**软件测试笔记（7/4\_8.7-9）**

**1-1-3 软件测试基础理论**

**测试基础理论**

* 功能测试、接口测试、性能测试
* 黑盒测试、白盒测试、灰盒测试
* 易用性测试、UI界面测试、兼容性测试
* 冒烟测试、回归测试
* 单元测试、集成测试、验收测试……

**测试文档的编写**

* 测试计划
* 测试策略
* 测试方案
* 测试用例
* 测试报告

**Web功能测试**

* 功能 + 业务流程
* 理解需求，提取功能点
* 梳理业务流程图
* 测试用例的方法
* 边界值
* 等价类
* 场景法
* 错误推测法
* 执行测试
* BUG的类型
* BUG的状态
* BUG管理工具
* 测试报告的输出
* 测试范围
* 缺陷模块分析
* BUG严重等级分析
* 风险分析
* ……

**App专项测试**

* 弱网测试
* 权限测试
* 资源争用
* 消息推送
* 安装 / 卸载 / 更新
* 离线测试
* ……

**Linux基本命令**

**MySQL**

* 数据库
* 数据库的语句（增删改查）

**接口测试**

* Postman

**性能测试**

* Jmeter
* 功能稳定
* 并发测试
* 负载测试
* 压力测试

**自动化测试**

* Python

脚本去执行功能测试

**软件测试的概念**

* 概念：软件测试是用来促进鉴定软件的正确性、完整性、安全性和质量的过程
* 预期结果和实际结果做对比
* 账号、密码正确，登录系统
* 预期：登录成功
* 实际：需要去测试
* 登录成功，ok
* 登录不成功，一个Bug，就要去提Bug
* 正确性：预期与测试实际相符，登录QQ成功
* 完整性：输入账号、密码，信息是否显示完整
* 安全性：安全的问题（黑客攻击）——有专门岗位
* 质量的过程：功能等一系列的操作，确保质量

**软件测试工程师**

* 保卫质量、守卫质量
* 是系统质量最后的一道关卡

**测试方法**

**黑盒测试**

* 黑色的盒子，盒子指的是软件
* 软件模拟成不可见的盒子
* 对外部暴露出来的功能测试，不关注内部的结构
* 只验证程序是否能够按照需求规格说明书可以正常使用
* 输入错误的时候，页面给出正确的提示
* 输入正确，页面正确保存或者发送
* 只要结果不正确，测试是不通过的
* 所有的功能测试都是黑盒测试方法
* 验证的是功能
* 不关注内部的结构
* 举例：遥控器的开关机、换台、切换音量

**白盒测试**

* 看得见的盒子
* 又被称之为透明盒测试、结构测试、基于代码的测试
* 需要对程序的结构去进行测试，关注模块与功能的关联
* 具有一定的透明性，软件的内部应用，源代码进行测试
* 要求懂代码、读代码
* 通过检查内部的结构来做白盒测试
* 白盒测试是用在单元测试阶段
* 开发人员自己测的

**灰盒测试（中级测试要求）**

* 介于白盒和灰盒之间的测试
* 灰盒测试关注功能的正确性，同时内部的表象、事件、标志来判断内部运行的状态
* 关注输入输出的正确性
* 同时关注内部的结构
* 关注接口、日志
* 测试中，结合接口去看，看接口的返回是否正常
* 提供接口去看
* 发现错误、定位Bug，Linux命令查看日志
* 提供接口去分析该问题是因为什么导致的
* 提供日志去看
* Linux——文件夹——tail-f——exception

**功能测试**

* 根据（项目经理）需求文档 / 规定要求去对产品的功能逐项验证，看是否达到标准
* 举例：车载系统超速提醒，若超速未提醒则为未达到要求

**性能测试**

* 性能测试工具：Jmeter
* 性能：一个软件运行的最大指标（极限）
* 压力测试（大的数据量去压测）、负载测试（对系统不断加压）、并发测试
* 时机：功能稳定后做性能测试，用户量有了一定的要求
* 举例
* 汽车极速
* 淘宝用户最高访问数
* 内存满的情况下，去运行某软件
* 举重运动员不断挑战加重

**接口测试**

* 软件：前端和后端的连接通过接口去连接
* 硬件：手机和电脑的连接是通过USB接口连接的
* 测试工具：Postman / Jmeter
* 前端：显示的展示的东西
* 后端：涉及到逻辑的、改动的
* 接口和端口的区别
* 端口
* ip
* 协议和服务
* 一个端口可以调用无数接口
* 接口对应的资源

**UI界面测试（Bug较多）**

* 界面的样式是否完整、美观
* 风格是否统一
* 文字是否正确
* 界面显示测试主要包括界面输入（输入域内是否可以完成正常的输入）、界面显示无变形、界面文字无乱字符、排版显示是否正常、普通文本显示是杏正常、图片显示是否正常、图片缩放是杏正常、图片链接是杏正常打开、表格显示是否正常、文字链接能否正常打开等；（好细啊~~~）
* 带有一定的主观性
* UI规范的要求，按照UI的要求测

**易用性测试**

* 好不好用
* 操作步骤越少越好
* 学习成本越低约好
* 指引用户操作正确
* 难的名词给出注释
* 图标功能给出注释
* 下拉框的地方有搜索的功能，方便查找
* 数据较多时，有分页的功能
* 举例
* 微信易用性
* 折叠置顶功能
* @/引用
* 群聊——回复消息——@人——搜索——搜到这人
* 群聊——回复消息——引用的消息来自张三——第一个就可以选择到张三
* 抖音/微博
* 一个视频播放完了，重播 / 跳转下一个

**兼容测试**

* 产品在浏览器上或者App上是否显示正常，有没有错误，是否加载失败
* 记录bug的同时，记录浏览器种类及其版本号
* 对不同系统做兼容性测试
* web：访问网站功能历遍
* App：下载、运行、卸载App，功能历遍测试
* web兼容性测试
* 对不同浏览器不同版本进行测试
* 测试功能是否正常
* 乱码、错码、排版、崩溃、闪退
* 记录版本号对应的浏览器
* 在谷歌最新版本，新增项目功能排版显示不全，火狐没有这个问题
* 浏览器有哪些：谷歌、火狐、edge
* 浏览器怎么选择
* 根据用户的需求
* 不同版本：最新版本、最低版本
* 根据公司的要求，没有要求则主动去问
* 面试提问：当小众浏览器出现兼容性bug时，需不需要做兼容性测试？用户反馈在这浏览器上面的问题，们公司怎么去处理的？答：建议使用其他浏览器；评估bug严重性并检查其他浏览器是否有相似问题，有必要则为客户定制化测试小众浏览器
* App兼容性测试
* 不同手机型号、不同设备、不同版本
* 一般选择市场占有率比较高的手机做测试
* 需要去调研
* 由项目经理和产品来决定
* 手机系统
* IOS
* 安卓
* 覆盖面非常大
* 版本6.0（参考小红书和中国移动App）
* 鸿蒙
* 手机系统版本、来源：采购机、借用
* Q&A 针对过低版本，建议升级版本。
* 安装测试、卸载测试、功能历遍测试、运行稳定性测试、界面显示测试、流畅度测试、分辨率测试……

**App专项测试**

* 区别：只用App有，Web没有
* 弱网测试
* 工具：Charles / fiddler
* 测试环境：网络信号比较差的时候
* 权限测试
* 拍照、麦克风、位置、通讯录、存储、蓝牙，测试开启/关闭权限
* 场景交互测试
* 并行
* 用微信的时候，同时看b站
* 资源争用测试
* 互斥
* 微信语音—打电话（争用麦克风资源）
* 消息推送测试
* 开启消息推送
* 联网时推送
* 离线重连时，历史消息的推送
* 消息数量
* 折叠
* 消息数量的正确性
* 锁屏状态下消息推送
* 删除消息推送，消息数据结果检验
* 关闭消息推送
* 安装 / 卸载 / 更新测试
* 不同系统、不同版本安装卸载更新是否正常
* 安装空间不足是否有提示
* 卸载App是否有二次删除的提示
* 更新成功后，检查客户的数据是否保留
* 更新提示的版本和升级成功的版本是否一致
* 安装/更新过程中，做中断测试操作（断网、重启、杀进程）
* 从头开始
* 断点续传
* 离线测试
* 断网后某些功能是否可以正常使用
* 缓存数据
* 缓存的数据是否可以正常查看
* 面试题举例
* 同样是发图片功能，慕课和App上有什么区别？
* 不同手机系统有什么区别？分辨率、版本不能降级、安装便利程度、系统的开源……
* 稳定性测试
* 长时间使用App和Web，看系统会不会崩溃
* 前提：功能正常
* 条件：一定时间内，一定人数
* 验证点：验证功能点稳定性
* 结果：减少系统的崩溃、闪退，保证系统的稳定性
* 前端+后端都要测
* 工具
* Monkey
* jmeter
* 安全测试
* 测试安全，以发现隐患为目标
* 暴力破解
* 密码过于简单
* Sql注入
* 程序员编写的时候疏忽，通过sql语句，实现无账号登录，篡改数据库
* 跨域攻击
* 点击不安全的网站链接
* 获取的cookies
* 流量攻击/DDOS攻击
* 大数据量、大流量攻击来压垮网络设备和服务器
* 制造大量的无法完成的请求，消耗服务器的资源，让服务器挂掉
* 渗透攻击
* 长期有计划、有组织、逐步渗透网络，最终完全控制网络
* 安全工具
* 安全测试公司
* 安全按次付费也可以
* 绿盟
* 漏洞盒子

**单元测试**

* 测试代码、函数、方法、逻辑
* 一步步去拆分代码逻
* 白盒测试的方法
* 一般是开发自测

**集成测试**

* 组装测试/联合测试
* 接口
* 数据经过接口有没有丢失
* 满足功能要求
* 两个模块去组合测试被叫作集成测试
* 集成功能测试
* 集成出来的功能是否ok

**系统测试**

* 详细的测试
* 系统测试包含大部分内容
* 功能测试/ui界面测试/易用性测试/性能/兼容性/稳定性/app 专项

**验收测试**

* 客户验收
* α测试—内测—内部员工使用
* β测试—公测—真实用户体验

**软件测试的测试过程**

* 软件模块进行单元测试(开发自测),经过单元测试的各个 模块组成起来进行集成系统(接口),集成测试的系统合成软件版本，需求说明书， 在测试环境下，进行系统功能验证(系统测试),进入要验收测试(客户验收)

**冒烟测试**

* 冒烟测试是对软件的基本功能进行快速验证的测试，确保系统的核心功能正常工作。通常在开发人员提交新版本后执行，用于决定是否继续进行更详细的测试。

**回归测试**

* 回归测试是在修改代码后（如修复缺陷或添加新功能）重新执行之前的测试用例，确保原有功能未被破坏。

**1-4 测试实战1-如何测试电梯**

**课程引入**

* 主题：如何测试一个电梯
* 测试维度：功能UI界面测试、易用性测试、兼容性测试、性能测试、稳定性测试、安全测试、接口测试等
* 理论闭环：与软件测试理论知识形成闭环

**软件测试基本概念回顾**

* 软件测试定义：鉴定软件的正确性、完整性、安全性和质量的过程
* 对比：预计结果和实际结果
* 软件测试工程师角色：系统质量的最后一道把关

**测试方法回顾**

* 黑盒测试：只验证功能，不关注内部结构
* 白盒测试：代码测试，需要懂代码
* 灰盒测试：介于黑盒和白盒之间，关注功能和内部结构

**测试类型回顾**

* 功能测试：页面上的每一个可操作部分
* 性能测试：软件或物品运行的最大指标
* 接口测试：硬件通过接口连接，软件通过接口连接前后端
* UI界面测试：输入和输出是否完整、界面是否变形等
* 易用性测试：站在用户角度关注产品是否好用
* 兼容性测试：Web端（不同浏览器和版本）、APP（不同手机型号、设备和版本）

**APP专项测试**

* 弱网测试、权限测试、场景交互测试、资源占用测试、消息推送测试、安装卸载更新测试、离线测试等

**稳定性测试**

* 定义：长时间运行系统是否稳定
* 测试时间：7×24小时

**安全测试**

* 定义：发现隐患，杜绝bug出现
* 测试内容：暴力破解、SQL注入、跨域攻击、流量攻击、渗透攻击等

**单元测试、集成测试、系统测试和验收测试**

* 单元测试：开发自己进行，测试代码、函数、方法和逻辑
* 集成测试：接口测试，检查数据经过接口是否丢失
* 系统测试：详细的功能测试，包括所有功能
* 验收测试：系统测试没问题后进行，交付上线

**冒烟测试和回归测试**

* 冒烟测试：功能测试开始前的测试
* 回归测试：所有功能都跑一遍，占比40%到50%的工作量

**电梯测试实例讲解**

* 功能测试：上升和下降、开门和关门、指定楼层、报警按钮、电话、灯光、指示灯、障碍物感应
* UI界面测试：光滑程度、是否容易生锈、形状是否美观、标识承重和人数
* 易用性测试：按钮设计、通风和照明、按键灵敏度、扶手和空调
* 兼容性测试：不同电话类型、不同电压
* 性能测试：人员超载、物品超载、负载测试、运行速度和反应时间
* 稳定性测试：长时间运行电梯
* 安全测试：暴力拦门、障碍物停留、停电反应机制、报警装置、电梯速度
* 集成功能测试：两台电梯联动测试、就近原则、同行方向且顺路的电梯优先运行、满载不停靠、先接后上

**测试思维总结**

* 考虑多端：多个电梯运行
* 功能拆分：大的功能—>小的功能
* 多维度测试：UI界面测试、易用性测试、兼容性测试、性能测试、安全测试、稳定性测试、集成测试

**课程总结与作业布置**

* 总结：测试电梯的测试思维与功能测试相同
* 作业：梳理如何测试一个电梯，结合软件测试的基本概念和方法

**1-5 测试实战2-如何测试视频播放器&测试思维**

**核心功能拆解**

以播放视频功能为例，需覆盖以下关键测试点：

1. 基础播放控制

* 封面交互：验证轮播图左右切换功能及选集跳转逻辑（如直接跳转第10集或自动续播下一集）。
* 历史记录：测试播放记录按时间维度（最近1天/7天/30天/1年）的筛选功能，确保记录准确性。
* 清晰度选择：
* 会员特权：蓝光/高清需与会员状态关联，非会员切换蓝光时触发购买弹窗。
* 闭环验证：支付成功后蓝光权限生效，取消支付则维持原有清晰度。
* 播放形式：支持进度条拖拽与屏幕滑动切换，需校验播放位置准确性。

1. 互动功能

* 评论与点赞：验证评论发布、回复通知及点赞数实时更新（如评论获赞后+1提示）。
* 弹幕权限：测试登录后弹幕开关状态切换，关闭弹幕后内容不可见。
* 收藏与分享：
* 收藏：验证视频在“我的收藏”列表的增删同步。
* 分享：覆盖微信、微博等渠道，测试分享成功/取消场景。

1. 高级控制

* 播放速度：支持0.25倍速步进调节（如1.0→1.25→1.5→2.0）。
* 投屏与多端同步：
* 投屏：验证同一WIFI下设备连接及播放状态同步。
* 多端登录：手机/平板/电脑端播放记录实时共享（如手机端观看10分钟，平板端续播）。

1. 异常处理

* 网络切换：测试4G/WIFI切换时的播放连续性。
* 弱网场景：模拟卡顿、黑屏、闪退等异常，验证容错机制。
* 进程中断：如来电、锁屏导致播放暂停，恢复后能否从断点续播。

**APP专项测试**

针对移动端特性，需补充以下测试维度：

1. 安装与更新

* 正常流程：验证安装/卸载/更新完整性。
* 异常场景：内存不足时中断更新，恢复后是否支持断点续传。

1. 权限与资源

* 麦克风权限：测试语音评论功能的开关控制。
* 资源冲突：视频播放时启动音乐播放器，验证音频优先级逻辑。

1. 消息推送

* 推送机制：测试多条消息堆叠、删除及离线接收功能。
* 锁屏通知：验证消息在锁屏界面的展示与交互。

**非功能测试**

1. 性能测试

* 压力测试：模拟万人同时在线，验证服务器承载能力。
* 稳定性测试：持续播放72小时，监测崩溃率及内存泄漏。

1. 兼容性测试

* 浏览器：覆盖Chrome/Firefox/Safari等主流浏览器及版本。
* 移动端：适配Android 6-13、iOS 14-16及主流机型（如华为、小米、iPhone系列）。

1. 易用性测试

* 界面布局：验证视频标题、返回键、清晰度切换按钮的位置合理性。
* 播放记忆：退出后重进是否自动续播。
* 提示友好性：如加载动画、断网提示的及时性与清晰度。

**测试思维建模**

以会员体系为例，构建业务逻辑树：

1. 登录状态

* 已登录：
* 会员：免广告、蓝光权限、跨集选播。
* 非会员：试看5分钟VIP内容后触发购买。
* 未登录：
* 免费内容：直接播放。
* 付费内容：强制跳转登录页。

1. 购买流程

* 支付成功：立即解锁会员权限，续播VIP内容。
* 支付失败：保持原有权限，提示余额不足或网络异常。

**总结**

测试视频播放器需遵循“主线功能+专项场景”策略：

* 主线功能：以播放流程为核心，覆盖选集、清晰度、互动等场景。
* 专项场景：针对移动端特性补充权限、兼容性、异常处理测试。
* 思维延伸：通过会员体系、多端同步等业务逻辑，构建闭环测试用例。

**课后作业**

请选择以下任一主题进行测试设计：

1. 微信发朋友圈功能（覆盖权限、内容类型、隐私设置）。
2. 微信聊天功能（重点测试消息送达、多端同步、文件传输）。

要求：从功能拆分、异常场景、非功能测试（如兼容性、安全性）等维度展开，形成完整测试方案。

**1-6 互联网公司结构和组成**

**为什么要学习互联网公司结构和组成**

进入公司后，需要清楚与打交道的同事有哪些，他们的职位以及工作内容是什么。这样，在对接这些同事时，心里大概会有一个判断，而不会发懵，比如“这个同事是干啥的？”或者他在跟我讲一些内容时，我可能对他的职位还不清楚。

**为什么没有讲到自动化测试**

自动化测试是功能测试的一个进阶。自动化测试是一门单独的课程，会把所有功能测试内容学完了，性能测试也学完了之后，再回来开始学习自动化测试。因为自动化测试是把人工测试转换成代码去进行测试，也就是它依靠代码去做测试，所以它不在功能测试这一部分内容里面。

**前情回顾**

软件测试基础理论知识、两个实战（测试电梯和播放器）

之前的课程里，已经跟大家把整个软件测试基础理论知识都讲解了，还引入了两个实战，就是怎么样测试电梯和怎么样测试播放器。通过这两个例子，带大家去熟悉和掌握了功能测试、接口测试、性能测试、UI界面测试、易用性测试、稳定性测试，以及单元测试、集成测试、系统测试、验收测试，还有回归测试和冒烟测试。

**架构师**

架构师工资非常高，架构师是做什么呢？主要是管理开发规范、需求，让整个研发效率变得高起来，开发容易维护、简单升级。整体一个开发框架的把控也是由架构师来控制。因为有一些代码，越写越多，框架就会显得更加复杂，或者堆积起来不好用。架构师会对整个代码框架去进行把控，包括一些疑难杂症，都可以找到架构师。

**项目经理**

项目经理是对整个项目负责。通俗一点来讲，就是所在产品需要测试，然后产品经理、研发、测试其实都听命于项目经理，但是自己也会有自己直系领导。因为项目经理是统筹这个项目，他是直接去对接客户。客户想要什么东西，一般会告诉项目经理，然后项目经理会安排产品经理去进行对接。会发现项目经理对整个项目负责，比如说项目质量、进度、人员规划。比如项目经理会说，现在接到了一个项目，非常开心，就需要去立项。去成立这个项目时候，产品经理就会去整理需求。

需求，是来自于两部分：一个是根据竞品去做需求，还有一个是根据客户需求。比如客户说我现在想做一个贷款功能，们参考一下这个竞品，比如阿里，或者是某银行就行了。或者是我自己有一些需求，比如我对客户要求比较高，所以需要详细一些身份信息，需要他工作单位等。会把客户需求告诉，然后产品经理就整理相关需求，也就是说一个系统做成什么样子，这个软件功能流程是什么样子，都是由产品经理来负责。

比如说像贷款功能，需要输入姓名，上传身份证号码去做实名认证，留紧急联系人电话，留单位电话等，还要备注什么。像这个功能这些点，都是由产品经理去决定，产品经理会对标竞品或者是客户需求来进行分析。

**产品经理需求整理好，UI就去画设计图，然后前后端同步写代码**

当产品经理需求整理好了之后，产品会把项目经理、研发、设计、测试叫到一起，一起去讨论这个需求有没有问题。如果没有问题，大家都一致通过，说这个逻辑没有什么问题。这个时候，UI就会去画设计图。研发就准备开始去写代码了。

在这个里面，UI工程师，他就是一个画图。研发，就会去分成前端以及后端。前端，它是需要等到设计去做好设计图之后，设计出来东西和前端要呈现东西是一样。等设计图出来之后，前端才干活。设计图出来之后，前端就会和后端去同步写代码。后端，因为后端逻辑代码可能会比较复杂，所以后端代码逻辑会先写，前端就是等设计图出来之后，然后再开始去写这个代码。

这个时候，就知道，前端代码一般是用JavaScript或者是用Vue，后端就是写Java或者是写C++，他们一般都是写逻辑代码。

**测试呢**

这个时候，测试干什么呢？大家会不会觉得，测试完全没有事情做就是在公司里面摸鱼。其实不是，就是当需求出来之后，测试在做计划，写测试用例。等测试用例写得差不多了时候，开发代码可能也就写得差不多了，然后就会提测，然后就会进入到测试里面去。

**测试职位**

在这个里面，需要详细地来给大家讲一下测试。测试，分为测试经理、测试组长、测试工程师。大家在面试时候可能会面初级测试、中级测试、高级测试。基本上来说，中级以上就可以当一个小组长了。测试和开发配比是一比五，也就是说五个开发配一个测试。

**测试的日常工作**

测试日常工作是什么呢？就是产品经理出了一个需求，然后根据这个需求去梳理功能点、测试点，去编写测试用例。然后等这个系统出来之后，发现了bug，就提bug。开发解决了这个bug之后，就关闭这个bug。还有一个是我发现了这个bug，我也提了，开发解决了这个bug，我验证了之后，发现它并没有解决，所以又去重新打开这个bug。

**熟悉需求、分析需求和理解需求**

再一个就是刚提到，在产品整理出来需求时候，要干什么？要熟悉需求、分析需求和理解需求。怎么样熟悉需求、分析需求和理解需求呢？第一个就是提取它功能点。提取功能点方式，我可以画思维导图，对吧？再一个是设计测试用例、写测试用例。然后写完测试用例之后，去做执行测试，然后去做系统测试。然后再一个是做一个验收测试。这个就是日常测试工作，其实相对来说并不复杂。

**运维岗位**

运维，是负责生产环境和预生产环境搭建，包括本地一个测试环境和研发环境，也是运维同时负责。在这里，需要跟大家来讲一讲就是什么是研发环境，什么是测试环境，什么是验收环境，UAT环境以及什么是生产环境。

开发环境就是开发自己有一个网址，有一个地址来写它代码。它这个本地环境代码是随时在变动，bug一大堆，非常不稳定。开发把它这个功能写好了之后，它会上传到测试环境里面去。测试环境就是测试工程师测试一个环境，这个环境是比较稳定，不会频繁地发版本。也就是说，某一个需求如果要上线，它会经过这么几个步骤。

**步骤**

首先是开发在开发环境上把代码写出来，但他们自己不知道代码有没有问题，所以需要测试来验证。开发会把代码部署到测试环境，让测试人员进行测试。如果测试人员说没有问题，代码就会上到预生产环境，也就是UAT（User Acceptance Testing）环境。这个UAT环境一般是客户预生产环境，相当于是一个彩排环境。代码上到UAT环境后，大家可以把它理解成是客户边一个测试环境。在这个环境里做一个彩排，如果发现确实没问题，客户也觉得没问题，这个时候，代码就会上到客户生产环境。生产环境就是线上环境，是面对用户真实环境，比如大家现在看到京东、淘宝、抖音、小红书等，都是线上环境。在这个环境里，不能随意改动客户数据，也不能随便进行测试。如果说但凡在生产环境上面发现了一个bug，就需要倒推。

生产环境上面发现一个bug，怎么样去解决呢？倒推到UAT环境，预生产环境里面，看它会不会也有。一般来说，生产环境有了bug，UAT环境肯定也会有。这个时候就要去复现，看UAT环境里有没有问题。如果有，这个时候就要看本地测试环境有没有，测试环境有，不用说，开发环境100%有。所以说这个时候，是一步地去反推，反正就是不动用户生产环境。

最后，如果发现有bug，开发就去改这个bug。改完之后上到测试环境，测试环境验证之后没问题，再发到UAT环境，UAT环境也没问题，再更新到生产环境。这个就是整个关于线上出现bug之后解决流程。所以大家会发现，运维工程师是负责生产环境和预生产环境搭建。

**运营的工作**

再来讲一下运维搭建环境，如果说上到预生产环境下面来是没问题情况下，公司自己产品就会去做验收测试。验收测试通过之后，运营同事就会去宣传，去吸引更多用户去使用这个软件。运营怎么样去做宣传就是运营事情了，比如说投放流量、送礼品等，这些都是运营自己去考虑。

在这个过程里，运营会收集一些用户意见和反馈。运维收集到用户意见和反馈，会转交给测试。需要测试去判断该意见属于bug还是建议。如果是bug就交给开发去改，如果不是bug，是提取一些新想法和功能，就转给产品经理去开发和设计。

这个就是一个典型互联网公司结构，以及每一个结构里负责角色和具体做工作。就给大家介绍到这里。大家就会有一个比较清晰认识了。在工作里面，打交道比较多，最多就是产品经理和研发，其次是运维、项目经理，还有UI设计师。好，在这个过程里，其实可以把流程梳理一下。

**流程梳理**

首先有一个项目，产品经理说要立项，然后项目经理也说要立项。产品经理整理需求，需求整理出来之后，就去评审这个需求，从而去画UI图。这个时候UI设计师干什么呢？他去画图，前端和后端写代码。测试干什么呢？要分析需求，对不明确地方进行整理和汇总，进行需求评审，写测试用例。然后代码搞好了之后，测试进行测试，有bug就提bug，提完bug之后追踪bug，进行bug管理。OK，没有bug了，可以上线了，就输出测试报告。这个时候运维会搭建预生产和生产环境，代码部署到预生产环境里面去进行验收。最后，上到生产环境，这个时候运营去引流，吸引用户来使用，收集用户意见，然后转交给测试。测试确定它是bug还是需要优化，或者是新需求。

好，这个就是整个一个互联网公司构成和它每一个工作岗位，需要打交道，他们具体工作内容。好，接下来，就是进入到测试流程一个学习。测试流程，实际上就是把测试工作单独拎出来去做一个讲解。也就是说，把测试工作流程单独拎出来跟大家讲。

测试工作，也是会把它分成不同阶段来做测试工作怎么样去进行？怎么样分成不同阶段呢？说到阶段，其实大家应该非常熟悉，因为在之前课程里面提到了，软件测试按照阶段划分，可以分成哪些阶段。大家想一想，是不是可以分成单元测试阶段、集成测试阶段、系统测试阶段和验收测试阶段？但是这里说是软件测试流程阶段，而不是软件测试阶段。

软件测试流程阶段，是怎么分呢？把它分成需求分析阶段、测试设计阶段、测试执行阶段以及测试总结阶段。明天，就要开始进入到需求分析阶段一个学习。需求分析阶段，主要来学习什么呢？主要来学习理解什么是需求。在需求这个阶段，作为测试，我应该怎么样去理解、梳理和分析需求，把需求里面我懂地方、我知道地方提取出来，我不懂地方去进行一个汇总和整理，然后去弄清楚这个软件有什么样功能，流程是什么样子。

在需求这个阶段，会教会大家怎么样去提取功能点，怎么样去梳理业务流，怎么样去画业务流程图，从而去把所有关系线都给它整理出来。好，今天关于互联网公司结构和组成介绍就给大家介绍到这里。明天正式进入到需求分析阶段一个学习。

在需求分析阶段学完之后，就会进入到大家比较关心测试用例方法学习。怎么样去写测试用例，测试用例内容、目的，以及测试用例评审等。在学习测试用例方法之前，其实会来学习测试计划方案编写、测试策略编写等，然后再来学习测试用例。这个是测试设计阶段一个内容。

测试执行阶段是干什么呢？测试执行阶段，会来学习bug一个管理，就首先是bug六要素、bug平台、bug类型、bug生命周期以及bug状态、版本迭代、回归测试等。会来学习bug这样一个管理。最后，就是所有测试通过了之后，会学习测试总结阶段，也就是测试报告编写，以及测试报告内容一个涵盖。

好，今天关于互联网公司结构组成课程，就给大家介绍到这里了，谢谢大家。

**1-7 需求概念&登录规则、提取功能点**

**课程引入**

* 项目流程概述
* 产品经理梳理需求
* 开发、测试、设计分工
* 测试流程：系统测试、验收测试、运维部署、运营宣传

**测试工作阶段划分**

* 四个阶段

1. 需求分析阶段
2. 测试设计阶段
3. 测试执行阶段
4. 测试总结阶段

* 与软件测试阶段区分
* 单元测试、集成测试、系统测试、验收测试

**需求分析阶段重点**

* 目标
* 理解需求、分析需求
* 提取不合理或逻辑不明确的地方并汇总
* 目的
* 弄清楚软件功能
* 预期结果与实际结果对比
* 了解操作流程、业务流程、功能规则限制
* 易用性考虑
* 从用户角度出发，判断功能是否好用

**需求的形式**

* 原型图
* 类似效果图，展示界面布局和交互效果
* 需求文档
* 详细记录功能参数和规则
* 口头/书面沟通
* 不规范，易引发麻烦
* 最正规方式
* 原型图 + 需求文档 或 原型图包含需求规格说明

**需求文档与原型图解析**

* 需求文档内容
* 修改版本、系统功能需求简介、业务流程、操作交易规则、菜单、功能点等
* 字段说明：是否必填、是否可为空、字符长度等
* 原型图特点
* 元素可点击，模拟系统交互
* 前端根据原型图搭建框架和界面
* 测试根据原型图编写测试用例

**功能点提取方法**

* 功能点定义
* 包含不同操作步骤，可梳理出业务流程
* 页面上可点击的元素均可理解为功能点
* 提取目的
* 理解业务含义，串联业务流程
* 提取步骤
* 按模块梳理（如登录、主页、进入教室等）
* 提炼功能点，记录业务规则
* 思考测试点，记录潜在问题

