1. 用java语言进行编程，遵循开发基本的规范，熟悉git的版本控制工具和IDEA、eclipse开发工具的使用。

Git 是分布式版本控制系统，那么它就没有中央服务器的，每个人的电脑就是一个完整 的版本库，这样，工作的时候就不需要联网了，因为版本都是在自己的电脑上。如何多人协作，两之间只需把各自的修改推送给对方，就可以互相看到对方的修改了

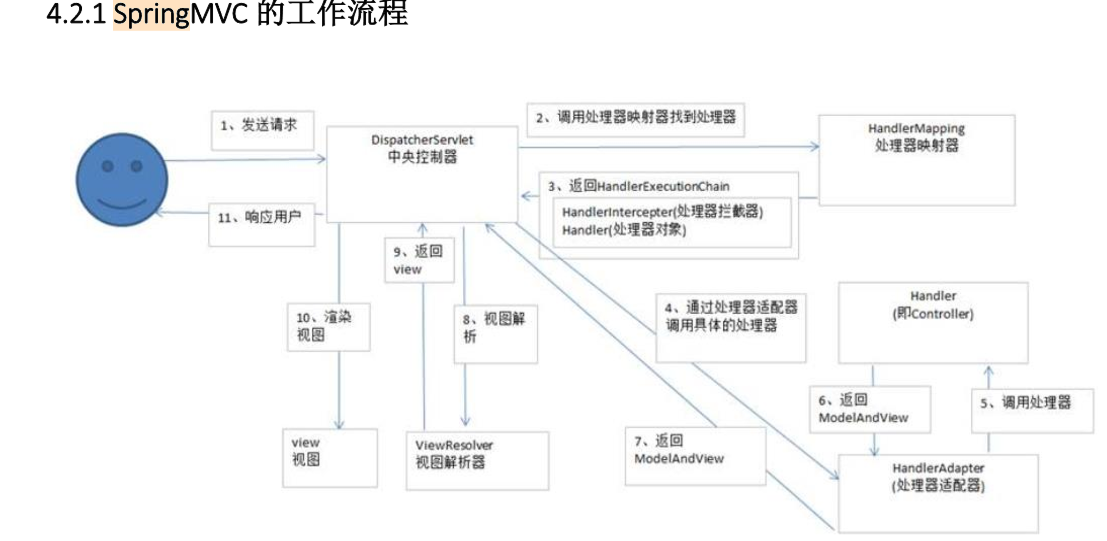
在idea配置git后，使用流程：

1. 从远程仓库克隆：在 idea 的欢迎页上有“Check out from version control”下拉框，选择 git>clone, 代码就会导入到 idea 中。
2. 修改了部分源码，提交到远程仓库：菜单中选择 vcs→git→push，
3. 从服务端拉取代码：如果需要从服务端同步代码可以使用工具条中的“update”按钮
4. 获取更新有两个命令：Fetch和Pull，Fetch是从远程仓库下载文件到本地的origin/master，然后可以手动对比修改决定是否合并到本地的master库。

如何从GitHub获取源代码：

1. 项目的源代码可以直接在代码浏览窗口查看，感觉有用的话，直接复制。
2. 克隆（Clone）：克隆之后，会吧源代码下载到本地，创建一个本地的代码库，可以任意在本地修改代码并使用git所提供的命令操作代码。有代码对应的历史记录和分支。
3. 熟练掌握常用的开源框架SSM（Spring、SpringMVC、Mybatis）等开源框架应用知识，能快速熟悉框架整合下的项目开发，理解并使用Spring的核心思想IOC、AOP，深刻理解SpringMVC核心运行流程。

Spring MVC 是一个使用了MVC架构模式思想的Web框架



1、用户发送请求至前端控制器 DispatcherServlet

2、DispatcherServlet 收到请求调用 HandlerMapping 处理器映射器。

3、处理器映射器找到具体的处理器，生成处理器对象及处理器拦截器(如果有则 生成)一并返回给 DispatcherServlet。

4、DispatcherServlet 调用 HandlerAdapter 处理器适配器

5、HandlerAdapter 经过适配调用具体的处理器(Controller，也叫后端控制器)。

6、Controller 执行完成返回 ModelAndView

7 、 HandlerAdapter 将 controller 执 行 结 果 ModelAndView 返回给 DispatcherServlet

8、DispatcherServlet 将 ModelAndView 传给 ViewReslover 视图解析器

9、ViewReslover 解析后返回具体 View

10、DispatcherServlet 根据 View 进行渲染视图（即将模型数据填充至视图中）。

11、DispatcherServlet 响应用户

2． spring中的IOC、AOP和DI及应用场景

spring是一个开源框架，可以管理web层，持久层，业务层， spring可以配置各个层的组件，并且维护各个层的关系。Spring即使一个AOP框架，也是一IOC容器。

