1. IDEA中提交项目到GitHub：选中整个项目而不是module去add，然后commit，commit后只是提交到本地git仓库，然后再push才是提交到远程GitHub仓库。
2. 修改服务器的IP地址(**centos7**)： vim /etc/sysconfig/network-scripts/ifcfg-ens33，配置完成后需要重启网络服务：service network restart。
3. 通用mapper的使用(**tk.mybatis.mapper**)：在pom.xml文件中配置通用mapper的依赖，在启动类上添加@MapperScan注解去扫描指定包下的mapper文件，然后在application.properties配置文件中配置数据库的url、用户名、密码、以及控制台打印的日志级别。通用mapper的接口文件中，要继承tk.mybatis.mapper的mapper接口，接口的泛型是需要的实体类。
4. 包名或项目为bean、pojo、entity、domain、vo、dto的都为实体类，实体类的字段与数据库中表的字段要一一对应，属性命名时采用驼峰命名规则，使用@Id注解标注的属性表示数据库表中的主键，@Column标注的属性表示表中的普通字段，有@GeneratedValue注解的属性，表示获取主键自增，@GeneratedValue(strategy = GenerationType.*IDENTITY*)表示获取MySQL的主键自增，@GeneratedValue(strategy = GenerationType.AUTO)表示获取Oracle的主键自增。
5. 普通接口的实现类要标注spring的@Service注解，还要使用spring的@Autowired注解注入mapper或其它，去调用数据访问层操作数据库，即服务层调用数据访问层。@Autowired即自动注入，从spring容器中获取。
6. 控制器要标注spring的@Controller或@RestController表明这是一个控制器，@RestController等价于@Controller和@ResponseBody两个注解，使用@ResponseBody注解可以返回json字符串，也可以将数据直接显示到页面。控制器还要使用@Autowired注入服务层接口，调用服务层。
7. 垂直应用架构中controller、service接口、接口的实现都在同一个工程， 在controller和接口的实现使用@Autowired注入就可以实现每一层之间的相互调用。
8. 分布式应用架构：将各个应用通过分层独立出来，可以利用RPC实现controller与service、service与service之间的互相调用，提高了代码的复用性。分布式架构隶属于不同的工程，可以使用RPC、Dubbo、spring cloud实现工程之间的相互调用。

SOA分布式架构：由于服务节点越来越多，服务之间的调用和服务之间的依赖越来越复杂，，所以使用注册中心来调度、路由、管理所有的服务，注册中心可以使用Dubbo官网推荐的zookeeper。

1. Dubbo的使用分为服务提供者和服务消费者，服务提供者相当于service的实现类，只需要在实现类上使用阿里的@Service注解，同时在对用的application.properties配置文件中配置服务名、注册中心协议、注册中心的地址和端口号、注解方式要扫描的包、协议名称dubbo，服务消费者相当于controller，同时在对应的application.properties配置文件中配置服务名、注册中心的地址及端口号、要扫描的包、协议名称dubbo，还要配置服务消费者访问服务提供者的超时时间、启动消费者时是否检查服务提供者能够正常访问，如果配置为true，必须保证服务提供者正常，否则接口无法注入，此处配置为false。服务消费者使用只需要在控制器中把原来spring的@Autowired注解换为@Reference注解，即可实现不同模块的调用。
2. 线上部署vue页面：在vue项目所在路径使用npm run build 命令进行编译，生成一个dist文件夹，将此文件夹放入nginx根目录做一个静态代理，然后修改nginx的配置文件指定即可。
3. 整体业务简介：首页、分类查询、全文搜索、商品详情、后台管理、购物车、单点等登录、支付、订单等模块。
4. 性能优化：一是提高数据库SQL的性能，首先是优化SQL，包括使用索引，减少不必要的表关联，还可以分库分表；二是尽量避免直接查询数据库，使用缓存，此项目使用redis。在项目中redis的使用需要在application.properties配置文件中配置redis的主机地址、端口号、适应哦·使用的数据库号、访问的超时时间。
5. token是由JWT工具生成的，包括三部分，公共部分，私有部分和签名部分，公共部分包括加密算法、格式类型过期时间等，私有部分包括用户id、用户昵称、用户登录的IP地址等，签名部分包括服务器的IP地址，即salt值，主要用于JWT生成字符串的时候进行加密，因为salt值不同。三部分最终经过base64编码生成token。
