JavaWeb

Servlet

1. 问题：编写Web应用程序即处理HTTP请求并响应，但处理TCP连接和解析HTTP协议效率很低；
2. 解决：麻烦的事交给Web服务器去做，基于Servlet接口编写HTTP请求处理逻辑，Web服务器实例化Servlet并运行；
3. 实现：
    - 通过注解将路径映射到Servlet；
    - Servlet中的字段要注意多线程安全的问题；
    - 重定向与转发；
    - Session和Cookie；

JSP

1. 问题：通过Writer输出HTML文本比较麻烦；
2. 解决：使用JSP编写HTML，内嵌JAVA代码；
3. 实现：
    - JSP会被Web服务器编译成Servlet；
    - 无需配置路径；

MVC开发

1. JavaBean充当模型，Servlet负责业务逻辑，JSP负责页面渲染；
2. MVC高级：
    - 问题：Servlet过于底层，需要实现Servlet接口；
    - 解决：创建一个接收所有请求的Servlet，使用普通JAVA类作为Controller，而非强制继承Servlet;
    - 一个MVC框架是基于Servlet基础抽象出更高级的接口，使得上层基于MVC框架的开发可以不涉及Servlet相关的HttpServletRequest等接口，处理多个请求更加灵活，并且可以使用任意模板引擎，不必使用JSP。

Filter

1. 问题：多个功能组件具有相同的逻辑判断，代码冗余重复；
2. 解决：将重复的逻辑放到Filter中，对HTTP请求进行统一的预处理，适用于日志、登录检查等；

Listener

1. 监听Web应用程序的生命周期，获取某些对象的创建、销毁等事件。

Spring

1. 作用：
    - 简化开发
    - 框架整合

IoC

1. 问题：耦合度高；组件依赖其他组件功能时，需要内部通过New创建对象，管理对象的生命周期；浪费资源，多个组件可共享同一个对象。
2. 解决：控制翻转，程序不再采用new而是**依赖注入**的方式获取对象，控制权交给IoC容器，而不在程序本身。就是说对象的创建和配置交给IoC容器，程序只需关注使用。
3. 实现：
    - IoC创建的对象称为Bean；
    - 通过XML文件配置Bean，把XML文件转换为JAVA类，并通过注解配置Bean更简单；
    - Bean的实例化方法：构造方法、静态工厂、实例工厂；
    - 依赖注入的方式：构造方法、setter方法、自动装配（反射实现）；
    - 依赖注入的内容：Bean、基本类型、配置文件、第三方Bean；

AOP

1. 问题：不同的方法可能包含相同的子功能（打印日志、登录检查等），出现代码冗余的情况；
2. 解决：将相同的功能抽取出来，再不改变原始代码的情况下增强功能，即无侵入式编程；
3. 实现：
    - 相同的功能称为通知，要增强功能的方法就是切入点，切面的作用就是绑定通知和切入点；
    - **核心为代理，创建目标对象的代理对象**；
    - 切入点表达式、五种通知类型；
    - 应用：作为拦截器，进行权限检查、登录检查、参数格式化等；

整合第三方框架MyBatis、JUnit等

1. 作为第三方Bean被Spring感知，配置文件转为JAVA类；

Spring事务

1. 作用：在数据层或业务层保障数据一致性；
2. 实现：事务管理员和协调员，事务的传播；
3. 应用：转账；

SpringMVC

1. 问题：Servlet开发效率不高，SpringMVC编程更简单，效率更高。
2. 实现：
    - Web程序启动过程：启动Tomcat服务器、根据配置初始化容器、加载SpringMVC配置类、加载Bean、执行方法；
    - 单次请求处理过程：Tomcat服务器捕获请求，交给SpringMVC处理，解析路径并执行相应方法；

MyBatis

1. 问题：JDBC编程繁琐，需要硬编码，手动配置，手动封装结果；
2. 解决：采用MyBatis持久层框架，简化JDBC开发；
3. 主要工作：将数据库记录映射为java bean
4. 采用代理模式跟踪Java bean的修改
5. 二级缓存：一级缓存在session范围内
6. 实现：