优点：

1. 方便解耦，简化开发
2. AOP编程的支持
3. 声明式事务的支持：这意味着你可以将业务代码和事务管理分离，你只需用注解和XML配置来管理事务
4. 方便程序的测试
5. 方便集成各种优秀框架

(6) 降低 JavaEE API的使用难度

IOC（控制反转）：利用反射的原理将对象创建的权利交给了spring容器，Spring在运行的时候，根据配置文件来动态的创建对象和维护对象之间的关系，实现了松耦合的思想。

AOP（面向切面）：简单的说它就是把我们程序重复的代码抽取出来，在需要执行的时候，使用动态代理的技术，在不修改源码的基础上，对我们的已有方法进行增强。

通过配置可以把业务逻辑和系统服务分离，业务逻辑只关心业务，而不在去处理其他事情。

DI（依赖注入）：由Spring容器将对象注入到使用它的地方，被注入的对象只提供对应的方法接受就行，由容器来决定对象之间的依赖关系。

应用场景：

IOC（控制反转）：项目中的Bean都可以交给Spring容器来维护，这样Bean的创建及销毁以及生命周期都由Spring来处理。service、dao层。

AOP（面向切面）：事务，日志，权限等。

具体使用：写一个切面类，对所需方法进行增强。

1在类上面加上@Aspect @Component注解，说明这是一个切面类。

2 写一个切点签名方法。在上面加上切入点表达式@Pointcut("execution(public \* com.example.controller.\*.\*(..))") 就是指定对哪些类的哪些方法进行增强。（这样做可以避免每次都写很长的切入点表达式）

3 @Before（签名）表示：在每次执行切入点表达式的方法之前会执行此方法。

后置异常通知 @AfterThrowing("webLog()") 后置通知 @After("webLog()")

DI（依赖注入）：比如Service层需要调用Dao层访问数据库，这时可以把Dao层的Bean给Spring管理，我们只需要在Service中定义对应的方法来接收由Spring容器负责注入的Dao层的Bean即可。

常用的注解：组件@Controller @Service @Repository @Component 注入@Autowired

javaben@Bean 事务@Transactional

事务：它是一个操作序列，要么都执行，要么都不执行：原子性，一致性，隔离性，持久性

隔离级别：读已提交，读未提交，可重复读，可串行化

1. 熟悉使用基于springBoot的springClould全局治理框架的常用组件：Eureka、 Feign、Hystrix、Zuul、SpringCloudConfig、SpringCloudBus。

Spring Boot是为了简化Spring应用的开发。Spring Boot专注于快速、方便集成的单个微服务个体。因为它提供了一堆依赖打包，并且按照使用习惯解决了依赖问题。能够自动配置，整合依赖，嵌入式容器。做到专注于Spring应用开发，无需过多关注于xml配置。

SpringCloud是一系列框架的有序集合，是一个关注全局的服务治理框架。

如服务发现 Eureka （相当于dubbo框架的zookeeper）：将所有的微服务都注册到Eureka中，这样所有的微服务之间都可以互相调用了。

Eureka Server提供服务注册服务，各个节点启动后，会在Eureka Server（相当于dubbo）中进行注册，这样Eureka Server中的服务注册表中将会存储所有可用服务节点的信息

Eureka Client（相当于zookeeper）是一个java客户端，在应用启动后，将会 向Eureka Server发送心跳,默认周期为30秒，如果Eureka Server在多个心跳周期内没有 接收到某个节点的心跳，Eureka Server将会从服务注册表中把这个服务节点移除(默认90 秒)。，Eureka还提供了客户端缓存机制，即使所有的Eureka Server都挂掉，客户端依然可以利用缓存中的信息消费其他服务的API。

这时候就用到feign了：他可以实现服务间的调用，还有负载均衡的作用。比如讨论微服务和文章微服务同时调用多个基础微服务，它会轮流调用这些基础微服务。

在微服务架构中通常会有多个服务层调用，有一些微服务出现故障（比如基础服务的故障可能会导致级联故障，） 进而造成整个系统不可用的情况，这种现象被称为服务雪崩效应。

Hystrix防止服务级联失败，同时提供失败回退机制，使你的系统能更快地从异常中恢复。能够自动的开启和关闭。出现异常，自动开启。异常解决，自动关闭。

zuul微服务网关是介于客户端和服务器端之间的中间层，所有的外部请求都会先经过微服务网关， Zuul组件的核心是一系列的过滤器。等于把所有微服务看成是一个整体。就可以解决：