**功能点、测试点与系统关系**

* 功能点
* 包含测试点，是测试用例编写的依据
* 测试点
* 最小颗粒，不可再划分
* 系统
* 由业务流程组成，业务流程由功能点组成

**实例：小红书评论发帖功能**

* 功能点
* 输入评论、取消评论、确定评论
* 测试点
* 不输入评论、评论长度超出、评论包含敏感话题
* 业务流程
* 浏览文章 → 找到评论按钮 → 进行评论 → 查看评论留言 → 删除留言

**实例：教育培训系统登录模块**

* 功能点提取
* 输入账号密码登录、小眼睛显示/隐藏密码、登录按钮点击、记住密码、忘记密码、第三方登录
* 业务规则记录
* 账号为手机号码，密码包含大小写且不少于八位
* 密码连续错误五次锁定十分钟，30分钟内错误十次锁定24个小时
* 测试点思考
* 合规/不合规手机号码、密码包含大小写情况、提示信息统一文案等

**兼容性测试考虑**

* 登录页面适配
* 电脑端、笔记本、平板、手机端

**作业布置**

* 提取忘记密码功能点
* 输入、获取验证码、确定、返回登录
* 拓展思考
* 验证码和密码测试点
* 笔记整理
* 理解需求、需求形式、功能点核心含义、功能点与测试点关系等

**课程总结**

* 需求分析阶段是测试工作的重要环节
* 通过提取功能点和记录业务规则，为后续测试工作打下基础
* 强调实战中的测试思维和方法应用

**1-8 教培个人主页功能点提取&慕课网功能点实战**

**一、需求分析阶段**

**1.1 需求基本概念**

* 什么是需求
* 需求阶段测试要做什么
* 弄清楚软件功能
* 梳理业务流程
* 明确规则与限制（如字段必填、长度要求、流程合理性等）

**1.2 功能点提取方法**

* 以教育培训系统为例
* 登录功能点
* 主页功能点（如打卡时长统计、课时统计、近期课表等）
* 总结类方法提取功能点
* 归纳页面展示内容的本质功能
* 关注数据来源与校验（如数据库校验）
* 发现并记录业务规则不明确处，及时与产品沟通

**二、典型功能点提取实战**

**2.1 打卡功能**

* 打卡成功提示
* 打卡成功赠送雪碧
* 连续打卡五天额外奖励
* 打卡记录查看需求
* 需与产品确认的业务规则（如不足一小时如何统计）

**2.2 课时统计功能**

* 总课时数
* 已上课时数
* 剩余课时数
* 完课率
* 数据动态变化特性

**2.3 近期课表功能**

* 课表统计总数展示
* 颜色区分（灰色=无课，粉色=有课）
* 默认高亮当天
* 支持切换日期、月份
* 规则总结与场景覆盖

**2.4 课程提醒功能**

* 只提醒当天课程
* 进入教室按钮高亮/置灰规则
* 不支持分页，最多显示三个模块，支持滑动查看更多
* UI排版检查

**2.5 评论与消息功能**

* 评论流程（A发消息，B回复）
* 内容规则限制（如敏感词、长度、类型等）
* 规则限制与内容限制的区分

**三、功能点提取的总结方法**

**3.1 规则限制**

* 长度、必填、格式等

**3.2 内容限制**

* 敏感词、内容类型（图片/视频/表情等）

**四、慕课网功能点提取案例**

**4.1 课程报名流程**

* 认证学习模式（年付/半年付）
* 文案对比
* 查看全部权益
* 付费按钮
* 付费成功/失败场景
* 获得认证成绩单
* 免费学习模式
* 文案对比
* 立即参加按钮
* 协议勾选与报名流程
* 勾选/未勾选/取消按钮对应结果

**4.2 业务流程梳理**

* 登录→选择课程→报名（认证/免费）→学习→获得结果
* 付费失败/协议未勾选等分支场景

**4.3 其他功能点举例**

* 评论功能点（本次课程/全部课程、发表评论、编辑、删除、点赞等）

**五、需求分析阶段的核心任务**

**5.1 熟悉需求的方式**

* 查看原型图
* 结合需求文档
* 提取功能点
* 梳理业务流程

**5.2 功能、功能点、测试点的关系**

* 功能→功能点→测试点
* 大小功能点组合成业务流程
* 多条业务流程组成系统

**六、课程总结与作业**

**6.1 课程总结**

* 需求阶段要弄清楚功能、流程、规则、限制
* 规则限制与内容限制的区分
* 功能点提取与业务流程梳理的重要性

**6.2 课后作业**

* 教育培训系统：进入教室功能点提取
* 有余力可提取全部功能点
* 慕课网：任选模块提取功能点

**1-9 需求分析阶段&业务流程实战**

**一、业务流程图的意义与作用**

- 1.1 需求分析阶段的最后一步

- 1.2 工作和面试中的重要性

- 梳理业务流程，理清功能逻辑

- 面试时主要介绍业务流程

**二、业务流程图的基本概念**

- 2.1 业务流程图定义

- 用规定符号和连线表示业务处理过程

- 主要描述业务走向

- 2.2 以业务处理过程为中心

- 没有数据的概念

- 只描述流程，不涉及具体数据

**三、业务流程图的常用符号**

- 3.1 跑道椭圆：开始/结束

- 3.2 矩形：过程

- 3.3 菱形：判断/分支

- 3.4 连线：流程走向

**四、业务流程图的绘制工具**

- 4.1 Web端工具（如ProcessOn）

- 4.2 纸笔手绘也可

**五、业务流程图的绘制方法与步骤**

- 5.1 明确开始和结束

- 5.2 梳理主要流程节点

- 5.3 判断分支与闭环

- 5.4 结合实际业务场景（以慕课网为例）

**六、典型业务流程梳理（以慕课网为例）**

- 6.1 未登录用户流程

- 游客浏览

- 想报名课程→提示登录

- 不报名→继续浏览

- 6.2 登录用户流程

- 登录成功→进入系统主页

- 选择课程：免费/付费

- 免费课程：同意协议→报名成功→可学习

- 付费课程：支付成功→可学习，支付失败→不可学习

- 订单/课程状态联动

- 6.3 反向流程

- 退出课程→需重新报名

- 订单取消/失败场景

**七、业务流程图与测试的关系**

- 7.1 冒烟测试：先打通主流程

- 7.2 细节功能测试：主流程通后再测

- 7.3 单功能点与流程组合测试

**八、熟悉系统和业务的技巧**

- 8.1 按模块、列表、功能点逐层拆解

- 8.2 关注数据源头与页面数据关系

- 8.3 先熟悉局部，再扩展到整体

**九、系统的组成与业务流程的本质**

- 9.1 需求→功能→功能点→测试点

- 9.2 功能关联形成场景，场景丰富系统

- 9.3 业务流程图帮助理解系统全貌

**十、面试与实际工作中的应用**

- 10.1 面试时主要介绍业务流程

- 10.2 细节测试点靠实际工作积累

- 10.3 业务流程图是项目介绍的核心

**十一、作业与学习建议**

- 11.1 梳理教育培训系统业务流程

- 11.2 选择其他模块（如直播、个人中心）练习

- 11.3 多画、多写、多总结，提升上手速度

**十二、需求分析阶段的回顾**

- 12.1 需求分析的重要性

- 12.2 功能、流程、规则、限制的梳理

* - 12.3 功能点提取与业务流程梳理的关系
* **1-10 测试计划实战**
* **测试设计阶段4个项目，测试用例为重点**
* hello，各位同学大家好，今天学习测试设计阶段的内容。那测试设计阶段一般干什么呢？测试设计阶段一般就是写文档。
* **测试阶段需要编写的文档**
* 那么，在这个阶段里面，有几个文档要写。第一个是测试计划文档，再一个是测试策略的文档，以及测试方案文档。接下来就是大家非常熟悉的测试用例文档。这四个文档就是测试设计阶段的核心文档。
* **不同角色对文档编写的要求**
* 在这个里面，大家对测试用例应该是比较熟悉的。测试用例是每一个测试人员都要会写的，都需要掌握的。而前面三个文档——测试计划、测试策略、测试方案，这些通常只需要由组长或经理来掌握就可以了。也就是说，这三个文档一般是由测试组长来编写的。
* **普通测试人员是否需要学习前三类文档？**
* 那么，我如果不是测试组长，不是测试经理，那我是否可以不学习这些文档呢？答案是否定的。肯定是要学的。因为当进入公司之后，很有可能就会担任测试组长的角色。在一些公司中，甚至只有一名测试人员的情况下，就既是测试人员，也是测试组长和测试经理。
* 在这种情况下，测试计划怎么出？测试策略、测试方案如何制定？就需要自己来掌控。因此，就还是需要学会编写测试计划、测试策略和测试方案。
* **测试用例的学习重点**
* 至于测试用例这一块，主要学习的是测试用例的一些编写方法，以及测试用例中的具体内容，比如测试用例怎么写、其中有哪些细节需要注意、测试评审是什么样的等等。等学到测试用例这一块的时候，会进行一个详细的讲解。
* hello，各位同学大家好，今天继续学习测试设计阶段的内容。现在来重点讲一下\*\*测试计划\*\*。
* **测试计划概述**
* 其实测试计划比较简单，因为在日常生活中经常会做各种计划。比如最常见的旅行计划、学习计划、生活计划等等。
* 比如说打算在五一或国庆节的时候和朋友去旅游，那就会做一个详细的旅行计划：什么时候出发、什么时候到达、到了之后要做什么事情，按步骤一一列出。这就是一个典型的“计划”行为。
* 同理，测试计划就是对\*\*测试工作的一个安排\*\*，也就是要怎么去做测试，测试的目标是什么，时间如何安排，人员怎么分配等。
* **测试计划的核心方法论：5W1H（六合分析法）**
* 在编写测试计划时，会用到一个通用的方法论：\*\*5W1H 六合分析法\*\*。它包括：
* **What**：做什么？
* **Why**：为什么这样做？
* **When**：什么时候做？
* **Who**：谁来做？
* **Where**：在哪里做？
* **How**：怎么做？
* 下面逐一进行讲解。
* **What：做什么？测试内容是什么？**
* 这是测试计划中最基础的部分。需要明确本次测试的任务是什么。
* 测试内容一般是根据\*\*需求文档\*\*或者公司的具体安排来确定的。常见的测试内容包括：
* 熟悉需求、理解需求
* 功能测试（主流程、逆向流程、详细功能点）
* 回归测试（系统整体功能覆盖）

冒烟测试（核心流程验证）

举个例子：如果是一个新需求，但功能还没开发出来，那当前的测试内容可能就是熟悉需求、提取需求中不明确的地方；如果功能已经上线，则是进行详细的功能测试、提Bug、跟踪Bug、验证修复等。

**Why：为什么要这么做？**

这一部分需要解释为什么选择这样的测试策略。

比如，为什么要做冒烟测试而不是直接进入详细的功能测试？可以这样解释：

|  |
| --- |
| “因为目前主要流程还没有通过测试，如果直接做详细功能测试，在流程阻塞的情况下是没有意义的。所以先打通冒烟流程，再深入测试。” |

* 这部分也可以在面试中体现的思考逻辑和技术深度。
* **When：什么时候做？时间安排**
* 时间安排也是测试计划中的关键部分。需要明确每个测试阶段的时间节点和持续周期。
* 例如：
* 需求评审阶段：X月X日 - X月X日
* 用例编写阶段：X月X日 - X月X日

执行测试阶段：X月X日 - X月X日

在面试中，面试官可能会问：

|  |
| --- |
| “如果追求更高的质量，但时间不够了怎么办？” |

可以回答：

|  |
| --- |
| “在这种情况下，应该抓大放小，优先保证主流程和核心功能的测试。在时间充足的前提下，再进行更全面细致的测试。” |

* **Who：谁来做？人员安排**
* 这一部分涉及测试团队成员的分工与职责。
* 比如：
* 张三负责模块A的测试
* 李四负责接口测试
* 王五负责回归测试等
* 同时也要明确每个人员的具体任务，这又回到了“What”的问题——他们具体要做什么事情。
* **Where：在哪里做？测试环境说明**
* 这里的“哪里”并不是指物理位置，而是指\*\*测试环境\*\*。
* 比如：
* 在UAT环境做测试
* 在内网环境测试
* 或者是在公司搭建的演示环境中测试
* 不同的测试阶段可能对应不同的测试环境，这一点要在测试计划中提前确认并说明。
* **How：怎么做？采用什么方式测试？**
* 最后一个是“How”，也就是要使用哪些技术和手段来进行测试。
* 比如：
* 功能测试
* 接口测试
* 自动化测试脚本的使用
* 工具的选择（如JIRA、Postman、TestLink等）
* 这部分体现了在技术层面的掌握程度，也能展示的测试能力。
* **总结：测试计划的核心要素**
* 综上所述，测试计划并不复杂，它主要包括以下几个方面：
* 做什么（What）：测试内容
* 为什么做（Why）：测试目的和依据
* 何时做（When）：时间节点安排
* 谁来做（Who）：人员分工
* 在哪做（Where）：测试环境
* 怎么做（How）：测试方法与技术
* 掌握了这些内容，就能够写出一份结构清晰、逻辑严谨的测试计划文档了。
* **测试计划总体思路**
* 那么在这个里面的话，测试计划到底应该怎么写？首先，不管是做接口测试的计划还是做系统测试的计划，都需要对接口测试和系统测试有一个了解。
* **系统测试内容举例**
* 在这里，以系统测试来举例，比如说如果我来做功能测试、做系统测试的话，那么我首先肯定是要做UI测试、功能测试、兼容性测试、稳定性测试、入网测试、性能测试等等。我要确定以上哪些是确认要做的，哪些可能因为时间或者还没有到节点，比如说性能功能没有做好的情况下，不能做性能测试，没有到这个节点，那就不需要做测试，是吧？
* **测试时间与人员安排**
* 比如说现在确定要做功能测试、UI测试、兼容性测试，那么在做这三项测试的基础之上，要安排时间，比如说需要一个星期测完，还是两个星期测完；然后是人员安排，比如如果一个星期测完，我可以安排两个人来测；如果两个星期测完，我可以安排一个人来测。人员安排其实是根据时间规划去安排的。因为给的时间越长，我可以放一个人全职过来，但时间越少，就需要更多人协助完成这个事情。
* **模块与工具分工**
* 这一个是人员的安排，比如谁负责哪个模块的测试。如果要做接口测试的话，现在用什么技术、什么工具去做？比如用Postman工具去做接口测试，使用相应的技术手段去做。
* **测试目的与环境说明**
* 为什么要做接口测试？从测试角度可以给出专业的解答：为了保证软件后端不会出现问题，有利于提升软件质量，让软件更不容易出问题，从而提高用户体验。在哪里做呢？需要说明测试环境和所用测试工具。
* **总结**
* 这就是5W1H分析法，这个方法不仅适用于测试行业，任何行业都可以用来做分析。
* **实操**
* **测试计划与策略文档的关系**
* 一般来说，有的公司会把测试计划、测试策略、测试方法放到一个文档中统一编写；但也有的公司会将其分别拆分为三个文档。今天的重点是测试计划，因此接下来将带领大家具体了解测试计划该如何撰写。
* **测试计划文档结构**
* 现在可以打开一个文档，从零开始撰写测试计划。测试计划首先需要包含任务信息。然后，需要指定具体负责人、计划开始时间等基本信息。可以对这些字段进行适当排序，例如：
* 编号
* 任务名称
* 负责人
* 计划开始时间
* 计划完成时间
* 实际完成时间
* 状态
* 遇到的问题
* 此外，可以根据需要设定任务的编号与分类，例如：
* 产品名称
* 测试内容（如功能测试、UI测试、兼容性测试等）
* 任务描述
* **示例：教育培训系统测试计划**
* 以教育培训系统为例，假设有两个主要需求：
* 用户报名付费课程
* 用户查询已报名的课程

2. 那么对应的任务包括：
3. 熟悉需求
5. 理解功能点
7. 编写测试用例
9. 执行功能测试

* 接下来需要明确每一项工作的分配情况。例如：
* **测试需求流程**：云海 负责明确测试环境、启动时间
* **需求熟悉与用例编写**：小华 负责功能点提取和用例编写，时间范围为 6 月 1 日到 6 月 5 日
* **测试阶段时间安排**
* 测试流程的典型时间安排如下：
* **6月6日**：开发提测，开始冒烟测试（1-2天）
* **6月7日-6月10日**：冒烟测试通过后，进行详细功能测试、提Bug与跟踪Bug
* **风险提示**：若冒烟测试未通过，则测试时间整体顺延，需标记为风险项
* **6月11日-6月12日**：进行第二轮测试（回归测试），确认修复未引发其他问题
* **6月13日**：出具最终测试报告，测试工作收尾
* **风险与时间节点的管理**
* 在测试计划中要特别关注\*\*风险管理\*\*：
* **风险点1**：冒烟测试未通过，导致测试时间延迟
* **解决方式**：尽早暴露风险，协调研发返工与重新提测，确保整体进度受控

**总结**

通过上述规划，可以看出，一个完整的中型需求从提测到出报告大约需要 10 天到 2 周的时间。测试负责人需合理安排资源，分配任务，设定时间节点，并在早期识别并应对潜在风险，从而保障测试工作的高效有序开展。

**测试时间的预估与责任边界**

在编写测试计划时，测试人员需要对\*\*测试所需时间进行合理预估\*\*。不能把所有项目进度风险都压在测试这边。尤其在研发完成后，如果产品或客户发现实现的内容与预期不一致——无论是产品表达不清，还是开发理解有误——都需要返工，而返工不只是开发重做，测试也需重新梳理用例与测试流程。这类情况\*\*会影响整体进度，但并非测试的责任\*\*。

相对而言，如果是\*\*冒烟测试未通过\*\*，影响的是后续所有计划节点。这是测试侧需要\*\*尽早识别并上报的风险点\*\*，以便项目管理及时响应，避免后续问题扩大。

**测试任务的安排示例**

以下是几个具体任务安排的示例，说明了如何在测试计划中详细填写每一项：

* **任务一：熟悉需求**
* **产品**：教育培训系统
* **任务**：熟悉“付费课程报名与查询”需求
* **负责人**：云海
* **计划开始时间**：6月1日
* **计划完成时间**：6月5日
* **实际完成时间**：6月5日
* **状态**：进行中
* **问题点**：无

* **任务二：全系统功能测试（演示环境）**
* **产品**：教育培训系统
* **任务**：演示环境全系统功能测试
* **负责人**：小明、小米
* **计划时间**：6月2日 – 6月6日
* **实际完成时间**：6月5日（提前）
* **状态**：已完成
* **问题点**：无
* 若此任务未完成（例如环境未就绪）则应备注如下：
* **问题点**：演示环境未搭建成功，导致测试延期，原因归属于运维团队

* **任务三：平台模块回归测试**
* **产品**：教育培训系统
* **任务**：登录、课程、直播等模块回归测试
* **负责人**：A、B
* **计划时间**：6月7日 – 6月8日
* **实际时间**：6月8日
* **状态**：已完成
* **问题点**：无

* **任务四：大需求测试用例编写**
* **产品**：教育培训系统
* **任务**：大需求测试用例编写
* **负责人**：羽墨
* **计划时间**：6月15日 – 6月17日
* **实际时间**：延期（6月18日）
* **状态**：已完成
* **问题点**：需求不清晰、逻辑混乱，导致用例梳理时间延长

* **测试执行与用例管理技巧**
* 测试用例编写完成后，需进入\*\*测试执行阶段\*\*，并可能伴随用例的动态调整。以下是一些实用技巧：
* **红色字体标注已更新的测试用例**，便于后期识别
* **补充遗漏的测试点**，完善测试覆盖
* **为未通过的测试用例标注对应的 bug 编号**，便于追踪分析
* **记录用例调整的原因**，利于回溯和复盘

这些方法虽非“问题点”，但属于\*\*用例管理的实战技巧\*\*，可有效提高测试可追踪性与可维护性。

**总结**

测试计划不仅要体现清晰的任务分工与时间安排，更要能提前识别风险与明确责任边界。测试人员需合理预估工作量、灵活应对突发状况，并在文档中准确记录问题与变化情况。测试计划既是项目管理工具，也是团队协作和信息同步的重要载体。

**交叉测试**

**交叉测试的重要性与实践方式**

在实际工作中，强调\*\*交叉测试（Cross Testing）\*\*\*\*\*\*的一个重要原因，是为了\*\*\*\*\*\*打破思维固化，提升系统测试的全面性和鲁棒性\*\*。

长期负责固定模块的测试人员，容易陷入“熟悉区域”的惯性思维，这种熟悉度虽然提高了效率，但也可能形成\*\*盲区\*\*，导致遗漏隐藏的缺陷。因此，需要通过\*\*交叉测试机制\*\*来激发新的思考视角。

**什么是交叉测试？**

**交叉测试**是指：

* 不同测试人员互相测试彼此原本负责的模块，以此带来不同的观察视角。
* 例如：
* 小明长期负责模块 A，小米负责模块 B。
* 在交叉测试阶段，小明来测试模块 B，小米来测试模块 A。
* 这样彼此可以从“陌生视角”中发现被忽略的问题。

1. **交叉测试的好处**
2. **发现新问题**

* 他人视角更容易打破默认假设，找出未覆盖或被忽略的 bug。

1. **提升系统熟悉度**

* 所有人都了解整个系统结构和逻辑，减少依赖。

1. **增强团队抗风险能力**

* 某位同事请假或离职，其他人能快速接手，保障测试不中断。

1. **促进知识共享**

* 模块之间的交叉熟悉，有助于统一团队的测试标准和认知水平。

* **常见疑问与应对**
* **疑问**：“交叉测试是不是浪费时间？”

* 是的，\*\*在初期确实会带来时间成本\*\*，因为需要重新熟悉他人的模块。
* 但从长期来看，一旦团队成员都对整个系统熟悉，整体效率反而提升，\*\*避免了单点依赖和返工\*\*的成本。
* **疑问**：“工作量是不是会增加？”

* 实际上只是前期学习曲线的问题，\*\*并不会真正增加人均测试量\*\*，只是把测试资源做了更科学的轮换分配。

* **实施建议**
* 每一轮测试后做\*\*模块轮换表\*\*，安排下轮交叉分配；
* 对于新项目，在初期阶段就设定\*\*交叉测试的制度\*\*；
* 保持\*\*交叉测试记录表\*\*，方便追踪测试历史与交接；
* 在交叉测试之后，可以进行\*\*问题归纳会议\*\*，分析彼此发现的问题和遗漏点。

* **总结**
* 交叉测试是一种\*\*提升测试质量、减少依赖风险、强化团队能力\*\*的重要手段。虽然前期需要投入时间培训和适应，但长期来看，这是一种值得实施的团队测试机制，有助于打造\*\*高韧性、高质量\*\*的测试体系。
* **测试中的风险预估与需求返工问题**
* 那么，在这个里面的话，作为测试来说，是需要去预估测试时间的。不能把所有的风险都放在测试这边，是吧？再一个，就是当开发把这个需求做出来了，也就是研发完成了，但如果和产品，或者说和客户想要的不一样，那么这个时候需求就会返工。
* 比如说，有可能是产品没有表达清楚，也有可能是开发没有理解产品的需求。总之，这个需求需要返工、重新制作。这个时候进度肯定会受到影响，因为不仅仅是开发要重新做，测试这边的用例等所有内容也都需要重新梳理，然后重新进行测试，时间节点必然会被影响。
* 这一步，其实和测试关系不大。因为需求做出来不一样，并不是测试能决定的。但是如果冒烟测试没有通过，影响到了后面的节点，那就需要尽快上报这个风险。
* **测试计划的制定流程与示例说明**
* 大家会发现，测试计划怎么做？测试计划的主要流程就是这样的。现在就来排测试计划。比如说，有一个付费课程查询的需求，安排负责人云海来做，也提到了计划开始的时间和需要完成的时间。
* 例如，计划开始时间是6月1号，计划完成时间是6月5号。实际完成的时间也可以是6月5号，也可以不是。实际完成的时间可以是提前，也可能延后，按照实际情况填写。在这里暂时写6月5号。任务状态是“进行中”。
* 这个任务有没有问题点？如果有问题点就写明，如果没有就写“无”。这就是理解需求阶段的计划，现在已经排出来了。
* **全系统功能测试安排与应对延期说明**
* 接下来的第二步，是安排其他人做事。比如说，在这个教育培训系统中，需要做全系统的功能测试，安排小明和小米两人从6月2号测试到6月6号。如果他们提前一天完成任务，实际完成时间是6月5号，状态就可以标记为“完成”。
* 但如果换一个场景，比如原计划在演示环境上做全系统功能测试，安排小明和小米参与，但因为环境没有搭好，导致测试延期，没有实际完成时间，那么就要在备注里写明延期原因，例如“运维环境未搭建好”。
* **平台回归测试与时间安排**
* 接下来是平台的回归测试，比如登录、个人中心、课程模块、直播模块等都要进行回归测试。安排两个人执行。用时两天，实际完成时间由执行人自行填写，若无问题点就不写，有问题点就如实备注。
* **大需求测试用例编写与执行**
* 6月15号后，有一个非常大的需求出来，安排羽墨来编写测试用例。需要明确计划时间，如果需求不明确或逻辑不清楚，会导致延期，那么这些问题要在备注中说明。
* 测试用例编写完成后，需要进行执行与验证。根据排期填写实际完成时间，例如6月17号。如果没有问题点就不写，有问题点就标注清楚。
* **测试用例管理技巧**
* 在测试用例执行过程中，如果因为需求变更导致用例需要更新，可以用红色字体标识。这样可以清楚知道哪些用例被修改过。同时，缺失用例在补充测试点后也要更新完善。
* 没有通过的测试用例要标记 bug 编号。这些是工作中的一些小技巧，不算问题点。例如：
* 用红色字体标识更新的测试用例；
* 补充缺失测试点；
* 标记 bug 编号。
* **测试思维固化与交叉测试机制**
* 说过，固定负责某一个模块，久而久之容易形成思维固化，导致测不出 bug。因此，在工作中一个非常重要的点就是进行\*\*交叉测试\*\*。
* 比如说，小明和小米分别负责不同模块，然后互相测试对方模块。这样可以发现彼此未注意的问题。同时也有好处：两人都熟悉系统全部模块。若一人请假，另一人可以独立完成整个系统的测试。
* **交叉测试时间安排与 bug 去重技巧**
* 时间安排上，要根据实际工作来定。比如第一轮测试各自模块，第二轮进行交叉测试。
* 交叉测试时要注意避免重复提交 bug。比如，小明提了一个 bug 还未修复，小米在交叉测试中再次发现同样的 bug。这样会造成重复提交，开发只会认第一次提交的 bug，第二次会被打回。
* 因此在提交前，要和测试队友沟通确认：
* “之前遇到这个问题了吗？”
* “是否已经提交了 bug？”
* 如果对方已提交，就不再重复提交；如果没有遇到，是新的问题，就可以由自己来提交。
* **用户操作手册的编写角色说明**
* 有些公司因为测试熟悉系统，会要求测试人员编写用户操作手册；但也有公司由产品经理负责这项工作。
* 所以大家进公司后要看实际情况。若需要测试编写，可以作为能力锻炼；若不需要，那也轻松一些。只是提醒大家，用户操作手册编写可能由测试负责。
* **项目负责人在测试计划中的角色与职责**
* 好，那么在这里可以看到，当我作为一个项目负责人，来做这一块的计划安排时，需要明确地把每一个人要做的事情整理清楚，包括他们是否已经完成了对应的任务。作为负责人，需要对每一个节点进行把控。
* **节点把控与每日沟通机制**
* 当某个任务快要到节点时，就需要去询问：“这个做完了没有？”“现在有什么困难？”也可以采用“每日一问”或“每日一更”的方式，了解任务的进展与可能存在的问题。
* **风险识别与团队支持**
* 如果团队中有一些经验不足的同事加入，那么也可以帮助他们识别和规避潜在风险。这些，都是作为负责人应该去承担和执行的职责。
* **风险前置与公司损失预防**
* 因为作为负责人，不能等到风险已经临近上线，或者快要结束的时候才发现问题。那样一来，这个“风险”就会变成真正的问题，而问题则可能会给公司带来经济上的损失。
* **小结：测试计划编写的意义**
* **测试计划的重要性与作用**
* 所以，这一部分正是在\*\*测试计划\*\*中需要考虑和落实的内容。测试计划的编写不仅是排任务，更是把控风险、协调进度、保障质量的重要手段。
* **测试计划文档的模板与结构**
* 那么再来看一下测试计划文档。在实际工作中，测试计划有很多种模板，无论使用哪一种模板，核心内容都是相似的。一般来说，会把这些模板发给大家作为参考。
* 这些模板的本质都是类似的：围绕产品任务、进度安排、执行跟踪，最后再加上备注说明。文档中最关键的部分包括：
* 计划开始时间
* 计划完成时间
* 实际完成时间
* 相关负责人
* **测试类型的详细分类**
* 在文档结构上，测试计划通常会分为多个阶段或类型，例如：
* 集成测试
* 功能测试（如业务流程功能测试）
* 前两轮问题产生后的回归测试
* 兼容性测试
* 性能测试
* UAT测试（用户参与测试，验证客户反馈的问题）
* 这些只是一些常见的测试说明，和今天带领大家整理的测试计划其实大同小异。
* **测试计划的文档形式：图表 vs. 文字**
* 在一些公司中，测试计划可能采用文档类的格式进行输出，比如写成正式的文档。虽然这种方式可能看起来更严谨，适合在交付客户时使用，但阅读体验可能偏枯燥。
* 更推荐在实际工作中使用图表类的测试计划，因为它更直观，能清晰展示：
* 测试流程安排
* 问题点位置
* 项目是否延期
* 当前是否正常进行

**实践任务与总结**

好，那么今天的测试计划就讲解到这里。任务是：

请大家根据自己在\*\*面试项目\*\*或\*\*当前实际项目\*\*中，尝试\*\*编写一份完整的测试计划\*\*。

* 在这份计划中，需要考虑：
* 如何安排团队成员的具体任务
* 如何设置时间节点
* 如何进行风险控制与进度跟踪
* 因为未来当作为团队的 leader，这正是必须完成并承担的核心任务。而写好测试计划，就是做好这一切的第一步。
* **实战作业：编写测试计划**
* 那么作业的话呢，大家就需要实际写一份\*\*测试计划\*\*。如果一开始不会写，也没有关系——可以\*\*参考提供的文档模板\*\*，仿照着写一写，这对自己的能力提升也是非常有帮助的。
* **分层建议：初学者与有经验者**
* 如果是\*\*初学者\*\*或暂时缺乏经验的同学，可以\*\*依照模板结构\*\*，逐项完成任务，逐步建立起对测试流程的理解。
* 如果是\*\*有经验的同学\*\*，或者在这方面已经比较熟悉，那么可以\*\*独立编写一份完整的测试计划\*\*，不依赖模板，结合自己的项目实际情况，思考以下内容：

