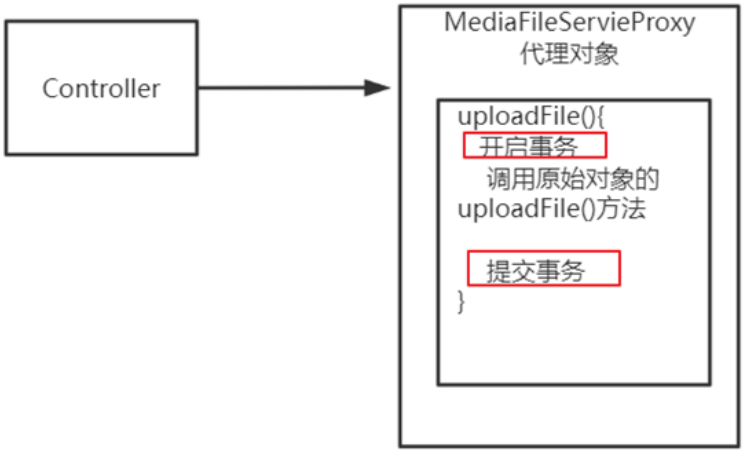
问题

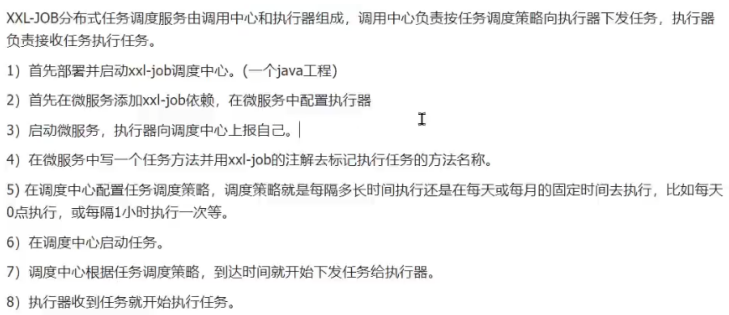
- Git代码冲突怎么处理?
  - 我们在使用Git时难免会出现代码冲突的问题，出现冲突的原因是因为当本地文件的版本与目标分支中文件的版本不一致时当存在同一行的内容不同时在进行合并时会出现冲突。
  - 代码冲突一般发生在以下情况：1、多个分支向主分支合并时2、同一个分支下pull或push操作时。
- 你是在哪个分支开发?
  - 我们不是直接在主分支开发，由技术经理创建独立的开发分支，我们是在独立的开发分支中进行开发，最后由技术经理将开发分支合并到主分支。
- maven依赖版本冲突怎么处理?
  - maven依赖版本冲突一般是由于间接依赖导致一个jar包即有多个不同的版本，比如：A依赖了B的1.0版本，C依赖了B的2.0版本，项目依赖A和C从而间接依赖了B的1.0和2.0版本，此时B有两个版本引入到了项目中，当存在版本冲突时可能会出现ClassNotFoundException、NoSuchMethodError等错误。
  - 处理版本冲突可以使用以下方法：1、使用exclusions 排除依赖比如：我们只依赖B的1.0版本，此时可以在依赖C时排除对B的依赖。2、使用dependencyManagement锁定版本号。通常在父工程对依赖的版本统一管理。比如：我们只依赖B的1.0版本，此时可以在父工程中限定B的版本为1.0。
- maven的常用命令
  - mvn clean //清除target目录中的生成结果mvn compile //编译源代码mvn test //执行单元测试mvn package //打包mvn install //打包并把打好的包保存到本地仓库mvn deploy //打包并把打好的包上传到远程仓库
- MySQL常见的存储引擎及区别?
  - 1.InnoDb
    - 1、支持事务。2、使用的锁粒度默认为行级锁，可以支持更高的并发；也支持表锁。3、支持外键约束；外键约束其实降低了表的查询速度，增加了表之间的耦合度。
  - 2.MyISAM
    - 1、不提供事务支持2、只支持表级锁3、不支持外键
  - 3.memory
    - 数据存储在内存中
  - 总结
    - • MyISAM管理非事务表，提供高速存储和检索以及全文搜索能力，如果在应用中执行大量select操作，应该选择MyISAM• InnoDB用于事务处理，具有ACID事务支持等特性，如果在应用中执行大量insert和update操作，应该选择InnoDB
- MySQL建表时注意什么?

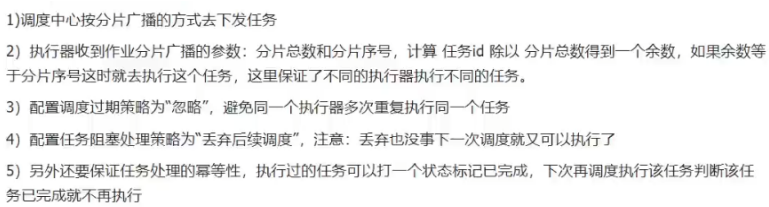
- 1、注意选择存储引擎，如果要支持事务需要选择InnoDB。
- 2、注意字段类型的选择，对于日期类型如果要记录时分秒建议使用datetime，只记录年月日使用date类型，对于字符类型的选择，固定长度字段选择char，不固定长度的字段选择varchar，varchar比char节省空间但速度没有char快；对于内容介绍类的长广文本字段使用text或longtext类型;如果存储图片等二进制数据使用blob或longblob类型；对金额字段建议使用DECIMAL；对于数值类型的字段在确保取值范围足够的前提下尽量使用占用空间较小的类型，
- 3、主键字段建议使用自然主键，不要有业务意义，建议使用int unsigned类型，特殊场景使用bigint类型。4、如果要存储text、blob字段建议单独建一张表，使用外键关联。5、尽量不要定义外键，保证表的独立性，可以存在外键意义的字段。6、设置字段默认值，比如：状态、创建时间等。7、每个字段写清楚注释。8、注意字段的约束，比如：非空、唯一、主键等