客户端会多次请求不同微服务，增加客户端的复杂性，跨域请求的问题，认证复杂问题。调微服务不用写微服务端口号，写微服务名字就可以了。简单理解为跟nginx，是一个代理。

两个网关，一个网站前台网关，用户调用，一个管理后台网关，管理员调用。

zuul的过滤器：zuulfilter

zuul中定义了四种不同生命周期的过 滤器类型

pre ：可以在请求被路由之前调用

route ：在路由请求时候被调用

post ：在route和error过滤器之后被调用

error ：处理请求时发生错误时被调用

spring cloud config在分布式系统中，由于服务数量巨多，了方便服务配置文件统一管理，实时更新，所以需要分布式配置中心组。我们把配置中心放到GitHub上，用SpringCloudBus来实现配置的自动更新，用到消息中间件rabbitMQ，当GitHub上的配置文件发生改变时，会生产消息到MQ。其他微服务是消费者。当监听到消息的时候。会跟新配置。

常用的注解：springboot：

启动类：@SpringBootApplication ；

controller层： @RestController @CrossOrigin @RequestMapping @Autowired @PathVariable @RequestBody Service层： @Service dao层是个接口：@Query注解JPQL方式查询

pojo类：@Entity 实体类 @Table 类对应的表 @Id 主键id

SpringCloud:启动类：服务节点 @EnableEurekaClient 服务注册@EnableEurekaServer @EnableFeignClients 调用者

@FeignClient注解用于指定从哪个服务中调用功能@RequestMapping注解用于对被调用的微服务进行地址映射

1. 熟练使用关系型数据库MySQL、Oracle。熟悉MySQL数据库性能调优、熟悉innodb和Myisam存储引擎的锁机制。

我

1. 表、字段设计阶段。遵循三个范式：字段原子性，消除对主键的部分依赖，消除对主键的传递依赖。
2. 存储引擎的选择。

Innodb支持事务，支持外键，支持行级锁，锁定力度小，并发能力高。默认使用。

MyISAM 不支持事务，不支持外键，支持表级锁不支持行级锁，并发能力较低。

1. 利用好MySQL自身提供的索引功能。为了快速匹配where条件中匹配的列。普通索引：对关键字没有限制，唯一索引：要求关键字不能重复，主键索引：要求关键字唯一且部位null。建立了索引要避免失效。如where后面不要使用like。不要用不等于。
2. 横向扩展：MySQL集群，负载均衡，读写分离
3. SQL语句的优化（收效甚微）主要是尽量避免全表扫描，利用explain查看执行计划。
4. 熟悉Redis、MongoDB的特点和应用场景。了解redis的RDB、AOF持久化机制。

Redis的五种数据类型：string、hash、set、list、 zset。

redis是一个内存版的数据库，写性能和读性能都远高于MySQL，这就解决了高并发下的性能问题。

RDB：该机制可以在指定的时间间隔内生成数据集的时间点快照

AOF：安全性高，修改后立即同步或每秒同步一次。

1.主要应用在门户网站首页广告信息的缓存。因为门户网站访问量较大，将广告缓存到redis中，可以降低数据库访问压力，提高查询性能。

2.应用在用户注册验证码缓存。利用redis设置过期时间，当超过指定时间后，redis清理验证码，使过期的验证码无效。

SpringDataMongoDB：json格式的数据类型：独立于编程语言的文本格式来存储和表示数据，特点是结构简介清晰，易于人阅读和机器解析，提高传输效率。面向集合存储。一个json格式的数据相当于关系型数据库的一行，多个json数据的集合，相当于关系型数据库的一个表。几乎可以实现关系型数据库的单表查询的。

使用场景：

适合用于数据量大，写入操作频繁，价值低的数据。文章评论，吐槽

1. 网站实时数据处理
2. 缓存性能高
3. 用于对象及json数据的存储
4. 高伸缩性的场景：服务器随着访问量可以多可以少
5. 熟悉分布式搜索引擎ElasticSearch和solr的特点和使用场景，及Ik分词器的配置。