* 如何分配各个测试模块
* 团队成员之间的任务安排
* 遇到过哪些具体问题，如何解决
* 有哪些经验可以迁移到当前测试项目中

这些内容都可以自然地延伸到软件测试工作的各个方面。

**总结**

好的，那么今天的课程就到这里。

感谢大家的参与，也希望们通过这个测试计划的编写练习，真正建立起项目思维和实战能力。

**1-11 测试策略&测试方法**

**测试设计阶段概述**

各位同学大家好，今天继续学习测试设计阶段的内容。在上一节课学习了如何编写测试计划，这节课将重点讲解测试策略和测试方案。

**测试策略核心概念**

测试策略的核心在于确定：

* 任务优先级（先做什么后做什么）
* 必要与非必要任务
* 资源合理分配

时间优化管理

**测试策略具体内容**

1. **明确测试目标**：

* 定义预期结果

1. 建立预期与实际结果的对比机制
2. **确定测试范围**：

* 划分功能模块

1. 按优先级安排测试顺序
2. **选择测试方法**：

* 白盒测试
* 灰盒测试

1. 黑盒测试
2. **制定详细测试计划**：

* 时间安排
* 资源分配

1. 问题跟踪机制
2. **准备测试数据**：

* 正常数据
* 异常数据

1. 边界数据
2. **测试用例编写**：

* 关注输入输出正确性

1. 明确执行步骤
2. **测试执行与缺陷管理**：

* 按计划执行用例
* 记录测试结果

1. 缺陷跟踪与优先级处理
2. **测试报告编写**：

* 总结测试结果

分析发现问题

**风险分析与应对**

1. **需求风险**：

* 问题：需求不明确/频繁变更
* 解决方案：
* 尽早沟通确认
* 建立变更规范流程

1. 保持开发测试信息同步
2. **时间风险**：

* 问题：进度延迟
* 解决方案：
* 合理预留缓冲时间
* 环境问题及时排查

1. 必要时人员借调
2. **人员风险**：

* 问题：人手不足/流动
* 解决方案：
* 团队人员储备

1. 跨部门协作机制
2. **技术风险**：

* 问题：技能欠缺
* 解决方案：
* 技术摸底
* 针对性培训

建立知识共享机制

**测试方案详解**

测试方案主要明确：

* 测试内容
* 实施方法

完成标准

示例（接口测试）：

* 使用技术：Postman等工具
* 测试思路：正向/异常场景

完成标准：覆盖率达标、缺陷修复率等

**测试计划vs测试方案**

|  |  |  |
| --- | --- | --- |
| 维度 | 测试计划 | 测试方案 |
| 关注点 | 做什么（整体规划） | 怎么做（具体实施） |
| 内容 | 全过程组织与约束 | 方法选择与环境规划 |
| 包含要素 | 时间、资源、风险管理等 | 工具、用例设计等 |

**实践建议**

1. 文档编写可根据公司要求合并或分开
2. 风险防控要提前规划

保持质量与效率平衡

**课后作业**

1. 整理课程笔记
2. 模拟练习：

* 假设作为测试组长
* 设计测试计划/策略/方案
* 进行口头表述练习
* **1-12 测试方法\_边界值**
* **测试用例方法概述**
* hello，各位同学大家好，今天来学习测试用例的方法。提到测试用例的方法，其实在前面已经学过测试方法。大家还记得之前学到的测试方法有哪些吗？对，大家应该想起来了，之前学到的测试方法有黑盒、白盒、灰盒这三种测试方法。那今天在学习测试用例编写时，首先要了解测试方法。
* **测试用例的具体方法**
* 这里所说的测试方法，指的是测试用例的具体、可操作性的测试方法。那么这些具体的、可操作性的测试方法都包括哪些呢？首先是边界值，其次是等价类、场景法和错误推测法。此外，可能有同学还了解过因果图、判定表以及路径覆盖法等。
* **重点学习的测试方法**
* 面对这么多测试方法，是不是每一种都要学习呢？其实不用。只需要重点学习前面四个方法，就已经可以应对工作中90%的场景了。至于后面提到的三种方法，主要是因为它们难度较大，而且在实际工作中用得比较少。因此，掌握前面四种方法，基本上就不会出现漏测的情况。
* **边界值方法的定义与意义**
* 首先来学习一下边界值这种测试方法。那么，什么是“边界”呢？在生活中，常说万事万物都有边界，也就是说，没有规矩不成方圆。可以把“规矩”理解为一种边界。在遵守规矩的过程中，规矩通常有一个最大上限和最小下限，这就是边界的含义。
* **边界值测试方法简介**
* 边界值是一种针对输入输出边界进行测试的方法，属于黑盒测试方法。并不关注系统的内部结构，而是关注外部输入输出结果的正确性。边界值测试的核心在于：大量的错误往往发生在输入输出范围的边界上。也就是说，bug并不是发生在输入输出的内部范围，而是发生在边界的外部或临近点上。因此，需要针对边界值设计测试用例，这样可以发现更多潜在错误。
* **边界值Bug示例**
* 举个例子，比如有一个真实的姓名输入场景。先对姓名进行正常保存，之后进行边界测试。如果输入的数据过长，系统无法保存，就会提示异常。这种情况就是典型的边界bug。应该对边界进行限制，比如最多只能输入30个字符。到达30个字符后，后面的内容可以截断，或者在输入框下方提示用户“最多只能输入30个字符”。这样用户才能明确知道问题出在哪里，而不是输入超长字符后页面报错却不明所以。这就是边界值相关的bug。
* **边界值测试的具体做法**
* 那么，边界值测试具体怎么做呢？边界值测试其实比较简单，就是取域内和域外的值进行测试。简单总结，边界值测试通常取域内和域外边界上的四个数，然后在中间随意选择一个数即可。如果需求或功能规定了取值范围，就取该范围的边界值，以及刚刚超越该范围边界的值作为测试数据。
* **上点、离点、内点的概念**
* 在边界值测试中，有“上点”“离点”“内点”这几个概念。有些同学可能会疑惑它们的含义。实际上，“上点、离点、内点”指的是：选取正好等于边界、刚好小于边界、刚好大于边界的点作为测试数据。可以将边界值测试划分为上点、离点、内点三类。
* **上点、离点、内点的定义**
* **上点**是指选取在边界上的点。实际工作中，大家经常容易只选左边的点，忘记右边的点。因此要记住，上点指的是边界上两侧正好等于边界值的点，这两个点都需要测试。
* **离点**是指距离上点左右两边最近的点。也就是说，离点通常有两个，分别在上点的左右两侧，都是需要测试的数据。
* **内点**则比较简单，是指在上点范围内的任意一个点，也就是在取值域内的一个中间值或相近的值，用于测试。
* **上点、离点、内点举例说明**
* 以年龄填写为例，假设允许的年龄范围是18到35岁，只有在这个范围内填写信息才能报名成功，小于18或大于35都不行，18和35包含在内。
* 上点：18和35
* 域内离点：离18最近的是19，离35最近的是34
* 域外离点：17和36
* 内点：如24
* 通过这个例子，可以更直观地理解上点、离点和内点的含义。
* **拟点优化口诀：开内闭外**
* 在优化离点选择时，有一个口诀：“开内闭外”。意思是：
* 开区间时，选择内部的离点
* 闭区间时，选择外部的离点
* 以闭区间为例（如18到35岁，包含18和35），选择域外的离点（17和36），而域内的离点可以省略。这样可以更清晰地梳理有效值和无效值。
* **区间类型说明**
* **开区间**：上点的值不包含在内，如(18, 35)，18和35都不包括。
* **闭区间**：上点的值包含在内，如[18, 35]，18和35都包括。
* **半开半闭区间**：一边包含一边不包含，如(18, 35]或[18, 35)。
* 实际工作中，输入框等场景多采用闭区间，便于用户理解和输入。
* **有效值与无效值的梳理**
* 以闭区间[18, 35]为例：
* 有效值：18、24、35（边界值和内点）
* 无效值：17、36（域外的离点）
* 通过这种方式，可以确保不遗漏任何边界上的测试场景。
* **边界值测试用例案例：名称长度为2到10位**
* 再来看一个案例，并将边界值测试用例写出来。假设有一个需求：输入名称必须为2到10位数字，才能新增项目成功。也就是说，只要名称是2到10位数字，就可以新增项目。
* **边界点、内点、离点的确定**
* 在这个需求下，依然需要先找出上点、内点和离点：
* **上点**：2位和10位数字
* **内点**：任意一个在区间内的值，比如5位数字的名称
* **离点**：对于闭区间，选择外部的离点，即1位和11位数字
* 这样，有效值为2位、10位、5位数字，无效值为1位、11位数字。
* **测试点梳理与用例编写**
* 有同学可能会疑惑测试用例应该怎么写。其实，在测试行业中，通常是先确定测试数据和预期结果，其他内容用文字表达即可。
* 输入2位数字的名称，新增项目成功（上点）
* 输入10位数字的名称，新增项目成功（上点）
* 输入5位数字的名称，新增项目成功（内点）
* 输入1位数字的名称，提示不符合规范（离点，域外）
* 输入11位数字的名称，提示不符合规范（离点，域外）
* 这里的“提示不符合规范”是因为实际工作中产品经理会给出具体文案，如果没有，可以先用概括性描述，并备注让产品经理补充文案，后续再核对。
* **关于测试用例的粒度**
* 有同学可能会问，为什么不测0位或12位？因为只要1位和11位都不符合规范，其他超出范围的情况也会被控制。
* 这种写法下，通常需要写5条测试用例，每条对应一个测试点，后续再补充详细步骤。这种方式较为简单，适合初学者或需要细致覆盖的场景，比如银行类项目。写得越细，越不容易遗漏。
* **测试用例的简化与总结**
* 实际工作中，可能因为时间等原因，无法写得很细。这时可以采用简化写法：
* 输入2到10位数字的名称，新增项目成功
* 输入不符合2到10位数字的名称，提示不符合规范
* 这种总结类写法适合有经验的测试人员，但初学者容易遗漏边界场景，因此建议先从详细用例写起，等测试思维成熟后再尝试总结类写法。
* **方法总结**
* 本案例用到的都是边界值方法。通过边界值，可以系统整理测试数据，并丰富测试用例内容。以上就是本案例的完整说明。
* **边界值的基本总结**
* 在这里来对边界值进行一个总结。边界值在实际工作中应用非常广泛，通常具有“左右”之分，也就是有最小值和最大值。在考虑边界时，一定要同时关注左边界和右边界，这样才能避免遗漏。
* **边界值常见应用场景**
* 边界值常见于以下几类功能：
* **长度边界**：如输入框的字符长度，常见于用户名、密码等字段。
* **大小边界**：如图片大小、分辨率、文件大小、视频大小等。例如，图片上传时会有最大和最小支持的图片大小。
* **数量边界**：如微信中图片最多可上传多少张，超过数量如何处理。

**高低边界**：如高度的最大值和最小值等。

这些边界不仅仅局限于数字类型，很多功能都涉及边界值的测试。

**边界值的类型与适用范围**

一般来说，输入框相关的边界值以整数为主，因为字符数通常无法出现“半个字符”这种情况。小数边界主要用于金额等场景，比如微信发红包时，最小金额可以是0.01元，这类场景下需要关注小数边界，常见于金融、银行等业务。

**边界值在各类功能中的应用**

边界值不仅适用于文本框的长度限制，还适用于如报表的行数（第一行、最后一行）、消息发送（字符数、语音时长、图片/视频数量和大小）等多种功能。以微信发消息为例，涉及文本、语音、图片、视频等多种类型，每种类型都存在不同的边界值需要测试。

**边界值与等价类的关系**

边界值通常与等价类划分结合使用。边界值主要用于确定长度、大小、数量等的极限值，等价类则用于确定输入的类型。在编写测试用例时，建议将边界值和等价类结合起来使用，这样可以更全面地覆盖测试场景。后续课程会详细讲解等价类的内容。

**课堂作业与实践建议**

今天的作业包括：

1. 针对微信发消息功能，梳理并写出相关的边界值场景，包括文本、语音、图片、文件、视频等。

* 整理本节课的学习笔记，并尝试默写加深记忆。
* 微信发消息的边界值场景也是面试中常见的高频问题，建议大家提前准备，提升测试思维能力。
* 本节课内容到此结束，谢谢大家。
* **1-13 测试方法\_等价类**
* **等价类方法简介**
* hello各位同学，大家好！在上一节课中，学习了测试用例中的第一个方法——边界值法。也提到，边界值与等价类密切相关。本节课将进一步学习等价类，探讨边界值与等价类之间的关系。
* **等价类的基本概念**
* 在软件测试中，测试数据是无穷无尽的。例如，微信发红包时，金额可以从0.01到200元，理论上可以输入两万个不同的值。实际工作中，不可能把所有的值都测试一遍，这样既耗时又低效。因此，需要使用等价类的方法。
* 等价类方法的核心思想是：通过选取少数具有代表性的值，去覆盖所有的值。这样可以将大量的测试数据，归纳为少数几个测试数据，从而大大减少测试工作量。
* **等价类的分类与作用**
* 等价类可以分为有效等价类和无效等价类。实际上，就是将测试过程中一些输入、输出、操作等相似内容进行分组，从每组中挑选具有代表性的内容，作为测试用例。
* 有效等价类：能够被系统识别的输入数据。例如，要求输入2到10个汉字，只有2到10个汉字的输入才是有效的。

无效等价类：不能被系统识别或导致系统功能无法实现的输入数据。例如，输入英文、数字或长度不在2到10之间的内容。

**等价类的适用场景**

等价类方法适用于多种场景，比如输入框、单选框、多选框等。例如：

* 输入框：输入或不输入，输入有效或无效内容
* 单选框：选择正确、错误或不选择
* 多选框：不选、单选、全选等
* **微信发红包案例分析**
* 以微信发红包为例，说明等价类的应用：
* 边界值：0.01元和200元
* 有效等价类：金额在0.01到200之间，且余额充足时，可以成功发红包。例如，0.01、200、0.02、199.99、88.88等，这些值可以覆盖全部两万个可能的输入。
* 无效等价类：输入中文、英文、标点符号等非数字内容，或输入超出200元的金额、不输入金额、输入格式不正确（如“0.”后面留白）等，这些都属于无效等价类。
* 在实际测试时，只需用实际数据进行验证即可，无需每个用例都写具体的测试数据。用简洁、概括性的语言总结测试点即可。
* 以上就是微信发红包的等价类分析示例。
* **为什么要测试无效等价类？**
* 有同学可能会有疑问：无效等价类既然不在规定范围内，为什么还要测试？可不可以不测？其实这是不可以的。因为如果不测试超出范围的情况，就无法确认系统是否真的禁止了这些输入。例如，只测试了200元的上限，但没有测试超过200元的情况，就无法保证系统能正确处理。因此，无论是有效等价类还是无效等价类，都需要进行测试。
* **如何通过边界值划分等价类？**
* 可以通过边界值来找到有效等价类和无效等价类。具体做法是利用“上点、里点、内点”等方法，结合输入的长度和数据类型进行划分。按照长度划分时，采用的是边界值方法；按照类型（内容）划分时，采用的是等价类方法。通常，边界值和等价类是结合使用的。
* **案例：输入两到十位数字的中文名称**
* 以“输入两到十位数字的中文名称”为例：
* 长度要求：两到十位数字

1. 类型要求：内容为中文
2. 只有同时满足长度和内容要求，才能算作有效输入。
3. **有效等价类的梳理**
4. 有效等价类包括：
5. 两到十位的中文名称（如两位、六位、十位的中文名称），均可新增项目成功。
6. 这些测试点可以用边界值法来覆盖，如测试两位、六位、十位的中文名称。
7. **无效等价类的梳理**
8. 无效等价类包括：
9. **长度不符合要求但内容符合规则**

* 一位中文名称

1. 十一位中文名称
2. **长度符合要求但内容不符合规则**

* 两到十位的非中文内容（如特殊字符、字母、空格等）

1. 中文与英文、特殊字符的组合
2. **长度和内容都不符合要求**
3. 非中文内容且长度不在两到十位之间
4. **空输入**

* 输入框为空时，若该项为必填，则应提示错误；若为非必填，则允许为空。
* **测试用例示例**
* 有效等价类：
* 输入两位中文名称，新增成功
* 输入六位中文名称，新增成功
* 输入十位中文名称，新增成功
* 无效等价类：
* 输入一位中文名称，页面提示不满足条件
* 输入十一位中文名称，页面提示不满足条件
* 输入两到十位的特殊字符、字母或空格，页面提示内容不符合要求
* 输入中文与特殊字符、字母的组合，页面提示内容不符合要求
* 输入框为空，页面提示必填项（如适用）
* **关于重命名校验**
* 在实际工作中，还需考虑“重命名校验”这一特殊场景。并非所有名称都需要唯一性校验，例如身份证号码需要唯一性，但姓名可以重名。是否需要做重命名校验，取决于产品需求。建议在测试用例中默认考虑重命名校验，避免遗漏场景。如果产品不要求，可以删除相关测试点。
* 重命名校验测试点示例：用户输入已存在的名称，页面提示“该名称已存在”。
* 以上内容即为有效等价类和无效等价类的梳理及测试点总结。
* **总结类等价类的编写方法**
* 也可以采用总结类的方式来梳理等价类。具体来说，依然分为有效等价类和无效等价类两部分。
* **有效等价类**：输入两到十位的中文名称，新增项目应成功。新增成功后，页面会新增一条数据，这一点在测试用例中也会被覆盖。通常只需用一条用例即可覆盖该场景。
* **无效等价类**：可以这样总结：

1. 输入长度不在两到十位之间的中文名称（如一位或十一位中文名称）。
2. 输入非中文格式的名称（如特殊字符、字母、空格等，或它们的组合）。
3. 不输入名称，页面应给出提示。这个场景可以选择写在结尾，也可以省略，视实际需要而定。

* 存在重名时，进行重命名校验。可以写为“不能保存成功”或“页面给出提示”，具体提示内容根据产品需求而定。测试时只需验证该场景是否被覆盖即可，至于产品是如何处理的，不影响测试用例的设计。
* 通过上述总结，可以清楚地知道需要测试哪些内容。实际上，这样的边界值和等价类的编写方法非常简单明了。
* **等价类的作用**
* 等价类的主要作用和目的，是帮助设计测试数据，将无限的数据集合转化为有限的、具有代表性的测试数据。通过有效和无效等价类的划分，使测试用例具有代表性和完整性，从而提升测试效率和覆盖度。
* **边界值与等价类的关系**
* 边界值和等价类之间的关系密切。通常通过边界值来确定等价类，边界值一般作为等价类的补充，两者经常一起使用。可以说，边界值和等价类是“好朋友”或“搭档”，缺一不可。
* 在测试方法上，边界值和等价类都属于黑盒测试方法，即只关注输入和输出结果，不关注内部实现结构。两者的区别在于：边界值分析会将等价类的每一个边界都作为测试条件，而不是随意从等价类中选取一个代表。边界值分析不仅考虑输入条件，也考虑输出条件，关注等于、大于、小于边界的情况。
* 通常，边界值是等价类的补充，测试用例往往来自等价类的边界。大家要记住，边界值和等价类是相辅相成、成双成对出现的关系。
* **等价类的两个划分维度**
* 等价类是从两个维度进行划分的：长度和内容。长度通常对应边界值，内容则是规则限制，这部分由产品经理来决定。需要注意的是，边界值和等价类主要适用于单个功能点的测试，比如登录、搜索、新增等。如果需要串联多个场景，则需要使用场景法（将在后续课程讲解）。
* **手机号码输入框测试用例举例**
* 以手机号码输入框为例，设计测试用例时同样要区分有效等价类和无效等价类。
* 有效等价类：输入11位数字，且符合手机号码规则的号码。
* 无效等价类：
* 非11位数字的手机号码（如10位、12位等）。
* 输入非数字内容（如中文、字母、特殊字符，或与数字的组合）。
* 输入框为空（需根据是否为必填项判断其有效性）。
* 输入11位数字但不符合手机号码规则的号码（如“12345678901”虽然是11位数字，但不一定是有效号码）。
* 需要注意，是否符合手机号码规则通常由开发通过正则表达式进行校验。只需知道有正则表达式这种校验方式即可，不必掌握具体写法。
* **空值场景的特殊说明**
* 如果手机号码为必填项，输入为空属于无效等价类；如果不是必填项，输入为空则属于有效等价类。保存后页面显示为空白即可。
* **特殊规则场景**
* 像手机号码、邮箱、身份证号码、银行卡号、车牌号等字段，通常会用正则表达式进行规则校验。对于这类单一功能点的场景，边界值和等价类方法都适用。
* **边界值与等价类方法复盘**
* 本节内容回顾如下：
* 边界值：选取输入输出范围的边界进行测试，因为大量错误发生在边界上。边界值分析属于黑盒测试方法，并涉及上点、离点、内点等概念。只要功能涉及长度、大小、数量等，都与边界值相关。
* 等价类：用于将无限的测试数据集合转化为有限且有代表性的测试数据。通过划分有效等价类（系统能识别的）和无效等价类（系统不能识别的），提升测试用例的代表性和完整性，提高测试效率和覆盖度。

边界值和等价类常常结合使用，互为补充。通过边界值可以找到等价类，两者成双成对、相辅相成。

**概念性内容的掌握与应用**

只要能熟练表达上述概念，面试时一般不会有问题。实际工作中，若因功能复杂而遗漏某些边界或等价类也无需担心，随着经验积累和重复实践，自然会熟练掌握。

**课后作业**

请大家课后在慕课网随便找一个功能，简单写出其有效等价类和无效等价类。

本节课内容到此结束，谢谢大家。

**1-14 测试方法\_场景法**

* **场景法简介**
* hello，各位同学大家好，今天学习测试用例方法中的场景法。前面的两节课程，学习了边界值和等价类的方法。这两种方法主要适用于单个功能的测试。如果需要把整个场景串联起来，边界值和等价类方法就不太合适了。因此，今天要学习的场景法，是通过运用场景来对系统的功能点或者业务流程进行测试，这样可以提高测试的效果。
* **场景法的定义与核心思想**
* 那么，场景法到底是一种什么方法呢？其实，场景法的全名也叫做用户场景法。也就是说，把自己当做一个最终用户，在使用软件时，可能会遇到哪些情况，把这些场景一一列举出来。场景法的主要目的是测试软件主要业务流程、主要功能的正确性，以及其处理异常的能力。
* **场景法的应用举例**
* 比如说，像微信发红包这样的功能，作为用户，会如何操作这个功能？可能会遇到哪些情况？每一个情况就是一个场景。通过用户场景法，把所有场景全部找出来。在实际运用场景法的初期，可能会不熟练，可能会有一些场景被漏掉。遇到这种情况也不用担心，漏掉就漏掉，不要害怕背锅。后面也会讲到，如何避免漏掉一些场景。
* **场景法的概念与作用**
* 再来总结一下场景法的概念：通过运用场景，对系统的功能点或业务流程进行测试。场景法可以提高测试结果的有效性。在这里也提到，场景法也叫做用户场景法，就是站在用户的角度去考虑，把自己当做真实的用户，在使用软件时，可能会遇到哪些问题，把这些场景一一列举出来。
* **场景法的目标**
* 场景法的主要目的是测试软件的主要流程的正确性，以及其处理异常的能力。实际上，场景法可以分为正确的场景和错误的场景。在工作中，同事们在梳理这块内容时，可能会说“梳理一下正向的场景”或“有效的场景”，其实都是指正确的场景。有的同事会说“梳理一下错误的场景”“失败的场景”“逆向的场景”或“无效的场景”，本质上就是正反向场景的梳理。
* **场景法的本质与等价类的关系**
* 有些同学可能会疑惑，场景法的本质到底是什么？其实，场景法的本质是基于等价类划分的测试技术。从技术层面来说，场景法也是用等价类的技术。比如，场景可以分为正向和逆向，这在等价类中可以对应为有效等价类和无效等价类。
* **场景法与等价类的区别**
* 场景法和等价类的差异在于：场景法应用于对软件业务的深入了解。从业务层面来说，场景法需要了解业务。它是基于对软件业务的深入理解。软件业务是什么？业务一定来源于需求，所以业务可以理解为需求。从技术层面来说，就是对需求和业务的深入了解。
* 这就是场景法区别于等价类的地方。等价类主要针对单个功能点，不需要把整个业务串联起来。而场景法则需要把整个业务流程串联起来。
* **微信发红包场景法举例**
* 以“微信发红包”为例，来说明场景法的应用。假设有张三和李四两位用户，张三给李四发红包。先只考虑成功的场景，不涉及单个功能点（如金额输入是否合法等，这些可用等价类或边界值法测试）。在所有前置条件都满足的情况下，张三微信发红包成功。
* 具体流程如下：张三输入正确的金额（如100元），选择任意支付渠道且余额充足，支付成功。此时，张三的余额应减少相应金额，李四接收红包后，余额增加相应金额。双方的红包状态也会发生变化，如张三发出红包、李四接收红包后，系统会有相应的提醒和状态变更。
* **失败及特殊场景举例**
* 除了上述成功场景，还需考虑其他情况。例如，张三发红包成功，但李四未在24小时内接收，系统会将红包金额退回张三，张三余额增加，李四余额不变。如果李四在24小时内接收，则流程回到成功场景。
* 还有一种情况，张三发出红包后，李四未接收，张三主动撤回红包，张三余额增加，李四余额不变。这个场景在面试时也可以作为补充说明。
* 此外，微信发红包还包括群发红包（如拼手气红包、定额红包等），但面试时建议优先讲解单个红包的流程，因其业务流程更简单，便于突出重点。
* **场景法的核心概念**
* 通过上述例子，可以更好地理解场景法。场景法就是站在用户角度，梳理在使用功能时可能遇到的各种情况，列举所有场景，目的是测试系统主要流程的正确性及异常处理能力。
* **场景法的三种流程**
* 场景法可分为三类流程：
* **基本流**：即正确流程，模拟用户按照正常操作完成业务，验证软件流程和主要功能的正确性。
* **备选流**：即错误流程，模拟用户因误操作等原因走错流程，但可通过修正回到正确流程，验证软件的错误处理能力。例如导航走错路后纠正，最终仍能到达终点。
* **异常流**：即流程终止，用户因连续错误操作无法完成业务。例如购票系统三次未支付后当天无法再购票，或银行系统密码输错三次被锁定，必须线下处理。
* **场景法与等价类、边界值的关系**
* 场景法主要关注业务流程的梳理，优先考虑流程而非单个功能点。单个功能点可用等价类或边界值法设计测试用例。实际工作中，流程优先于功能点，场景法用于流程测试，等价类和边界值法用于功能点测试。项目时间紧张时，优先保证场景覆盖，功能点可适当放宽。
* **场景法的用例设计原则**
* 每一个场景都可以视为一个测试用例。在设计场景时，不需要考虑软件以外的因素（如网络、硬件等），只需关注软件本身的业务流程。不是所有功能都有失败场景，例如朋友圈点赞、评论点赞等单向功能没有失败场景。设计时只需说明该功能无失败场景即可。
* **共享单车业务流程场景举例**
* 以共享单车为例，业务流程可从是否登录、是否会员、是否购买等角度梳理。会员和非会员的流程略有不同：
* **会员流程**：未登录时提示登录，登录后扫码、开锁、骑车、还车。会员有效期内可免费骑行，过期后需付费。
* **非会员流程**：未登录时提示登录，登录后每次骑行需付费，付款成功后可继续骑行，未付款则下次无法骑行。
* **多端设备场景**：需考虑不同设备、不同渠道（如APP、小程序、支付宝、微信等）的使用限制。

**异常场景**：如扫码信号差、未开蓝牙、扫码非本平台车辆等，都需作为异常场景考虑。

此外，会员购买周期（月卡、年卡、季卡、周卡等）不同，需分别测试。一般来说，周、季、年卡更容易出bug。

**其他业务流程场景举例**

作业建议：可尝试梳理微信群发红包、慕课网业务流程、淘宝购物车业务流程等。以淘宝购物车为例，涉及单个/多个商品购买、不同商家组合付款、优惠券叠加或互斥、满减活动、订单数量、平台规则等，流程较为复杂。

**场景法的总结与注意事项**

场景法的核心是把自己当作最终用户，列举所有可能遇到的场景，验证主要流程的正确性和异常处理能力。场景法强调流程优先，细节功能靠后。场景法本质上也用到了等价类技术，将场景分为正确和错误两类（对应有效和无效等价类），但更关注整体流程。

在实际工作中，场景覆盖不全时，需区分遗漏的是大场景还是小场景。小场景可补充，大场景需反思并举一反三，避免后续重复犯错。没有人能100%覆盖所有场景，关键在于积累经验、不断完善测试思维。

