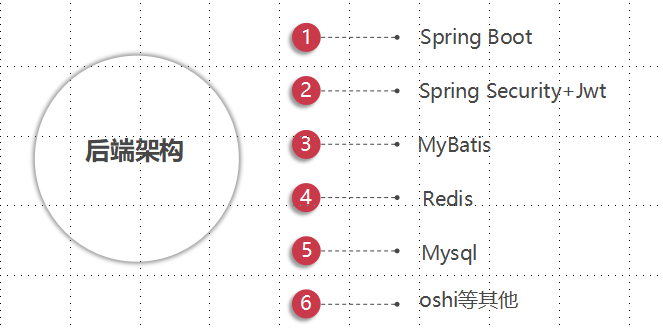
作业答案

第一章:

1-2、打开git仓库，并将源码下载到本地，方便后期学习。

1-3、思考：社会中，还有哪些系统可以进行融合！

1-4、请问本系统使用的主要后端技术栈有哪些？



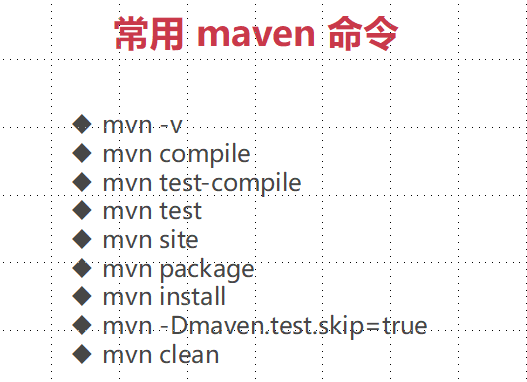
第二章:

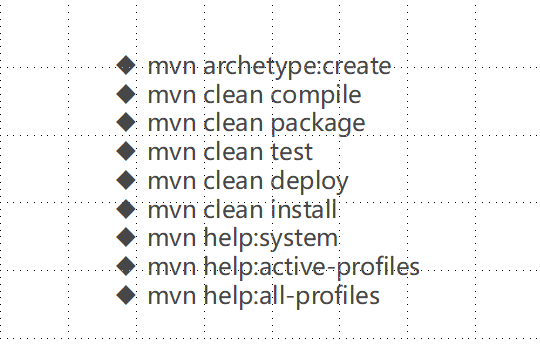
2-1、在本地亲手搭建开发环境

2-2、在本地亲手搭建mysql和redis环境

2-3、请问常用的maven命令有哪些？







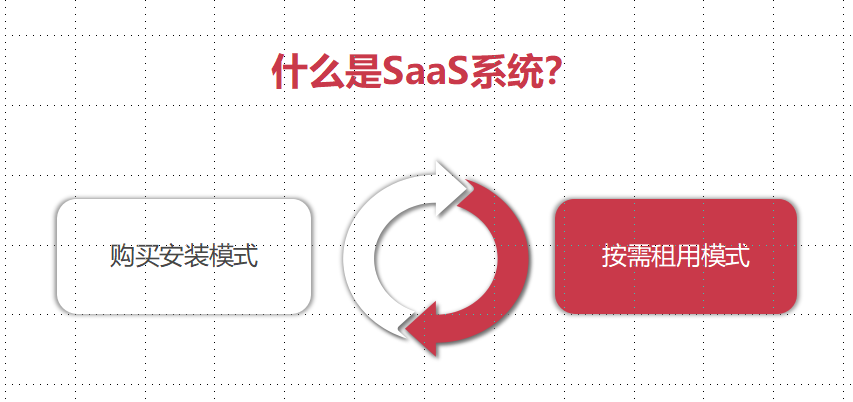
2-4、请为自己的IDEA配置合适的字符编码？

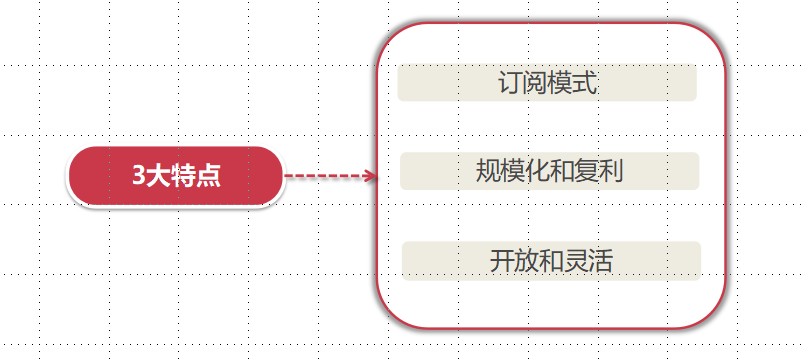
需要自己手动操作练习

第三章:

3-1、请阐述一下自己对saas系统的理解？

以下只是借鉴，按照自己的理解进行阐述：





3-2、saas总体上使用了哪几种数据库架构模型？

行隔离

表隔离

数据库隔离

3-3、数据库设计的方法有哪些？

直观设计方法

规范设计法

计算机辅助设计法

3-4、熟悉powerdesigner，并练习如何设计pdm ！

需要自己手动操作练习

第四章:

4-1、下载代码，并进行启动，注册租户，并登录

4-2、梳理父子工程创建的步骤

4-3、spi机制原理是什么？

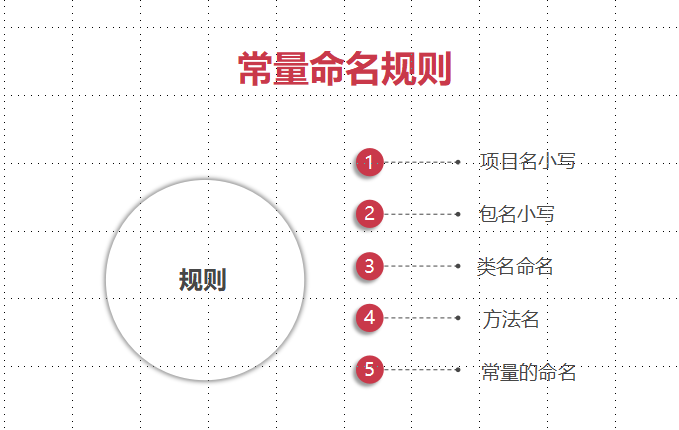
服务提供商接口（SPI）机制是一种将服务接口与服务实现分离以进行解耦的机制，大大提高了程序的可伸缩性。引入服务提供商是一种引入SPI接口的实现，该接口通过易于插入的本地记录发现获得特定的实现类。

4-4、@ConfigurationProperties注解原理是什么？

application.yml 和 [application.properties](https://links.jianshu.com/go?to=http://application.properties" \t "_blank)等配置文件方便我们进行配置的管理，但是我们可以通过@Value的形式去使用，

但是使用@ConfigurationProperties配置之后，可以简化@Value的使用

4-5、请阐述一下常量的命名规则！



4-6、@InitBinder注解的作用是什么？

由@InitBinder注解修饰的方法用于初始化WebDataBinder对象，能够实现：从request获取到handler方法中由@RequestParam注解或@PathVariable注解修饰的参数后，假如获取到的参数类型与handler方法上的参数类型不匹配，此时可以使用初始化好的WebDataBinder对获取到的参数进行类型处理。

4-7、Entity（Entity Class）的作用是什么？

Entity（Entity Class）的作用通常是映射数据表。

4-8、自己梳理一下注册和登录的流程

4-9、cod、data、message来存放什么信息？

状态码放到code里面，数据放在data里面，message来存放备注信息。

4-10、邻接表的优缺点是什么？

优势

结构简单；

缺点

如果不递归使用所有父节点、所有子集，则无法查询节点

4-11、请手动梳理TableSupport的逻辑流程

4-12、手动梳理RedisCache的重点逻辑

4-13、什么是字符集？

简单地说，字符集就规定了某个文字对应的二进制数字存放方式（也就是编码）和某串二进制数值代表了哪个文字（也就是解码）的转换关系。

4-14、Converter接口怎么定义？

Converter接口的定义：

public interface Converter<S, T>{

T convert(S source);