6. 检查用户是否需要登录，需要把cookie中的token和当前登录人的IP发送给远程服务器进行登录验证，返回结果为true还是false。
7. spring boot1.x与spring boot2.x版本的区别：spring boot1.x遵循spring4.0的规则，因为Spring与springMVC为父子容器关系，所以springMVC的版本也为4.0。启动时控制台会显示自带的error控制器，自带的tomcat版本为8，对thymeleaf支持有变化，html标签必须为双标签。spring boot2.x遵循spring5.0的规则，因为Spring与springMVC为父子容器关系，所以springMVC的版本也为5.0。启动时控制台不会显示自带的error控制器，自带的tomcat版本为9。
8. spring4.0与spring5.0的区别：spring4.0：拦截器实现方式有：继承和实现两种方式。spring5.0：拦截器实现方式有：要求JDK版本最低为Java8，支持JUnit5并且兼容JUnit4，支持函数式编程。
9. 功能开发流程：第一步(数据访问层)：从数据库中找对应的表，创建对应的bean，然后写interface，再写mapper(使用通用mapper可以把interface和mapper合并)；第二步(服务层)：创建service与serviceImpl ；第三步(控制层(承载数据模型与页面渲染))：创建控制器controller。
10. 通用mapper的使用：1. 添加通用mapper的坐标，添加依赖；2. 创建实体类；3. 创建接口并继承通用mapper(泛型为实体类)；4. 在启动类添加mapper扫描；5. 编写服务层及实现类，实现类调用mapper；6. 编写控制器，控制器调用服务层；7. 配置数据源；8. 测试。
11. 通用mapper的用法：select(UserInfo userInfo) 相当于属性的等值查询，selectOne(UserInfo userInfo)登录，selectAll()查询所有，selectByExample(Object example) 查询范围数据或等值数据，selectByPrimaryKey() 根据主键查询，insertSelective(userInfo) 选择性的添加，delete(UserInfo userInfo) 等值删除，deleteByPrimaryKey(Object key) 根据主键删除，deleteByExample(Object example) 范围或等值删除，updateByExampleSelective(userInfo, example) 根据条件修改(第一个参数表示要修改的数据，第二个参数表示修改的条件)。
12. 使用debug模式下的热部署功能，必须在配置文件中配置thymeleaf的缓存关闭。多表操作除了要编写xml文件，还要开启事务。redisson工具可以设置分布式锁，redisson底层采用的是netty框架。
13. 用户点击销售属性值切换商品的思路：**第一种方法**：第一步：先获取到页面传递过来的销售属性值id({"123|126"})，组成一个json字符串( {"123|126|131":"37", "124|127|132":"30"...} )，第二步：后台页面做一个map集合，然后把map集合也转换为json字符串，key=销售属性值id拼接而成，value=skuId，map.put("123|126","37")，JSON.toJSONString(map);，第三步：由页面选择的销售属性值id组成key，与后台拼接好的字符串组合。**第二种方法**：通过SQL语句实现拼接，使用GROUP\_CONCAT函数。
14. 通用mapper多表关联的xml文件： [gmall-manage-service\src\main\resources\mapper\BaseAttrInfoMapper.xml](gmall-manage-service/src/main/resources/mapper/BaseAttrInfoMapper.xml)。springMVC的四大组件：核心控制器(DispatcherServlet)、适配器(HandlerAdapter)、映射器(@RequestMapping())、视图解析器。thymeleaf标签：th:text 页面显示的文本，th:value 表示如果有提交的时候，value是提交的数据。th:utext 表示解析文本样式(全文检索时高亮使用)。
15. 1.服务启动后，后台页面无法显示数据，是因为没有跨域，可以在控制器添加跨域注解@CrossOrigin。(A项目直接访问B项目的控制器，称之为跨域)。页面与后台数据交互：在后台通常以对象的形式接收数据，在平台属性对象中存储一个平台属性值的集合属性，传递数据的时候，把平台属性对象直接传递到后台即可。平台属性管理：保存的时候，前台页面传递过来的是json数据，在后台使用@RequestBody来接收数据，将其转换为java对象。