以上就是本次关于场景法的课程内容，谢谢大家。

**1-15 测试方法\_错误推测法&测试用例基本概念**

* **场景法回顾与引入错误推测法**
* hello，各位同学，大家好。上一节课学习了场景法，并通过案例了解了场景法在功能测试中的应用。场景可以分为正向和逆向，正确和错误的场景。场景法的技术基础也是等价类划分，但从业务层面来看，场景法更专注于业务本身。
* 今天要学习的是错误推测法。什么是错误推测法？它是基于自身经验，判断哪些地方容易出错，从而有针对性地进行测试。比如，多个查询条件组合查询时容易出错，根目录下有内容删除但未做校验等，这些都是常见的错误点。
* **错误推测法的特点与应用**
* 错误推测法有些推测无法直接写进测试用例，往往是在用例执行后，结合经验设计和执行一些探索性、主观性的测试用例。如果在执行中发现了bug，可以再补充到用例中。错误推测法属于探索性测试，依赖测试人员的经验，经验越丰富，发现的问题也越多。
* 其实，错误推测法也有一定的规则可循，比如从应用性、用户体验角度进行测试，也能发现许多问题或产品缺陷。例如，登录时长、界面跳转时屏幕亮度、色彩、饱和度等，尤其在移动端如夜间人脸识别时，亮度适配就很重要，这些都属于探索性测试的范畴。
* **常见bug实例总结**
* 下面通过实例总结工作中常见的bug，这些内容在第一节课的易用性测试中也有提及：
* **输入框字符长度控制**所有可输入标题的地方应设置长度限制。例如，某些bug管理工具未限制输入长度，用户不清楚可输入多少字。建议给出提示或截断，超出部分不允许输入。
* **唯一性校验**是否需要唯一性校验取决于功能和需求。比如通过ID区分而非内容校验唯一性也是可以的。
* **必填项星号标记**所有必填项前应有星号标记，便于用户识别。
* **规则验证与特殊字符限制**根据产品规则验证输入内容。部分产品对特殊字符不做限制，但如微信昵称等则有限制，需根据需求决定。
* **新增数据后显示问题**新增数据成功后，不能再次新增时还显示上一次的数据，或无操作时显示提示信息。
* **新增与编辑页面文案不一致**新增与编辑弹窗的文案、内容应保持一致，避免混淆。
* **提示信息与操作类型不符**新增项目时提示“编辑项目信息不对”，属于文案不一致。
* **列表文案与列头不一致，排序问题**新增界面文案与列表列头应一致，列表应按时间倒序排列，最新数据在最上面。
* **回车键搜索支持**列表搜索时应支持回车键触发搜索。
* **分页与数据一致性**切换分页后查询操作，列表数据和分页指示应一致。
* **有子目录或数据时禁止删除**有子目录或数据时不能直接删除，需二次确认，防止误删。
* **多选删除提示**多选删除时应有相应的提示文案。
* **文件上传与下载校验**文件上传需校验格式和大小，下载后文件名应与数据名一致。
* **登录权限与超时提示**登录超时后应有明确提示“登录超时，请重新登录”，而不是无提示直接退出。
* **无权限菜单访问控制**没有权限的菜单，通过URL或模块访问也不能进入。
* **已选项不可重复添加**已选择的复选框或单选框应置灰或过滤，防止重复添加。
* **下拉列表支持搜索**非固定数据的下拉列表应支持搜索，便于数据量大时查找。
* **下拉列表显示名称而非值**下拉列表应显示名称，不显示数据值。
* **分类页签定位**树结构或分类页签下操作后，应定位到当前分类或页签。
* **弹框可拖动**弹框界面最好支持拖拽，便于查看被遮挡内容，保持产品一致性。
* **鼠标悬停提示**鼠标悬停时应有提示，如关闭按钮、刷新、收藏等。
* **页面布局与美观**页面布局、样式、美观性很重要，影响用户第一印象。即使功能完善，UI不美观也会被吐槽。
* **版本兼容与数据兼容**新版本需兼容旧数据，不能因升级导致旧数据无法使用。
* **新建或修改记录排序**新建或修改成功的记录应按时间顺序排在列表首页。
* **同级目录查重**查重只针对同级目录，不对整个产品查重。例如，同部门只能有一个“张三”，但不同部门可以。
* **查重时只查本分类**查重时只查本部门（分类）下的数据，不查其他部门。
* 这些都是基于经验总结的错误推测法，具有一定主观性，但在实际工作中非常实用。大家如果担心工作中发现不了bug，可以参考这些内容，面试时也可以用实例与面试官交流。
* **测试用例基本内容**
* 接下来学习测试用例的内容。测试用例可以用思维导图或表格编写，实际工作中多用表格。测试用例的主要内容包括：
* **编号**可自动生成或自定义，如“功能001”、“功能002”。
* **名称**简明表达测试点，如“账号密码正确，登录成功”。
* **前置条件**如登录成功、报名成功等。部分用例前置条件简单，如进入网站、点击模块等。尽量不要留空。
* **优先级**分为高、中、低。优先级高的先执行，低的可后执行，便于合理安排测试进度。
* **重要级别**也分为高、中、低，指是否为核心业务。例如银行的转账、存款、取款等。
* **测试数据**一般采用控制变量法，如测试姓名时，其他字段均为正确值，便于定位问题。
* **测试步骤**如实描述操作步骤：第一步登录，第二步选择用户，第三步点击转账，输入金额，点击下一步，支付等。
* **预期结果**明确期望，如转账成功，对方收到提醒，余额减少等。
* **实际结果**实际测试结果，若与预期一致则通过，否则记录bug。
* **备注**可选项，补充说明。
* **测试用例设计流程与目的**
* 在需求分析阶段，需梳理需求、提取功能点，对每个功能点设计场景（测试集），再为每个场景设计测试数据。
* 写测试用例的目的有：
* 避免漏测，有据可查。
* 量化测试工作量，如写了500条用例，执行250条即完成50%。
* 便于后续回溯和经验积累。
* 更加精准梳理需求，加深对需求的理解。
* 外包公司可能写用例和执行用例不是同一人，自研公司则可能是同一人。对需求不熟悉时，可对照用例完成测试，加深系统理解。
* 所有测试用例都与时间相关，时间紧可简写，执行时多补充；时间充裕可写得更详细。
* **编写测试用例的注意事项**
* 用例越细致越好，覆盖越全面越好。
* 功能流程、页面美观、UI交互都要关注。
* 前后端联动、交叉测试也需考虑。
* 编写步骤：熟悉需求→提取功能点→设计场景→设计数据。
* **课后作业与总结**
* 本节课作业：整理以上错误推测法的笔记，常见bug总结共29条，建议理解并记忆20条以上，便于面试时举例说明，丰富项目经验。
* 本节内容主要介绍了错误推测法、测试用例的内容和目的。希望大家能非常流利地回答测试用例包含哪些内容。作业建议有条件的同学进行默写。
* **1-16 测试用例带写实战**
* **测试用例编写的意义与基本结构**
* hello，各位同学大家好，今天学习测试用例的编写。我会带着大家一起写一个模块的完整测试用例。这样大家以后进入工作后，也能很快上手编写自己负责模块的用例。
* 在上一节课中，已经学习了测试用例的基本内容，包括：编号、名称、前置条件、优先级、重要级别、测试数据、测试步骤、预期结果和实际结果。学习和编写测试用例的目的，主要是为了避免测试过程中出现漏测，评估测试工作量，更加精准地梳理需求，同时帮助更快熟悉系统。
* **测试用例的两种编写形式**
* 测试用例有两种常见的编写形式：一种是使用模板表格，另一种是用思维导图。今天先学习用模板表格来写，之后再整理成思维导图。实际工作中，通常只需选择一种方式即可。表格形式更正式，交付客户时也更规范美观，所以本节课主要讲解如何用表格编写测试用例。
* **测试用例模板结构说明**
* 测试用例模板一般包括：编号、模块、测试点（名称）、前置条件、测试步骤、优先级、重要级别、预期结果、实际结果等。优先级和重要级别有时可以合并。实际结果可以用勾或叉表示是否符合预期，不需要重复写一遍。
* **业务流程测试用例实战——以慕课网为例**
* 以慕课网为例，串联写一条完整的业务流程测试用例，包括：跳转国家精品课程、轮播图切换、报名课程、评论、编辑、删除等。
* **1. 跳转国家精品课程页面**
* 测试点：跳转国家精品课程页面成功
* 前置条件：登录成功
* 测试数据：无（本用例无需测试数据）
* 测试步骤：

1. 点击课程
2. 选择国家精品课程

* 预期结果：正确跳转到国家精品课程页面，页面内容完整美观，可上下滑动
* **2. 轮播图切换展示**
* 前置条件：已进入国家精品课程页面
* 测试步骤：

1. 观察轮播图是否自动切换展示图片

* 预期结果：轮播图图片自动切换，展示正确
* **3. 轮播图跳转课程链接**
* 前置条件：已进入国家精品课程页面
* 测试步骤：

1. 点击轮播图中的某一图片

* 预期结果：正确跳转访问对应课程链接，页面内容完整美观，可上下滑动
* **4. 课程报名流程**
* 步骤一：点击“立即参加”按钮
* 预期：弹出认证学习模式和免费学习模式选择
* 步骤二：选择认证学习模式
* 预期：弹出认证学习模式页面，可查看全部权益，页面内容完整美观
* 可选择半年或年度付费，进入后续付费流程（本节不展开）
* 步骤三：选择免费学习模式
* 预期：进入免费学习模式页面，页面文案正确
* **5. 免费课程报名**
* 前置条件：已进入免费学习课程页面
* 测试步骤：

1. 点击“立即报名”按钮
2. 弹出是否同意协议
3. 勾选协议
4. 点击“确定”按钮

* 预期结果：报名成功，页面按钮显示为“已参加”，可进入学习页面
* 场景补充：报名成功的课程可在个人中心查看
* **6. 报名失败场景**
* 步骤：点击“立即报名”但未勾选协议
* 预期结果：确定按钮置灰，无法报名，不能学习该课程
* **7. 取消或关闭报名弹窗**
* 步骤：点击“取消”或“关闭”按钮
* 预期结果：退出报名页面，未报名成功，不能学习该课程
* **课程评论相关测试用例**
* **8. 课程评论区默认展示**
* 前置条件：无（登录与否均可）
* 测试步骤：

1. 进入相关课程页面
2. 查看课程评价

* 预期结果：评论区默认展示全部课程评论
* **9. 评论课程成功**
* 前置条件：用户已登录并报名课程
* 测试步骤：

1. 点击“提交评价”按钮
2. 选择星级评分
3. 输入5-500字评价内容
4. 点击“提交”按钮

* 预期结果：评论成功，评分和内容正确展示，已评论用户无法再次评价
* **10. 评论必填项校验**
* 情况一：评分和内容均未填写
* 预期：页面提示“请输入评价星级”和“请输入评价内容”
* 情况二：评分未选但内容已填，或评分已选但内容未填
* 预期：页面分别提示缺失项
* **11. 评论内容边界值校验**
* 输入少于5个字
* 预期：页面提示“请输入至少五个字符”
* 输入超过500个字
* 预期：页面提示“已超过xxx字”，并提示“请输入内容”
* **12. 取消或关闭评论操作**
* 步骤：在评论页面点击“取消”或“关闭”按钮
* 预期结果：退出评论页面，再次进入时不保留上次操作数据
* **13. 未报名用户评论限制**
* 前置条件：用户未报名该课程
* 预期结果：不显示评论功能
* **编辑与删除评论测试用例**
* **14. 编辑评论**
* 步骤：点击“编辑”按钮，修改内容后提交
* 预期结果：编辑后内容和评分正确展示
* **15. 删除评论**
* 步骤：点击“删除”按钮，弹出确认，点击“确定”
* 预期结果：评论删除成功，页面重新显示“提交评价”按钮
* 取消删除或关闭弹窗
* 预期：评论未被删除，依然展示在页面上
* **点赞与取消点赞测试用例**
* **16. 点赞功能**
* 步骤：点击点赞按钮
* 预期结果：点赞成功，点赞数量增加，个人中心可查看获赞数量
* **17. 取消点赞**
* 步骤：再次点击点赞按钮
* 预期结果：取消点赞成功，点赞数量减少，个人中心同步更新
* **其他说明与方法总结**
* 分页功能等类似场景可按“点击页码跳转，页面内容正确显示”方式编写。遇到不明确的地方或唯一性校验需求，应及时与产品经理沟通并备注。
* 本节课用到的测试用例设计方法包括：所见即所得、场景法、等价类、边界值等。大家可以结合实际需求灵活运用。
* **课后作业与下节预告**
* 课后建议大家在慕课网任选一个模块练习编写测试用例。下节课将把这些用例整理成思维导图，并介绍如何在需求变更或功能迭代时补充测试用例。
* **1-17 测试用例评审&目的**
* **思维导图与测试用例编写**
* hello，各位同学大家好，今天来学习如何用思维导图来学习和编写测试用例。在上一节课中，已经用表格的形式编写了测试用例，今天会更简单一些，把这些思路迁移到导图上即可。
* 用思维导图编写测试用例时，通常分为以下几个部分：首先是名称，然后是前置条件，接着是测试步骤，最后是预期结果。按照这个结构来梳理，比如上一节课学到的“跳转国家精品课程成功”，首先前置条件是登录成功，然后点击课程选择国家精品课程，最后写出预期结果。
* **优先级与导图标注**
* 在导图中，可以备注优先级。如果觉得某条用例很重要，可以用图标或标记表示其重要性，比如标注为“一级”。有些导图工具支持重点标注，也可以备注优先级。这样可以更直观地表达用例的重要程度。
* **执行与结果标记**
* 执行测试用例时，如果有问题，可以备注“未通过”或打叉，测试通过则打勾。可以根据自己的习惯选择用导图还是表格，导图会更清晰直观。比如我写几条，大家可以参考。
* **常规测试用例分享**
* 在工作中，经常会用到一些常规的测试用例，主要包括新增、编辑、查询、分页、时间等。下面简单介绍：
* 新增：如点击“新增分类”按钮，弹框内容完整，数据保存成功，页面正确显示新增数据。还包括列表排序、输入框长度校验、特殊字符校验、必填项校验、唯一性校验等。
* 子集新增：有的产品不允许在子集下新增，需要根据实际产品设计。
* 编辑：编辑用例与新增类似，需注意编辑后显示最新数据，数据多时支持滚动展示。
* 删除：分为单条删除和目录下有内容时的限制删除。单条删除后数据不再显示，目录下有子集时不允许直接删除。多选删除、已发布与未发布组合删除等也需覆盖。
* 查询：包括精准查询、模糊查询、输入框长度校验、错误信息提示、清空查询内容等。
* 分页：包括页码跳转、每页显示条数切换、分页统计等。
* 时间：起始时间不能大于结束时间，支持时间段查询、清空、年月日切换和手动输入等。
* 这些常规用例希望能帮助大家，文档会发给大家，遇到不会写时可以参考。
* **测试用例评审的意义与流程**
* 在工作中需要对测试用例进行评审，目的是规避风险，防止漏测，让所有人都认可测试用例的完整性。评审通常有产品经理、研发、测试组员参与。产品经理请假并非真的休息，而是因事务无法参加。
* 评审内容是基于产品需求，对编写的测试用例进行讨论。如果不评审，责任全在测试；评审后即使有遗漏，也能分摊责任。评审是再次讨论测试点和问题的机会，大家无异议后即可进入测试。
* **评审中的沟通与补充**
* 评审时，测试用例的编写人需要快速抛出关键点，而不是逐字朗读。几十条用例如果逐条读，时间会很长。要简明扼要地表达测试点和预期结果。如果有遗漏，现场补充即可，不用紧张。比如有同学指出某个功能没写，可以说明原因并补充上去。
* **评审参与人员与会议安排**
* 评审参与人包括产品经理、相关研发、测试组员。不是所有研发都要参加，只需涉及该需求的前后端。测试组长或经理有时间也会参加。评审通常通过会议形式，约定好时间，尽量保证大部分人能到场。
* **评审内容与方式**
* 评审内容是基于产品需求，对测试用例进行讨论和补充。一般漏掉的点由组员或经理提醒，大家氛围轻松，目的是提前发现和解决问题。评审方式可以是文档（表格）或思维导图，导图更直观，重点更突出。
* **迭代测试用例的处理**
* 当需求变更时，测试用例也要迭代。比如原本用手机号+密码登录，后改为手机号+验证码+密码登录。此时需补充新的测试用例，原有能用的继续用，不能用的重新写，并注明版本号。
* **测试用例的基本要素与面试应答**
* 测试用例的难点在于反复编写和熟悉内容，基本概念并不复杂。面试时，需能清楚表达测试用例的编号、名称、前置条件、优先级、重要级别、测试数据、测试步骤、预期结果、实际结果等内容。每家公司大同小异，主要区别在于项目和业务场景。
* 测试用例的编写方法包括边界值、等价类、场景法、错误推测法等，要熟练掌握并能举例说明。测试用例的目的在于避免漏测、评估工作量，更好地梳理需求和熟悉系统。
* **导图与表格的选择**
* 实际工作中，我更倾向于用思维导图编写测试用例，客户交付时则多用表格。导图可以导出为表格，大家可以根据实际情况选择。测试用例评审是内部评审，参与人员有产品经理、研发、测试同事，通过会议讨论和优化测试用例。
* **评审的价值与追溯**
* 评审让大家对测试工作有认可，明确进度和熟悉度。上线后如有问题，也能通过测试用例追溯责任。评审方式包括文档和思维导图。
* **总结与作业**
* 以上就是测试用例的全部内容。测试用例的概念和评审本身不复杂，难点在于方法和与测试计划、方案、策略的结合。课程到此形成闭环，大家要能清楚表达所学内容，熟练掌握测试用例的方法和要素。
* **1-18 bug的基本概念和形成**
* **测试用例的编写与评审**
* hello，各位同学大家好。在上一节课中，学习了如何用思维导图编写测试用例。思维导图和表格编写测试用例的本质区别只在于表现形式和编写方式不同，但提取测试点、梳理步骤、整理预期结果与实际结果并进行对比的思维方式是一致的。
* **测试用例的评审与补充**
* 测试用例评审时，参与人包括产品经理、研发和测试组员。会约定会议时间进行评审，内容基于产品需求或变更需求，对编写的测试用例进行评审。若无问题，则以此用例进行功能测试。评审方式可用文档或思维导图呈现。需求变更或更新时，也要及时补充和更新测试用例。
* **测试用例的内容与分类**
* 测试用例内容包括：编号、名称、前置步骤、优先级、重要级别、测试数据、测试步骤、预期结果和实际结果。优先级和重要级别分为高、中、低，依据功能或需求的重要性判断。编写测试用例的主要目的是避免漏测、评估工作量，并在出现漏测时可回溯用例。测试用例有助于梳理需求、熟悉系统，也是测试工作的成果物，需对内外交付。
* **功能测试用例与接口测试用例**
* 实际工作中，既要编写功能测试用例，也要编写接口测试用例。接口测试用例通常在功能测试前编写，开发完成接口后即可进行接口测试。接口测试用例的设计思路和方法与功能测试一致，主要用到等价类、边界值等方法。后续会分享接口测试用例的模板和编写方法。
* **测试用例的执行与bug的发现**
* 测试用例编写完成后，在系统测试阶段需逐条执行。执行过程中，若预期结果与实际结果不一致，即发现了bug。此时需要记录和管理bug，通常借助专业工具进行。
* **bug管理的基本概念**
* bug管理是指对开发、测试、设计等过程中出现的bug进行记录、审查、跟踪、分配、修改、验证、关闭、整理、分析、汇总及删除等一系列活动状态的管理。最终会生成相应的图表统计，或以邮件等方式通知相关修改者。
* **bug管理流程与意义**
* bug管理流程包括记录、审查、跟踪、分配、修改、验证、关闭、整理、分析、汇总和删除。部分bug可能重复或并非真正的bug，需要删除。bug管理的统计结果可用于会议汇报，也可通过邮件或工具通知修改者。发现bug后要及时管理，这是完整的管理流程。
* **bug在各阶段的产生**
* bug不仅在测试阶段产生，开发和设计阶段也会有bug。例如：
* 需求分析阶段：需求描述不清、难以理解、有错误、文案模糊等都可视为bug。
* 设计阶段：设计文档有误或缺失，也属于缺陷。
* 开发（编码）阶段：代码错误，可通过白盒测试、单元测试发现。
* 运行（测试执行）阶段：编译后软件缺陷。
* **bug的判定标准**
* 衡量是否为bug，需有对照物（如需求规格说明书）。常见标准包括：
* 少功能/多功能
* 功能实现错误（如预期A效果，实际为B或未达A）
* 隐性功能错误

1. 不易使用（用户角度易用性差）
2. 系统或产品中任何影响使用的问题都可视为缺陷，即bug。
3. **bug的六要素**
4. bug的六要素与测试用例内容类似，主要包括：
5. 编号：可自定义或由bug管理系统自动生成。
6. 名称：需严谨准确，如“输入正确账号密码登录失败”。
7. 优先级：根据功能或需求的重要性、业务流程主次判断，越重要优先级越高。
8. 严重等级：分为致命、严重、一般、轻微、建议等。

* 致命：影响核心功能，系统崩溃、闪退、挂掉，或涉及资金安全。
* 严重：主要功能无法使用，如报名功能失效。
* 一般：功能中某些异常场景有问题，反向场景测试失败。
* 轻微：如前端文案错误、易用性差、非核心UI问题等。
* 建议：非bug，仅为优化建议，优先级低但可能被采纳。

1. 复现步骤：需详细描述，便于开发复现。包括环境地址、账号、密码、具体操作步骤（可直接引用测试用例步骤）、预期结果和实际结果。

附件/证据：如截图、录屏、日志、测试数据等，便于佐证bug的存在。

**bug与测试用例的关系**

bug的六要素与测试用例内容高度相似。测试用例有编号、名称、优先级、严重等级、测试数据、步骤、预期结果和实际结果。不同之处在于，bug需形成证据（如附件）。两者在编写时可相互借鉴。

**bug的发现与提交**

发现bug的方式主要是按测试用例执行，预期与实际结果不一致即为bug。但测试用例无法穷尽所有场景，实际工作中还会因未覆盖场景或资源有限而发现bug，这些都应及时提交。

**bug的修复与管理**

需求分析、设计、开发、系统执行各阶段的缺陷都需修复。bug修复流程包括审查、跟踪、分配、修改、验证等。后续课程将学习bug的生命周期及专业工具的管理方法。

**总结与作业**

本节内容知识点不多，但需与前面课程串联理解。bug管理需与测试用例结合：编写完测试用例并执行，发现预期与实际结果不一致即为bug，需记录和管理。理解bug管理的概念、出现阶段、判定标准和六要素，是本节的重点。

作业：整理本节笔记，结合上一节测试用例内容，将两者结合理解。时间充裕的同学可多默写、流利表达，为面试时能清晰回答bug六要素及内容做准备。

**1-19 bug管理工具实操**

**课程回顾与本节目标**

hello，各位同学，大家好！在上一节课中，学习了bug的六要素、如何衡量bug的标准，以及在什么阶段会出现bug、bug的定义和管理的基本概念。那么这一节课要学习什么呢？本节课主要讲解如何提bug。在提bug之前，首先要学会如何发现bug。

**如何发现bug**

找bug可以分为两种情况：

1. 按照测试用例逐条执行，发现bug。这种方式在时间和成本允许的情况下较为系统，但实际工作中往往没有充足的时间编写完整的测试用例。

在没有完整测试用例的情况下，直接在系统中操作，发现bug。无论哪种方式，发现bug后都需要及时记录。

另外，刚进入公司时，可能因为交接不充分导致部分模块缺少用例。在熟悉模块的过程中发现bug，也需要及时提报。

**发现实际bug案例分析**

以中国大学慕课网的国家精品课程页面为例，页面轮播图展示正常。随意点击一个课程，查看其状态和内容，发现课程状态与实际情况一致，没有问题。

但在全部课程分页中，发现总共有66页。切换到“已结束”筛选时，页面显示没有内容，但实际上存在已结束的课程，说明搜索筛选未生效。同理，“正在进行”筛选也有类似问题。

进一步测试发现，只有在第一页时，“已结束”课程能正常显示，翻到最后一页或中间某些页时，“已结束”或“正在进行”都查不到内容。分析原因是分页数量与实际数据不匹配，导致查询不到内容。

**记录bug的流程**

发现bug后，需要截图并备注问题。例如：“存在已结束的课程，但查询不出对应内容。”截图保存后，便于后续记录和沟通。

**为什么要用bug管理工具**

直接将bug发给开发虽然快捷，但在实际工作中会遇到时间差、沟通混乱、信息被覆盖等问题，容易导致bug被遗忘。因此，推荐使用bug管理工具进行记录、跟踪和管理。开发只需登录平台即可查看待处理的bug，实现全流程跟踪。

**常见bug管理工具介绍**

常见的bug管理工具有：禅道、Jira、TAPD、TestLink、QC等。不同工具本质上功能类似，只是名称和界面不同。就像购物平台一样，核心功能一致。

本节课以TAPD为例进行讲解。TAPD由腾讯开发，支持免费和付费版本，应用广泛。学会TAPD后，其他工具也能快速上手。

**TAPD平台基本操作演示**

进入公司后，领导会根据姓名和手机号分配TAPD账号。登录后，主页会有功能介绍和仪表盘。

* “我的代办”显示分配给的任务或bug。
* “我的工作”可添加工作项，如登录需求研发，设置处理人、预计开始和结束时间后提交。
* “我的文档”可上传、分享、收藏项目相关文档，所有项目成员可见。
* “协作空间”用于项目协作和办公。

可创建项目看板，如“教育培训系统”、“银行系统”等，便于缺陷管理和文档整理。

**提交bug的标准流程**

以“教育培训系统”为例，创建缺陷时需填写以下内容：

1. **标题**：如“慕课网-全部课程查询已结束课程时，存在内容但查询不出来”。
2. **环境信息**：包括环境地址、账号、密码。
3. **复现步骤**：

* 登录网站
* 进入菜单选择“国家精品课程”
* 下拉到全部课程页面
* 左右切换“即将开始”和“已结束”课程
* 截图并备注每一步

1. **实际结果**：存在内容但查询不出来。
2. **预期结果**：只要存在内容，无论在第几页，都应能查询出对应内容。
3. **优先级与严重程度**：如优先级高，严重程度一般。
4. **处理人**：如分配给开发“唐小鹏”。

* **证据**：上传相关截图或日志。
* **bug生命周期与状态流转**
* 提出bug后，开发可接受或拒绝处理。
* 开发接受后，bug状态变为“处理中”，处理完成后标记为“已解决”。
* 测试验证通过则关闭bug，验证不通过可重新打开。
* 若开发认为不是bug，可拒绝并备注原因，如“设计如此”，必要时转交产品经理。

bug在开发和测试之间可多次流转，直到最终关闭。

**bug管理工具的优势**

通过bug管理工具，bug的提出、处理、验证、关闭等流程一目了然，所有变更和评论都有记录，便于追溯和协作。每个bug有唯一编号、名称、优先级、严重程度、处理人、复现步骤、实际与预期结果、证据等信息，管理更规范。

**实践与作业**

建议大家注册TAPD账号，自己模拟提bug和处理流程。作业要求：

* 在慕课网或其他项目中找到10个bug，提交到TAPD平台。

熟悉bug的记录、跟踪和管理流程。

本节课为实战内容，重点在于实际操作和流程体验，文字笔记相对较少。

1. 最后，附上bug记录样例，便于大家参考。
2. **1-20 bug生命周期和回归测试**
3. **bug生命周期概述**
4. hello，各位同学大家好，今天来学习bug的生命周期。在上一节课中，通过具体的例子学习了如何发现和提交bug，这部分内容已经讲过了。今天将更详细地讲解bug的生命周期相关理论知识。
5. **bug的发现**
6. bug的生命周期首先从发现bug开始。如何发现bug呢？比如在执行测试用例时，如果预期结果和实际结果不一致，就可以发现bug。此外，测试用例无法穷尽所有情况，探索性测试时也可能发现bug。有时因为成本或时间问题，无法编写或执行所有测试用例，也可能在这些情况下发现bug。
7. **bug的提交**
8. 发现bug后，需要提交bug。提交bug时，要尽可能详细地描述缺陷的属性，包括复现环境、账号密码、bug类型、等级、优先级、详细复现步骤、实际结果与预期结果等。提交时还要确保该缺陷没有被重复提交，尤其在交叉测试时要注意。
9. **bug的指派**
10. 提交bug后，需要指派bug。指派有两种方式：一种是指派给开发负责人（当不清楚具体由谁修复时），另一种是直接指派给具体开发人员。实际工作中，通常采用第一种方式，开发负责人或组长确认后再分配给具体开发人员，以节省流转时间。如果测试同事穿插在不同项目和团队中，不清楚具体模块，也可以指派给对应开发人员。如果固定在一个项目中，则可以直接指派给具体开发人员。
11. **bug的确认**
12. 开发人员收到bug后，会分析和重现。如果认为不是缺陷（如需求理解有误或无法重现），会将问题退回并注明原因，即“拒绝”；如果确认是缺陷，则“接受”。确认缺陷后，进入修复阶段。
13. **bug的修复**
14. 确认是缺陷后，开发会判断是否需要立即修复，还是可以推迟处理。如果推迟处理，会给出预计处理时间，优先级较低的bug会延后。推迟处理一般需项目经理和测试经理协商决定。如果及时修复，开发会在每日例会中更新进度，修复完成后通知测试人员验证。
15. **bug的验证与关闭**
16. 测试人员验证bug修复情况。如果修复通过，则关闭bug；如果未通过，则重新打开。验证结果无非是“通过”或“不通过”。最终，缺陷处理完成后，bug被关闭。
17. **bug生命周期总结**
18. bug的生命周期可以总结为：从发现、提交、指派、确认、修复、验证到关闭。整个过程就是bug从被发现到被关闭的完整流程。
19. **bug生命周期流程梳理**
20. 新建bug
21. 指派bug
22. 开发确认bug
23. 解决bug
24. 验证bug
25. 关闭bug
26. 如果验证不通过，则重新打开，流程重复。开发确认bug时，可能会拒绝（认为不是bug）、延期处理（优先级低）、或因设计如此等原因否决。拒绝后需判断是关闭还是重新打开。
27. **bug被拒绝或无法复现的处理**
28. 开发拒绝bug的原因有：设计如此、不是bug、无法重现、问题描述不清、需求变更、挂起等。对于无法复现的bug，可以定期测试、观察一段时间，或让开发定位分析，或在不同环境下重现。有些功能与发布管理等有关。设计如此需与产品经理确认，不能仅凭开发说法。
29. **bug的状态**
30. bug的状态包括：新建、待解决、已解决、验证中、关闭、重新打开、拒绝、延期、挂起、外部因素等。每种状态对应不同的处理流程。面试时要熟悉这些状态及其含义。
31. **bug的类型**
32. 常见bug类型有：功能问题、设计缺陷、界面优化、UI问题、性能问题、兼容性问题、客户相关、用户体验、标准规范、安全相关、环境部署、安装部署等。部分bug管理工具在提交bug时可选择类型，注意区分功能和性能等类型。
33. **开发不认可bug的处理思路**
34. 如果开发不认可提的bug，首先要明确开发不认可的理由。听取开发意见后，告知自己判断bug的依据。对开发理由进行校验，参考需求文档或与产品经理确认。如果确认不是bug，则关闭；如果是bug，则重新打开交给开发处理，确保产品质量。这样既有理有据，也有助于与开发良好沟通。
35. 如果开发和测试、产品经理等多方意见不一致，可由产品经理、项目经理、研发经理、测试经理共同定夺。面试时如被问到“这么小的bug也要改吗”，要明确回答“bug无大小，只要影响系统使用就应处理”，具体是否紧急、延期或不处理由领导层决定。
36. **需求变更、环境问题等特殊情况**
37. 如果开发认为是需求变更，需找产品经理确认；如果是测试环境问题，让开发说明并验证；如确实如此则关闭，否则继续激活bug。若开发认为不存在该用户场景，可召集产品和研发共同确认是否需要修改。
38. **版本迭代与回归测试**
39. 版本迭代在回归测试中也很重要。每次bug修复后，需在项目中创建新版本，并将bug标记为已解决。版本管理用于统计bug出现在哪个版本。测试环境搭建时，需选择发现bug的版本，修复后选择新版本。开发会提供新安装包，测试环境更新后进行回归测试。
40. **迭代的概念**
41. 迭代是重复反馈的过程，每次迭代结果作为下一次的初始值。迭代不是一次性完成，而是不断改进、调整和优化的过程。版本管理记录每次变化，便于追溯。现在版本管理多由开发或运维负责，测试只需关注迭代的本质。
42. **功能迭代举例**
43. 比如最初报名课程只需点击报名，后续增加了弹出协议并同意后才能报名，这就是功能的迭代。
44. **回归测试的类型**
45. 回归测试分为全量回归和增量回归。全量回归是所有功能都测一遍，适用于时间充足或变更频繁的场景。增量回归只测改动相关的功能，未改动部分可通过自动化测试覆盖。实际敏捷开发中，通常采用全量回归，只有特别稳定的功能（如银行转账、支付）才考虑自动化。
46. **课程小结与作业**
47. 本节内容包括：bug生命周期、流程梳理、无法复现bug的处理、bug状态和类型、开发不认可bug的应对、版本迭代和回归测试。
48. **1-21 测试报告**
49. **测试总结阶段介绍**
50. hello，各位同学大家好，今天学习测试总结阶段。测试总结阶段的核心内容就是学习如何编写测试报告。在正式学习测试报告编写之前，先简单回顾上一节课的内容，包括bug的生命周期、bug的状态、bug的类型，以及遇到开发不认可bug时的处理方法，还有版本迭代和回归测试。这里补充一下回归测试的相关内容。
51. **回归测试与迭代概念**
52. 在回归测试阶段，首先要理解“迭代”这个名词。一个版本就是一次迭代，有的公司会问“做了几轮测试”，其实就是问迭代次数，比如一周、两周或一个月迭代一次。一般来说，一周迭代一次的公司加班会比较多，两周或一个月迭代一次则压力相对分散。在面试时，若被问到测试了几轮，需要根据项目周期和迭代频率简单计算，比如一年每月迭代一次，除去春节假期，大约有11到12次迭代。
53. **功能测试与自动化测试的关系**
54. 课程中还没有详细讲自动化测试，先补充一下功能测试。很多同学觉得功能测试简单，其实不然。功能测试包括数据准确性、表单值与逻辑分析、业务梳理、交互易用性、逆向思维、UI兼容性、cookie等内容。比如一个表单，涉及文本框、多行文本框、密码框、隐藏框、复选框、单选框、下拉选框等，用于采集用户输入或选择的数据。
55. 自动化测试虽然效率高，但本质上依赖人工编写或录制脚本，只能覆盖主要业务流程，无法完全替代详细的功能测试。自动化测试是提升效率的手段，而不是功能测试的替代品。所以即使专注于功能测试，也完全可以有很好的发展。
56. **前后端bug的区分**
57. 面试中经常会被问到如何区分前端bug和后端bug。一般来说，界面或兼容性相关的问题属于前端bug；功能错误则需要进一步区分，比如接口报错通常是后端bug。实际工作中可以通过抓包查看请求和响应，结合接口测试进一步判断bug归属。
58. **测试总结阶段的意义**
59. 在系统测试中，会参与很多测试，涉及哪些模块、发现了多少bug，尤其是重要级别的bug，这些都可以作为工作绩效的展示。因此，测试报告非常重要。如果项目中没有专门的领导负责测试报告，建议大家主动争取，因为会写测试报告是职场中的一项优势，有助于让领导记住。
60. **测试报告的主要内容**
61. 测试报告主要是对工作和bug的总结。编写测试报告时，通常包括以下几个部分：
62. **概述**：说明项目的基本情况和测试目的。
63. **测试环境描述**：在哪个环境上进行测试，包括平台访问地址、服务器操作系统、数据库信息、测试数据统计等。
64. **测试总体安排**：每一轮测试的时间、人员安排等。
65. **测试范围**：涉及哪些产品模块，比如直播、报名、商课、个人中心等。
66. **转测版本分析说明**：每一轮测试的情况，比如第一轮发现了多少bug，第二轮产品是否有好转等。
67. **回归测试情况**：包括遗留bug、修复bug、关闭bug的数量统计。需要注意，重新打开的bug不计入关闭数量。
68. **缺陷等级分布分析**：统计严重、高、中、低、轻微、建议等不同等级的bug数量。有些公司用P0-P3表示优先级，P0为紧急，P1为高，P2为中，P3为低。
69. **缺陷模块分析**：分析bug分布在哪些模块，比如登录、订单管理、课程报名等，判断是业务复杂还是开发能力问题。
70. **系统缺陷类型分布分析**：分析bug类型，如代码错误、界面优化、设计缺陷、环境问题、操作问题、性能问题、兼容性问题等。
71. **遗留问题分析**：统计高中低等级的遗留bug数量，列出遗留问题清单，包括编号、创建者、处理人、状态（如待办、延期等）。