- 采用配置文件、SQL映射文件+Mapper接口；
- 查询结果自动转换为实体类；

SSM整合

1. AOP用于异常处理；
2. 拦截器；

Maven

基础

1. 项目管理、包管理
2. 依赖管理（依赖传递、可选依赖、排除依赖）
3. 生命周期与插件

高级

1. 模块拆分
2. 聚合（整合模块成项目）和继承（简化配置、版本管理）

Spring Boot

1. 作用：简化Spring配置、应用搭建

瑞吉外卖：

1. 登录
    - session保存用户信息；
    - 过滤器实现登录拦截（@webfilter、实现filter接口）；
2. 新增员工
    - 密码进行MD5加密处理；
    - AOP实现全局异常（唯一值重复）捕获（@ContollerAdvice、@ExceptionHandler）；
    - 自定义业务异常
3. 分页查询员工信息
    - MybatisPlus分页构造器；
4. 启用/禁用用户
    - 权限控制（v-if）
    - js丢失Long精度
    - 自定义消息转换器，转换字段格式，json对象与java对象互转

5. 编辑员工信息
    - MybatisPlus自动填充公共字段（修改人、修改时间）@TableField；
    - 自动填充时采用ThreadLocal获取session信息（ThreadLocal实现线程内资源共享）；
6. 删除分类
    - 检查是否有关联的菜品
7. 文件上传
8. 新增菜品、分页查询菜品、修改菜品
    - 操作多个表，定义DTO封装前端数据
    - 加入事务，保证数据一致性。
9. 新增套餐、分页查询套餐、删除套餐
10. 菜品展示、购物车、下单
    - 计算金额使用AtomicInteger保障原子性
11. 项目部署
    - 手工部署vs利用Shell脚本自动部署
12. 缓存优化
    - 用户数量过多，频繁访问数据库，系统性能下降
    - redis String类型缓存数据，Key作为菜品类型，Value作为对应类型的菜品信息列表。
    - **增删改操作需要清除缓存**
    - spring cache框架简化缓存功能的实现，整合的缓存产品自己选
    - 实体类需要实现序列化接口
13. 读写分离（主从复制）
14. nginx部署静态资源、实现反向代理（正向和反向代理）、负载均衡
15. 前后端分离（YApi、Swagger、前后端分离部署）
16. 点赞（采用set记录）
    - redis事务
    - 分布式存在session共享问题，可以用redis解决
17. 消息队列（解耦、异步、削峰）

大数据场景题

1. 海量数据：位图 + hash分治 + 堆
2. 抢红包怎么设计：单机（volatile + CAS），多机（分布式锁）
3. 消息队列来缓冲高并发的请求

常见智力题

智力题：阿秀的学习笔记。

面试软实力：口语表达+着装。

车辆定损系统

用户表、定损单表、车辆图片表

工作流程

用户注册登录→添加车辆信息→上传多张受损车辆图片 →生成定损单→检测算法返回结果→更新定损单、生成损伤表 →返回给用户； 员工修改、确认损伤； 管理员管理后台数据。

1. app端，用户使用；
2. web端，后台控制数据

数据库表

1. 用户表：用户id 邮箱 姓名 年龄 性别 。。。
2. 员工表：员工id 账号 密码 身份
3. 定损单表：定损单id 用户id 车牌号 上传时间 状态 定损的描述信息、修改人id、修改时间、
4. 车辆图片信息表：图片id 定损单id 图片文件路径 检测结果json 状态 修改人id 修改时间

项目可能遇到的问题：

1. 邮箱注册登录的实现，为什么使用MD5进行加密？过滤器实现登录状态检查
    - 邮箱启用SMTP服务，Spring Email配置邮箱参数，使用JavaMailSender发送邮箱验证码。
    - MD5是不可逆的加密算法，加密后的数据无法破解，防止数据库的密码泄露。密码设置简单可以通过密文反向查询，通过加入盐值共同加密。
    - @WebFilter注解，实现Filter接口，定义路径通配符，重写doFilter方法，调用filterChain.doFilter方法放行
    - web.xml配置顺序，springboot的话采用filterRegistrationBean设置优先级
2. websocket如何实现服务器通信，很多人上传图片怎么办？凉拌，阻塞着等。
    - WebSocket配置类，设置接受的url路径。
    - 继承TextWebSocketHandler接口实现处理器。三个ConcurrentHashMap，一个存储WebSocketSession，一个<string，thread>匹配字符串和thread，一个<string, result>匹配字符串和检测结果。
    - 实现HandshakeInterceptor实现websocket拦截器，url携带token判断是不是GPU服务器。
    - GPU服务器作为客户端连接应用服务器，应用服务器发送携带当前线程和图片路径的消息，GPU服务器返回检测结果和当前线程。

