要改。

base-package 注解方式要扫描的包

#### 消费者配置：

**dubbo.application.name=gmall-order-web  
dubbo.registry.protocol**=**zookeeper  
dubbo.registry.address**=**192.168.67.159:2181  
dubbo.scan.base-package**=**com.atguigu.gmall  
dubbo.protocol.name**=**dubbo**

#### 使用注解

@Service、@Reference

**【如果没有在配置中写dubbo.scan.base-package,还需要使用@EnableDubbo注解】**

## 6、整合hystrix

Hystrix 旨在通过控制那些访问远程系统、服务和第三方库的节点，从而对延迟和故障提供更强大的容错能力。Hystrix具备拥有回退机制和断路器功能的线程和信号隔离，请求缓存和请求打包，以及监控和配置等功能

#### 1、配置spring-cloud-starter-netflix-hystrix

spring boot官方提供了对hystrix的集成，直接在pom.xml里加入依赖：

|  |
| --- |
| <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>  </dependency>  <dependencyManagement>  <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-dependencies</artifactId>  <version>Finchley.SR1</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies>  </dependencyManagement> |

然后在Application类上增加@EnableHystrix来启用hystrix starter：

|  |
| --- |
| @SpringBootApplication  @EnableHystrix  public class ProviderApplication { |

#### 2、配置Provider端

在Dubbo的Provider上增加@HystrixCommand配置，这样子调用就会经过Hystrix代理。

|  |
| --- |
| @Service(version = "1.0.0")  public class HelloServiceImpl implements HelloService {  @HystrixCommand(commandProperties = {  @HystrixProperty(name = "circuitBreaker.requestVolumeThreshold", value = "10"),  @HystrixProperty(name = "execution.isolation.thread.timeoutInMilliseconds", value = "2000") })  @Override  public String sayHello(String name) {  // System.out.println("async provider received: " + name);  // return "annotation: hello, " + name;  throw new RuntimeException("Exception to show hystrix enabled.");  }  } |

#### 3、配置Consumer端

对于Consumer端，则可以增加一层method调用，并在method上配置@HystrixCommand。当调用出错时，会走到fallbackMethod = "reliable"的调用里。

|  |
| --- |
| @Reference(version = "1.0.0")  private HelloService demoService;  @HystrixCommand(fallbackMethod = "reliable")  public String doSayHello(String name) {  return demoService.sayHello(name);  }  public String reliable(String name) {  return "hystrix fallback value";  } |

# 六、创建项目模块

gmall-parent[pom]

🡪gmall-common（通用模块）【工具类，通用依赖】（jar）

🡪gmall-api（dubbo分包建议，接口层&Model层抽取）（jar）

🡪gmall-cms（Content Manage System：内容管理系统[文章、评论等]）[service]

🡪gmall-pms（Product Manage System：商品管理系统）[service]

🡪gmall-oms（Order Manage System：订单管理系统）[service]

🡪gmall-wms（Warehouse Manage System：仓库管理系统[库存，出入库等]）[ service]

🡪gmall-ums（User Manage System：用户管理系统[会员/管理员、登陆、注销等]）[service]

🡪gmall-sms（Sale Manage System：营销管理系统[优惠券、活动等]）[service]

🡪gmall-admin-web（后台管理web-restapi层，对接前端的Vue项目）[boot-web]

==========================为后台提供服务的系统===============================

🡪gmall-list（商品列表系统）[boot- service]

🡪gmall-cart（购物车系统）[boot-service]

🡪gmall-search（检索系统）[boot- service]

🡪gmall-cas（中央认证系统）[boot- service]

🡪gmall-order（订单系统）[boot- service]

🡪**gmall-ware（库存系统）**[boot- service]

🡪gmall-seckill（秒杀系统）[boot- service]

========================为前端提供服务的系统===============================

🡪gmall-portal-web（前端web RestAPI层，对接app以及web）[boot-web]