16. 1.fastDFS：使用C语言编写，由阿里巴巴的余庆独立开发，可以使用在Linux系统上，存储文件大小在4K-500M之间。文件上传流程：client发起上传文件的请求，tracker获取上传信息，并查询storage是否有可存储的空间，如果有，则返回IP和端口，storage生成文件id，将文件写入磁盘。fastDFS结合项目必须使用nginx做代理文件服务器。
17. Git提交文件时必须先add再commit后然后push。未add时，在IDEA中文件名为红色的，add后，文件名为绿色的。commit后文件名变为白色，表示最近修改的文件。第一次commit新文件时，不会提示修改位置，再次commit时才会显示修改文件的位置，并且显示修改前与修改后的差别。在未commit前修改文件后文件名变为蓝色，commit后变为白色。
18. redis：redis是一个开源的、用C语言写的高性能的分布式内存数据库，基于内存运行，支持持久化，即可以将内存中的数据保存到磁盘中，重启的时候可以再次加载使用。redis支持数据的备份，即主从模式的数据备份。redis支持异步持久化，即将数据写到磁盘上的同时不影响服务的正常运行。redis默认有16个库(0-15)，使用统一的密码管理，默认使用0号库，默认端口为6379。redis服务启动后，可以使用redis-benchmark测试机器性能，redis-check-aof：修复有问题的AOF文件，redis-check-dump：修复有问题的dump.rdb文件，redis-cli：客户端操作入口，redis-cli shutdown 单实例关闭，redis-cli -p 6379 shutdown 多实例关闭，指定端口关闭。
19. redis命令(没有分号)：select 库编号 切换数据库；dbsize 查看当前库的key的数量；flushdb 清空当前库；flushall 清空所有库；keys \* 查询当前库的所有数据；exists key的名字，判断某个key是否存在；expire key 秒：为给定的key设置过期时间为多少秒；ttl key 查看key还有多少秒过期，-1表示永不过期，-2表示已过期；type key 查看key是什么类型；del key 删除这个键。
20. redis的五种数据类型及使用场景(redis5.0后增加了一个stream数据类型)：String：最基本的类型，一个key对应一个value，是二进制安全的，可以存储任何数据，value最多可以是512M。一般使用时只存储常量、变量；List：简单的字符串列表，按照插入顺序排序，底层是双向链表。可以用于做简单的队列处理；Set：无序的集合。一般用于获取两个数据集的交集、补集、差集；Hash(类似于java的Map)：是一个键值对集合，value还是一个键值对。一般用于存用户对象(便于修改)；Zset：有序的集合，不允许重复的元素，每个元素都会关联一个double类型的分数，但是分数可以重复。一般用于排序。
21. 错误笔记：**1.** 启动类中的@MapperScan指定到包名即可，不要指定到包下面的文件，否则会报……that could not be found.错误。**2.** 使用通用mapper等框架时，需要在启动类添加@MapperScan扫描mapper包，有作为配置文件的类时，需要在启动类添加@ComponentScan注解指定包名，把配置文件交给spring容器管理，否则都会报not be found异常。**3.** 克隆FastDFS项目时，需要在GitHub上打开源码，从GitHub上复制项目地址，末尾以.git，结尾，否则测试文件上传时，会出现无法解析getConnection()方法，网络差时也会丢失getConnection()方法，项目的文件上传功能不可用，需要复制源码解决。**4.** 查询一个数据库中不存在的商品时，页面出现500的错误并显示错误信息，造成了缓存穿透。**5.** spring boot的测试类上需要添加@RunWith(SpringRunner.class)注解，否则会导致空指针异常，因为没有读取到配置文件。**6.** 多表关联查询时，需要在配置文件中配置mybatis.mapper-locations=classpath:mapper/\*Mapper.xml和

mybatis.configuration.mapUnderscoreToCamelCase=true，否则会报ibatis.binding.BindingException异常。

1. Spring boot与maven的区别：spring boot整合了spring与springMVC，自带tomcat。创建spring boot项目时artifact不能大写，必须全部小写。maven只有一个空的pom.xml，需要依赖时必须手动添加依赖。maven的主要作用是：依赖，聚合(pom)，继承，聚合下有jar、war。只需要bean和接口的时候，创建maven项目即可。