ES和solr都是分布式搜素引擎。基于lucene搜索服务器。建立索引减轻mysql的访问压力。

特点：

1、 Solr查询快，但是更新删除慢。Solr一边建立索引一边搜索会产生io阻塞，查询性能较差。多用于电商查询多的应用。

ES建立索引快，查询稍慢，但实时搜索效率明显高于solr，索引建立不影响搜索速度。

2、Solr利用zookeeper进行分布式管理，而ES自身带有分布式协调管理功能。

3、solr支持更多格式的数据，而ES仅支持json格式数据。

两个都要注意把默认的分词器换成IK分词器。

使用logStash定时同步功能，把mysql中的数据文章同步到ES中。

1. 熟悉分布式系统中消息中间件RabbitMQ和ActiveMQ的使用，和RabbitMQ的路由规则及ActiveMQ的两种模式。

消息中间件是分布式系统中的重要组件，主要作用是：异步处理，应用解耦，流量削峰和消息通讯。实现高性能，高可用，可伸缩和最终一致性。

ActiveMQ的消息传递有两种类型：一种是点对点，一个生产者对应一个消费者。一种是发布/订阅模式，一个生产者生产的消息可以由多个消费者消费。

RabbitMQ:生产者将消息发送给Exchange，由Exchange将消息路由到一个或多个Queue队列中。有路由规则有：直接模式，分列模式，主题模式，headers。用户注册发送短信验证码

1. 能基于SpringSecurity做BCrypt密码加密和用户鉴权JWT。

使用SpringSecurity提供的BCryptPasswordEncoder类。使用BCrypt强哈希方法加密密码。

jwt（json web token）

1 头部:用于描述关于该JWT的最基本的信息，例如其类型以及签名所用的算法等。

2 载荷:就是存放有效信息的地方，如jwt的签发时间 ，过期时间，jwt的唯一身份标识，主要用来作为一次性token,从而回避重放攻击。

3签证:这个部分需要base64加密后的header和base64加密后的payload使用.连接组成的字符串，然后通过header中声明的加密方式进行加盐secret组合加密，然后就构成了jwt的第 三部分

JWT基于 Token 的身份验证方法，在服务端不需要存储用户的登录记录

1. 客户端使用用户名跟密码请求登录

2. 服务端收到请求，去验证用户名与密码

3. 验证成功后，服务端会签发一个 Token，再把这个 Token 发送给客户端

4. 客户端收到 Token 以后可以把它存储起来，比如放在 Cookie 里

5. 客户端每次向服务端请求资源的时候需要请求头带着服务端签发的 Token

6. 服务端收到请求，然后去验证客户端请求里面带着的 Token，如果验证成功，就向客户端返回请求的数据

优点：支持跨域访问，用户认证信息通过HTTP头传输.。

CSRF:因为不再依赖于Cookie，所以你就不需要考虑对CSRF（跨站请求伪造）的防范。

1. 能够基于linux系统进行项目的部署以及常规操作，docker容器的简单部署。

比如说查看日志：tail -f 默认查看后十行日志。

解压：tar -zxvf 软件安装命令rpm

改变文件夹：cd 创建文件夹：mkdir

编辑文件vim

搜索镜像docker search

查看本机上有什么镜像 docker images

拉取镜像：docker pull mysql

启动mysql容器 docker run -di --name=容器的名称 -p 33306:3306

所有运行的进程docker ps -a

编辑文件vi

1. 了解一些常见算法如快排、选择、冒泡。

java代码直接排序写Collection.sort，底层用的是归并排序。

冒泡排序：

1、比较相邻的元素。如果第一个比第二个大，就交换他们两个。

2、 对每一对相邻元素做同样的工作，从开始第一对到结尾的最后一对。在这一点，最后的元素应该会是最大的数。

3、针对所有的元素重复以上的步骤，除了最后一个。

4、持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。

public static void bubbleSort(int []arr) {

int[] arr = {12,23,34,56,56,56,78};

for(int i =0;i<arr.length-1;i++) {

for(int j=0;j<arr.length-i-1;j++) {//-1为了防止溢出

if(arr[j]>arr[j+1]) {

int temp = arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

}

快速排序：（Quicksort）是对冒泡排序的一种改进。  
通过一趟排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一部分的所有数据都要小，然后再按此方法对这两部分数据分别进行快速排序，整个排序过程可以递归进行，以此达到整个数据变成有序序列。

选择排序：（Selection sort）

它的工作原理是每一次从待排序的数据元素中选出最小（或最大）的一个元素，存放在序列的起始位置，然后，再从剩余未排序元素中继续寻找最小（大）元素，然后放到已排序序列的末尾。以此类推，直到全部待排序的数据元素排完。 选择排序是不稳定的排序方法。