**测试产品评估**：对产品是否可以上线进行总体评估，并指出存在的风险。评估内容包括测试过程的充分性、测试用例覆盖情况、版本质量评估和风险评估等。

**测试报告的撰写建议**

测试报告内容较多，实际工作中可以根据需要适当删减。比如用图表展示项目计划、需求达成率、功能测试进度、本周发现和关闭的缺陷、测试进度和人员支持等，能更直观地反映测试情况。

如果时间有限，可以采用简化版的测试报告模板。公司通常会提供模板，按模板填写即可。但在面试时，往往没有模板，需要清楚测试报告应包含哪些内容。

**测试报告内容总结**

测试报告应包含：概述、测试环境、测试总体安排、测试范围、转测版本分析、回归bug情况、缺陷分布（模块和等级）、系统缺陷类型分布、遗留问题、版本上线分析及风险等。

**课后作业**

请大家以慕课网或自己手头的项目为例，编写一份测试报告。可以参考项目测试周报的形式，也可以按照上述详细测试报告的结构来写。

**2-1 Mysql概念+DDL语言**

**1. 数据库的基本概念**

hello，各位同学大家好，今天学习MySQL数据库。在学习MySQL数据库之前，需要先了解一下数据库的基本概念。MySQL数据库只是众多数据库类型中的一种。那么，数据库到底是什么呢？如果用比较官方的说法，数据库是指长期存储在计算机内、有组织的数据集合。可以简单理解为，数据库就是一个数据仓库，是用来存储数据的地方。

在数据库中，会存储很多表，每一条数据都存储在数据库的表里。表是数据库中存储数据的基本单位。数据会根据不同的分类存储到不同的表中，这样做的目的是为了高效、快速地查询数据。因此，数据库可以看作是一个存储数据的仓库，里面有不同的数据表，方便快速高效地查询表中的数据。

**2. 数据库与表的关系**

还可以把数据库理解为文件系统上的一组文件，等同于schema（模式）。表就像数据库中的一个电子表格，数据库中的内容是以电子表格的形式存储的。在数据库中，还会有字段、索引、主键、记录等内容，这些会在后续进行讲解。

**3. 为什么要学习数据库**

有同学可能会问，自己不是开发，为什么要学习数据库？只需要把功能测试做好就可以了。其实，测试工程师的工作不仅仅是找出软件的不足并告知开发，按照步骤复现问题。优秀的测试工程师（如中高级测试工程师）在发现问题后，还能分析问题原因并给出解决方案。这就要求测试人员除了会测试，还需要了解开发相关的知识，其中很重要的就是数据库。

绝大多数软件都与数据密切相关，比如微博、小红书、京东等，用户的各种操作都会涉及数据的存储和读取。这些数据记录都会存储在数据库中，数据库会把数据反馈到软件上，同时保存用户输入的数据。因此，测试工程师在测试系统时，不仅要在界面上操作，还需要检查数据库中的数据是否正确。这样可以在软件出现问题时，快速定位问题原因。

**4. 测试工作中的数据操作**

在工作中，测试人员经常需要造数据，这也需要用到数据库。如果不会用数据库，就无法造数据。例如，需要模拟用户的购买习惯（如晚上某个时间下单），可以通过修改数据库中的时间字段来实现，而不必等到实际时间。需要注意的是，造数据和修改数据一般都在测试环境中进行，不能在生产环境中修改用户的账号数据和信息。

**5. 数据库存储的数据结构**

数据库中存储的是一条一条的数据。例如，在电商平台（如京东、淘宝、天猫）中，每个商品在数据库中都有对应的数据。这些数据会以类似“name（商品名）、价格、时间、地点、是否购买”等字段的形式存放在表中。数据库会存储所有用户相关的信息。

**6. 不同行业的数据库应用**

今后大家进入不同的行业，会接触到不同的产品，数据库中存放的信息也会有所不同。

**7. 数据库的类型**

数据库类型主要分为两类：关系型数据库和非关系型数据库。主流学习的是关系型数据库。关系型数据库是建立在关系模型基础上的，利用集合、代数等数学概念和方法来处理数据。常见的关系型数据库有MySQL、Oracle、SQL Server等。非关系型数据库将在后续知识点中介绍。

**8. 本节学习重点**

本节课的学习重点包括：

* 数据库的基本概念
* 数据库的可视化界面操作工具
* SQL语言的分类和使用，尤其是查询类语句
* **9. 关系型数据库的结构**
* 关系型数据库的数据以“行”为单位，每一行是一条记录。数据表由多行数据组成，数据库则是多个数据表的集合。MySQL、Oracle、SQL Server等都是关系型数据库，不同厂商有不同的数据库产品，但本质结构类似。
* **10. MySQL的安装与配置**
* 以MySQL为例，介绍其安装过程。安装步骤很简单，按照教程一步步操作即可。以Windows 11为例，首先在MySQL官网下载对应版本（一般为8及以上），下载后进行安装。安装过程中，建议目录保持默认，不要修改。设置密码时，建议设置简单易记的密码。安装完成后，需要配置MySQL的环境变量。方法是：右击“此电脑”→“属性”→“高级系统设置”→“环境变量”，将MySQL的安装路径添加到Path中。
* **11. 验证MySQL安装**
* 安装完成后，可以通过快捷键Win+R，输入cmd打开命令行，输入mysql -u root -p，回车后输入密码。如果成功进入MySQL命令行界面，说明安装成功。
* **12. SQL语言简介**
* SQL（结构化查询语言）是数据库中进行操作的语言。所有关系型数据库都支持SQL语句。通过SQL可以操作Oracle、SQL Server、MySQL等数据库。SQL语言主要分为以下几类：
* DQL（数据查询语言）：用于数据查询
* DML（数据操作语言）：用于数据的增删改查
* DDL（数据定义语言）：用于数据库和表的定义与管理，如创建、删除数据库和表

TPL（事务处理语言）：用于事务处理

**5. SQL语言分类及事务处理语言简介**

其中，DQL主要用于查询，DML用于对数据进行增删改查，DDL用于数据库和表的管理，TPL用于事务处理。

可以简单了解一下事务处理语言（TPL）是什么意思。事务处理语言，顾名思义，就是对数据库事务进行处理，比如回滚等操作。它主要用于保证数据操作的完整性和一致性。

**5.1 初学者对SQL分类的理解**

对于刚接触这些概念的同学来说，可能会觉得陌生，其实记住这些分类并不难，可以通过记忆的方式掌握。下面重点来看一下DDL相关的内容。

**5.2 DDL语句的基本操作**

DDL主要用于数据的定义和管理，比如建表等。在MySQL安装好后，首先会用到的就是查看数据库的命令。

例如，使用show databases;可以查看当前有哪些数据库。

* 其中，像information\_schema、mysql等是系统库，是MySQL安装后自动生成的，test 1则是自己创建的数据库。
* **5.3 常用数据库操作命令**
* show databases;：查看所有数据库。
* 注意：命令结尾要加英文分号;，否则命令无法正确结束。有些同学可能会误用中文分号，导致命令无法执行。
* 如果命令输入有误，MySQL会提示错误信息，指出出错的位置。可以通过键盘的“上箭头”键快速调出上一条命令进行修改，无需重新输入。
* **5.4 进入指定数据库与查看表**
* use 数据库名;：进入指定数据库。例如use test1;，系统会提示Database changed，表示已切换到该数据库。
* show tables;：查看当前数据库下的所有表。例如在test1下，可能会看到“学生表”和“students1”两张表。
* **5.5 创建数据库**
* create database 数据库名;：创建新数据库。例如create database test2;。
* 如果数据库名已存在，系统会提示错误，因为数据库名是唯一的。
* 创建成功后，可以用show databases;再次查看，确认新数据库已添加。

也可以通过可视化工具查看数据库的创建情况。

**5.6 创建表**

create table 表名 (字段名 类型 约束, ...);：在数据库中创建表。例如：

|  |
| --- |
| Plain Text create table test\_dp (  id int(5),  name varchar(10),  age int(16) ); |

* 如果没有指定数据库，表会默认创建在当前选中的数据库下（如test1）。
* 也可以指定数据库创建表，例如create table test3.test\_dp (...);，这样表会被创建在test3数据库下。
* **5.7 查看指定数据库下的表**
* 进入目标数据库后，使用show tables;即可查看该数据库下的所有表。
* 如果需要在不同数据库下操作，记得先用use 数据库名;切换。
* **5.8 可视化工具辅助操作**
* 除了命令行，也可以使用可视化工具进行数据库和表的创建、管理等操作。
* 可视化工具通常会自动生成对应的DDL语句，方便复制和复用。

如果担心命令行操作出错，使用可视化工具是一个更简单直观的选择。

**5.9 小结**

本节课主要介绍了DDL相关的基本操作，包括数据库和数据表的创建、切换和查看等内容。建议大家课后用命令行多加练习，熟悉这些基本命令。下节课将学习如何使用可视化工具操作数据库，操作会更加简单直观。本节内容到此结束。

**2-2 Navicat工具使用&DML语言(数据增删改查)**

**Navicat连接数据库**

大家下载好Navicat后，首次进入页面，需要点击连接来连接数据库。以连接MySQL为例，连接名可以随便起，但在工作中，建议根据环境来命名，如测试环境的MySQL、用户环境的MySQL、生产环境的MySQL等。

主机部分，localhost代表本地。用户信息一般不要修改，3306是MySQL默认的端口号，工作里非必要不会改动。用户名一般是root或管理员账号，也不建议修改。密码则每个人不同，需要输入自己的密码。

点击连接MySQL后，测试连接成功，点击确定即可。连接成功后，双击连接就表示已经连接上了数据库。

**使用Navicat建表**

连接上数据库后，可以通过Navicat在数据库里新建表。找到数据库，点击“表”，再点击“新建表”，就会进入操作页面。

在操作页面，可以定义表的字段。例如，先定义一个ID字段，类型为int，长度可以设为5（长度为零表示不做限制），还可以设置是否为主键、添加注释等。接着可以定义name字段，类型一般为varchar，默认长度255，如果觉得太长可以修改。主键的概念会在下一节课讲解。

再比如，建一个部门表，包含姓名、编号、岗位等字段。岗位字段一般也是字符串类型。还可以加入入职时间字段，类型可以选择time或date等。

保存表时，需要输入表名，比如“info\_test”（公司信息表，测试部门）。输入表名后点击保存，表就建好了。

**Navicat表操作**

使用Navicat建表和用命令行建表效果是一样的，但Navicat更简洁高效。

在Navicat中，选择表后，可以打开表、设计表、新增表、删除表、清空表（清空表数据）、复制表、导出数据（如导出为CSV或文本格式）以及导入数据（最好导出为SQL脚本再导入）。

例如，导出为SQL脚本到桌面，然后可以在另一个库中导入。导入时要注意数据行数，如果数据行数为零，导入可能会报错。

还可以对表进行重命名等操作。在表的位置，可以看到对应的SQL语句，这就是接下来要学的DML语言，用于对数据进行增删改查。

**DML语言之添加数据**

在Navicat中，可以点击表的位置新建查询来运行SQL语句。例如，添加数据的语句是“INSERT INTO 表名(字段名1, 字段名2, ...) VALUES(值1, 值2, ...)”。

比如，向“info\_test”表添加一条数据，字段为ID、name、岗位，值分别为2、“张三”、“测试”。运行语句后，刷新表就可以看到数据插入成功。

**DML语言之修改数据**

修改数据的语句是“UPDATE 表名 SET 字段名1 = 值1, 字段名2 = 值2, ... WHERE 条件”。

例如，将“info\_test”表中ID为1的记录的name改为34，语句为“UPDATE info\_test SET name = 34 WHERE ID = 1”。如果不加WHERE条件，会把表中所有记录的name都改为34。所以修改数据时一定要加上条件。

**DML语言之删除数据**

删除数据有两种形式，一种是删除指定数据，语句是“DELETE FROM 表名 WHERE 条件”。

例如，删除“info\_test”表中ID为4的记录，语句为“DELETE FROM info\_test WHERE ID = 4”。需要注意的是，字段名和值要对应，如果字段名和值不匹配，可能会报错。

另一种是删除整张表的数据，语句是“DELETE FROM 表名”。例如，删除“info\_test”表的所有数据，运行语句后，表中的数据就会被全部删除。

**DML语言之查询数据**

查询数据的语句是“SELECT \* FROM 表名”，表示查询表中的所有数据。也可以查询具体字段，例如“SELECT name FROM info\_test”，表示只查询name字段的数据。还可以同时查询多个字段，如“SELECT name, 岗位 FROM info\_test”。

**数据库表字段总结**

数据库表的字段包括名称、类型、长度、是否为空、主键、注释、自动递增、索引和外键等。

名称通常以英文命名，可以是英文加数字或纯英文。类型有数字类型（如整数、浮点型、小数类型）、字符串类型、时间类型等，根据数据场景选择。长度是指字段允许输入的最大字符数，用户输入内容需遵守长度规则。

是否为空勾选后，表示该字段不能为空，必须有内容。主键表示唯一性校验，字段值不可重复。注释是对字段的描述，方便理解字段含义。

自动递增勾选后，不输入该字段的值，系统会自动填充数值。索引和外键会在后续课程讲解。

**课程总结与作业**

今天这一节课主要介绍了Navicat的使用，包括连接数据库、建表、表操作以及DML语言（数据操作语言）的增删改查语句。这些内容非常重要，因为在实际工作中会经常用到。

作业是操作今天学的DML语句，在Navicat上进行实际操作，将相关问题整理出来，进行总结和积累。

**2-3 SQL高级查询(函数、分组、查询、运算符)**

**课程引入与回顾**

Hello，各位同学大家好呀！这一节课来学习SQL的高级语言。在上一节课里面，学习了DML的数据操作以及NoSQL的基本使用，还有数据库表的字段。在DML里面，学习了添加、修改、删除数据。今天，要来补充一下关于删除的内容。

**删除操作详解**

上一节课学的删除操作是DELETE FROM 表名，它其实指的是清除这张表的信息，但并没有删除表本身。删除表的操作是在DDL里面，使用DROP TABLE 表名。DROP TABLE会删除整张表的表结构，而DELETE只是删除表的数据。

演示一下：

|  |
| --- |
| SQL -- 删除ID=1的数据 DELETE FROM 表名 WHERE ID=1;  -- 删除整张表 DROP TABLE 表名; |

* 另外，还有一个TRUNCATE TABLE语句，它也是删除表的所有数据，但事务不能回滚，可以把它理解成截断。
* **重点区别：DELETE与DROP TABLE**
* DELETE FROM 表名：清除表的数据，属于DML语句。

DROP TABLE 表名：删除表的结构，属于DDL语句。

**数据查询高级操作**

今天主要学习数据查询的高级操作，包括函数、条件查询、运算符、排序、模糊查询等。

**简单查询回顾**

在上一节课的最后，学习了使用SELECT语句进行简单查询，比如查询所有字段或指定字段。

|  |
| --- |
| SQL -- 查询所有字段 SELECT \* FROM 表名;  -- 查询指定字段 SELECT name, id FROM 表名; |

**取别名**

可以使用AS关键字给字段取别名，AS可以省略。

|  |
| --- |
| SQL -- 取别名 SELECT name AS 姓名, id AS 编号 FROM 表名;  -- AS省略写法 SELECT name 姓名, id 编号 FROM 表名; |

**条件查询**

使用WHERE子句进行条件查询。

|  |
| --- |
| SQL -- 条件查询 SELECT \* FROM 表名 WHERE ID=3; |

**运算符**

**比较运算符**

比较运算符包括大于、小于、大于等于、小于等于、等于、不等于等。

|  |
| --- |
| SQL -- 大于 SELECT \* FROM 产品表 WHERE 价格 > 10;  -- 小于 SELECT \* FROM 产品表 WHERE 价格 < 100;  -- 大于等于 SELECT \* FROM 产品表 WHERE 价格 >= 10;  -- 小于等于 SELECT \* FROM 产品表 WHERE 价格 <= 100;  -- 等于 SELECT \* FROM 产品表 WHERE 价格 = 100;  -- 不等于 SELECT \* FROM 产品表 WHERE 价格 != 100; -- 或 SELECT \* FROM 产品表 WHERE 价格 <> 100; |

**空值处理**

使用IS NULL和IS NOT NULL处理空值。

|  |
| --- |
| SQL -- 查询为空的值 SELECT \* FROM 产品表 WHERE 数量 IS NULL;  -- 查询不为空的值 SELECT \* FROM 产品表 WHERE 数量 IS NOT NULL; |

**范围查找**

使用BETWEEN...AND...和NOT BETWEEN...AND...进行范围查找。

|  |
| --- |
| SQL -- 区间范围查找 SELECT \* FROM 产品表 WHERE PID BETWEEN 1 AND 3;  -- 不在区间范围查找 SELECT \* FROM 产品表 WHERE PID NOT BETWEEN 1 AND 3; |

**集合范围**

使用IN和NOT IN进行集合范围查找。

|  |
| --- |
| SQL -- 集合范围查找 SELECT \* FROM 产品表 WHERE PID IN (2, 3, 4);  -- 不在集合范围查找 SELECT \* FROM 产品表 WHERE PID NOT IN (2, 3, 4); |

**逻辑运算符**

逻辑运算符包括AND和OR。

|  |
| --- |
| SQL -- AND：两个条件都要满足 SELECT \* FROM 产品表 WHERE PID=1 AND 数量 >= 10;  -- OR：满足其一即可 SELECT \* FROM 产品表 WHERE PID=1 OR 数量 >= 10; |

**模糊查询**

使用LIKE进行模糊查询，%表示任意字符。

|  |
| --- |
| SQL -- 查询以某字符结尾的内容 SELECT \* FROM 产品表 WHERE name LIKE '%s';  -- 查询以某字符开头的内容 SELECT \* FROM 产品表 WHERE name LIKE 'I%';  -- 查询包含某字符的内容 SELECT \* FROM 产品表 WHERE name LIKE '%手机%'; |

**排序**

使用ORDER BY进行排序，ASC表示正序，DESC表示倒序。

|  |
| --- |
| SQL -- 正序 SELECT \* FROM 产品表 ORDER BY PID ASC; -- 或 SELECT \* FROM 产品表 ORDER BY PID;  -- 倒序 SELECT \* FROM 产品表 ORDER BY PID DESC; |

**函数**

SQL函数包括MAX（最大值）、MIN（最小值）、COUNT（计数）、AVG（平均值）、SUM（求和）等。

|  |
| --- |
| SQL -- 最大值 SELECT MAX(价格) FROM 产品表;  -- 最小值 SELECT MIN(价格) FROM 产品表;  -- 计数 SELECT COUNT(\*) FROM 产品表; -- 或 SELECT COUNT(数量) FROM 产品表;  -- 平均值 SELECT AVG(价格) FROM 产品表;  -- 求和 SELECT SUM(价格) FROM 产品表;  -- 查询数据库版本号 SELECT VERSION(); |

**分组**

使用GROUP BY进行分组，常与函数一起使用。

|  |
| --- |
| SQL -- 查询每个班级的男生女生个数 SELECT class ID, 性别, COUNT(\*) AS 人数 FROM 学生表 GROUP BY class ID, 性别;  -- 分组后条件查询 SELECT class ID, AVG(分数) AS 平均分 FROM 学生表 WHERE 分数 > 60 GROUP BY class ID HAVING 平均分 >= 95; |

**课程总结与作业**

今天学习了SQL的高级查询，包括删除操作的区别、简单查询回顾、取别名、条件查询、运算符、排序、模糊查询、函数和分组。作业是练习这些高级查询方法，并简述DROP和DELETE的区别。

上一节课的作业是简述MySQL的数据类型，并创建一张包含ID、教师名称、学科、班级、电话等字段的表，进行增删改查操作。如果对SQL非常感兴趣的同学，也可以用命令行去操作一遍。

**2-4 Mysql多表查询**

**课程介绍**

各位同学大家好！这节课将学习MySQL中非常重要的知识点——多表查询。在学习多表查询之前，需要先理解外键的概念。

**外键的概念**

在项目中，通常会有多张表存储不同的数据。例如：班级表、学生表和分数表。这些表虽然拆分开了，但它们之间存在关联关系。为了定义这种关系，需要使用约束，这就是外键。

**主键与外键的区别**

* **主键**：能够唯一确定一条记录的字段，具有唯一性和非空性。例如：身份证号码、工号等。

**外键**：用于与另一张表关联，能够确定另一张表记录的字段，用于保持数据一致性。外键通常是另一张表的主键，可以重复或为空值。

**示例说明**

以两张表为例：

1. **部门表（dept）**：包含部门编号（主键）、部门名称和地址。
2. **员工表（emp）**：包含员工ID（主键）、姓名、部门编号（外键）等。

部门表的部门编号是主键，员工表的部门编号是外键，通过部门编号关联两张表。

**多表查询（SQL JOIN）**

多表查询用于关联多张表的数据。常见的关联方式包括：

1. **内连接（INNER JOIN）**：取两表的交集数据。
2. **左连接（LEFT JOIN）**：以左表为主，返回左表所有数据，右表无匹配则为NULL。

**右连接（RIGHT JOIN）**：以右表为主，返回右表所有数据，左表无匹配则为NULL。

**内连接示例**

|  |
| --- |
| SQL SELECT \* FROM dept  INNER JOIN emp ON dept.dnumber = emp.dept\_number; |

只返回两表中部门编号匹配的数据。

**左连接示例**

|  |
| --- |
| SQL SELECT \* FROM dept  LEFT JOIN emp ON dept.dnumber = emp.dept\_number; |

返回左表（dept）所有数据，右表无匹配则为NULL。

**右连接示例**

|  |
| --- |
| SQL SELECT \* FROM dept  RIGHT JOIN emp ON dept.dnumber = emp.dept\_number; |

返回右表（emp）所有数据，左表无匹配则为NULL。

**总结**

* 主键是唯一标识，外键用于关联其他表。
* 多表查询通过JOIN实现，包括内连接、左连接和右连接。

内连接取交集，左连接以左表为主，右连接以右表为主。

**2-5 多表高级查询、索引、外键**

**多表查询回顾**

各位同学，继续学习多表查询。上节课学习了内关联（INNER JOIN）、左关联（LEFT JOIN）和右关联（RIGHT JOIN）。

* **内关联**：取两表交集数据。
* **左关联**：以左表为主，返回左表所有数据，右表无匹配则为NULL。

**右关联**：以右表为主，返回右表所有数据，左表无匹配则为NULL。

**查询A表存在但B表不存在的数据**

语法示例：

|  |
| --- |
| SQL SELECT \* FROM a表  LEFT JOIN b表 ON a表.主键 = b表.外键  WHERE b表.外键 IS NULL; |

**示例**：

|  |
| --- |
| SQL SELECT \* FROM dept  LEFT JOIN emp ON dept.dnumber = emp.dept\_number  WHERE emp.dept\_number IS NULL; |

此查询会返回部门表中存在但员工表中不存在的部门数据（如运营部）。

**查询B表存在但A表不存在的数据**

语法示例：

|  |
| --- |
| SQL SELECT \* FROM a表  RIGHT JOIN b表 ON a表.主键 = b表.外键  WHERE a表.主键 IS NULL; |

**示例**：

|  |
| --- |
| SQL SELECT \* FROM dept  RIGHT JOIN emp ON dept.dnumber = emp.dept\_number  WHERE dept.dnumber IS NULL; |

此查询会返回员工表中存在但部门表中不匹配的数据（如无部门的小白龙）。

**表别名（AS）**

别名可简化SQL语句，AS可省略：

|  |
| --- |
| SQL SELECT d.dname AS 部门名称, e.ename AS 员工姓名  FROM dept d  LEFT JOIN emp e ON d.dnumber = e.dept\_number; |

**事务（Transaction）**

事务用于处理操作量大、复杂度高的数据操作（如银行转账），确保数据一致性。

**事务控制语句**

1. **BEGIN**：开启事务。
2. **COMMIT**：提交事务（数据永久修改）。

**ROLLBACK**：回滚事务（撤销未提交的操作）。

**示例**：

|  |
| --- |
| SQL BEGIN;  DELETE FROM emp WHERE id = 1001;  ROLLBACK; -- 撤销删除操作 |

|  |
| --- |
| SQL BEGIN;  DELETE FROM emp WHERE id = 1001;  COMMIT; -- 永久删除数据 |

**索引（Index）**

索引是一种数据结构，用于加速查询，但会增加存储开销并影响增删改性能。

**索引操作**

1. **查看索引**：

|  |
| --- |
| SQL SHOW INDEX FROM 表名; |

1. **创建索引**：

|  |
| --- |
| SQL CREATE INDEX 索引名 ON 表名(字段名); |

**示例**：

|  |
| --- |
| SQL CREATE INDEX idx\_ename ON emp(ename); |

**索引原则**

* WHERE条件频繁使用的字段应加索引。
* 排序或分组字段可加索引。

避免过多索引，影响性能。

**课程总结**

重点内容：

1. 多表查询（JOIN）的三种方式及差异查询。
2. 事务的开启、提交与回滚。

索引的作用与创建方法。

**作业**

1. 练习多表查询语句。
2. 整理主键与外键的区别。
3. 练习事务控制语句（BEGIN/COMMIT/ROLLBACK）。

整理索引概念并练习创建索引。

**面试题五则（6/18）**

**一、做性能测试的原因是什么？**

1.互联网行业发展快，用户量大增，有些用户反馈的问题是多用户并发操作导致的；

2.业务和系统的架构越来越复杂，简单架构没有的性能问题，往往在复杂架构中更容易出现；

3.用户量剧增的情况下，导致数据库存的数据量也跟着陡增，这种情况下也会更容易出现性能瓶颈；

4.最重要的是用户越来越不仅仅只满足于按钮能用，还关注点点点后的反应速度。

**二、订单测试方案**

（基于 UI、功能、可靠、易用、效率、维护、可移植）

**0. UI 界面测试**

* 打开页面后，检查页面布局是否合理，显示是否完整。

验证按钮及其他链接是否可以正常点击。

**1. 功能测试**

* **订单取消**：用户下单后，测试取消订单功能。
* **订单超时处理**：
* 下单后一直不付款，检查订单超时不付款的场景下是否会自动取消订单。
* 在订单提醒超时时付款，验证系统处理逻辑。
* **多端操作**：在不同终端登录，一端取消订单，同时另一端对该订单进行付款，观察系统行为。
* **弱网测试**：在弱网状态下，多次点击提交订单按钮，检查是否会生成多个订单。
* **退款申请**：
* 用户付款后，申请退款，验证退款流程。
* 卖家发货后，买家申请退款，检查系统处理。
* 买家收货后，买家申请退款，验证退款机制。
* **自动确认收货**：买家长时间不确认收货，检查系统是否自动确认收货。
* **评论处理**：收货后超时不评论，验证系统是否自动设为好评，订单状态是否为已结束。
* **优惠券与积分**：
* 下单时使用优惠券，检查优惠券是否能叠加使用（考虑平台优惠券、店铺优惠券及它们之间的互斥）。
* 下单时使用积分扣减，验证积分使用逻辑。
* **地址管理**：下单界面临时新增地址，测试勾选与不勾选地址的场景。
* **商品数量**：测试单个商品下单和多个商品下单的场景。