2. 线上部署vue页面：将vue的页面进行编译(npm run build命令 生成dist文件夹)，将dist文件夹放入nginx(或者在nginx中创建一个文件夹，将dist文件夹内的东西放入创建的文件夹内)，做一个静态代理，然后修改nginx.conf，最后修改hosts文件(IP地址 域名)。
3. Spring boot核心注解：@SpringBootApplication：spring boot的启动配置注解，此注解的源码中的@EnableAutoConfiguration为自动配置自动装配功能。@SpringBootApplication源码中的注解有：@Target({ElementType.TYPE}) @Retention(RetentionPolicy.RUNTIME) @Documented @Inherited @SpringBootConfiguration @EnableAutoConfiguration @ComponentScan。
4. spring(不重复造轮子)：IOC：IOC不是一种技术，只是一种思想，一个重要的面向对象编程的法则，它能指导我们如何设计出松耦合、更优良的程序。传统应用程序都是由我们在类内部主动创建依赖对象，从而导致类与类之间高耦合，难于测试；有了IOC容器后，把创建和查找依赖对象的控制权交给了容器，由容器进行注入组合对象，所以对象与对象之间是 松散耦合，这样也方便测试，利于功能复用，更重要的是使得程序的整个体系结构变得非常灵活。AOP的使用场景：日志(记录用户的行为、商品浏览的次数、后台用户的操作等)、事务。
5. 项目注意：项目上线之前一定要做压力测试。@PropertySource("classpath:alipay.properties")，spring的注解，指定当前类读取哪个配置文件。一个类上添加了@Configuration注解，等价于xml文件。项目的后台管理系统使用的vue前后端分离技术，而项目的前台系统使用thymeleaf技术渲染。首页使用nginx做静态代理，将首页资源放到nginx配置文件中指定的文件夹内，并配置域名，配置文件中的upstream表示代理一个服务器。final是用来定义常量的，finally是用来处理异常的关键字。实体类必须序列化而且提供get和set方法。添加事务(启动类添加@EnableTransactionManagement注解，并且在实现类添加@Transactional注解)。包名为bean、pojo、entity、domain、vo、dto的，里面存放的都是实体类。service和controller在同一个项目中可以使用@Autowired注解相互调用，在不同项目中使用RPC框架调用。mapper通常都写在服务层(gmall-user-manage、gmall-xxx-service)而不把他写在控制层(gmall-xxx-web)。gmall-xxx-web通常只写控制器。确保服务提供者能够发送到注册中心可以通过dubbo的监控中心查看，也可以使用zookeeper的客户端查看。跨域方案：@CrossOrigin、httpClient(restful)、jsonp等。事务分为两种：声明式事务、编程式事务。thymeleaf：spring boot官方推荐使用thymeleaf，是一种前端模板技术，vue是一种前端框架，thymeleaf可以不依赖tomcat服务器运行。RPC：远程过程调用(web应用与服务器不在同一个网络之内，不同服务之间相互调用),可以使用nginx中的IPhash算法映射出服务器的位置。单一架构(访问量小)：JSP + Servlet + Tomcat。 mybatis(半自动的ORM框架，手动编写sql语句)，mybatis底层是对jdbc的高度封装，封装了connection、PrepareStatement、recourseSet，并且配置了数据源和连接池，通过xml映射实现，替换了单一架构的DBUtils工具类。springMVC可以与spring无缝整合，是spring的子容器，而且相对于struct2安全性好。项目架构采用SOA架构，使用dubbo做远程过程调用，使用zookeeper做注册中心。以bean、pojo、entity、domain命名的包中存放的都为实体类。
6. Dubbo：dubbo本质是一个jar包，是一个应用程序。dubbo的网络传输层使用netty(基于事件驱动的异步网络应用框架)框架传输数据。dubbo官网推荐使用zookeeper做注册中心。spring cloud中可以把nacos作为注册中心替换eureka。
7. Dubbo如何工作：把provider加载到容器中由spring管理，把provider注册到注册中心上，consumer从注册中心异步的获取到provider的地址，consumer得到地址后去同步的调用provider。