}

4-15、不看源码，自己手动编写字符串格式方法

4-16、如何比较两个枚举成员定义的顺序？

答：compareTo(Enabled enabled) 比较两个枚举成员在定义时的顺序

4-17、自己手动梳理全局异常定义的方法步骤

第五章:

5-1、请阐述一下XSS攻击的基本原理！思考如何防止XSS攻击！

XSS攻击是一种黑客通过特殊方式将恶意JavaScript脚本插入网页的攻击，导致cookie数据盗窃、会话劫持、网络钓鱼以及用户浏览网页时对浏览器的其他攻击。

XSS的原理是，web应用程序将用户提交数据的代码限制与JS脚本混淆，导致浏览器将用户输入作为JS代码执行。XSS的目标是浏览器一端的普通用户。

XSS的防御

（1）编码：用户输入数据的HTML实体编码

（2）过滤器：删除用户上传的DOM属性，如oneerror、删除用户上传样式节点、script节点、iframe节点等。

（3）校正：避免直接编码HTML实体，使用DOM PRASE转换，并修复不匹配的DOM标记。

总结

这节课我们学习了XSS的核心原理，也了解了如果进行防御，防御方法是我们学习的重点，接下来我们还自定义了xss效验注解，也对xss效验注解进行了实现，这块内容我们要会灵活应用

5-2、请阐述一下springbean。

Spring Bean是Spring最基本的单元，其中构成应用程序核心并由Spring IOC容器管理的对象称为Bean。Bean是由Spring IOC容器实例化、组装和管理的对象。

5-3、梳理图片处理的方法逻辑

5-4、梳理EscapeUtil的重点方法和逻辑

5-5、不看源码，自己手动编写http发送方法

5-6、不看源码，自己手动编写获取IP方法

5-7、XSSF 的功能模块作用是什么？

XSSF － 提供读写 Microsoft Excel OOXML XLSX 格式 (Microsoft Excel XML (2007+)) 文档的功能。

5-8、不看讲解，用自己的语言复述Reflect工具里面常用的方法及作用

5-9、Base64原理是什么？

Base64编码是一种基于64个可打印字符的二进制数据表示的表示方法。它通常用于传输、存储的二进制编码数据！它可以用作电子邮件的传输编码。当使用记事本打开这些exe、jpg和pdf文件时，我们都会看到一片混乱代码，因为二进制文件包含许多无法显示或打印的字符。因此，如果你想让记事本这样的文字处理软件能够处理二进制数据，你需要一种二进制到字符串的转换方法。Base64是最常见的二进制编码方法。

5-10、 ApplicationContext是什么？

BeanFactory是Spring底层IOC容器的实现，在实际应用场景中，BeanFactory容器很简单，它不适合生产环境，我们通常会选择ApplicationContext。ApplicationContext是著名的Spring应用程序上下文，它不仅继承了BeanFactory系统，还提供了更高级的功能，更适合我们的正式应用程序环境。它有如以下功能：

5-11、谈一下sql注入的原理

SQL注入是什么呢，它是指web应用程序对用户输入数据的合法性没有进行判断，导致攻击者在web程序中原先定义好的查询语句的结尾上，添加额外的SQL语句，借此来实现欺骗数据库服务器，执行非授权的查询，得到相应的数据信息。

5-12、Redis的incr（key）API用于什么？

Redis的incr（key）API用于将key的值进行递增，并返回增长数值。

5-13、手动梳理Arith的重点逻辑

5-14、手动梳理字典缓存、字典标签的重点逻辑方法

5-15、已面试者的身份，口述一下对springSecurity原理的理解

SpringSecurity 本质其实是一个过滤器链，它采用的是责任链这个设计模式，因此它有很长的一条过滤器链。

(Authentication和用户授权(Authorization两部分，这两点也同样是SpringSecurity的核心功能。

客户端发起一个请求，并且进入Security 过滤器链。

当到LogoutFilter时，确定它是否是登出路径。如果是登出路径，就转到logoutHandle。如果登出成功，就转到logoutSuccessHandle进行成功登出处理。如果登出失败，就转到ExceptionTranslationFilter；如果它不是登出路径，它将直接进入下一个过滤器。

