# 技术

Springmvc

Springbooot:简化xml配置（mybatis的sql文件除外），提高开发效率

Springcloud:微服务框架，RPC远程调用框架(仅遵循http协议+restful)

Dubbo:分布式服务治理框架，RPC远程调用框架(遵循多种协议)

Zookpeer:分布订阅，事件通知，配置管理，分布式锁

Mq:消息中间件(点对点 p2p一对一 和发布订阅 一对多)

Redis:nosql缓存数据库

Nginx:反向代理工具，负载均衡

# 传统项目与电商项目

传统项目：

缺点：容易单点故障，抗压能力差。（成千上万人访问，容易宕机）

优点：适合小团队

电商项目：

分布式：根据业务需求进行拆分成N个子系统，多个子系统相互协作才能完成业务流程子系统之间通讯使用RPC远程通讯技术。

集群：将一个项目部署到多个tomcat上。（集群好处：解决高并发。注意：session共享问题。怎么做集群：nginx，F5（硬件），lvs，服务框架集群dubbo,springcloud,rebin）

# 电商项目考虑哪些问题

支持移动端（android,ios），H5端，PC端，微信端

分布式，高可用，高并发，安全

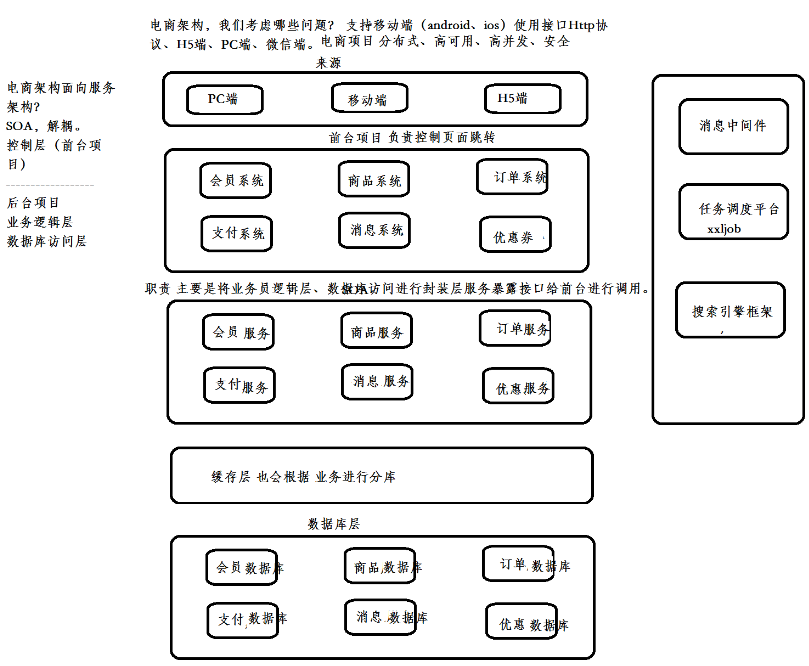
面向服务SOA。（传统项目分三层：控制层，业务逻辑层，数据访问层；电商项目面向服务：分前台项目（控制层），后台项目（业务逻辑层，数据访问层）。即后台项目对外暴露接口，前台项目调用接口）（SOA职责：主要是将业务逻辑层，数据访问层进行封装成服务暴露接口给前台进行调用）

# 分库分表

垂直拆分：按照模块进行划分数据库。

水平拆分：在同一个数据库，把一张表，拆分成很多表。

# 电商项目架构



# 项目构建层次结构

|  |
| --- |
| itmayiedu-shopp-parent  ├─itmayiedu-shopp-common // 抽取的公共模块  ├─itmayiedu-shopp-eurekaserver // eureka注册中心  ├─itmayiedu-shopp-api // api服务  ├─itmayiedu-shopp-member-api // 会员服务  ├─itmayiedu-shopp-order-api // 订单服务  ├─itmayiedu-shopp-goods-api // 商品服务  ├─itmayiedu-shopp-pay-api // 支付服务  ├─itmayiedu-shopp-member // 会员系统  ├─itmayiedu-shopp-nessage // 消息系统  ├─itmayiedu-shopp-mobile-web // h5端工程 |