8. Dubbo如何使用：两个注解，两个配置。(接口实现类使用@Service注解暴露服务提供者，application.properties配置文件中添加共享配置(服务名称、协议、端口等)，接口使用@Reference注解指定服务消费者，application.properties配置文件中添加共享配置)。 dubbo中建议在provider端配置的consumer端属性有：timeout：方法调用的超时时间；retries：失败重试次数，缺省是2；loadbalance：负载均衡算法(dubbo为什么可以实现调度和治理服务的原因)，缺省是随机random，还可以配置轮询、最不活跃优先和一致性哈希等；actives：消费者端的最大并发调用限制。dubbo在provider端配置的provider端属性有：threads：服务线程池大小；executes：一个服务提供者并行执行请求上限，即当provider对一个服务的并发调用达到上限后，新调用会阻塞，此时consumer可能会超时。
9. 把项目从eclipse迁移到IDEA：**第一步**：在application.properties中配置以下内容：spring.mvc.view.suffix=.jsp，spring.mvc.view.prefix=/WEB-INF/jsp/。**第二步：**在pom.xml中添加jsp的依赖。
10. 如何搭建dubbo环境：安装jdk：安装之前卸载掉原有的jdk。解压jdk的压缩包，修改profile文件(vim /etc/profile),刷新文件(source /etc/profile)。安装zookeeper：导入压缩包并解压，复制zoo.cfg(cp zoo\_sample.cfg zoo.cfg),然后启动zookeeper(./zkServer.sh start)。测试(ps -ef |grep zookeeper)。安装监控中心：dubbo-admin-2.6.0.war就是监控中心(本质是一个动态web工程，war包)，运行必须依赖tomcat服务器。，启动tomcat，然后将dubbo-admin-2.6.0.war放在webapps目录中，然后关闭tomcat，并删除dubbo-admin-2.6.0.war包。再重启tomcat。访问监控中心(服务器的IP:8080/dubbo-admin,默认用户名为：root，密码为：root)，此项目的dubbo版本为2.6.0。
11. 商品的基本知识：商品分类：前台的分类数据信息可以从json文件获取，后台管理的分类信息从数据库获取。对应的数据库表：baseCatalog1，baseCatalog2，baseCatalog3(不设计为一张表是因为项目为互联网项目，速度要快)。平台属性与平台属性值为1：n的关系，用于过滤。表结构(与表baseCatalog3相关)：baseAttrInfo：baseAttrValue。每个商品的每种属性都有对应的属性值。销售属性在商品详情页面，销售属性与销售属性值为1：n的关系，用于给用户提供商品选择。例如：销售属性：颜色 销售属性值：幻影蓝、幻夜黑、冰岛白、蓝水翡翠(四组图片来自于SPU图片)。SPU：一组可复用、易检索的标准化信息集合。(例如：手机、笔记本、小米、华为、锤子(品牌)手机、小米、华为笔记本)。SKU：每种商品均对应有唯一的编号。(例如：Apple iPhone 11 (A2223) 128G 红色 移动联通电信4G手机 双卡双待、荣耀MagicBook 2019 英特尔酷睿i5 14英寸轻薄笔记本电脑 (i5 8G 512G MX250 FHD) 冰河银) 例如：SKU01：8 + 128G，官方标配，蓝水翡翠 SKU02：8 + 256G，官方标配，蓝水翡翠。SKU的图片资源都是来自SPU。制作商品的时候，先有SPU，才会有SKU。
12. Spring boot1.x与spring boot2.x版本的区别：spring boot1.x遵循spring4.0的规则，因为Spring与springMVC为父子容器关系，所以springMVC的版本也为4.0。启动时控制台会显示自带的error控制器，自带的tomcat版本为8，对thymeleaf支持有变化，html标签必须为双标签。 spring boot2.x遵循spring5.0的规则，因为Spring与springMVC为父子容器关系，所以springMVC的版本也为5.0。启动时控制台不会显示自带的error控制器，自带的tomcat版本为9。
13. Spring4.0与spring5.0的区别：spring4.0：拦截器实现方式有：继承和实现两种方式。spring5.0：拦截器实现方式有：要求JDK版本最低为Java8，支持JUnit5并且兼容JUnit4，支持函数式编程。
14. 功能开发流程：**第一步(数据访问层)：**从数据库中找对应的表，创建对应的bean，然后写interface，再写mapper(使用通用mapper可以把interface和mapper合并)；**第二步(服务层)：**创建service与serviceImpl；**第三步(控制层(承载数据模型与页面渲染))：**创建控制器controller。