当到 UsernamePasswordAuthenticationFiltelr这个过滤器的时候，就会判断是是不是登录路径，如果是，就进入这个过滤器进行登录操作，假如登录失败的话，就到 AuthenticationFailureHandler 登录失败处理器处理，假如登录成功则到 AuthenticationSuccessHandler 登录成功处理器处理，如果不是登录请求，就不进入这个过滤器。

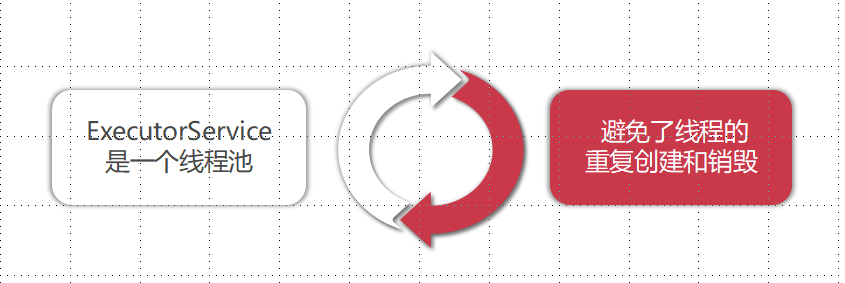
当到 FilterSecurityInterceptor 时会拿到 uri ，然后根据 uri 去找对应的鉴权管理器，通过鉴权管理器做鉴权工作，鉴权成功了就到 Controller 层，否则就到 AccessDeniedHandler 鉴权失败处理器处理。

5-16、自己手动编写字符串渲染处理中的重点方法

5-17、startsWith 这个方法作用是什么？

检查字符串是不是以指定字符开头

5-18、ExecutorService是做什么用的？



5-19、手动梳理生成验证码的逻辑步骤

第六章:

6-1、请简单阐述一下对Spring Security授权机制的核心原理的理解！

以下只是借鉴提示，按照自己的理解进行阐述：

在spring-boot+security+jwt框架中，权限控制是通过一系列的内置或自定义过滤器来实现的。

FilterSecurityInterceptor是过滤器链中的最后一个筛选器，它根据资源权限配置来确定当前请求是否有权访问相应的资源。如果访问受到限制，将引发相关的异常，由此产生的异常将被上一个过滤器ExceptionTranslationFilter捕获并处理，

FilterSecurityInterceptor是Spring Security身份验证和授权中的最后一个过滤器。在这个过滤器之后是真正接口请求的服务，随后将被执行到SpringMVC的拦截器链中。

6-2、@DS在方法和类上，方法上注解是否优先于类上注解？

@DS可注解在方法或者类上，同时存在就近原则，方法上注解优先于类上注解。

6-3、梳理LogAspect配置的逻辑

6-4、限流在什么的场景下会用到？

限流在高并发的场景下会用到

6-5、梳理CaptchaConfig配置的关键属性作用

6-6、手动梳理Redis限流脚本的业务逻辑

6-7、@Configuration的作用是什么？

@Configuration用于定义配置类，可替换xml配置文件，被注解的类内部包含有一个或多个被@Bean注解的方法，这些方法将会被AnnotationConfigApplicationContext或AnnotationConfigWebApplicationContext类进行扫描，并用于构建bean定义，初始化Spring容器。