递归算法：是一种直接或者间接的调用自身的算法。

1. 了解jQuery，Ajax， Angularjs，vue等前端技术。

JQuery的常用选择器，都是用$（）

基本选择器（标签选择器 ID选择器 类选择器）

标签选择器 $("html标签名") 获得所有匹配标签名称的于元素

id选择器 $("#id的属性值") 获得与指定id属性值匹配的元素

类选择器 $(".class的属性值") 获得与指定的class属性值匹配的元素

层次选择器、

属性选择器

提供静态页面，能看懂和简单修改。写一些简单的事件。如点击事件，键盘事件.

angularjs常用的指令：

ng-init : 初始化（打开页面即执行该函数）。 ng-click：点击事件（点击该按钮就执行该函数）。 ng-repeat：循环。ng-model双向绑定。

ng-options：动态显示下拉框的内容。 ng-if ： 显隐的控制（执行的函数为true就显示）

vue 常用指令

v-on vue的监听事件

v-on:click 鼠标点击事件 v-on:keydown 键盘按下事件 v-on:mouseover 鼠标的悬停

v-model vue的双向绑定

通过AJAX与服务器进行数据交换，AJAX可以使用网页实现局部更新，这意味着可以在不刷新整个网页的情况下，对网页的某部分进行更新。

怎么来实现AJAX？jQuery框架的ajax

使用XmlHttpRequest这个对象可以异步向服务器发送请求，获取响应，完成局部更新

直接在页面写一个函数，将ajax放进去。

GET请求 $.get(url, [data], [callback], [type])

url 请求的服务器端url地址

data 发送给服务器端的请求参数，格式可以是key=value，也可以是js对象

callback 当请求成功后的回掉函数，可以在函数体中编写我们的逻辑代码

type 预期的返回数据的类型，取值可以是 xml, html, script, json, text, \_defaul等

POST请求 $.post(url, [data], [callback], [type])

url 请求的服务器端url地址

data 发送给服务器端的请求参数，格式可以是key=value，也可以是js对象

callback 当请求成功后的回掉函数，可以在函数体中编写我们的逻辑代码

type 预期的返回数据的类型，取值可以是 xml, html, script, json, text, \_defaul等

微服务项目一的问题：

介绍一下淘股吧社区论坛？

我在一个外包公司承包了淘股吧社区论坛的项目。我们团队有9个人，2前端，6后端，和1个测试。今日推荐：回复数或者点赞数比较多的文章、和爬取得新闻。广告：就是一些证券公司图，一点就会跳转到相应的公司官网。问答：就是一些简短的提问或者回答。文章：就是一些牛人分享的经验，下面可以评论。交友：就是可以跟论坛上的用户互粉添加好友。搜索：就是可以对股票，标题，作者，主题关键字进行搜索，出现相应的文章。公告：就是官方发布一些消息，比如维护更新。热门股吧：就是爬的一些股票数据。用户中心：用户的注册登录等。

这个项目是前后端分离的，我只给以上的功能提供RESTfull API 接口，供前端去调用

用户搜索功能的开发

使用实时的分布式搜索和分析引擎elasticsearch对搜索微服务进行开发，并使用轻量级日志搜索处理框架Logstash将Mysql中的数据定时同步到ElasticSearch服务器。

这个模块里面，有文章的添加和文章的搜索。

文章添加直接保存在ES中，文章搜索使用方法命名查询的方法。

在dao层的访问接口中，添加方法public interface ArticleSearchDao extends  ElasticsearchRepository<Article,String> {

   public Page<Article> findByTitleOrContentLike(String title, String  content, Pageable pageable);

}

注意两点：

1用图形化界面来实现Elasticsearch的日常管理，最常用的就是Head插件

2 默认的中文分词是将每个字看成一个词，ik分词器，是一个分局中国词语进行分词的分词器。

IK提供了两个分词算法ik\_smart 和 ik\_max\_word

其中 ik\_smart 为最少切分，ik\_max\_word为最细粒度划分

添加好友模块开发：

1 当用户登陆后在推荐好友列表中点击“心”，表示喜欢此人 ，在数据库tb\_friend表中 插入一条数据，字段islike 为0

2 当你点击了喜欢过的人，也喜欢了你 , 表示互粉成功，就成为了好友。也向tb\_friend表中插入一条 数据，字段islike为1 ，并且将你喜欢她的数据islike也修改为1

3 当你点击了不喜欢某人（点击了叉），向tb\_nofriend添加记录.