15. 数据库表结构：base\_attr\_info 平台属性表；base\_attr\_value 平台属性值表；base\_catalog1 一级分类表；base\_catalog2 二级分类表；base\_catalog3 三级分类表；**SPU表：**spu\_info 商品表 小米、华为、荣耀、pixel(商品的品牌)；spu\_sale\_attr 商品的销售属性表；spu\_sale\_attr\_value 商品的销售属性值表；base\_sale\_attr 销售属性字典表；spu\_image 商品图片表；spu\_poster 商品海报表 (存储的是图片的url路径)；**SKU表：**sku\_info 库存单元表 (sku\_info.spu\_id=spu.id)；sku\_image 库存单元图片表(sku\_image.spu\_image\_id=spu\_image.id)；sku\_sale\_attr\_value sku与销售属性值的中间表 (每个商品都有自己的销售属性值)；sku\_attr\_value sku与平台属性值的中间表。 **多表关联查询：**1.找出业务需要的表；2.找出表与表的关联方式(根据业务需求)；3.找出关联条件；4.测试SQL语句。
16. mybatis多表关联： 使用xml配置文件或者使用注解，重要的两个配置文件：mybatis-cfg.xml(核心配置文件，里面配置数据源、映射文件。映射文件：<mappers>

<mapper resource="/com/brevity/mapper/xxxMapper.xml">

<mapper resource="/com/brevity/mapper/xxxMapper.xml">

……<mappers>。包扫描(包下面有xxxMapper.xml(相当于接口的实现类)，xxxMapper.java文件(相当于接口文件)，并且两个文件名必须一致)：<mappers> <package name="com.brevity.mapper">)，<mappers>。 xxxMapper.xml(映射文件，实体类和mapper的映射文件，里面使用select、insert、update、delete标签，标签内写SQL语句)，且在配置文件中配置扫描mapper.xml文件、驼峰命名转换，使xml文件与java文件关联起来。

1. 缓存：设置缓存是为了防止高并发情况下对数据库的访问压力太大，设置读写分离也可以减轻数据库的压力。缓存管理：先更新数据库，再删除缓存，然后从数据库查询后放入缓存。缓存中有数据，直接返回给用户，缓存中没有数据，从数据库中获取数据放入缓存中，再返回给用户。与用户息息相关的数据从数据库中获取，例如商品价格，其它的数据，例如商品详情可以放入缓存，直接从缓存中获取。本地缓存：例如浏览器的缓存数据，将文件数据存储到本地磁盘。redis：一般都是集群模式，是基于内存的缓存，没有开启持久化的话，在关机或者宕机的时候redis的数据会丢失。redis没有开启或者redis宕机，其中的一种方法可以使用try-catch-finally处理。数据访问先从本地缓存读取，读取不到会从redis中读取，redis也读取不到，会从tomcat等服务器缓存读取，都读取不到才会从数据库里获取，然后把数据一级一级的放入各个缓存。redis开机自启动：vim /etc/rc.d/rc.local，文件末尾添加：/usr/local/redis/bin/redis-server /usr/local/redis/bin/redis.conf。
2. 缓存击穿：在高并发的情况下，如果某一个key失效，会给数据库造成巨大压力。 解决办法：使用分布式锁。**方法1.** 使用redis命令加锁 setnx() setex() 例如：set k1 v1 px 10000 nx 返回OK说明操作成功，返回(nil)说明有人操作了，还没有到过期时间；最后需要手动释放锁，避免其它线程不必要的等待。**方法2.** 使用redisson加锁，redisson底层采用的是netty框架，与redis不同的是不需要手动释放锁，可以自动解锁，解锁操作被封装了。
3. 缓存穿透：当用户查询一个在数据库中不存在的数据时，会造成缓存的穿透。 解决办法：设置一个空值给缓存，不暴露错误页面给用户。
4. 缓存雪崩：在缓存中大量的key同时失效，数据库会宕掉。 解决办法：key设置不同的失效时间，不设置在同一个时间失效。
5. 全文检索(ES)：**1.** 使用数据库检索可能会导致索引失效，检索速度降低，用户体验差，导致用户减少。**2.** 全文检索使用的是倒排索引，数据库检索是通过id去匹配value，ES是通过value找id，先将检索条件分词，得到分词的id，用id去匹配值。**3.** 不使用solr的原因是solr适用于数据稳定不变的情况，而ES在数据动态变化的时候检索速度比solr快的多，而且solr是一个war包，需要服务器去运行且需要搭建集群，而ES默认就是集群模式。页面的数据来源是ES，而ES的数据来源于MySQL数据库。在ES中定义一个mapping相当于创建了一张表，需要把数据库的数据存到ES中，需要创建一个与ES中type对应的实体类作为数据的载体。热度排名：借助redis来存储用户访问的次数，当redis中存储的访问数量达到了一个规则后，再去更新ES的数据，不能直接更新ES。