**库存管理**：用户下单后，检查库存是否减少；当订单被取消，库存是否恢复。同时，实时检查库表对应订单记录的状态码变化。

**2. 可靠（安全性）**

* **信息拦截与篡改**：使用 Fiddler 等工具，检查是否能拦截篡改订单信息。
* **风控机制**：单用户多次提交订单，观察是否有风控提示。

**未登录下单**：测试未登录状态下下单，以及使用第三方账号（如微信、知乎）登录下单的场景。

**3. 易用**

* **界面整洁性**：检查订单界面是否整洁易懂。
* **功能提示**：下单、取消订单、申请退款等功能是否有友好的提示。
* **超时提示**：超时时长是否有倒计时提示，超时是否有明确提示。
* **发货状态展示**：只对订单的部分商品进行发货时，检查订单里的商品发货状态是否分开展示。

**快捷键支持**：验证是否支持 Enter、Tab 等快捷键操作。

**4. 效率（性能）**

**页面加载速度**：打开下单页面应较快，提交订单响应应迅速。

**5. 维护**

* **应用升级**：应用升级时，检查是否有提示用户安装更新（针对 app）。
* **更新与卸载**：验证是否能正常更新、卸载、重装（针对 app）。

**系统维护提示**：系统维护期间，检查是否能正常提示用户。

**6. 可移植（兼容性）**

* **浏览器兼容性**：在不同浏览器上测试是否能正常操作。
* **手机兼容性**：在不同手机上测试是否能正常操作。

**网络兼容性**：在不同网络（wifi/3G/4G/5G）下测试是否能正常操作。

**三、写好测试用例的关键是什么？**

1、业务知识要过关：对需求和设计文档的理解，对系统的熟悉。

2、测试理论要扎实：编写用例的方法要熟悉，xmind 和 excel 工具要用上。

3、用例评审不可少，需要产品和开发的指点。

**四、们公司的测试流程是什么?**

召开项目启动会明确项目背景、项目功能、项目时间、研发团队人员

项目会议结束后，有需求文档、原型、设计稿资料(会有资料但不一定完全有所有的看项目)

编写测试用例、发起测试用例评审、评审通过后等待技术提测

冒烟测试-->一轮测试-->二轮测试-->回归测试-->UAT 测试-->测试通过后提交测

试报告-->线上测试

**五、在测试中发现了一个bug，但是开发经理认为这不是一个bug，应该怎么解决?**

记录问题、判断问题(需求说明书、设计文档等)。如果没有就根据设计需求的人沟通确认是否是缺陷、合理陈述如果依然有问题可以反馈给领导决策

**MySQL 练习题（挑的简单的）**

**1. 查询"01"课程比"02"课程成绩高的学生的信息及课程分数**

|  |
| --- |
| SQL SELECT s.\*, a.s\_score AS score\_01, b.s\_score AS score\_02 FROM Student s JOIN Score a ON s.s\_id = a.s\_id AND a.c\_id = '01' JOIN Score b ON s.s\_id = b.s\_id AND b.c\_id = '02' WHERE a.s\_score > b.s\_score; |

**2. 查询"01"课程比"02"课程成绩低的学生的信息及课程分数**

|  |
| --- |
| SQL SELECT s.\*, a.s\_score AS score\_01, b.s\_score AS score\_02 FROM Student s JOIN Score a ON s.s\_id = a.s\_id AND a.c\_id = '01' JOIN Score b ON s.s\_id = b.s\_id AND b.c\_id = '02' WHERE a.s\_score < b.s\_score; |

**3. 查询平均成绩大于等于60分的同学的学生编号和学生姓名和平均成绩**

|  |
| --- |
| SQL SELECT s.s\_id, s.s\_name, ROUND(AVG(sc.s\_score), 2) AS avg\_score FROM Student s JOIN Score sc ON s.s\_id = sc.s\_id GROUP BY s.s\_id, s.s\_name HAVING AVG(sc.s\_score) >= 60; |

**3.1 总分超过200分的同学**

|  |
| --- |
| SQL SELECT s.s\_id, s.s\_name, SUM(sc.s\_score) AS total\_score FROM Student s JOIN Score sc ON s.s\_id = sc.s\_id GROUP BY s.s\_id, s.s\_name HAVING SUM(sc.s\_score) > 200; |

**4. 查询平均成绩小于60分的同学的学生编号和学生姓名和平均成绩（包括有成绩的和无成绩的）**

|  |
| --- |
| SQL SELECT s.s\_id, s.s\_name, IFNULL(ROUND(AVG(sc.s\_score), 2), 0) AS avg\_score FROM Student s LEFT JOIN Score sc ON s.s\_id = sc.s\_id GROUP BY s.s\_id, s.s\_name HAVING avg\_score < 60 OR avg\_score IS NULL; |

**5. 查询所有同学的学生编号、学生姓名、选课总数、所有课程的总成绩**

|  |
| --- |
| SQL SELECT s.s\_id, s.s\_name, COUNT(sc.c\_id) AS course\_count, SUM(sc.s\_score) AS total\_score FROM Student s LEFT JOIN Score sc ON s.s\_id = sc.s\_id GROUP BY s.s\_id, s.s\_name; |

**6. 查询"李"姓老师的数量**

|  |
| --- |
| SQL SELECT COUNT(\*) AS li\_count FROM Teacher WHERE t\_name LIKE '李%'; |

**7. 查询学过张三老师授课的同学的信息**

|  |
| --- |
| SQL SELECT DISTINCT s.\* FROM Student s JOIN Score sc ON s.s\_id = sc.s\_id JOIN Course c ON sc.c\_id = c.c\_id JOIN Teacher t ON c.t\_id = t.t\_id WHERE t.t\_name = '张三'; |

**8. 找出没有学过张三老师课程的学生**

|  |
| --- |
| SQL SELECT s.\* FROM Student s WHERE s.s\_id NOT IN (  SELECT sc.s\_id  FROM Score sc  JOIN Course c ON sc.c\_id = c.c\_id  JOIN Teacher t ON c.t\_id = t.t\_id  WHERE t.t\_name = '张三' ); |

**9. 查询学过编号为01，并且学过编号为02课程的学生信息**

|  |
| --- |
| SQL SELECT s.\* FROM Student s WHERE s.s\_id IN (  SELECT s\_id FROM Score WHERE c\_id = '01' ) AND s.s\_id IN (  SELECT s\_id FROM Score WHERE c\_id = '02' ); |

**10. 查询学过01课程，但是没有学过02课程的学生信息**

|  |
| --- |
| SQL SELECT s.\* FROM Student s WHERE s.s\_id IN (  SELECT s\_id FROM Score WHERE c\_id = '01' ) AND s.s\_id NOT IN (  SELECT s\_id FROM Score WHERE c\_id = '02' ); |

**28. 查询男女生人数**

|  |
| --- |
| SQL SELECT s\_sex, COUNT(\*) AS count FROM Student GROUP BY s\_sex; |

**29. 查询名字中含有"风"字的学生信息**

|  |
| --- |
| SQL SELECT \* FROM Student WHERE s\_name LIKE '%风%'; |

**31. 查询1990年出生的学生信息**

|  |
| --- |
| SQL SELECT \* FROM Student WHERE s\_birth LIKE '1990%'; |

**37. 查询不及格的课程**

|  |
| --- |
| SQL SELECT s.s\_id, s.s\_name, c.c\_name, sc.s\_score FROM Student s JOIN Score sc ON s.s\_id = sc.s\_id JOIN Course c ON sc.c\_id = c.c\_id WHERE sc.s\_score < 60; |

**44. 检索至少选修两门课程的学生学号**

|  |
| --- |
| SQL SELECT s\_id FROM Score GROUP BY s\_id HAVING COUNT(DISTINCT c\_id) >= 2; |

**47. 查询本周过生日的学生**

|  |
| --- |
| SQL SELECT \* FROM Student WHERE WEEK(s\_birth) = WEEK(CURDATE()); |

**48. 查询下周过生日的学生**

|  |
| --- |
| SQL SELECT \* FROM Student WHERE WEEK(s\_birth) = WEEK(CURDATE()) + 1; |

**49. 查询本月过生的同学**

|  |
| --- |
| SQL SELECT \* FROM Student WHERE MONTH(s\_birth) = MONTH(CURDATE()); |

**50. 查询下月过生的同学**

|  |
| --- |
| SQL SELECT \* FROM Student WHERE MONTH(s\_birth) = MONTH(CURDATE()) + 1; |

**3 redis**

**Redis 课程引入**

hello，各位同学大家好呀。这一节课来学习 Redis 数据库。Redis 是一个开源的内存数据库，它提供了一个高性能的键值存储系统。有些同学可能对键值对不是很了解，后面会详细讲解，也就是 key-value 的形式。在接口编程里，也会用到 key-value。

**Redis 简介**

Redis 是一个基于内存的 key-value 结构数据库，属于非关系型数据库。之前学 MySQL、Oracle、PostgreSQL 等数据库时，知道关系型数据库是用数据表组成数据库，通过表和表之间的关联进行查询。而 Redis 作为非关系型数据库，具有基于内存处理、读写性能高的特点，读写速度非常快，适合存储热点数据，比如热点商品、资讯新闻等场景。

**Redis 应用场景**

Redis 适合缓存、消息队列、会话存储等场景。例如，处理用户会话数据时，像用户的详细信息、用户状态等，都可以用 Redis 进行存储。在消息系统方面，Redis 支持发布订阅模式，也可用作实时消息系统，像社交网络、实时分析、实时通知等场景下，Redis 可以创建多个发布者和订阅者，实现消息的实时传输和快速传输。不过，像如何接收消息、支持发布订阅消息等具体开发细节，不在今天的学习内容里。

**Redis 优点**

* **极高性能**：Redis 以极高的性能著称，能够支持每秒数十万次的读写操作，是处理高并发请求的理想选择，尤其在需要快速响应的场景中，使用频率极高。
* **丰富数据类型**：Redis 不仅支持基本的键值对存储方式，还提供了丰富的数据类型，如字符串、列表、集合、哈希等。这为开发人员提供了比较灵活的数据操作能力，使 Redis 可以适应不同的应用场景。
* **原子性**：Redis 的数据操作要么执行，要么不执行，具有原子性，能够确保数据的一致性和完整性。
* **设置数据过期时间**：Redis 可以设置数据的过期时间，在工作里，开发人员经常会设置 token 的有效期、会话的有效期等。

**应用场景广泛**：Redis 支持跨平台使用，不管是在 Linux、Mac 还是 Windows 上，都能在不同的技术栈里部署。其应用场景包括但不限于缓存系统、会话存储、排行榜、实时分析等。在面试时，主要讲前面三个优点就够用了。

**Redis 数据类型及命令学习准备**

Redis 是一个非关系型数据库，也叫结构数据库、缓存数据库、内存数据库。关系型数据库的数据是放在磁盘里的，表和表之间有关联。而 Redis 作为非关系型数据库，具有极高的性能、丰富的数据类型、能保证数据完整性和一致性、可设置数据过期时间等优点。

今天主要学习 Redis 的字符串、列表、集合、哈希这四种数据类型的命令。在学习命令之前，需要先安装 Redis。老师发了两个工具包给大家，大家解压 Redis-x64-3.0.504 这个压缩包成文件夹，可以放在桌面、D 盘或其他盘。

**Redis 安装步骤**

* **第一步**：运行 redis-server。打开解压后的文件夹，运行 redis-server.exe，打开后会看到一个界面，这个界面不要关掉，关掉的话 Redis 就无法使用了。
* **第二步**：运行 redis-cli。还是打开刚刚的文件夹，运行 redis-cli.exe。

**第三步**：输入 ping，返回 pong，就表示 Redis 安装成功了。

**Redis 字符串命令**

* **创建添加数据**：Redis 的语法和 MySQL 数据库的语法不一样，它是 key-value 形式的。创建添加数据使用 set 命令，比如 set 部门 第一批 01，就创建了一个键为“部门 第一批”，值为“01”的数据。
* **查询数据**：查询数据使用 get 命令，比如 get 部门 第一批，就可以获取到对应的值“01”。
* **设置数据有效期**：使用 expire 命令设置数据的有效期，比如 expire test005 40，表示将键为“test005”的数据的有效期设置为 40 秒，返回 1 表示设置成功。

**查询剩余时间**：使用 ttl 命令查询键的剩余时间，比如 ttl test005，可以查看键“test005”还有多久过期。返回负一表示没有过期时间，返回负二表示数据的时间已失效。

**4-1 LINUX基本概念和安装**

**课程引入与Linux简介**

Hello，各位同学大家好呀！今天这一节课学习Linux。在学习Linux之前，首先要了解一下Linux是什么。一般来说，学一门课程，都会先讲讲这门课程相关技术的一些优点，以及可以用它来干什么。

Linux是一套免费的操作系统，它支持多用户、多任务、多线程和多CPU。它是基于Unix操作系统开发的，因此继承了Unix的许多优秀特性。

**Linux与其他操作系统的区别**

在生活和工作中，会接触到很多系统，比如Windows系统、安卓系统、iOS系统等。那么，Linux系统和这些系统有什么区别呢？

最大的区别就是Linux是命令行的操作系统，而Windows系统是图形化界面的操作系统。在Windows中，创建文件夹、新增文件等操作都是通过可视化的界面完成的；而在Linux中，需要输入相关的命令语句来执行这些操作。

另外，Windows系统是收费的（家庭版本除外），而Linux系统是免费的。此外，Windows系统经常需要更新安全补丁，而Linux系统相对更安全，不需要频繁更新补丁。

**Linux的优点**

Linux的优点有很多，首先就是它的稳定性很高。因为Linux是基于Unix操作系统开发的，所以它继承了Unix的稳定性和可靠性。

其次，Linux在系统服务器搭建方面应用广泛。许多程序部署的服务器都采用Linux系统。

再者，Linux是开源的，这意味着它是免费的。从公司成本控制的角度来看，使用Linux可以节约成本。

此外，Linux的安全性高、性能好、资源占用少。这些优点使得Linux在服务器领域得到了广泛应用。

**测试中Linux的应用**

那么，测试要用Linux来干什么呢？

首先，测试环境搭建和部署是Linux在测试中的一个重要应用。虽然现在一般的公司里面都有运维人员负责环境搭建和部署，但测试人员了解这一过程也是很有必要的。

其次，在测试过程中，如果发现后端问题，可以利用Linux服务器来定位问题并分析日志。通过查看报错的日志，可以更快地找到问题的根源。

此外，Linux还可以用于性能测试。在性能测试时，Linux能够监控服务器的性能资源使用情况，帮助评估系统的性能表现。

**云服务器的申请与使用**

为了学习Linux，需要申请一个云服务器。目前市面上有很多云服务器提供商，比如阿里云、华为云、腾讯云、百度智能云等。

对于完全没有用过云服务器的同学，我建议先申请阿里云、华为云或腾讯云的免费试用。申请步骤一般比较简单，只需要按照提示进行操作即可。

在申请云服务器时，需要注意选择正确的镜像版本。对于Linux服务器，一般选择CentOS 7.5版本。

申请成功后，需要登录云服务器平台进行实名认证等操作。然后，就可以开始使用云服务器来学习Linux了。

**Xshell的安装与连接**

为了方便地连接和操作Linux服务器，需要使用一个可视化的操作工具，比如Xshell。

Xshell的安装包一般可以在Linux的安装包中找到。只需要点击下一步、下一步进行安装即可。安装完成后，需要启动Xshell并新建一个连接。

在新建连接时，需要输入云服务器的公网地址、用户名和密码。用户名一般是root，密码则是申请云服务器时设置的密码（如果忘记了密码，可以在云服务器平台上重置密码）。

输入正确的账号密码后，就可以成功连接到Linux服务器了。此时，可以看到服务器的根目录下有一些默认的文件夹和文件。

**4-2 LINUX基本命令、查看日志**

**Linux 命令学习**

**1. Linux 文件基础**

**1.1 盘符与根目录**

* Windows 中有盘符概念，如 C 盘、D 盘等。
* Linux 中没有盘符，只有一个根目录，用斜杠 / 表示。
* 进入根目录命令：cd /

查看当前路径命令：pwd

**1.2 根目录下的文件与文件夹**

* 文件夹：蓝色
* 文件：白色

脚本等特殊文件：绿色（青色等，暂不深入）

**1.3 常见目录含义**

* /bin：存放常用命令
* /home：用户目录
* /etc：系统管理配置文件和子目录
* /tmp：临时文件存放目录

/usr：应用程序和文件存放目录，约等于 Windows 的 C 盘

**2. Linux 路径**

**2.1 路径表示**

* .：当前路径
* ..：上一级目录

/：根目录

**2.2 路径演示**

|  |
| --- |
| Bash # 查看当前路径 pwd # 当前在 root 目录下 cd . pwd # 仍在 root 目录 cd .. pwd # 返回到根目录 cd / pwd # 在根目录下 |

**3. Linux 常用命令**

**3.1 查看目录与文件**

* pwd：查看当前所在目录
* ls：查看当前目录内容
* ls -l：查看当前目录详细信息

ls -a：查看当前目录所有内容，包括隐藏文件

**3.2 创建文件夹与文件**

* 创建文件夹：mkdir

|  |
| --- |
| Bash mkdir test01 cd test01 ll # 查看当前目录（无内容） |

* 创建文件：可使用 touch 等命令，此处示例创建 r2.text 文件

|  |
| --- |
| Bash touch r2.text ll # 查看，有 r2.text 文件 |

**3.3 查看文件内容**

* cat：查看文件内容

|  |
| --- |
| Bash cat r2.text # 无内容无反应 |

**3.4 清空终端内容**

* clear：清空当前终端显示内容

|  |
| --- |
| Bash clear # 终端内容清空 |

**3.5 写入内容到文件**

|  |
| --- |
| Bash # 将当前页面内容写入文件 ls >> GG.TXT cat GG.TXT # 查看，写入了一些内容 |

**3.6 复制文件**

* cp：复制文件

|  |
| --- |
| Bash # 创建文件夹和文件 mkdir test09 cd test09 touch test.TXT ll # 查看，有 test.TXT 文件  # 创建目标文件夹 mkdir ss  # 复制文件（注意大小写） cp test.TXT /ss cd ss ll # 查看，有 test.TXT 文件  # 再次复制示例 cd .. touch 11.TXT cp 11.TXT /ss cd ss ll # 查看，有 11.TXT 文件 |

**3.7 移动文件**

* mv：移动文件

|  |
| --- |
| Bash # 创建文件夹 cd .. mkdir dep  # 移动文件 mv dep /ss cd ss ll # 查看，有 dep 文件夹  # 查看原目录 cd .. ll # test09 目录下无 dep 文件夹 |

**3.8 修改文件名**

* mv：也可用于修改文件名

|  |
| --- |
| Bash # 查看 test09 目录下文件 cd test09 ll # 有 11.TXT 文件  # 修改文件名 mv 11.TXT 222.TXT ll # 11.TXT 无，222.TXT 有 |

**3.9 查看日志**

* tail -f：动态查看日志

|  |
| --- |
| Bash # 示例操作 cd test09 echo "some data" >> 222.text cat 222.text # 查看，有数据  # 动态查看日志 tail -f 222.text # 在另一个终端或窗口写入数据 echo "more data" >> 222.text # 第一个终端会实时显示新数据 |

**3.10 日志报错关键词**

* 连接报错：如连接数据库（MySQL 等）错误
* 数据库报错：如 Redis 数据库报错
* SQL 语句报错
* 空指针报错
* 类型转换报错
* Spring 框架报错（org 开头等）
* **4. Linux 快捷键**
* Ctrl + C：操作退出
* Tab：自动补全
* 上下键：查找执行过的命令
* Shift + Insert：粘贴

Ctrl + L：清空当前终端数据

**4-3 Linux高级命令&xftp讲解**

**1. Vim 模式**

* Vim 模式可理解为编辑模式，Linux 提供了 Vim 编辑器。

进入 Vim 编辑模式：vim 文件名

**1.1 Vim 模式分类**

* **命令模式**：默认模式，用户不能直接编辑文件，但可使用命令操作文件，如输入信息、删除行、复制行、粘贴等。
* **插入模式**：用于输入内容，可通过以下方式进入：
* i：在当前位置输入
* o：换行输入
* a：从光标之后输入

s：删除光标所在字符并开始输入

**1.2 Vim 模式操作示例**

|  |
| --- |
| Bash # 进入 Vim 编辑模式 vim 222.TXT  # 进入插入模式（示例） i # 输入内容 esc # 退出插入模式 shift + : # 进入命令行模式 wq # 保存并退出  # 其他插入模式操作类似 |

**1.3 命令行模式操作**

* 保存：w
* 退出：q
* 保存并退出：wq
* 撤销上一步操作：u
* 显示行号：:set nu
* 取消显示行号：:set no nu

查找内容：/查找内容，按 n 下翻

**1.4 删除操作**

* 删除所在行：dd（需配合其他命令使用，或使用其他删除命令）

删除光标所在位置字符：x

**2. 删除命令**

* 删除文件或文件夹：rm -rf 文件名或文件夹名

|  |
| --- |
| Bash # 删除 222.TXT 文件 rm -rf 222.TXT |

**3. 系统信息相关命令**

* 查看系统中所有进程：ps -aux
* 查看进程及其依赖关系：ps -ef

动态显示进程：top（按 Ctrl + C 退出）

**4. 压缩相关命令**

* 打包文件：tar -zcvf 压缩包名.tar.gz 文件名
* 解压文件：tar -zxvf 压缩包名.tar.gz

|  |
| --- |
| Bash # 打包 111.TXT 文件为 number.tar.gz tar -zcvf number.tar.gz 111.TXT  # 解压 number.tar.gz rm -rf 111.TXT # 先删除原文件 tar -zxvf number.tar.gz |

**5. 管道命令**

* 管道符号 | 表示一个命令的输出作为另一个命令的输入。

|  |
| --- |
| Bash # 查看 JAVA 进程 ps -ef | grep JAVA  # 查看 etc 目录下的所有文件 ls /etc | grep '\*' |

**6. 安装命令**

* 从网络上自动查找并安装内容：yum
* 查看网络上所有的包：yum list
* 查看特定包（如 JAVA）：yum list | grep JAVA

安装包：yum install 包名（安装过程中按 y 确认）

**7. 文件传输（XFTP）**

* XFTP 是一个工具，帮助用户在 Linux 系统和 Windows 系统之间进行快速的文件传输。
* 安装 XFTP 后，通过新建连接，输入服务器公网 IP、用户名和密码进行连接。

连接成功后，可进行文件的上传和下载操作。

**8. Vim 非法退出处理**

* 当 Vim 非法退出时，可能会留下 .swp 文件。
* 处理方法：
* 找到 .swp 文件并删除。

重新进入 Vim 编辑模式。

**9 查看文件内容的其他命令**

* cat：查看全部文件内容。
* more：按进度查看大文件。
* head：查看文件前几行（默认前四行，可通过 -n 指定行数）。
* **5-1浏览器抓包及接口基础**
* **接口测试学习**
* **1. 接口的概念**
* 接口是两个模块与模块之间数据交互的一个通道。可以用硬件的例子来理解，比如电脑和手机设备通过USB接口连接。同样，软件的接口也是模块间数据交互的通道。
* **2. 抓包工具介绍**
* 在学习接口测试之前，先来了解一下抓包工具。抓包工具分为：
* 浏览器的开发工具
* Charles
* Fiddler
* **2.1 浏览器开发工具**
* 浏览器开发工具只能抓取外部端的接口，无法抓取APP或其他应用的接口。
* **2.2 Charles 和 Fiddler**
* Charles和Fiddler可以抓取所有接口，包括手机APP、PC端应用和外部端的接口，抓包更加详细和完整。
* **3. 通过浏览器开发工具认识接口组成**
* **3.1 打开浏览器开发者工具**
* 按 F12 打开开发者工具
* 如果打不开，可以按 Ctrl + F12

1. 初始界面可能是英文的，可以通过设置切换为中文
2. **3.2 开发者工具界面介绍**
3. 开发者工具界面包括：
4. 元素（Elements）
5. 控制台（Console）
6. 网络（Network）
7. 性能（Performance）
8. 内存（Memory）
9. 应用（Application）

* 安全（Security）
* 今天重点学习元素、控制台、网络和应用。
* **3.3 元素（Elements）**
* 也称为 elements
* 用于UI自动化测试
* 可以修改页面显示（前端修改，不影响后端）
* 可以定位元素，查看元素的尺寸、颜色等信息
* 可以进行兼容性测试和入网测试
* **3.4 控制台（Console）**
* 查看前端代码
* 运行JS代码
* 查看日志
* 正常日志：黑色字体
* 警告日志：黄色字体（感叹号）
* 错误日志：红色字体
* 日志等级：
* error：报错
* warning：警告
* info：正常打印输入输出信息
* debug：调试日志
* **3.5 抓包工具（Network）**
* 抓取数据包
* 接口地址（URL）：
* 请求协议：http 或 https
* IP端口/域名
* 接口地址
* 接口参数（问号后）
* 参数值（多个参数用 & 连接）
* 接口类型：
* GET：请求数据在URL中，不安全，用于查询类接口
* POST：请求数据在body中，隐藏且无长度限制，用于增删改操作
* 状态代码：
* 200：OK，接口运行正常
* 400：请求错误，数据格式不正确
* 403：访问权限问题
* 404：接口地址错误
* 405：请求类型错误
* 500：内部服务器错误，后端代码问题
* **3.6 应用（Application）**
* 储存cookies
* 查看历史cookies

1. 查看本地存储和缓存
2. **4. 总结**
3. 今天学习了接口的组成、接口类型、状态代码以及浏览器开发者工具的使用。具体包括：
4. 接口的概念：模块间数据交互的通道
5. 抓包工具：浏览器开发工具、Charles、Fiddler
6. 浏览器开发者工具：

* 元素（Elements）：UI自动化测试，修改页面显示
* 控制台（Console）：查看前端代码，运行JS代码，查看日志
* 抓包工具（Network）：抓取数据包，了解接口地址、接口类型、状态代码

应用（Application）：储存cookies，查看本地存储和缓存

**5-2 Postman基本操作**

**接口工具学习**

今天学习接口工具。常用的接口工具有Postman和Jmeter。

Postman可以用来做接口测试，Jmeter也可以用来做接口测试，甚至还可以用来做性能测试。

* 今天主要学习用Postman来做接口测试。为什么不学习用Jmeter做接口测试呢？因为相对来说Jmeter会麻烦一点，而且掌握了一个工具的使用后，另一个工具的使用也会自然而然地掌握。
* **1. Postman的安装**
* Postman的文件包已经发给大家了，直接安装即可。安装非常简单，只需无脑点击下一步，然后选择想要安装的磁盘即可。安装完成后，打开Postman，进入其页面。整个页面都是英文的，如果看不懂也没关系，接下来会一步步讲解。
* **2. Postman的基本操作**
* **2.1 创建请求**
* 点击“New”按钮
* 选择“HTTP Request”或点击加号新建请求
* 根据接口文档创建接口
* **2.2 示例：企业微信接口**
* 将以企业微信为例进行接口测试。首先，会把企业微信的接口文档发到群里面。接口文档会详细介绍每个接口的使用方法、参数说明等。
* **2.2.1 接口概述**
* 接口文档会告诉如何使用接口，包括开发者指南、开发前必读、API接口说明等。例如，所有接口使用HTTPS格式，JSON数据格式，UTF-8编码等。
* **2.2.2 基本概念**
* 接口文档中会有一些术语介绍，例如：
* CorpID：每个企业拥有唯一的ID
* UserID：每个成员拥有唯一的用户ID
* Secret：用于生成access\_token的重要票据
* access\_token是企业微信API接口的第一步，相当于创建了一个登录的凭证，其他的业务API接口都需要依赖于这个access\_token来验证接口的访问权限。
* **2.3 获取access\_token接口**
* 请求方式：GET
* 请求地址：包含CorpID和Secret的URL

1. 参数说明：CorpID和Secret都是必填项
2. **2.3.1 示例操作**
3. 在Postman中选择GET请求方式
4. 输入请求地址，替换CorpID和Secret为实际值
5. 点击“Send”按钮发送请求

* 查看返回结果，包括errorcode、errmsg、access\_token和expires\_in
* **2.3.2 错误排查**
* 如果返回错误码，例如40013，可以通过企业微信的错误排查码查询具体原因。例如，40013表示不合法的CorpID。
* **3. Postman的详细操作**
* **3.1 请求类型**
* Postman支持多种请求类型，如GET、POST、PUT、DELETE等。根据接口文档选择正确的请求类型。
* **3.2 请求地址和参数**
* GET请求的参数放在URL中
* POST请求的参数放在Body中
* **3.3 请求头**
* Postman会自动添加一些请求头，如果没有特别说明，一般不需要修改。
* **3.4 Body**
* POST请求的参数一般放在Body中。Body有多种格式，如form-data、x-www-form-urlencoded、raw等。如果没有特别说明，一般使用raw格式，并选择JSON。
* **3.5 响应**
* 响应数据可以在“Body”选项卡中查看
* 可以选择不同的格式查看响应数据，如Pretty、Raw、Preview
* 响应头可以在“Headers”选项卡中查看
* 测试结果可以在“Tests”选项卡中查看
* **3.6 保存接口**
* 创建完接口后，养成随手保存的好习惯。可以点击“Save”按钮保存接口，并命名接口。
* **4. 示例：创建成员接口**
* **4.1 接口说明**
* 请求方式：POST
* 请求地址：包含access\_token的URL