6-8、梳理DynamicDataSource的代码逻辑

6-9、梳理防止重复提交拦截器的方法逻辑

6-10、对照代码，手动梳理多租户数据源拦截器TenantInterceptor开发的方法逻辑

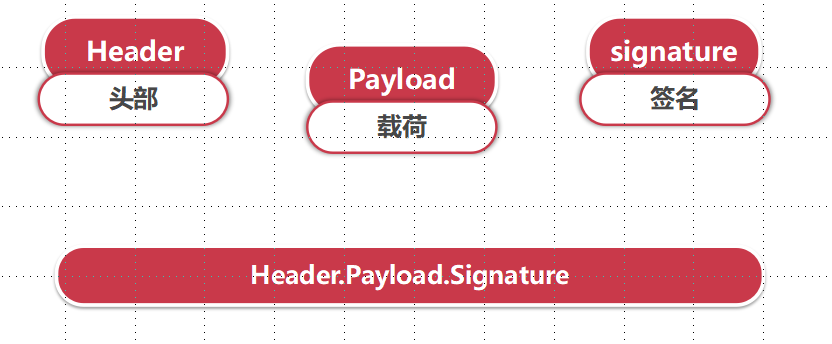
6-11、异步是什么？

异步是同步的相对概念。在我们正在学习的传统单线程编程中，程序的执行是同步的（同步并不意味着所有步骤都同时运行，而是指步骤在控制流序列中按顺序执行）。另一方面，异步的概念则是不保证同步，这意味着异步进程的执行将不再与原始序列有顺序关系。

简单地说，这意味着同步执行遵循您的代码顺序，而异步执行不遵循代码顺序，从而提高了执行效率。

6-12、思考AsyncManager和AsyncFactory的关联

6-13、JWT由哪三部分组成？



6-14、程序安全机制的入口是什么？

身份验证入口AuthenticationEntryPoint，这块是我们程序安全机制的入口

6-15、LogoutSuccessHandler椒做什么用的？

Spring Security框架提供了一个非常周到的接口，称为LogoutSuccessHandler，专门用于处理用户登出成功请求。当然，对于LogoutSuccessHandler接口，Spring Security框架有一些默认的实现，也可以进行自我扩展。

同用户成功登录的业务场景，我们还需要通过电子邮件、短信或微信通知用户何时何地退出了系统。甚至可以通知用户在登录过程中执行了哪些功能和操作。同时，这些信息也可以记录在日志文件中。所有这些需求都可以通过实现LogoutSuccessHandler接口来实现。

6-16、梳理HardwareAbstractionLayer在项目中在应用

6-17、@RestControllerAdvice相当于哪两个注解的组合？

@RestControllerAdvice相当于@ControllerAdvise和@ResponseBody的组合。

6-18、梳理Permission的关键逻辑

6-19、以面试者的身份，口述一下对AuthenticationManager的理解。

AuthenticationManager（接口）是与身份认证相关的核心接口，也是发起认证的起点。在实际需求中，我们可能会允许用户使用用户名和密码、电子邮件和密码、电话号码和密码登录。因此，AuthenticationManager通常不直接执行身份验证。AuthenticationManager接口的常用实现类ProviderManager在内部维护一个存储多种认证方式的list列表。事实上，这是委托者模式应用。

也就是说，始终只有一个核心认证入口：AuthenticationManager。不同的身份认证方式包括用户名+密码（UsernamePasswordAuthenticationToken）、电子邮件+密码和电话号码+密码登录，对应三个AuthenticationProvider。熟悉Shiro的朋友可以将AuthenticationProvider理解为Realm。在默认策略下，只需要通过一个AuthenticationProvider身份认证即可被视为成功登录。

ProviderManager中的List列表，会按照次序进行身份认证。如果身份认证成功，将立即返回。如果身份认证失败，它将返回null。下一个AuthenticationProvider将继续尝试身份认证。如果所有身份认证程序都无法成功进行身份认证，ProviderManager将抛出ProviderNotFoundException异常。

6-20、梳理一下用户权限Permission处理，思考是如何实现获取角色数据权限和菜单数据权限功能的

6-21、对照代码，手动梳理Tenant注册模块及租户数据源配置，实现TenantDatabase初始化，这部分的代码和业务逻辑。

6-22、对照代码，手动梳理Tenant数据库脚本开发，及初始化脚本执行方法这块的代码逻辑

6-23、以面试者的身份，谈一下对token的理解和运用，思考怎么防止使用中token过期和同个浏览器输入网址防止数据互串

Token是指客户端向服务器频繁请求数据，服务器频繁查询数据库中的用户名和密码，进行比较，确定是否正确，并进行相应提示。在这种情况下，出现了Token。

Token是服务器生成的字符串，作为客户端请求的Token。第一次登录后，服务器生成一个Token并将其返回给客户端。将来，客户端只需要携带此Token即可请求数据，而无需再次携带用户名和密码。