# itmayiedu-shopp-parent

|  |
| --- |
| **<parent>**  **<groupId>org.springframework.boot</groupId>**  **<artifactId>spring-boot-starter-parent</artifactId>**  **<version>1.5.3.RELEASE</version>**  **</parent>**  **<properties>**  **<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>**  **<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>**  **<java.version>1.8</java.version>**  **</properties>**  **<dependencies>**  **<!-- 集成commons工具类 -->**  **<dependency>**  **<groupId>org.apache.commons</groupId>**  **<artifactId>commons-lang3</artifactId>**  **<version>3.4</version>**  **</dependency>**  **<!-- 集成lombok 框架 -->**  **<dependency>**  **<groupId>org.projectlombok</groupId>**  **<artifactId>lombok</artifactId>**  **</dependency>**  **<!-- 集成redis -->**  **<dependency>**  **<groupId>org.springframework.boot</groupId>**  **<artifactId>spring-boot-starter-data-redis</artifactId>**  **</dependency>**  **<!-- 集成aop -->**  **<dependency>**  **<groupId>org.springframework.boot</groupId>**  **<artifactId>spring-boot-starter-aop</artifactId>**  **</dependency>**  **<!-- 集成web-->**  **<dependency>**  **<groupId>org.springframework.boot</groupId>**  **<artifactId>spring-boot-starter-web</artifactId>**  **</dependency>**  **<!-- springboot整合activemq -->**  **<dependency>**  **<groupId>org.springframework.boot</groupId>**  **<artifactId>spring-boot-starter-activemq</artifactId>**  **</dependency>**  **<!-- 集成发送邮件-->**  **<dependency>**  **<groupId>org.springframework.boot</groupId>**  **<artifactId>spring-boot-starter-mail</artifactId>**  **</dependency>**  **<!-- 集成mysql -->**  **<dependency>**  **<groupId>mysql</groupId>**  **<artifactId>mysql-connector-java</artifactId>**  **</dependency>**  **<dependency>**  **<groupId>org.springframework.boot</groupId>**  **<artifactId>spring-boot-starter-test</artifactId>**  **<scope>test</scope>**  **</dependency>**  **<dependency>**  **<groupId>org.mybatis</groupId>**  **<artifactId>mybatis</artifactId>**  **<version>3.4.0</version>**  **</dependency>**  **<dependency>**  **<groupId>tk.mybatis</groupId>**  **<artifactId>mapper</artifactId>**  **<version>3.3.7</version>**  **</dependency>**  **<!-- 阿里巴巴数据源 -->**  **<dependency>**  **<groupId>com.alibaba</groupId>**  **<artifactId>druid</artifactId>**  **<version>1.0.14</version>**  **</dependency>**  **<!-- httpclient -->**  **<dependency>**  **<groupId>commons-httpclient</groupId>**  **<artifactId>commons-httpclient</artifactId>**  **<version>3.1</version>**  **</dependency>**  **<dependency>**  **<groupId>org.apache.httpcomponents</groupId>**  **<artifactId>httpclient</artifactId>**  **</dependency>**  **<dependency>**  **<groupId>com.alibaba</groupId>**  **<artifactId>fastjson</artifactId>**  **<version>1.2.30</version>**  **</dependency>**  **<dependency>**  **<groupId>junit</groupId>**  **<artifactId>junit</artifactId>**  **</dependency>**  **<dependency>**  **<groupId>org.springframework</groupId>**  **<artifactId>spring-context-support</artifactId>**  **</dependency>**  **<dependency>**  **<groupId>commons-net</groupId>**  **<artifactId>commons-net</artifactId>**  **<version>3.3</version>**  **</dependency>**  **<dependency>**  **<groupId>org.springframework.cloud</groupId>**  **<artifactId>spring-cloud-starter-eureka-server</artifactId>**  **</dependency>**  **<dependency>**  **<groupId>org.springframework.cloud</groupId>**  **<artifactId>spring-cloud-starter-feign</artifactId>**  **</dependency>**  **</dependencies>**  **<dependencyManagement>**  **<dependencies>**  **<dependency>**  **<groupId>org.springframework.cloud</groupId>**  **<artifactId>spring-cloud-dependencies</artifactId>**  **<version>Dalston.RC1</version>**  **<type>pom</type>**  **<scope>import</scope>**  **</dependency>**  **</dependencies>**  **</dependencyManagement>**  **<build>**  **<plugins>**  **<plugin>**  **<groupId>org.springframework.boot</groupId>**  **<artifactId>spring-boot-maven-plugin</artifactId>**  **</plugin>**  **</plugins>**  **</build>**  **<repositories>**  **<repository>**  **<id>spring-milestones</id>**  **<name>Spring Milestones</name>**  **<url>https://repo.spring.io/milestone</url>**  **<snapshots>**  **<enabled>false</enabled>**  **</snapshots>**  **</repository>**  **</repositories>** |

# 创建Eureka服务注册中心项目

第一步：主程序

@SpringBootApplication

@EnableEurekaServer

**public** **class** EurekaServer {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(EurekaServer.**class**, args);

}

}

第二步：application.yml配置文件

|  |
| --- |
| **server:**  **port: 8761**  **eureka:**  **instance:**  **hostname: localhost** #eureka服务端的实例名称  **client:**  **registerWithEureka: false** #false表示不向注册中心注册自己  **fetchRegistry: false** #false表示自己端就是注册中心，我的职责就是维护服务实例，并不需要去检索服务  **serviceUrl:**  **defaultZone: <http://${eureka.instance.hostname}:${server.port}/eureka/>**  #设置与Eureka Server交互的地址查询服务和注册服务都需要依赖这个地址 |