4 当两个人互粉后，其中一人不喜欢对方了，删除好友表中的记录 ，向非好友表中添加记录

广告业务开发：

1 查询公告数据，现从数据库中查询

2 如果缓存中没有，再查询数据库，将查询出来的结果放入缓存

3 如果缓存中有数据，直接返回数据，不在查询数据库，大大减轻数据库的压力。

4 当广告数据发生变更时，需要将缓存数据清除，这样再次查询才能获取最新的数据。

什么是微服务？

微服务是系统机构上的一种设计风格，主旨是将一个原本独立的系统，拆分为多个小型服务，这些小型服务在各自独立的进程中运行。服务之间通过基于http/https协议的RESTful API进行通信。面向服务架构SOA强调的是异构系统之间的解耦合。而微服务降调的是系统按照业务边界做细粒度的拆分和部署。

公共子模块里面放的是什么？

实体类javabean：比如说，返回数据集的实体类，分页的结果类，状态码定义类。

工具类utils：比如说，分布式自增长的id的雪花算法。

poi插件的应用：如何将excel表中的数据导出输入数据库中，比如品牌数据。

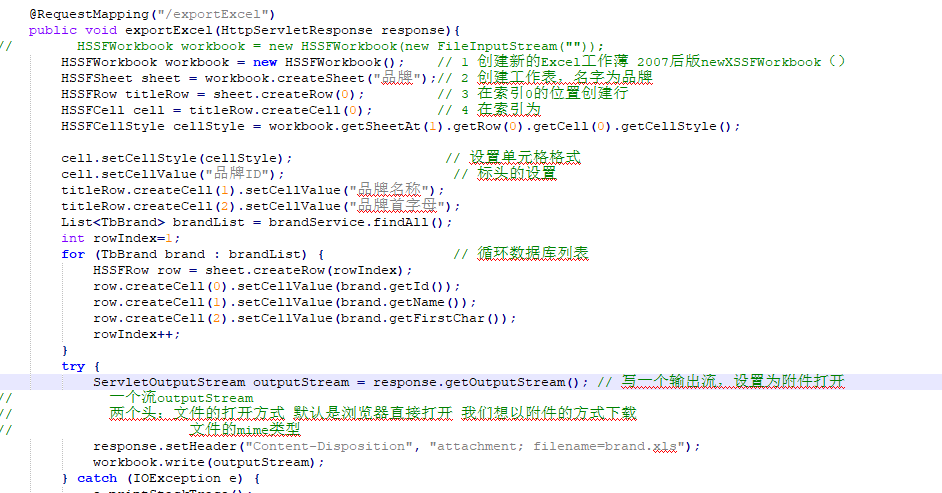
导出到excel表

1 创建新的Excel工作薄 2007后版newXSSFWorkbook（）

2 创建工作表，名字为品牌 3 在索引0的位置创建行 4 在索引0的位置创建单元格

5 设置单元格格式 6 写字段名 7 循环 数据库列表

8 写一个输出流，设置为附件打开

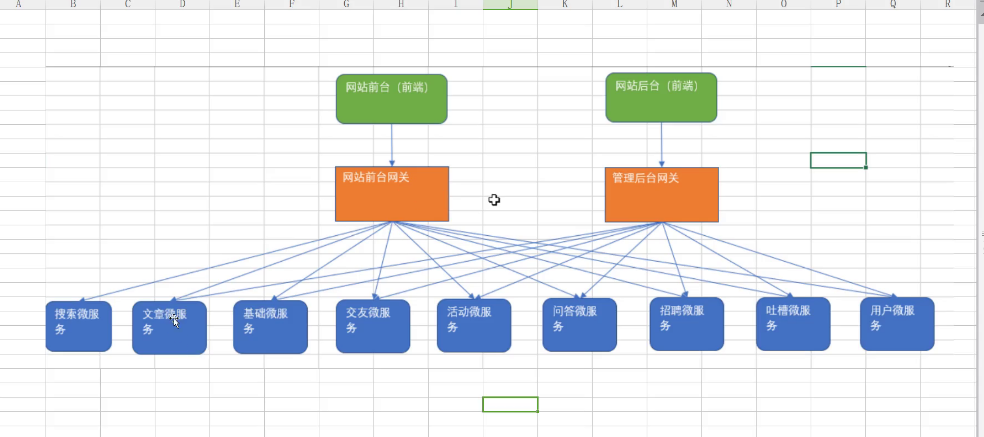


导入到数据库表

1 先写这个文件的输入流

2.创建一个工作薄，XSSFWorkbook

3 然后get表sheet 。写一个循环读取Excel表的数据。sheet.getLastRowNum



数据库：

1插入部分数据： INSERT INTO 表名 (字段名1, 字段名2, ...) VALUES (值1, 值2, ...);