3. 事务管理保障数据的正确性

    - 上传车辆图片同时更新两个表
    - 删除定损单同时删除图片表信息

4. Redis缓存邮箱验证码，缓存热点数据，Redis的持久化

5. 心跳包检查图片编辑状态

    - ConcurrentHashMap存储过期时间，收到心跳包刷新时间。
    - 两个人同时进入编辑页面。
    - 用redis的setnx对图片id加锁，首先判断是否存在key，存在则判断用户ID，如果是持锁对象则续期锁，不存在则用setnx设置key值。
    - key是图片id，value是用户id。setnx和set expire必须是原子操作，否则可能过期时间设置失败导致死锁。
    - 查询锁和删除锁必须是原子操作，否则可能会删别人的锁。
    - 判断id和更新缓存也必须是原子操作。

    if key existed: --lua-- if value == userid: return update expire else： return fail --lua-- else: if setnx + set expire: return success else: return fail

6. AOP切面在嵌套方法中调用不生效

7. 为什么DAO和Service层要用接口

    - 方便分离各层，上层可直接调用接口方法
    - 支持多实现
    - 支持AOP动态代理

8. Mysql 主从复制 读写分离导致的数据不一致问题

    - 同步复制
    - 利用缓存临时记录写操作的数据，过期时间设置为主从时延。

9. 项目中用到了什么设计模式

- 不同用户不同功能，策略模式

10. 做项目遇到的问题?

- 过滤器解决跨域问题，也可以用nginx代理解决
- spring 和 springMVC 容器不同，父子容器的关系，配置拦截器没用
- js丢失Long精度, 自定义消息转换器，转换字段格式，json对象与java对象互转

11. 有哪些收获?

- 不是一次考虑好，而是逐渐完善的。
- 复杂功能可以拆分成多个请求。
- 重点：提高系统性能、保障系统安全、处理异常；
- 每个员工有对应的车损订单处理，建索引的话可以员工id和车损id建立联合索引，字段信息少的话可以考虑覆盖索引。

12. 静态资源映射，请求图片
13. Mysql隔离级别设置为读提交

常见面试题

1. MySQL
    - 存储引擎(NDB用于集群，能不做集群就不做，分布式事务很麻烦)
    - 索引
    - 事务
    - 锁
2. Redis 数据类型、过期策略、淘汰策略、缓存穿透、缓存击穿、缓存雪崩、分布式锁
3. Spring IOC AOP MVC
4. 三要素：是什么，有什么作用/例子/过程，原理。

补充

1. 设计模式

- 创建型模式
    - 工厂方法：一个工厂创建不同的产品，（例如包装类型的 value of, hibernate框架切换数据库）。
    - 抽象工厂：一个抽象工厂，多个抽象产品，不同工厂负责不同抽象产品。
    - 单例：一个进程里有且仅有一个实例。
    - 原型：复制对象实例。
- 结构型模式：
    - 适配器：转换接口，inputStreamReader
    - 装饰器：解决子类过多的问题，通过组合获取功能。可以在运行期间增加功能。BufferedInputStream、DataInputStream，装饰器类需要跟原始类继承相同的抽象类或者实现相同的接口
    - 代理模式：适配器改变接口，代理不改变。装饰器为了增强功能，代理为了控制对象。
- 行为型模式：
    - 观察者模式：发布订阅模式，分离观察者和被观察者的耦合关系。NIO的文件目录监听服务
    - 模板方法：定义骨架。

2. 项目框架八股

- srping: IOC 依赖注入 自动装配（反射原理） AOP 三级缓存 事务（被aop增强的方法都应该是public的，而不能是private的）
- Mybatis
- 基于HTTP实现断点续传怎么做