1. 参数说明：包括userid、name、mobile等必填项
2. **4.2 示例操作**
3. 在Postman中选择POST请求方式
4. 输入请求地址，替换access\_token为实际值
5. 在Body中选择raw格式，并选择JSON
6. 输入请求参数，只测试必填项
7. 点击“Send”按钮发送请求
8. 查看返回结果
9. **4.3 错误排查**
10. 如果返回错误码，例如60104，表示手机号码已存在。可以更换手机号码或userid后重新测试。
11. **5. 企业微信接口测试作业**
12. **5.1 作业内容**
13. 整理今天的笔记
14. 使用Postman实现企业微信的管理，包括以下接口：

* 获取access\_token接口
* 创建成员接口
* 更新成员接口
* 删除成员接口
* 获取部门成员接口
* 获取部门详情接口
* **5.2 注意事项**
* 简化测试，只测试必填项参数

1. 在工作里面，全部的参数都需要去做测试
2. **6. 注册企业微信**
3. 要使用企业微信接口测试，首先需要注册一个企业微信公司。注册途径有两种：
4. 在官网点击立即注册，填写信息

* 用手机APP注册
* 注册完成后，获取企业ID，并进入企业管理的后台，获取Secret。
* **6.1 配置可信IP**
* 在企业微信的安全与管理页面，进入管理工具，配置可信IP。需要查询自己的公网IP，并设置到企业微信中。
* **7. 接口地址和排错码**
* 接口地址：[企业微信接口地址]
* 排错码地址：[企业微信排错码地址]
* 企业微信后台管理地址：[企业微信后台管理地址]

**8. 总结**

今天学习Postman的基本操作，包括创建请求、设置请求类型、输入请求地址和参数、查看响应等。还通过企业微信的接口示例，实际操作了获取access\_token和创建成员接口。

**5-3接口测试流程**

**1. Postman 导入导出操作**

**2. 接口测试流程**

**2.1 接口测试概述**

* 接口测试与功能测试流程相似，但测试对象为接口。
* 接口测试流程包括：需求分析、冒烟测试、业务逻辑串联、详细接口测试。

**2.2 冒烟测试**

* 按照接口文档进行冒烟测试，确保接口能正常返回数据。
* 冒烟测试通过后，说明接口调通，具有可测性。

**2.3 业务逻辑串联**

* 按照业务逻辑将接口串联起来，确保接口能跑通业务逻辑。
* 例如，获取接口凭证、创建成员、读取成员信息等接口的串联。

**2.4 详细接口测试**

* 对每个接口进行单独测试，包括正向场景和逆向场景。
* 使用场景法设计测试用例，考虑各种边界条件和异常情况。

**3. 接口测试实例：创建部门接口**

**3.1 接口文档分析**

* 接口类型：POST
* 请求地址：HTTPS
* 请求参数：access\_token、name、parentid（必填项）
* 参数说明：
* access\_token：调用接口的凭证
* name：部门名称，长度限制为 1-64 个 UTF-8 字符
* parentid：父部门 ID，32 位整型

**3.2 正向场景测试**

* 有效的 access\_token
* 部门名称长度在 1-64 个字符之间
* 父部门 ID 存在且为 32 位整型

**3.3 逆向场景测试**

* 过期的 access\_token
* 错误的 access\_token
* 空 access\_token
* 部门名称过长或过短
* 部门名称为空
* 部门名称包含特殊字符
* 部门名称重复
* 父部门 ID 不存在
* 父部门 ID 不是 32 位整型

**3.4 测试结果校验**

* 校验状态码是否为 200
* 校验返回结果与预期结果是否一致
* 校验数据库是否入库

**4. 接口测试的意义**

* 尽早发现后端数据问题，提高软件质量。
* 发现系统测试无法测试出来的 bug。

**5. 作业**

* 整理接口测试流程，准备面试时回答。
* 丰富接口测试的意义，准备面试时回答。

多找几个接口进行测试，加深理解。

**5-4 Postman高阶操作**

**Postman高级操作学习**

**1. 环境变量的使用**

首先打开Postman软件。当获取到一个token之后，每次创建成员都要去修改这个token，这样比较麻烦。比如现在拿到了一个token码，需要复制，然后在创建成员这里使用。如果token少复制了一个或者多复制了一个，token就会失效，就会报错。

有没有一个简单的方法，可以一致性地定义这个token，在token下一次失效之前可以一直比较方便地调用它，而不是每次创建成员、读取成员、更新成员、删除成员时都要进行token的复制粘贴，这样工作会比较重复机械，工作量也比较大。

在Postman里面，有比较简单的操作，那就是使用环境变量。

**1.1 环境变量的位置**

在Postman中，点击右上角设置，这里可以创建环境，点击快速创建。会发现有两个环境，一个是局部环境，一个是全局环境。局部环境和全局环境的区别后面会讲解。

**1.2 创建环境**

现在来创建一个环境，命名为“企业微信的token”。这里可以随便命名，然后定义一个key - value形式的名字。比如要获取access token，简单一点，命名为“TOKEN”（这里大写是因为里面已有小写的token，为了避免重复）。value的值就是获取到的token值。

|  |
| --- |
| Plain Text {  "key": "TOKEN",  "value": "获取到的token值" } |

后面的内容是自动的，中间的tab是默认的，不需要填写和修改，然后保存。

现在就可以看到有一个企业微信的token。把之前复制的token删掉，在这个地方就可以看到企业微信的token值。在需要使用token的地方，用两个大括号加上刚刚命名的token名称，例如{{TOKEN}}。

|  |
| --- |
| Plain Text {  "token": "{{TOKEN}}" } |

* 然后调用一下，如果手机号码已经存在，就在界面里改一下手机号码和名字（避免重复），再次点击发送。通过创建环境变量的方式，就简化了操作，不需要每次都非常麻烦地复制粘贴。
* **1.3 环境变量的操作步骤回顾**
* 不管是全局变量还是局部变量，都是点击右上角设置，然后添加到局部环境。
* 添加局部环境之后，配置对应的URL信息。
* 使用两个大括号加上变量名称进行调用。
* **1.4 查看和修改环境变量**
* 这里有一个小眼睛图标，点击它可以查看和编辑token。比如把token改成“token3”，按ctrl s保存或者点击上面的save保存。改了之后，再调用就会显示无效的token，需要对应地把使用的地方也改成“token3”。
* **2. 自动获取token**
* 手动获取token可以简化操作，但token是有有效期的，比如两个小时过期后，就要去环境变量这里修改，还是很麻烦。有没有全自动的方法获取token呢？
* 在上一节课的时候，讲到Postman有一个操作叫做tests。在请求的菜单上面有一个test菜单，tests是写代码的，控制响应的内容。虽然目前没有学习到编程，也不会自动化代码，但是Postman已经内置了很多方法，只需要修修补补就可以自动获取token。
* **2.1 自动获取token的代码编写**
* 选择一个方法，比如“json value check”。

选择之后，会自动生成四行代码，这四行代码实际上是一个断言的作用，但用不到断言，所以把第一行、第三行和第四行代码删掉，只保留第二行代码。

|  |
| --- |
| JavaScript // 原始生成的代码（示例） pm.test("Your test name", function () {  var jsonData = pm.response.json();  pm.expect(jsonData.value).to.eql(100); });  // 修改后的代码（保留第二行并修改） var jsonData = pm.response.json(); |

第二行代码的意思是把响应的值存储到json data变量里。

赋值之后，还需要创建一个token变量。

|  |
| --- |
| JavaScript var token2 = jsonData.access\_token; |

这里把json data变量里面的第一层的access token取出来。怎么看它是不是在第一层呢？比如花括号只有一层，就是在第一层；如果花括号下面还有一个子集，就是第二层，然后相应地去取。

现在要把token2放到环境变量里面去，选择“set”方法。

|  |
| --- |
| JavaScript pm.environment.set("token2", token2); |

选择之后，修改k值和value值。k值是定义的token2，value值是access token。

|  |
| --- |
| JavaScript // 完整代码示例 var jsonData = pm.response.json(); var token2 = jsonData.access\_token; pm.environment.set("token2", token2); |

* 点击发送，可能会有报错，比如access token没有被定义。检查代码，确保变量名正确。再次发送，就可以看到新增了一个token2，并且获得了token的值。
* 现在可以把获取access token和创建成员的代码中的token值改成新创建的token2（注意大小写）。比如之前建了一个token3，现在通过代码创建了一个token5，把创建成员的token值改成token5，就可以成功创建成员。
* **2.2 自动获取token的优势**
* 每次发送获取接口的时候都会自动运行一遍代码，代码会自动把新的内容覆盖掉老的内容，所以即使token过期了也没有关系，不需要手动去修改。
* **3. 断言的使用**
* 在Postman里面，能不能不用写代码就可以实现接口自动化的操作呢？也是可以的，那就是使用断言。
* **3.1 断言的概念**
* 断言就是自动判断是否符合条件，如果不符合条件，代码就会报错。
* **3.2 断言的代码编写**
* 依然选择“response body json value check”方法。选择之后，进行修修补补。
* 修改测试名字，比如“获取access token是否成功”。

修改json data里面的value和预期的状态码。比如用err code来判断，err code的状态码返回零表示断言成功，非零表示断言失败。

|  |
| --- |
| JavaScript pm.test("获取access token是否成功", function () {  var jsonData = pm.response.json();  pm.expect(jsonData.errcode).to.eql(0); }); |

* 运行之后，test的结果里面如果没有返回任何内容，但access token里面有pass的返回，就说明Postman实现了自动化，自动判断是否符合条件。
* **3.3 断言在创建成员接口中的应用**
* 把断言脚本放到创建成员接口里面，修改测试名字为“创建成员是否成功”。点击发送，如果返回fail，说明创建成员失败了，可以根据状态码等信息进行排查。比如手机号码重复了，账号已存在等。修改信息后再次发送，test里面结果就是pass。
* **4. 批量运行接口**
* **4.1 批量运行接口的操作**
* 找到文件夹，在文件夹旁边有一个三角形按钮，点击它，选择需要运行的接口。因为后面的接口没有定义变量和通行词，所以这里选择前面两个接口。点击运行之后，Postman会自动批量运行接口，也可以调整顺序。
* **4.2 批量运行接口的操作步骤总结**
* 找到文件夹。
* 在文件夹旁边点击三角形按钮。
* 选择需要运行的接口。
* 调整顺序。
* **5. 作业布置**
* 设置环境变量，尝试设置token的自动化。
* 进行接口的断言。

批量运行接口。

**5-5 接口测试面试题**

**postman 用过吗？**

用过，很简单，他是谷歌浏览器 Chrome 一个插件(一般是单独的安装包)，安装和使用起来比较方便，也是用来发送http 请求的接口测试工具，功能没有 jmeter 强大。开发工程师一般用 postman 调试接口，测试工程师用 jmeter比较多点。两者原理都一样，都是基于 http 协议发送请求得到响应，请求需要有请求地址、请求方式、请求参数。（PS：有些小公司可能会使用 postman 多点）

**请求体如果是 json 格式的数据，该怎么设置？**

在请求头设置内容类型 content-type：application/json，且必须使用 post 方式。（python+requests 在 post 方法中使用 json 参数名）

**哪些功能用到接口测试?接口测试作用是什么**

一般接口测试用于移动端APP、H5、小程序这些

接口作用：

1.1一般的功能测试，覆盖不到多种情况。

1.2从安全角度考虑，只依赖于前端的校验不能满足要求，需要服务端也要做校验，比如：充值的界面，充值金额输入框不能输入负数，测试时，要确保服务器也做了这个校验。有一万种手段绕过前端验证。

1.3当小程序的代码不更新，而服务端代码更新时，直接通过接口自动化测试就能快速知道是否影响APP的功能。

1.4很多情况下是没有界面的，只提供接口功能，没法通过界面的方式进行测试。

**如何编写接口测试用例**

1.拿到接口文档后,一般使用 xmind 提取接口的测试点，使用 excel 编写接口测试用例。

2.输入参数主要使用等价类边界值来分析，还要考虑必填和非必填项，空和非空

个业务状态码就是一个测试点，还要考虑接口的异常、接口的安全性。

3.其实这和功能测试用例的设计没有什么区别。

主要考虑 5 大点，1.输入校验;2.输出检査;3.接口逻辑;4.异常处理;5.接口安全。

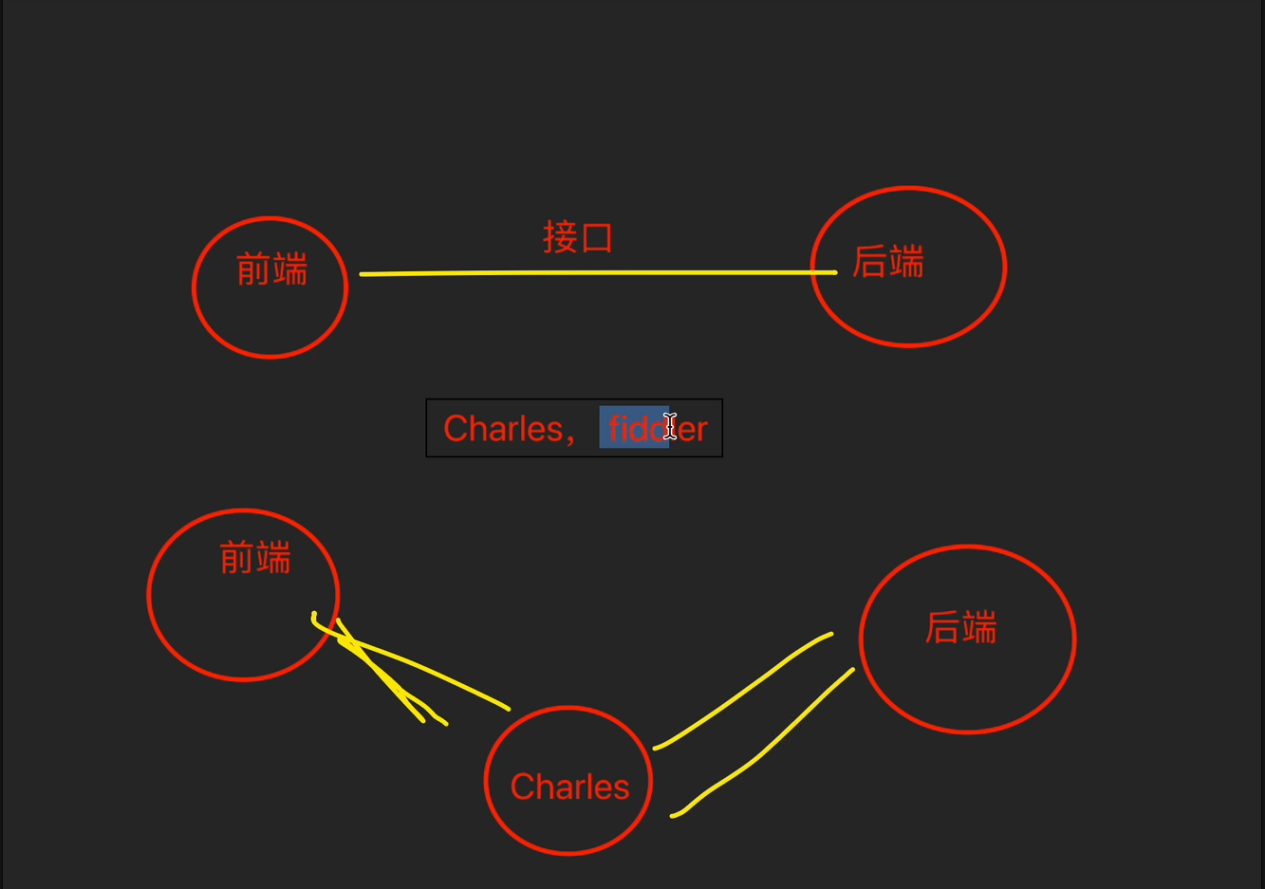
**接口用例包含哪些内容 ?**

模块名、用例名称、接口地址 URL、请求方式、请求头(headers)、请求参数(params)、请求主体(body)、取变量、预期结果。

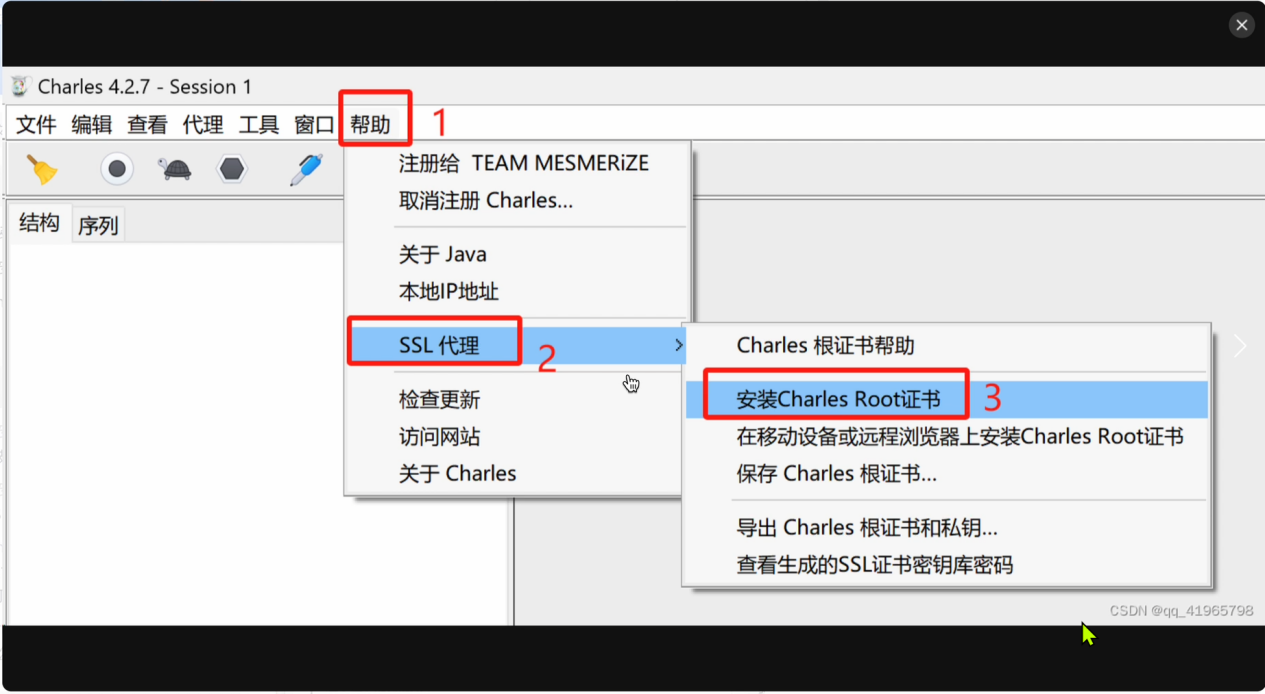
**6-1 Charles基本抓包设置**

**Charles 功能及应用**

* Charles 是一个强大的抓包工具，主要用于在前后端通信中拦截和修改数据。
* **前后端通信**：前端和后端通常通过接口进行连接。但在使用 Charles 时，前端会先连接到 Charles，Charles 再将数据转发给后端，同时可以拦截和修改数据。



* **抓包演示**：
* **HTTP 请求**：在浏览器中输入 http://h.cl.baidu.com，刷新页面后返回 Charles，会发现已经抓到了这个请求。
* **HTTPS 请求**：HTTPS 请求比 HTTP 多了一层加密。在抓包时，会发现界面上多了一个锁的图标，点击后显示的是未知的乱码。这是因为 HTTPS 加密了数据。
* **解决 HTTPS 加密问题**：
* **安装证书**：在 Charles 中选择“帮助”->“SSL 代理”->“安装 Charles 的证书”。按照提示安装证书，并确保证书在有效期内。



* **SSL 代理设置**：在“代理”->“SSL 代理设置”中添加一个规则，主机和端口都输入星号（\*），表示允许所有的主机地址和端口号。
* **重新打开 Charles**：设置完成后，关闭并重新打开 Charles。

**再次抓包**：在浏览器中访问百度，会发现 Charles 已经能够成功抓到 HTTPS 请求了。

**Charles 页面结构介绍**

* **结构和序列**：
* **结构**：将抓到的包按照网站进行归类，有层级目录。
* **序列**：将所有返回的结果全部列出来，不进行归类。

**过滤功能**：在页面上有一个过滤功能，可以筛选出想看的接口结果。例如，输入“baidu.com”可以筛选出所有与百度相关的接口。

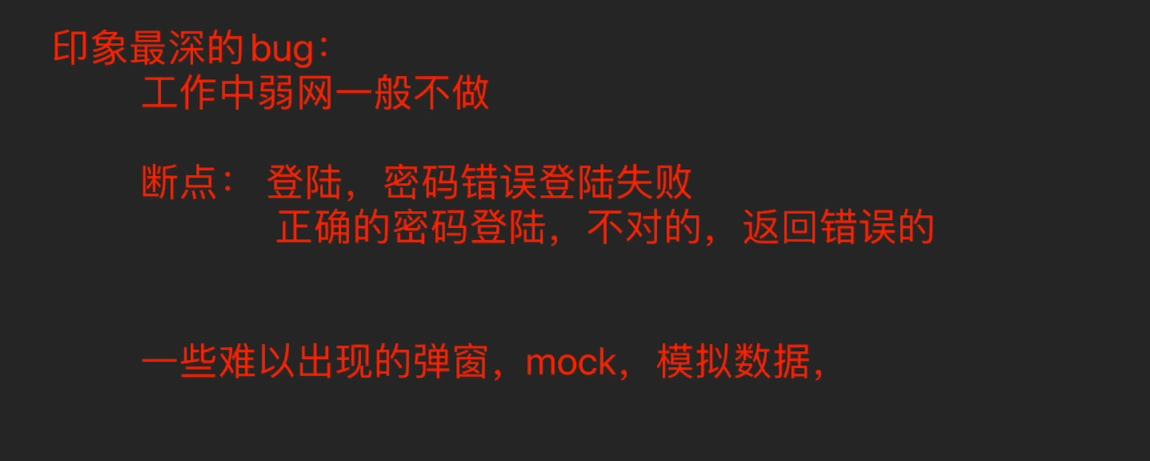
**Charles 小工具介绍**

* **清空按钮**：像扫把一样的按钮，用于清空抓到的包。
* **录制按钮**：开始记录按钮用于开始抓包，关闭后则不再抓包。
* **停止 SSL 代理**：停止 SSL 代理按钮用于停止抓包。
* **乌龟按钮**：乌龟按钮用于模拟弱网环境，做弱网测试时会用到。

**断点功能**：断点按钮用于拦截和篡改请求或返回的数据。例如，将正确的密码改成错误的密码，看看页面上会返回什么结果。

**Charles 在面试中的应用**

在面试中，可能会被问到印象最深的 bug。可以将使用 Charles 进行断点测试时发现的一个 bug 作为例子。例如，通过断点测试发现，即使输入正确的密码，页面也会返回密码错误的提示。



**Charles 高级设置**

**抓包时过滤**：除了在抓到包后进行过滤，还可以在抓包时就进行过滤。在“代理”->“记录设置”中，可以添加协议、主机、端口、路径和查询等条件进行过滤。例如，只抓取百度的 HTTPS 请求。

**6-2抓包工具的基本操作-必备**

**1. 代理记录设置补充**

之前讲解了代理记录设置中包括添加 at ps 协议，主机写的是 百度.com，只校验百度。现在进行延伸：

* 若想不管前面接口是什么（如 SP 100度、SP 200度），后边是 .cn、.cm 等都不用管，可以用星号 \* 代替，星号代表所有。
* 示例：\*百度.\*，这样筛选的就是 at ps 请求，只要中间是 百度，前后不管是什么都会被筛选出来。

操作：保存确定后，调用百度相关请求，会发现不管前面是什么（如 rmg、3w3w.），只要包含 百度 都会出来。

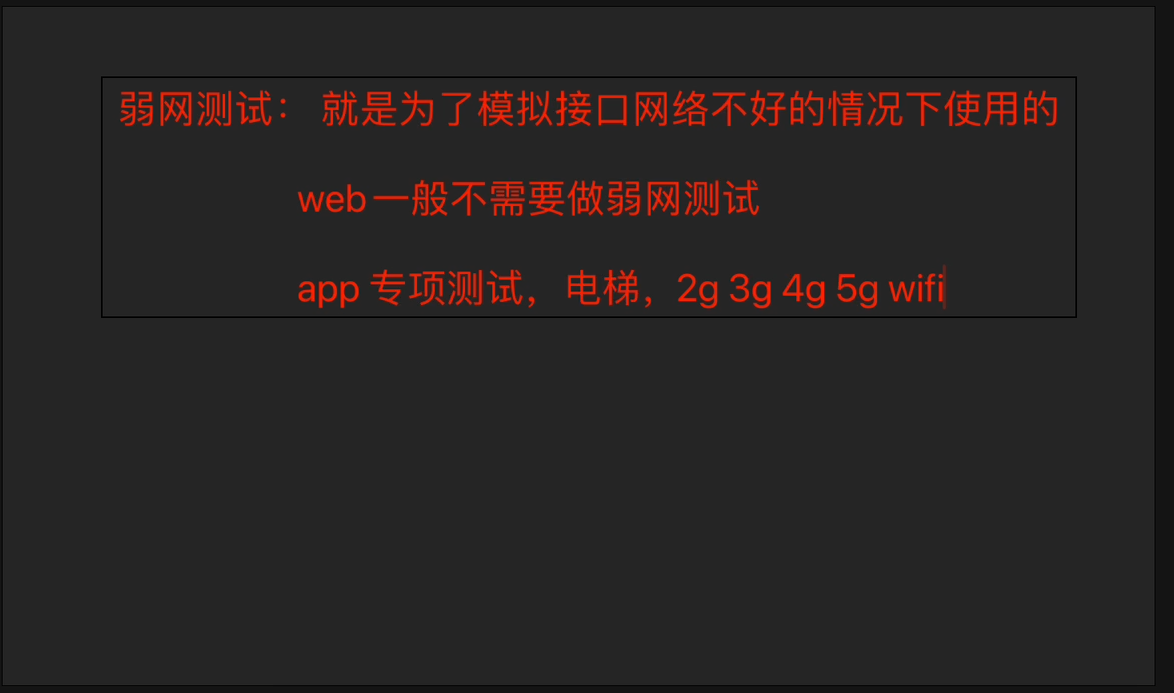
**2. 弱网测试**

**2.1 弱网测试概念**

* 在 JMeter 里，弱网测试使用小乌龟图标。
* 弱网测试目的是模拟接口在网络不好的情况下使用。

**2.2 弱网测试适用场景**

* 外部一般不需要做弱网测试，因为外部电脑只有 WiFi 和网线两种连接方式，WiFi 信号一般不差，宽带网速也不慢。
* 弱网测试可以作为 APP 的专项测试，Charles 也可以连接手机做弱网测试。
* APP 在手机端使用，会遇到电梯、地铁等场景下 2G、3G、4G、5G 以及 WiFi 信号不好的情况。



* **2.3 如何使用弱网测试**
* 弱网测试不在小乌龟图标处设置，而是在代理里边的节流阀设置。
* 操作步骤：

1. 打开节流阀设置，若不勾选“启动仅适用于选定的 host”，则所有网站节流都会慢。
2. 勾选“启动仅适用于选定的 host”后，添加 at tps 请求，只校验包含 百度 的地址（如 \*百度.\*），端口不用填。
3. 节流阀预设可以设置带宽利用率、延迟、MTU、可靠性和稳定性，主要影响网络的是带宽下载和上传的带宽。
4. 节流阀设置有 56 KB、256 KB、512 KB、两兆、8M、16M、32M、100M（推测）、2G、3G、4G 等选项。

若想模拟其他弱网，可修改下载和上传的速率（以 KB 为单位）。例如，模拟 2G 网，上传为 200 KB，下载速度为 15 到 20k。

**3. 使用手机进行抓包测试**

1. **3.1 操作步骤（以苹果电脑连接苹果手机为例）**
2. **在 JMeter 中设置**：

* 在帮助里边找到 sis sl 的代理，选择在移动设备或远程浏览器安装 Charles 跟证书。
* 弹出窗口后，将设备的配置为 Charles 作为 HTTP 代理，使用本地 IP 地址（如 192.168.101.5），端口为四个八。
* 浏览 Charles 的地址，下载并安装证书。

1. **在手机端设置**：

* 选择无线局域网，进入网络链接，选择配置代理，选择手动。
* 输入电脑的 IP 地址和端口（如 192.168.101.5 和 8888），注意手机和电脑的网络要在同一个局域网下。
* 保存设置后，手机可能没网，此时返回浏览器，输入 Charles 的地址（如 https://Charles.pro/ssl），点击前往，允许下载描述文件，下载完成后在设置中安装描述文件。



1. **抓包测试**：

* 安装证书后，就可以抓手机的包了。例如，在手机上操作某个 APP（如 csdn），在 JMeter 中可以看到抓到的包。

1. **还原网络**：

* 若要还原网络，将 WIFI 代理改成自动，网络恢复正常。

**3.2 操作步骤总结**

1. 手机和电脑要在一个局域网下，使用一个网络。
2. 在帮助 sl 代理里边，在移动设备安装 charts 证书。
3. 设置手机 WiFi 代理为手动，修改地址和端口，和第二步显示的地址一样。
4. 输入 https://Charles.pro/ssl 地址，下载证书。
5. 打开想要抓包的 APP 进行抓包。
6. 还原网络时，将 WIFI 代理改成自动。

**4. mock 介绍**

**4.1 Mock 的作用**

* 在工作中，前台和后台可能由不同的人开发，效率不同。若前台开发好，后台没有开发完，但又需要校验一些数据和特别场景，且需要使用到后台，这时就可以使用 Mock 模拟后台返回的结果。