INSERT INTO student (id, NAME, age, sex) VALUES (1, '张三', 20, '男');

2 更新表记 1. 不带条件修改数据 UPDATE 表名 SET 字段名=值;

UPDATE student SET sex='女';

2. 带条件修改数据 UPDATE 表名 SET 字段名=值 WHERE 字段名=值;

UPDATE student SET sex='男' WHERE id=2;

3 删除数据： 1. 不带条件删除数据 DELETE FROM 表名;

2. 带条件删除数据 DELETE FROM 表名 WHERE 字段名=值;

DELETE FROM student WHERE id=3;

4：查询 1查询指定 SELECT 字段名1, 字段名2... FROM 表名;

 SELECT NAME, age FROM student;

2. 别名查询 SELECT 字段名1 AS 别名, 字段名2 AS 别名... FROM 表名 AS 表别名;

3 条件查询： select 字段名 from 表名 where 条件

比较运算符，逻辑运算符，范围，like模糊查询，排序（放在where后的条件中）

4 聚合函数，SELECT COUNT(\*) FROM student3 WHERE age>40

之前我们做的查询都是横向查询，而使用聚合函数查询是纵向查询

五个聚合函数： count ： 统计指定列记录数，记录为NULL的不统计 sum ： 计算指定列的数值和，如果不是数值 类型，那么计算结果为0 max ： 计算指定列的最大值 min ： 计算指定列的最小值 avg ： 计算指定列的平均值，

5 分组查询： GROUP BY 将分组字段结果中相同内容作为一组

SELECT 字段1,字段2... FROM 表名 GROUP BY 分组字段 [HAVING 条件];

查询年龄大于25岁的人,按性别分组,统计每组的人数,并只显示性别人数大于2的数据

SELECT sex, COUNT(\*) FROM student3 WHERE age > 25 GROUP BY sex HAVING COUNT(\*)>2

having与where的区别

having是在分组后对数据进行过滤. where是在分组前对数据进行过滤 having后面可以使用聚合函数 where后面不可以使用聚合函数

6 limit 语句：LIMIT 的作用就是限制查询记录的条数，分页查询，orcle用rownumber

跳过前面两条，读取6条数据。

SELECT \* FROM student3 LIMIT 2,6

5 多表查询：内连接（隐式内连接，显示内连接）、外链接（左外连接，右外连接）

1内连接：用左边表的记录去匹配右边表的记录，如果符合条件的则显示,

显示内连接：SELECT 字段名 FROM 左表 INNER JOIN 右表 ON 条件；

eg 查询唐僧的信息，显示员工id，姓名，性别，工资和所在的部门名称，

SELECT e.`id` , e.`NAME` , e.`gender`, e.`salary`, d.`NAME` FROM  dept d INNER JOIN emp e ON e.`dept\_id`=d.`id` AND e.`NAME`='唐僧'

2左连接：用左边表的记录去匹配右边表的记录，如果符合条件的则显示；否则，显示NULL 可以理解为：在内连接的基础上保证左表的数据全部显示。

SELECT 字段名 FROM 左表 LEFT  JOIN 右表 ON 条件；

 SELECT \* FROM dept LEFT  JOIN emp ON emp.`dept\_id`=dept.`id`;

3子查询：一条SELECT语句结果作为另一条SELECT语法一部分（作为查询条件或查询结果或表） SELECT 查询字段 FROM 表 WHERE 查询条件;

1. 子查询的结果是一个值：

在WHERE 后面作为 条件 SELECT 查询字段 FROM 表 WHERE 字段=（子查询）;

SELECT \* FROM employee WHERE salary=(SELECT MAX(salary) FROM employee);

1. 子查询结果是单列多行的时候:

在WHERE 后面作为 条件 子查询结果是单例多行，结果集类似于一个数组，父查询 使用 IN 运算符 SELECT 查询字段 FROM 表 WHERE 字段 IN （子查询）;

SELECT \* FROM emp WHERE dept\_id IN (SELECT id FROM dept WHERE NAME IN('开发部'));

1. 子查询结果是多列多行的时候:

在 FROM 后面作为 表 SELECT 查询字段 FROM （子查询） 表别名 WHERE 条件; 子查 询作为表需要取别名，否则这张表没用名称无法访问表中的字段