3. Java八股
4. Linux常用命令
5. 了解其他中间件
6. 红黑树：范围查找需要中序遍历，平均复杂度更低；B+树：磁盘IO次数少；跳表：增删操作开销低，占用内存更少

项目

环境安装

1. maven——主要功能：管理项目和jar包依赖，进行项目编译、测试（生命周期）。
    - 下载解压缩；
    - 配置环境路径为bin；
    - conf/setting.xml配置本地仓库和镜像仓库（要注意配新版仓库地址）。
2. idea
    - 安装；
    - Build/Build Tools/Maven配置Maven；
    - 问题：生成maven项目没有src文件夹。原因：不要用JDK17以上版本！！！
3. springboot
    - 主要功能：起步依赖、自动配置、端点监控；
    - IoC：管理Bean的生命周期、自动注入DI。
4. springMVC
    - http文档：developer.mozilla.org；
    - Thymeleaf：以HTML为模板，不同于JSP文件。
5. MySQL
    - 下载服务器和客户端(WorkBench)；
    - mysql根目录创建my.ini配置mysql；
    - 配置环境变量为bin；
    - 初始化命令：`mysqld --initialize --console` 必须管理员身份打开，必须cd到对应Bin目录下执行！
    - 安装服务：`mysqld install`
    - 启动服务：`net start mysql`
    - 修改密码：`alter user root@localhost identifide by 'password';`
    - WordBench配置默认连接和数据库
6. Mybatis
    - 文档：www.mybatis.org/mybatis-3
    - 导入mysql和mybatis依赖，注意版本！
    - 配置application.properties
    - application.properties设置日志级别为debug
    - 新建Entity类、Mapper接口、Mapper映射文件
7. kafka
    - 下载安装
    - 配置zookeeper.properties/dataDir, server.properties/log.dirs
    - 使用Bin文件夹下的命令启动zookeeper和kafka
8. Git
    - IDEA Version Control配置git.exe路径
    - IDEA VCS 初始化本地仓库
    - IDEA VCS commit 提交暂存区的修改到本地仓库
    - IDEA VCS pull/push 拉取/提交到远程仓库
    - `git config --list` 查看用户配置
    - `git config --global user.name "<name>"` 配置全局用户信息
    - `git config --global user.email "<email>"`

    - `git init` 创建本地仓库
    - `git clone <url>` 克隆
    - `git status` 查看本地仓库状态
    - `git add *` 添加文件到暂存区
    - `git commit -m "<message>"` 提交暂存区到本地仓库
    - `git remote add <name> <url>` 添加远程仓库连接
    - `git branch --set-upstream-to=<remote>/<branch> <branch>` 建立本地分支和远程分支的链接
    - `git fetch <remote> <branch>` 抓取远程仓库指定分支
    - `git pull <remote>` 抓取远程仓库所有分支并merge到本地仓库
    - `git push <remote> <branch>` 将本地指定分支推送到远程仓库
9. 常用包

- lombok
- commons-lang3
- spring-boot-starter-mail

项目经验

1. 业务经验
    - **功能拆分成多个请求，分步实现**
    - 方法再简单也不要跨层调用，保证controller->service->dao
    - 对于复杂的前后端数据可以写个pojo进行封装，如分页信息，同时setter方法判断字段是否合法。
    - 重定向，以防服务器端功能耦合，例如删除后不要调用查询功能。
    - **整合框架三部曲：导依赖包、Application.properties配置、写config配置类（有必要的话）**
    - 根据需要判断工具类是否需要被容器托管。
    - 多做判断，多考虑各种情况，出现Bug比较恶心。
2. 项目调试
    - 查看状态码
    - 设置日志级别
    - 服务器和客户端断点
    - **可以设置日志级别和日志输出文件。**
3. 前端(Thymeleaf)
    - 相对路径采用th:src=@{path}
4. SpringMVC
    - 调用方法前会自动实例化Model对象，同时会把其他对象装到Model中
5. Mapper接口&映射文件
    - 方法考虑多种情况，采用动态sql实现。
    - 动态sql最好用@Param取别名。
    - 写sql注意字段不要写错，因为ide没提示。
    - **超大分页查询会走全表扫描，采用子查询优化，先查出id范围（因为查id是覆盖索引，不用回表效率高）。**
6. 功能点
    - 邮件功能；
    - **如何使用cookie和session进行会话管理？分布式session存在什么问题？如何解决？**
    - Kaptcha生成验证码；
    - 拦截器实现登录拦截，采用自定义注解标注哪些方法需要拦截；

- 前缀树实现敏感词过滤。
- **Spring事务处理**
- @ControllerAdvice实现统一异常处理；
- AOP实现统一日志记录；
- Redis存储验证码、登录凭证、用户信息、实现点赞关注等功能
- Kafka实现系统通知
- Spring Security实现权限控制，提高系统安全性
- Spring Quartz实现分布式定时任务