6. 循环面试题：普通循环需要判断循环的次数和是否要循环到末尾，固定取集合的具体数字时需要用普通循环。遍历时不对数据做操作，使用增强for循环。在需要删除集合的数据时，需要使用迭代器。
7. SSO(单点登录)：单一架构：一台服务器(用户信息存储在session中)。分布式架构：多台服务器，使用缓存来存储用户信息，需要将用户信息储存在一个公共的位置，保证每个模块都能访问，(例如存储在redis中，redis能够贯穿整个项目，浏览器也可以贯穿于整个项目，可以把用户名等不重要的信息存储在浏览器的cookie中)。单点登录的使用场景：最好在分布式架构中使用。单点登录的流程：用户访问项目的时候需要去认证中心(自定义的拦截器)判断是否需要登录，如果返回false，说明不需要登录，用户可以看到项目内容，用户继续访问下一个模块，需要去认证中心判断是否需要登录，返回false，说明不需要登录，用户接着访问下一个模块，同样需要到认证中心判断用户是否需要登录，认证中心返回true，同时看缓存中是否有登录的用户信息，如果有，说明已经登录了，如果缓存中没有用户信息，说明没有登录，此时跳转到登录页面，选择登录方式，相当于在访问登录模块。如果用户已经登录了，用户在访问其它模块的时候，页面会显示用户的昵称。单点登录的技术实现：用户访问web应用时先判断cookie中是否有token，如果cookie中没有token(token相当于令牌，相当于一半的门票)，重定向到登录页面去认证用户登录，如果认证登录成功，生成一个token，把token放入cookie中，用户可以继续访问其它模块。登录功能：获取用户输入的用户名和密码，与数据库的信息作比较，如果正确，将用户信息保存到redis中，同时使用JWT生成token令牌，将token写入cookie中。(JWT本质是一个jar包，是一个基于网络传输的json字符串，JWT由三部分组成，分别为公共部分，私有部分，签名部分，JWT主要作于防伪，公共部分：字符串；私有部分：存储用户信息；签名部分：记录服务器的IP地址(如何获取服务器的IP：在nginx的配置文件中配置："proxy\_set\_header X-forwarded-for $proxy\_add\_x\_forwarded\_for;")；三部分通过base64编码生成不可读的字符串，并不是加密)。用户认证：如果用户登录了，redis缓存中一定有数据，通过获取缓存中存储的用户的key(user:userId:info)来查询缓存的数据，其中最重要的是userId，其它的已经写死了。从返回的token中获取userId，如果token中有userId，用户可能已经登录了，去缓存中查询是否有用户数据。主要看redis中是否有用户信息，不能只看cookie。登录成功后显示用户昵称：用户昵称存储在token中。
8. session与cookie：session：存储在服务器，安全性高，生命周期通常是一次会话，默认是30分钟，存储的值是Object，键是String。cookie：存储在浏览器，和session比不安全，可以通过setMaxAge()方法设置，存储的是String，其中domain是cookie访问的作用域(例如：".jd.com" 表示只能是京东的服务访问)，path是存放cookie的路径。联系：JSESSIONID是通过cookie来设置的。
9. 购物车：使用技术：redis可以持久化但是有过期时间、购物车的商品设置过期时间、MySQL做数据的持久化、redis与MySQL数据的同步。登录状态需要把数据分别存在redis和MySQL中。未登录状态也需要把数据存放在redis和MySQL中。数据查询先从redis中查询，再从数据库查询。需要合并登录与未登录购物车的数据。**添加购物车：**判断用户是否登录，如果用户未登录，但是可以从cookie中取出临时的userId，说明之前向未登录的购物车添加过商品，查询购物车中是否已经有当前商品，如果有，数量相加，如果没有，直接添加到数据库，然后将数据放入缓存。如果用户未登录，而且cookie中也取不到临时的userId，说明用户是第一次向购物车添加商品，必须给一个临时的userId，并放入cookie中，把用户向购物车添加的商品存入缓存和数据库中。如果用户已经登录，查询购物车中是否已经有当前商品，如果有，数量相加，如果没有，直接添加到数据库，然后将数据放入缓存。**合并购物车：**未登录的数据合并到已经登录的数据，然后删除未登录的数据。