Token的目的是减轻服务器压力，减少频繁的数据库查询，使服务器更加健壮。

客户端请求使用用户名和密码登录；

服务器接收验证用户名和密码的请求；

验证成功后，服务器将发布一个Token并将其发送给客户端；

在接收到Token之后，客户端可以将其存储在cookie中；

客户端每次向服务器请求资源时，都需要携带服务器发布的Token；

服务器接收请求并验证客户端请求中携带的Token。如果验证成功，它将向客户端返回请求的数据。

推荐使用refresh token解决token过期问题，通常情况下，Refresh Token 的有效期会比较长，而 Access Token 的有效期比较短，当 Access Token 由于过期而失效时，使用 Refresh Token 就可以获取到新的 Access Token，如果 Refresh Token 也失效了，用户就只能重新登录了。

在 JWT 的实践中，引入 Refresh Token，将会话管理流程改进如下。

客户端使用用户名密码进行认证

服务端生成有效时间较短的 Access Token（例如 10 分钟），和有效时间较长的 Refresh Token（例如 7 天）

客户端访问需要认证的接口时，携带 Access Token

如果 Access Token 没有过期，服务端鉴权后返回给客户端需要的数据

如果携带 Access Token 访问需要认证的接口时鉴权失败（例如返回 401 错误），则客户端使用 Refresh Token 向刷新接口申请新的 Access Token

如果 Refresh Token 没有过期，服务端向客户端下发新的 Access Token

客户端使用新的 Access Token 访问需要认证的接口

如果使用session存储用户数据，可能会出现串数据的现象，对此可以搞一个全局过滤器，如果之前有用户A [session](https://so.csdn.net/so/search?q=session&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_35172715/article/details/_blank)存在了，新标签页进入该系统还是以用户A的身份进入系统。

就算用户直接进入登陆页面也自动进入首页。

第七章:

7-1、前端代码中vue.config.js文件的作用主要是什么！

vue.config.js 是一个 Vue CLI 项目的配置文件，用于配置各种开发和构建相关的选项。

7-2、手动梳理Tenant数据初始化和注册登录的逻辑关系

7-3、手动编写login和register关键前端view代码

7-4、手动编写个人中心和密码管理关键后端接口代码

7-5、手动编写个人中心和密码管理关键前端view代码

7-6、以User分页为例，梳理startPage代码逻辑功能实现

7-7、手动重写添加、修改、删除、导出用户功能mvc三层相关代码

7-8、手动重写添加、修改、删除、导出用户功能view和api相关代码

7-9、手动重写添加角色功能授权相关代码

7-10、手动重写添加role功能授权相关代码

7-11、手动重写buildDeptTree开发相关后端代码

7-12、手动重写Tree相关前端代码

7-13、Mybatis基础xml配置文件中resultType表示什么？思考和resultMap区别是什么？

resultType是直接表示返回类型的。总得来说resultType是直接表示返回类型的，而resultMap则是对外部ResultMap的引用。根据mybatis的官方文档，是这么说的：从这条语句中返回的期望类型的类的完全限定名或别名。 注意如果返回的是集合，那应该设置为集合包含的类型，而不是集合本身。可以使用 resultType 或 resultMap，但不能同时使用。下面我来举个栗子~  
resultType：  
当提供的返回类型属性是resultType时，MyBatis会将Map里面的键值对取出赋给resultType所指定的对象对应的属性。实际上MyBatis的每一个查询映射的返回类型都是ResultMap，只是当提供的返回类型属性是resultType的时候，MyBatis对自动的给把对应的值赋给resultType所指定对象的属性。

7-14、手动重写spread operator相关前端代码

7-15、思考内链域名特殊字符替换在其他地方的用处

7-16、操作菜单图标相关功能，并重写菜单图标使用的代码

7-17、梳理构建前端路由所需要的菜单和树结构代码逻辑

7-18、手动重写IconSelect和树形选择器组件的样式相关前端代码

7-19、手动重写refreshCache相关后端代码

7-20、手动重写refreshCache相关前端代码

7-21、手动重写SysConfig相关后端代码

7-22、手动梳理request拦截器代码逻辑

7-23、梳理Notice及租户管理业务层实现逻辑

7-24、手动编写Notice及租户管理前端关键代码

7-25、手动重写清空、删除、导出系统Log记录和Captcha操作处理关键代码

7-26、手动重写Log操作时间和操作详细关键代码

7-27、手动梳理通用请求处理代码逻辑

7-28、手动梳理响应拦截器代码逻辑

第八章:

8-1、手写Customer列表管理、移入公海、转移客户、跟进等功能后端接口关键代码

8-2、手写Customer列表管理、移入公海、转移客户、跟进等功能前端接口关键代码

8-3、手动编写成交客户、我的订单、订单列表及相关业务关系后端接口关键代码

8-4、手动编写成交客户、我的订单、订单列表及相关业务关系前端接口关键代码

8-5、手动编写客户公海及领取sql部分代码

8-6、手动编写客户公海及领取view部分代码

第九章:

9-1、按照自己的理解手动编写一个Stream管道流的map操作

9-2、不看源码，重写MyBatis动态sql之foreach标签构建in语句

9-3、按照面试者的身份，谈一下Mybatis 作用域和生命周期

生命周期和作用域，是至关重要的，因为错误的使用会导致非常严重的并发问题。

SQLSessionFactoryBuilder：

一旦创建了SQLSessionFactory，就不再需要它了

局部变量

SQLSessionFactory：

说白了就是可以想象为：数据库连接池

SQLSessionFactory一旦被创建就应该在应用的运行期间一直存在，没有任何理由丢弃它或重新创建另一个实例。

因此SQLSessionFactory的最佳作用域是应用作用域

最简单的使用方式就是单例模式或者静态单例模式

SQLSession：

连接到连接池的一个请求！

SQLSession的实例不是线程安全的，因此是不能被共享的，所以它的最佳作用域是请求或方法作用域。

用完之后需要赶紧关闭，否则资源会被占用！

9-4、SqlSource代表什么？

SqlSource就代表配置的SQL资源。

第十章:

10-1、编写Clues列表、转成客户、添加、展示和转移部分后端关键代码

10-2、编写Clues列表、转成客户、添加、展示和转移部分前端关键代码

10-3、梳理后端统计报表-Clues、Customer和公海代码逻辑

10-4、梳理重写后端统计报表-动态、跟进提醒关键代码

10-5、梳理重写统计报表-Clues、Customer、公海、动态和跟进提醒前端关键代码

第十一章:

11-1、@Transactional注解标注的方法表示一个什么方法？

@Transactional注解：这个注解表示该方法是一个事务方法，会在方法执行期间启动一个事务来管理数据库操作。

11-2、手动重写@PreAuthorize 部分的代码

11-3、用自己的语言谈一下mybatis动态条件查询的理解和应用

11-4、手动重写StorageFlow批量操作部分的代码

11-5、手动重写TaxInfo和StockOrder部分关键代码

11-6、手动梳理saas系统注册登录关键业务流程

11-7、按照自己的理解，进一步优化顶部菜单栏的前端组件逻辑及数据处理、仪表盘面板和网格布局

第十二章:

12-1、按照自己的语言复述一下前后端分离项目部署时注意事项及关键点

12-2、打开云服务器官网，浏览并搭配一个适合自己的服务器配置

12-3、自己动手进行Server基础环境搭建，及安装FileZilla等服务器连接工具12-4、手动搭建mysql数据库应用环境

12-5、手动配置安装Redis+Nginx等部署软件

12-6、手动打包及在服务器上部署前后端分离之后端springboot项目

12-7、build及在服务器上部署前后端分离之前端vue项目

12-8、按照面试者的角度，谈一下如何能第一时间高效解决线上的BUG

第十三章:

13-1、结合自己的实际，借鉴课程经验，总结自己的编程学习方法

13-2、按照自己的理解，梳理企业级软件开发流程和项目开发经验

13-3、结合自己的实际，借鉴课程经验，优化自己的简历

13-4、模拟面试形式，找个人当面试官，按照课程中的问题进行提问，自己作为面试者进行回答

13-5、结合自己的实际，借鉴课程经验，总结面试经验指南，以后面试前翻看一下