- SpringBoot接口开发的常用注解有哪些
  - @Controller 标记此类是一个控制器，可以返回视图解析器指定的html页面，通过@ResponseBody可以将结果返回json、xml数据。@RestController 相当于@ResponseBody加 @Controller，实现rest接口开发，返回json数据，不能返回html页面。
  - @RequestMapping 定义接口地址，可以标记在类上也可以标记在方法上，支持http的post、put、get等方法。@PostMapping 定义post接口，只能标记在方法上，用于添加记录，复杂条件的查询接口。@GetMapping 定义get接口，只能标记在方法上，用于查询接口的定义。@PutMapping 定义put接口，只能标记在方法上，用于修改接口的定义。@DeleteMapping 定义delete接口，只能标记在方法上，用于删除接口的定义。
  - @RequestBody 定义在方法上，用于将json串转成java对象。@PathVarible 接收请求路径中占位符的值.
  - @ApiOperation swagger注解，对接口方法进行说明。@Api wagger注解，对接口类进行说明。
  - @Autowired 基于类型注入。@Resource 基于名称注入，如果基于名称注入失败转为基于类型注入。
- 项目的开发流程是什么?
  - 1、产品人员设计产品原型。2、讨论需求。3、分模块设计接口。4、出接口文档。5、将接口文档给到前端人员，前后端分离开发。6、开发完毕进行测试。7、测试完毕发布项目，由运维人员进行部署安装。
- Mybatis分页插件的原理?
  - 首先分页参数放到ThreadLocal中，拦截执行的sql，根据数据库类型添加对应的分页语句重写sql，例如：(select * from table where a) 转换为 (select count(*) from table where a)和(select * from tablewhere a limit ,)计算出了total总条数、pageNum当前第几页、pageSize每页大小和当前页的数据，是否为首页，是否为尾页，总页数等。
- 树型表的标记字段是什么？如何查询MySQL树型表?
  - 树型表的标记字段是parentid即父结点的id。

- 查询一个树型表的方法：
  - 1）当层级固定时可以用表的自链接进行查询。2）如果想灵活查询每个层级可以使用mysql递归方法，使用with RECURSIVE 实现
- MyBatis的ResultType和ResultMap的区别？
  - ResultType：指定映射类型，只要查询的字段名和类型的属性名匹配可以自动映射。
  - ResultMap：自定义映射规则，当查询的字段名和映射类型的属性不匹配时可以通过ResultMap自定义映射规则，也可以实现一对多、一对一映射。
- #{} 和 ${} 有什么区别？
  - #{}是标记一个占位符，可以防止sql注入。
  - ${} 用于在动态 sql中拼接字符串，可能导致sql注入。
- 系统如何处理异常？
  - 我们自定义一个统一的异常处理器去捕获并处理异常。使用控制器增加注解@ControllerAdvice和异常处理注解@ExceptionHandler来实现。
  - 1) 处理自定义异常
    - 程序在编写代码时根据校验结果主动抛出自定义异常类对象，抛出异常时指定详细的异常信息，异常处理器捕获异常信息记录异常日志并响应给用户。
  - 2) 处理未知异常
    - 接口执行过程中的一些运行时异常也会由异常处理器统一捕获，记录异常日志，统一响应给用户500错误。
  - 在异常处理器中还可以针对某个异常类型进行单独处理。
- 如果javax.validation.constraints包下的校验规则满足不了需求怎么办？
  - 1、手写校验代码 。2、自定义校验规则注解。
- 请求参数的合法性校验如何做？
  - 使用基于JSR303的校验框架实现，SpringBoot提供了JSR-303的支持，它就是spring-boot-startervalidation，它包括了很多校验规则，只需要在模型类中通过注解指定校验规则，在controller方法上开启校验。
- 一个非事务方法调同类一个事务方法，事务无法控制，这是为什么？
  - 注入自己，作为代理对象去调用



- xxl-job的工作原理是什么？xxl-job是怎么样工作？
  - 

- xxl中如何保证任务不重复执行？
  - 

- 任务幂等性如何保证？
  - 它描述了一次和多次请求某一个资源对于资源本身应该具有同样的结果
  - 幂等性是为了解决重复提交问题，比如：恶意刷单，重复支付等。
  - 解决幂等性常用的方案：
    - 1)数据库约束，比如：唯一索引，主键。同一个主键不可能两次都插入成功。
    - 2)乐观锁，常用于数据库，更新数据时根据乐观锁状态去更新。
    - 3)唯一序列号，请求前生成唯一的序列号，携带序列号去请求，执行时在rdis记录该序列号表示以该序列号的请求执行过了，如果相同的序列号再次来执行说明是重复执行。
  - 比如：在数据库视频处理表中添加处理状态字段，视频处理完成更新状态为完成，执行视频处理前判断状态是否完成，如果完成则不再处理。

以上内容整理于 幕布文档