一些复杂实例步骤：

1. 确定要查询哪些表
2. 确定表连接条件
3. 确定查询字段

 SELECT \* FROM emp e INNER JOIN job j INNER JOIN dept d INNER JOIN salarygrade s ON  e.job\_id=j.id AND e.dept\_id=d.id AND e.salary BETWEEN s.losalary AND hisalary AND j.jname='经理;

 SELECT d.`id`, d.dname, d.`loc`, e.total FROM dept d INNER JOIN (SELECT dept\_id, COUNT(\*)

total FROM emp GROUP BY dept\_id) e ON e.dept\_id=d.`id`;

通过Mycat结点来管理不同服务器上的数据库。

查询计划：

MySql提供了EXPLAIN语法用来进行查询分析，在SQL语句前加一个"EXPLAIN"即可。

我们就能知道查询大致的运行时间了，如果查询里没有用到索引、或者需要扫描的行过多，那么可以感到明显的延迟。因此需要改变查询方式或者新建索引。mysql中的explain语法可以帮助我们改写查询，优化表的结构和索引的设置，从而最大地提高查询效率。

分布式事务：

1. 结合MQ消息中间件实现的可靠消息最终一致性，基于事务的mq方案。

第一种方案：可靠消息最终一致性，需要业务系统结合MQ消息中间件实现，在实现过程中需要保证消息的成功发送及成功消费。即需要通过业务系统控制MQ的消息状态

1. TCC补偿性事务解决方案

TCC补偿性，分为三个阶段TRYING-CONFIRMING-CANCELING。每个阶段做不同的处理。TRYING阶段主要是对业务系统进行检测及资源预留

CONFIRMING阶段是做业务提交，通过TRYING阶段执行成功后，再执行该阶段。默认如果TRYING阶段执行成功，CONFIRMING就一定能成功。

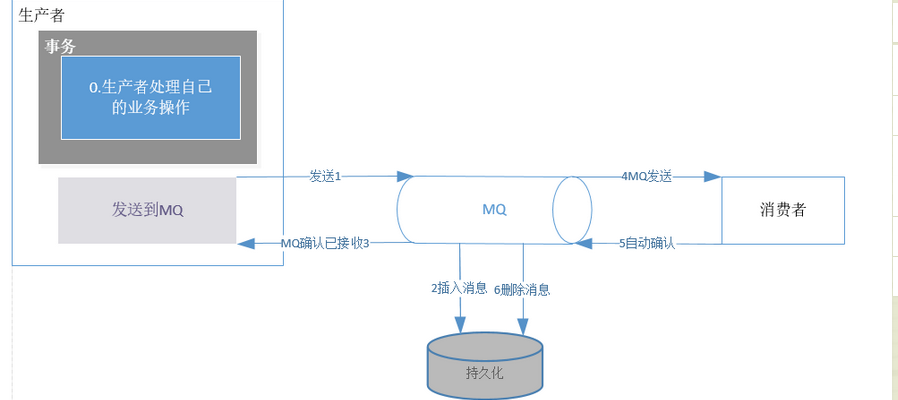
CANCELING阶段是回对业务做回滚，在TRYING阶段中，如果存在分支事务TRYING失败，则需要调用CANCELING将已预留的资源进行释放。

1. 最大努力通知型方案

最大努力通知xing型，这种方案主要用在与第三方系统通讯时，比如：调用微信或支付宝支付后的支付结果通知。这种方案也是结合MQ进行实现，例如：通过MQ发送http请求，设置最大通知次数。达到通知次数后即不再通知。

基于事务消息的MQ方案是目前公认的较为理想的分布式事务解决方案，各大电商都在应用这一方案。种方式适合的业务场景广泛，而且比较可靠。不过这种方式技术实现的难度比较大。目前主流的开源MQ（ActiveMQ、RabbitMQ、Kafka）均未实现对事务消息的支持，所以需二次开发或者新造轮子。

如何保证mq的消息一致性：



如果消费者，在自动确认消息后，在还没有消费消息时，若消费者挂掉了，当前产者不会重新发送，这就产生了消息不一致的情况。

RabbitMQ的 从conform模式，onform模式主要包含两种编程模式，一种同步的，一种异步通知的：

同步：通道Channel开启Conform，在发送完消息后可以通过WaitForConfirm等待消息的投递结果，这里有个可选参数，就是阻塞等待的时间，如果返回结果为false则表示消息投递失败，则发送端这时候也就可以采取重试之类的策略了。