10. 购物车实现：购物车实现分为登录和未登录的状态：登录情况下用户id一定不为空，数据存储使用redis和MySQL存储，使用redis是为了提高查询效率，使用MySQL是为了做数据的持久化，不使用redis做持久化是因为redis是基于内存的数据库，数据量大的时候，系统性能会下降，不适合大数据量的存储，redis存储时会有一个过期时间。登录的时候用户id一定不为空，首先判断缓存中是否有购物车的key，如果没有，需要从数据库中查询购物车的数据加载到缓存中，如果缓存有数据，直接获取，登录的时候添加购物车根据用户id查询数据库中是否已经有该商品，如果有则数量相加，如果没有，直接添加到数据库，添加到缓存，并设置过期时间。添加缓存时使用的redis的数据类型是hash，因为hash的存储方式是hset(key,field,value),key是谁的购物车，含有userId，field使用skuId，value是cartInfo的字符串。更改购物车的商品的时候可以单独操作。设置的过期时间与用户的过期时间保持一致。用户未登录的情况下给一个默认的过期时间。**购物车列表展示：**先判断未登录的购物车中是否有商品，如果有，判断商品id是否相同，如果相同则数量相加，如果不同，则直接添加到数据库，并放入缓存。未登录状态下没有购物车则直接查询数据库的数据。
11. 防止表单重复提交：**1.** 从页面入手，第二次提交的时候设置一个倒计时，将提交按钮设置为不可用或不可见。**2.** 从后台入手，在生成订单页面的时候制作一个流水号，使用一个隐藏域保存流水号到页面。将流水号保存到缓存或cookie中。比较提交订单的时候新获取的流水号与缓存或cookie中的流水号，如果不相同，说明是第一次提交，则下单成功，如果相同，则是重复提交，则下单失败。最终都要删除缓存的流水号。点击去结算时必须登录，下单的时候需要防止用户的重复提交，还需要验证库存，即订单明细的每个商品都需要进行库存数量校验。
12. 支付宝支付(保证支付的幂等性)：支付宝发送的数据是json格式。支付宝接口参数：支付宝网关(固定)、app\_id、开发者私钥、参数返回格式(只支持json)、编码集(GBK/UTF-8)、支付宝公钥(由支付宝生成)、生成签名字符串的算法类型(RSA2)。支付宝异步通知业务参数：trade\_no支付宝交易号、app\_id开发者的id、out\_trade\_no商户订单号、buyer\_id买家支付宝用户号、seller\_id卖家支付宝用户号、trade\_status交易状态、total\_amount订单金额等。异步返回结果的验签：第一步：在通知返回参数列表中，除去sign、sign\_type两个参数外，凡是通知返回回来的参数皆是待验签的参数；第二步：将剩下参数进行url\_decode，然后进行字典排序，组成字符串，得到待签名字符串；第三步：将签名参数使用base64解码为字节码串；第四步：使用RSA的验签方法，通过签名字符串、签名参数(经过base64解码)及支付宝公钥验证签名；第五步：按照如下描述校验通知数据的正确性(1.商户需要验证该通知数据中的out\_trade\_no是否为商家系统中创建的订单号；2.判断total\_amount是否确实为该订单的实际金额；3.校验通知中的seller\_id是否为out\_trade\_no这笔单据的对应操作方；4.验证app\_id是否为该商户本身)。只有交易状态通知为TRADE\_SUCCESS或TRADE\_FINISHED时，支付宝才会认定买家付款成功。保证支付的幂等性：通过第三方交易编号来保证支付的幂等性。
13. 分布式事务：CAP定理(一致性、可用性、分区容错性)：关系型数据库选择CA，非关系型数据库选择CP。BASE理论：是对CAP定理中一致性和可用性进行权衡的结果，核心思想是：无法做到强一致，但每个应用都可以根据自身的业务特点，采取适当的方式来使系统达到最终一致性。分布式事务使用场景：在订单、支付、库存模块使用分布式事务，基于BASE理论做数据的最终一致性，而不是强一致性。使用消息队列解决分布式事务其实就是将事务分隔。分布式事务的解决方案：1.基于XA协议的两阶段提交方案；2.TCC方案(try confirm cancel)；3.基于消息的最终一致性方案(activemq rabbitmq)；4.阿里的GTS；5.京东的JDTX。消息队列整合到业务中：当订单支付成功后，支付宝异步回调发送消息给订单，订单接收到消息后更新订单的状态为已支付，订单然后发送消息给库存，通知库存系统减库存，库存系统收到消息后开始减库存，库存系统将减库存的结果发送给订单，订单根据减库存的状态去更新订单的状态。消息队列的弊端：消息的不确定性。解决方案：延迟队列、轮询(定时任务)。延迟队列在项目的使用场景：支付过程中，在生成二维码的时候使用延迟队列，每隔一段时间去主动询问支付宝当前订单是否支付成功。轮询的使用场景：关闭过期订单。实现方式：spring boot整合spring task。
14. 多线程的使用：解决高并发，在秒杀的时候使用消息队列限流、降级。下订单：验证库存、验证价格、优惠券、活动、验证用户的身份(异步线程编排)。
15. 拆单：因为商品商家、物流、仓库不同，支付完成后拆单。在支付完成之后，支付宝异步回调，发送消息给订单，更新订单状态，同时订单发送消息给库存，减库存的同时进行拆单。