**4.2 Mock 的方式**

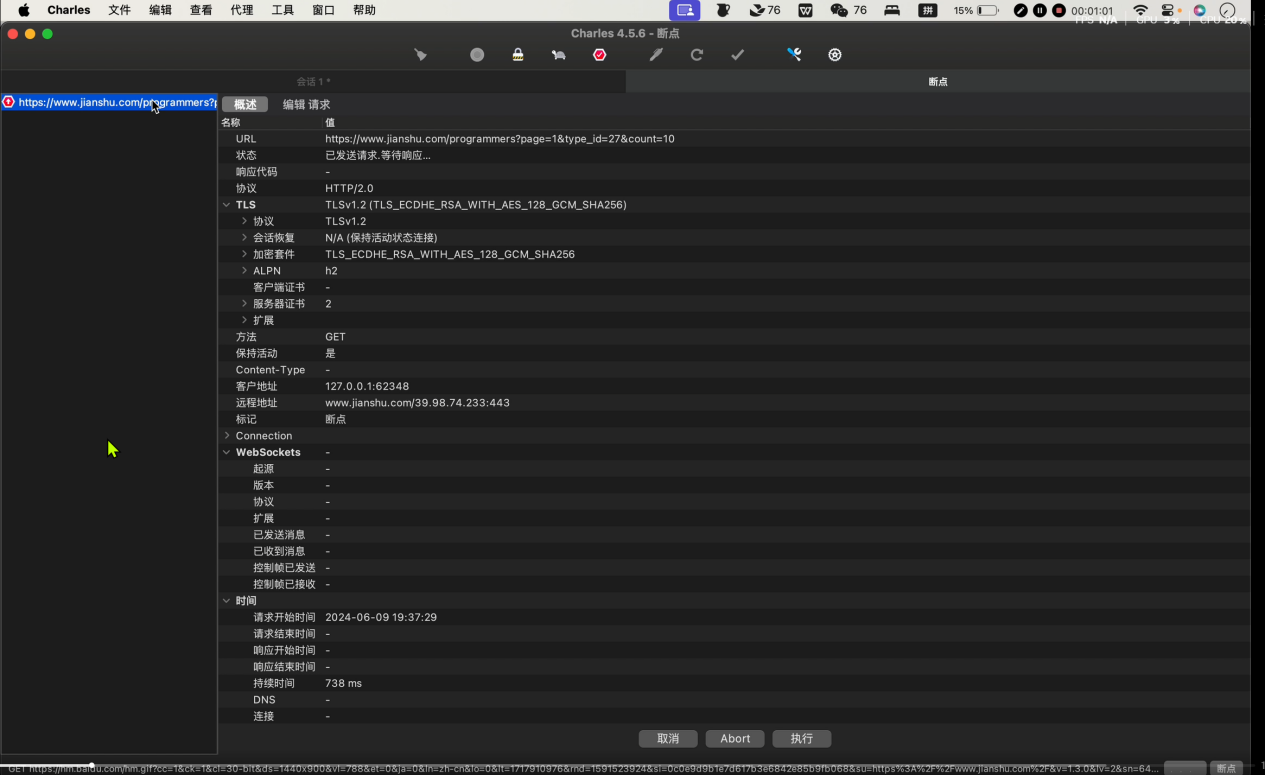
* 可以使用本地映射和远程映射，但主要关心打断点的方式。

**4.3 操作提示**

打开 JMeter 的断点设置，里面有启用断点和在什么地方打断点的选项。

**6-3 抓包断点及本远地映射**

断点、本地映射和远程映射



**1. 断点**

首先，演示一下断点的使用。

1. 打开简书网站，进入 IT 技术页面。
2. 使用 Charles 拦截简书的接口（这里我过滤了只有简书的接口）。
3. 右击选择断点，启用断点。

启用断点后，再次刷新页面，会发现页面好像网络不太好，一直在转圈。这是因为 Charles 拦截了请求，没有返回数据。

* **修改请求**：可以编辑请求，比如将一页显示多少条改成一条，然后执行。会发现页面上只显示了一条数据。
* **修改返回**：再次刷新，Charles 会重新拦截请求。默认执行后，会弹出一个响应数据。可以编辑响应，比如在 JSON 文本里修改响应结果，然后执行。会发现页面上的返回结果变成了修改后的内容。

断点有个缺点，就是它是临时修改。如果去掉断点，刷新页面后，数据就会恢复原样。再次启用断点，需要重新修改请求和响应。

**2. 本地映射**

接下来，演示一下本地映射的使用。

1. 清除之前的断点。
2. 再次请求接口，启用断点。
3. 修改返回，选择本地映射，然后执行。
4. 右击选择保存响应，给文件起个名字，比如“本地一”，然后保存。
5. 再次选择这个接口，右击选择本地映射，选择刚刚保存的文件“本地一”。

这个时候，无论怎么刷新页面，返回的都是本地映射的内容。只要 Charles 不关，它就会一直是本地映射。如果修改了本地文件，刷新页面后，返回的内容也会相应改变。

**如何关闭本地映射**：

* 在工具里选择本地映射，然后选择移除。
* 把上面的勾号去掉，再次确定。
* 会发现再次右击这个接口，本地映射就没了。

**3. 远程映射**

最后，演示一下远程映射的使用。

远程映射有点特殊，不能使用之前的接口来做演示。假设有一个互联网地址，比如“九六e4c...”，想要让它返回“林志玲婚变”的内容。

1. 打开 Charles，在代理里边把 macOS 的代理打开。
2. 刷新页面，拿到“九六e4c...”的返回数据。
3. 右击选择远程映射，把“林志玲婚变”的地址复制到主机里边。
4. Charles 会自动识别协议，点击确定。
5. 再次刷新“九六e4c...”的接口，会发现它变成了“林志玲婚变”的内容。

这就是远程映射的作用，把另外一个地址的数据映射到这个地址里去使用。

**总结**

* **断点**：临时修改请求和响应，用于检查数据是否正确。
* **本地映射**：将接口的返回映射到本地文件，实现持久化修改。
* **远程映射**：将一个地址的数据映射到另一个地址，实现数据的灵活使用。

Charles 还有其他功能，比如压力测试等，但在工作中主要用到的是这三种功能。

**7-1 ADB安装**

ADB是一个客户端到服务器的一个程序端，起到一个调试的作用。

* **客户端**：其实指的是的操作电脑。
* **服务端**：其实指的就是安卓的这样的一个设备。

大家可以把ADB理解成它是一个手机和电脑之间上传或者下载文件的工具，也可以去把它理解成可以对设备的应用去进行卸载、安装、更新的工具。当然，也可以把ADB理解成当的APP遇到一些bug的时候，可以通过ADB来抓取的日志。

**ADB是谷歌官方出品的，用来控制安卓手机的工具。**

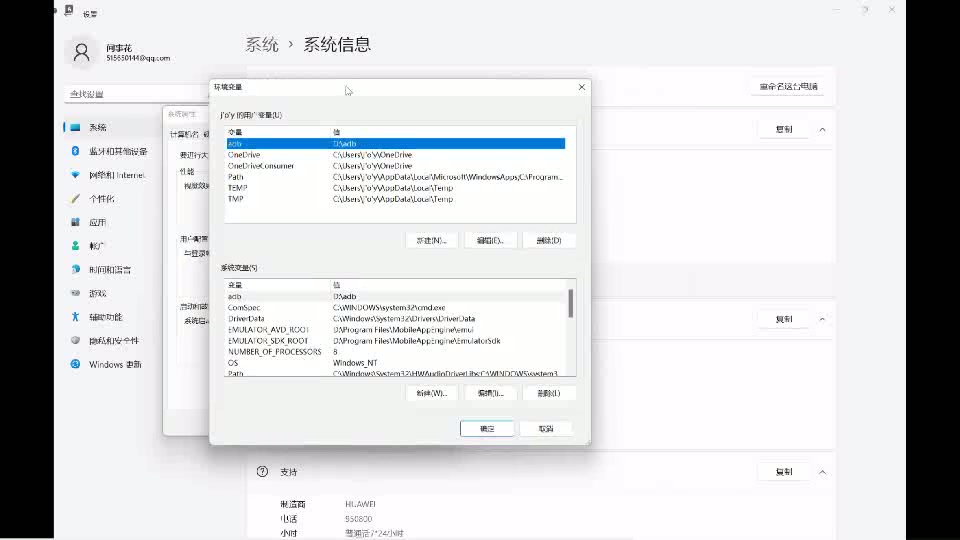
**1.1 ADB的作用**

ADB除了可以用来打印日志、定位bug之外，还可以做的一个稳定性测试。除了这两个之外，还可以运行这个设备的一些命令。

ADB是一个桥梁，它可以在计算机和设备之间上传、下载的文件。同时，也可以对的设备应用通过电脑进行卸载、安装、更新等操作。

ADB的功能非常的丰富且强大，包括也可以用ADB去抓包，去改一些权限之类的，等等的都可以。

**2. ADB的安装配置**



**7-2 ADB命令及稳定性测试**

**1. ADB 安装配置**

* **安装步骤**：
* 将压缩包解压到对应根目录下
* 配置环境变量
* 使用 adb -v 命令检查是否安装成功

|  |
| --- |
| Bash adb -v |

成功后会显示版本号和安装路径

**2. 连接手机**

* **准备工作**：
* 安卓手机和数据线
* 苹果手机无法使用 ADB
* 连接电脑和手机
* **USB 连接模式**：
* 选择充电、传输照片等模式均可
* **开启开发者模式**：

1. 进入设置 -> 关于手机 -> 版本信息
2. 连续点击软件版本号 5 次以上
3. 返回设置 -> 系统管理 -> 开发者选项

开启 USB 调试

**3. ADB 基本命令**

**3.1 文件传输**

* **从电脑上传文件到手机**：

|  |
| --- |
| Bash adb push <电脑文件路径> <手机目标路径> |

* 示例：

|  |
| --- |
| Bash adb push C:\adb1.txt /sdcard/test/ |

* **从手机下载文件到电脑**：

|  |
| --- |
| Bash adb pull <手机文件路径> <电脑目标路径> |

* 示例：

|  |
| --- |
| Bash adb pull /sdcard/test/adb1.txt D:\ |

**3.2 安装/卸载 APP**

* **安装 APK**：

|  |
| --- |
| Bash adb install <APK文件路径> |

* 示例：

|  |
| --- |
| Bash adb install C:\youtiaotiao.apk |

* **卸载 APP**：

|  |
| --- |
| Bash adb uninstall <包名> |

* 示例：

|  |
| --- |
| Bash adb uninstall com.whatsbug.youtiaotiao |

**4. 高级命令**

**4.1 查看已安装应用**

* **查看所有应用包名**：

|  |
| --- |
| Bash adb shell pm list packages |

* **查看第三方应用包名**：

|  |
| --- |
| Bash adb shell pm list packages -3 |

**4.2 日志管理**

* **查看所有日志**：

|  |
| --- |
| Bash adb logcat |

* **查看特定应用日志**：

1. 获取进程号：

|  |
| --- |
| Bash adb shell ps -A | grep <包名> |

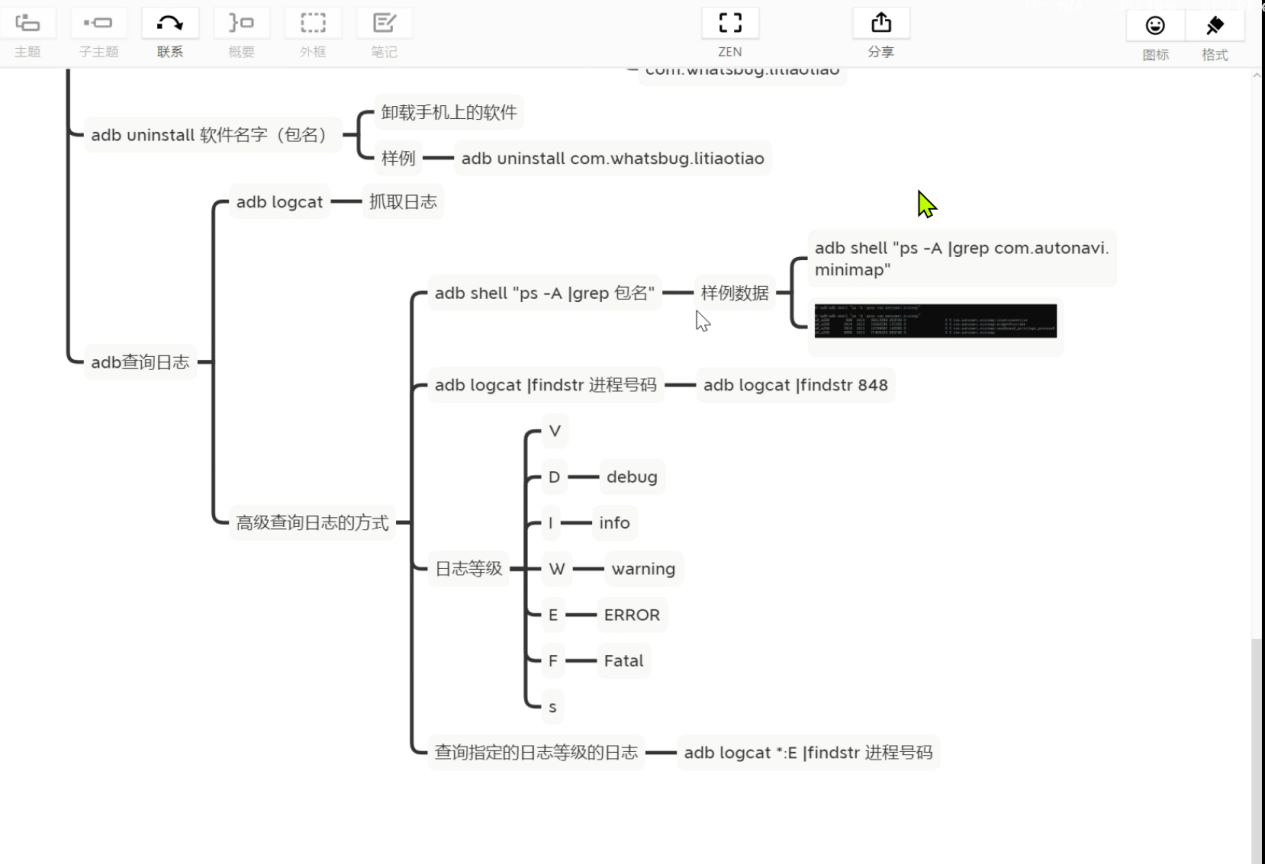
1. 查看日志：

|  |
| --- |
| Bash adb logcat | grep <进程号> |

* **按日志级别查看**：

|  |
| --- |
| Bash adb logcat \*:级别 <进程号> |

* 级别：V(Verbose)、D(Debug)、I(Info)、W(Warn)、E(Error)、F(Fatal)



**4.3 稳定性测试（Monkey）**

* **基本测试**：

|  |
| --- |
| Bash adb shell monkey -p <包名> -v -v -v <次数> |

* 示例：

|  |
| --- |
| Bash adb shell monkey -p com.autonavi.minimap -v -v -v 200 |

* **设置点击间隔**：

|  |
| --- |
| Bash adb shell monkey -p <包名> --throttle <毫秒> -v -v -v <次数> |

* 示例：

|  |
| --- |
| Bash adb shell monkey -p com.autonavi.minimap --throttle 300 -v -v -v 1000 |

**5. 系统信息查看**

**5.1 电池状态**

|  |
| --- |
| Bash adb shell dumpsys battery |

**5.2 性能监控**

|  |
| --- |
| Bash adb shell top |

按 Ctrl+C 退出

**6. 示例命令总结**

|  |  |
| --- | --- |
| 操作 | 命令 |
| 查看版本 | adb -v |
| 上传文件 | adb push <电脑路径> <手机路径> |
| 下载文件 | adb pull <手机路径> <电脑路径> |
| 安装 APK | adb install <APK路径> |
| 卸载 APP | adb uninstall <包名> |
| 查看所有应用 | adb shell pm list packages |
| 查看第三方应用 | adb shell pm list packages -3 |
| 查看日志 | adb logcat |
| 查看特定应用日志 | adb logcat | grep <进程号> |
| 按级别查看日志 | adb logcat \*:级别 <进程号> |
| Monkey 测试 | adb shell monkey -p <包名> -v -v -v <次数> |
| 设置点击间隔 | adb shell monkey -p <包名> --throttle <毫秒> -v -v -v <次数> |
| 查看电池状态 | adb shell dumpsys battery |
| 性能监控 | adb shell top |

**8-1JMeter工具基本熟悉**

本节重点为JMeter的安装配置与基础功能演示。

**二、环境准备**

1. **系统适配说明**

* Windows用户可直接跟随操作

1. Mac用户需自行查阅安装教程（推荐参考知乎专栏《测试网站》中的JMeter安装指南）
2. **JDK安装要点**

* 版本选择：根据系统架构下载x86/x64版本
* 安装路径：建议解压至C盘根目录
* 环境变量配置：
* 新建系统变量JAVA\_HOME指向JDK路径
* 修改Path变量追加;%JAVA\_HOME%\bin

1. 验证命令：java -version
2. **JMeter安装**

* 推荐使用最新版（示例使用5.4.1）
* 解压至C盘并配置环境变量：
* JMETER\_HOME指向JMeter根目录

更新Path变量追加;%JMETER\_HOME%\bin

**三、启动方式**

1. **命令行启动**

|  |
| --- |
| Plain Text jmeter |

1. **批处理文件启动**

* 双击bin/jmeter.bat文件

特别说明：Linux系统需通过命令行启动，这是无界面环境的必要操作

**四、界面汉化**

1. 路径：Options → Choose Language → 中文(简体)

提示：建议首次使用后保持默认语言设置

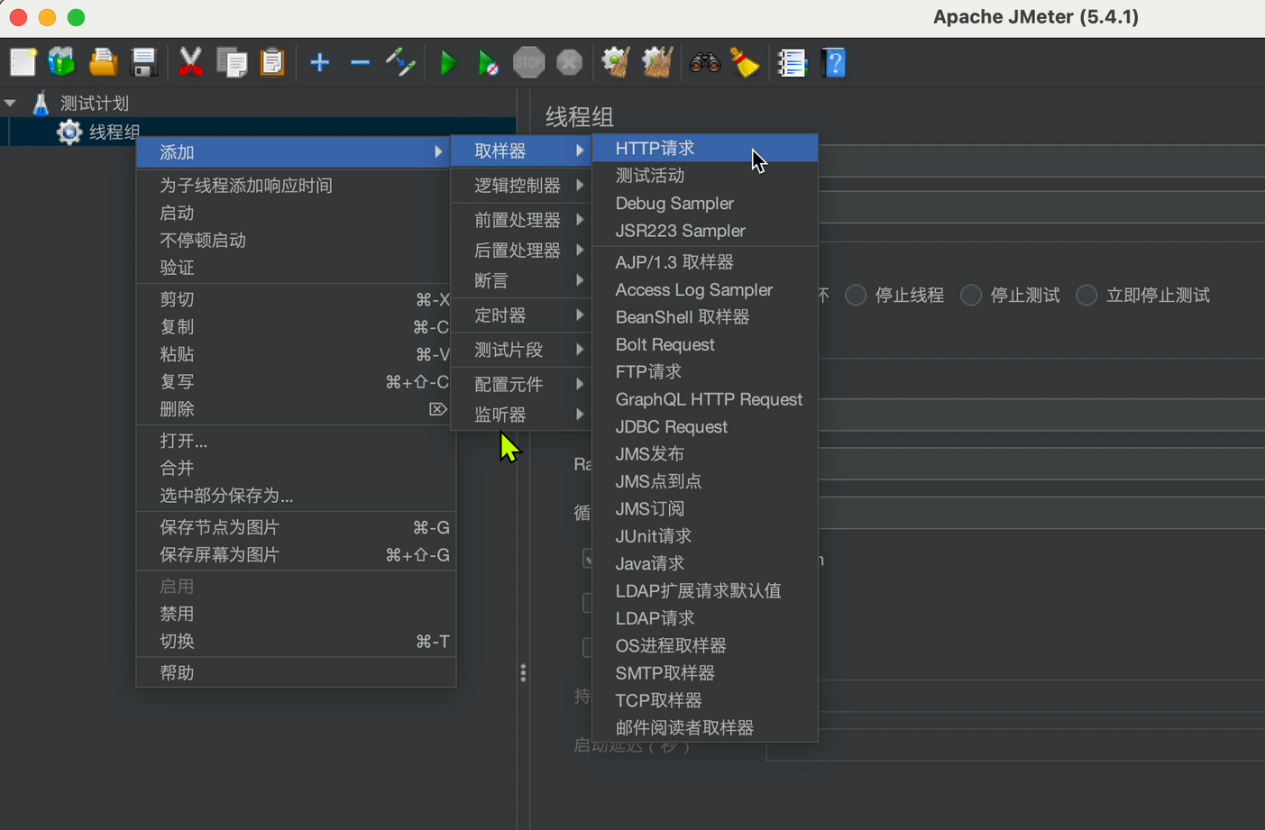
**五、基础测试示例**

1. **测试计划创建**

* 右键测试计划 → 添加 → 线程组
* 关键参数：
* 线程数：并发用户数（示例设为5）
* Ramp-Up时间：用户启动间隔

1. 循环次数：测试执行轮次
2. **HTTP请求配置**

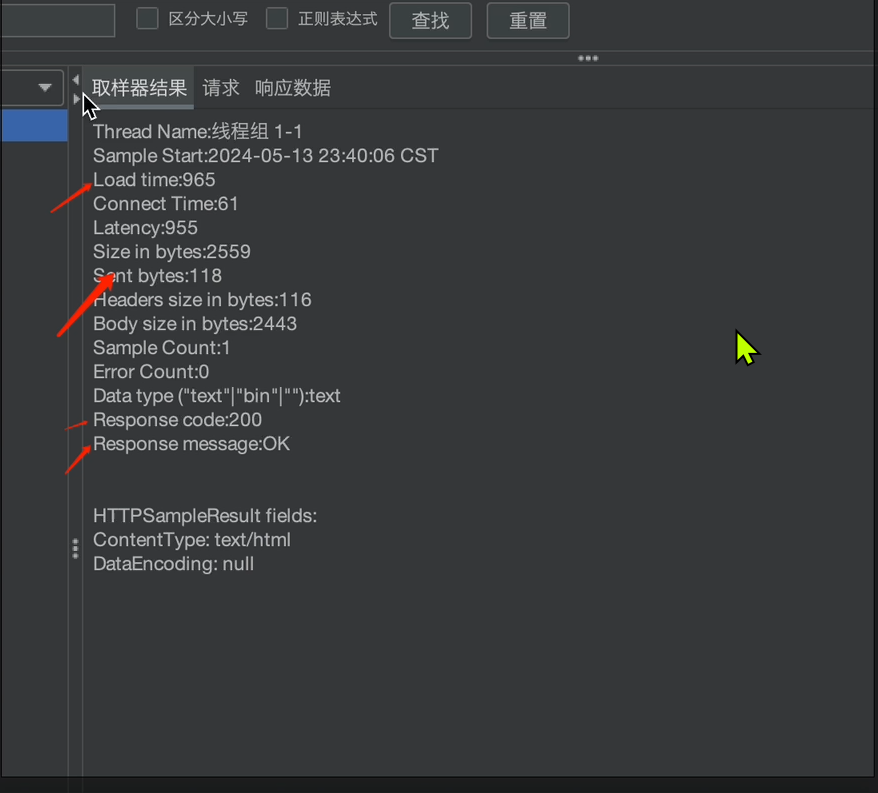
* 右键线程组 → 添加 → 取样器 → HTTP请求
* 参数设置：
* 协议：https（根据实际接口）
* 服务器：www.baidu.com
* 端口：默认80（HTTPS为443）
* 方法：GET
* 路径：/



1. **结果验证**

* 添加监听器：查看结果树
* 运行测试：点击绿色启动按钮
* 结果分析：
* 状态码200表示成功
* 响应时间反映接口性能

响应数据可查看原始返回内容



**六、进阶功能演示**

1. **多线程组控制**

* 独立运行模式：勾选"独立运行每个线程组"实现顺序执行
* 特殊线程组：
* setUp Thread Group：测试前置操作

1. tearDown Thread Group：测试后置清理
2. **中文乱码解决**

* 方法一：请求头添加Content-Type: application/json;charset=utf-8
* 方法二：后置处理器添加BeanShell脚本：

|  |
| --- |
| Java prev.setDataEncoding("UTF-8"); |

**七、企业微信接口测试示例**

1. 测试准备：

* 获取企业微信API文档

1. 准备测试账号与接口权限
2. 请求配置：

* 协议：https
* 服务器：qyapi.weixin.qq.com
* 路径：根据具体API（如/cgi-bin/gettoken）
* 参数传递：
* Query参数：corpid=XXX&corpsecret=YYY

1. Body参数（POST请求时）
2. 认证机制：

* 获取Access Token的流程演示

Token的缓存与复用策略

**八、面试高频考点**

1. 线程组核心参数：

* 线程数 vs 循环次数

1. Ramp-Up时间的计算方法
2. 分布式测试原理：

* Master-Slave架构

1. 负载生成器的配置要点
2. 性能监控指标：

* TPS/QPS的计算
* 响应时间分布分析
* 错误率监控

**8-2 JMeter接口请求**

* 实践企业微信接口调用
* 学习JMeter基础操作（GET/POST请求、参数传递、结果分析）

**一、企业微信接口测试流程**

**1. 获取Token接口**

* **步骤1：打开API文档**
* 路径：企业内部开发 → 服务端API → 通讯录管理
* 关键接口：获取Token（/gettoken）
* **步骤2：JMeter配置**
* **HTTP请求设置**
* 协议：HTTPS
* 服务器名称/IP：api.weixin.qq.com
* 路径：/cgi-bin/gettoken
* **参数传递**
* 企业ID（corpid）：从“我的企业”信息中获取
* 密钥（corpsecret）：应用管理→自建应用→查看密钥
* **步骤3：调试与结果验证**
* 常见问题：空格导致请求失败
* 成功响应：返回access\_token

**2. 网易外协工作台接口测试**

* **步骤1：注册与登录**
* 使用网易邮箱注册账号
* **步骤2：接口分析**
* 功能：搜索接口（GET请求）
* 关键参数：
* keywords：搜索关键词
* page：分页参数
* **步骤3：JMeter配置**
* **HTTP请求设置**
* 协议：HTTPS
* 服务器名称/IP：目标域名（如api.example.com）
* 路径：/api/search
* **参数传递**
* 通过“添加参数”按钮配置keywords、page等
* **Session管理**
* 问题：Session过期导致请求失败
* 解决方案：添加HTTP信息头管理器，携带Cookie

**三、JMeter核心功能详解**

**1. 请求类型与配置**

* **GET请求**
* 参数通过URL传递（问号后）
* **POST请求**
* 示例：企业微信创建部门接口
* 关键配置：
* 协议：HTTPS
* 路径：/cgi-bin/department/create
* 消息体数据：JSON格式（如{"name":"部门名称","parentid":1}）

**2. 参数化与动态数据**

* **企业ID与密钥**
* 从企业微信后台复制粘贴
* **Token复用**
* 通过变量引用避免硬编码

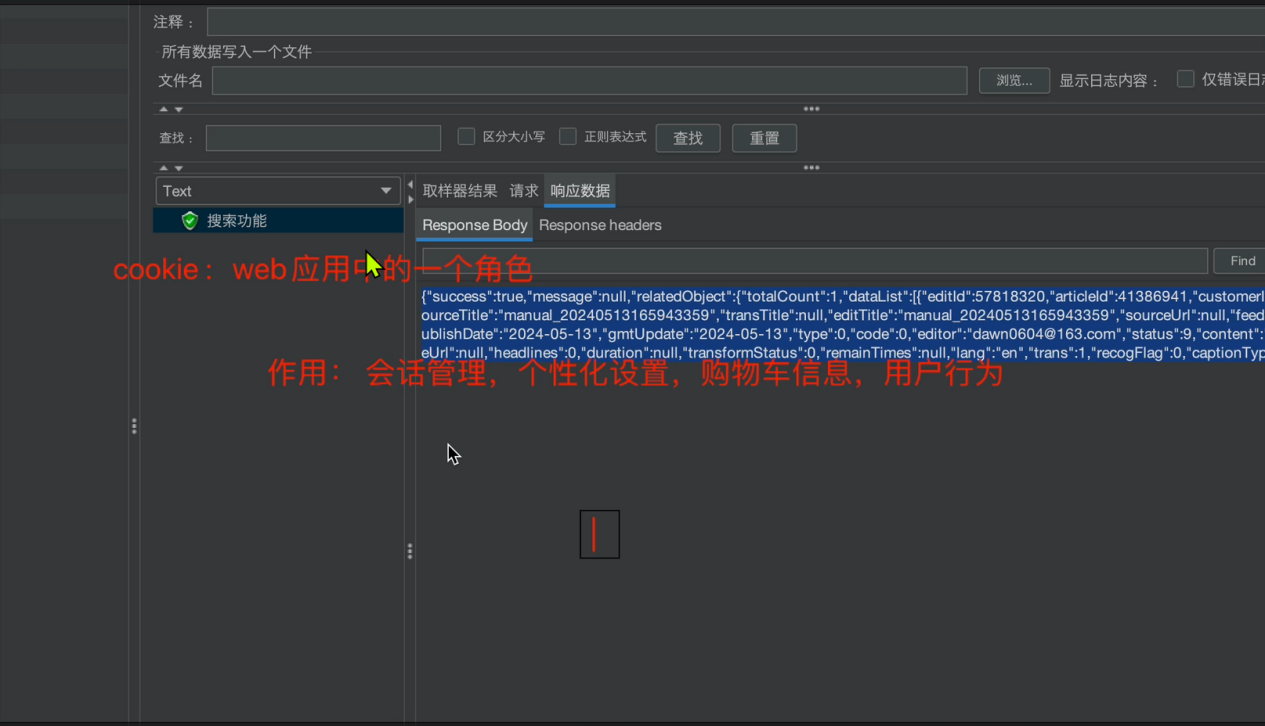
**3. 结果分析与调试**

* **常见错误**
* 空格导致参数解析失败
* IP白名单限制（需在企业微信后台配置可信IP）
* **调试技巧**
* 使用“查看结果树”监听器
* 检查请求与响应

**四、Cookie与会话管理**

**1. Cookie基础概念**

* **作用**
* 会话管理：保持用户登录状态
* 个性化设置：存储用户偏好（如语言、布局）
* 购物车信息：电商网站常用
* 用户行为分析：记录用户操作习惯



**2. JMeter中的Cookie处理**

* **HTTP信息头管理器**
* 添加Cookie头，值从浏览器开发者工具复制
* **示例：网易接口会话保持**
* 登录后复制Cookie到JMeter请求头

**五、线程组与并发控制**

**1. 线程组基础**

* **作用**
* 模拟多用户并发请求
* **关键配置**
* 线程数：并发用户量
* 循环次数：请求执行频率

**2. 高级配置（进阶）**

* **定时器**
* 固定定时器：控制请求间隔
* **断言**

响应断言：验证返回结果是否符合预期

**8-3 JMeter并发及报告**

**一、线程组核心配置**

**1. 基础参数**

* **线程数**
* 并发用户量（如设置为5表示模拟5个用户）
* 压测机性能限制：需根据本地电脑配置调整
* **循环次数**
* 每个线程的重复请求次数（如设置为2表示每个用户请求2次）
* 总请求量 = 线程数 × 循环次数

**2. 高级调度**

* **启动时间（Ramp-Up Period）**
* 线程启动间隔（如设置为3秒表示3秒内逐步启动所有线程）
* 避免瞬间高并发导致压测机崩溃
* **调度器（Scheduler）**
* **启动延迟（Startup Delay）**：延迟多久开始执行
* **持续时间（Duration）**：测试总时长（单位：秒）

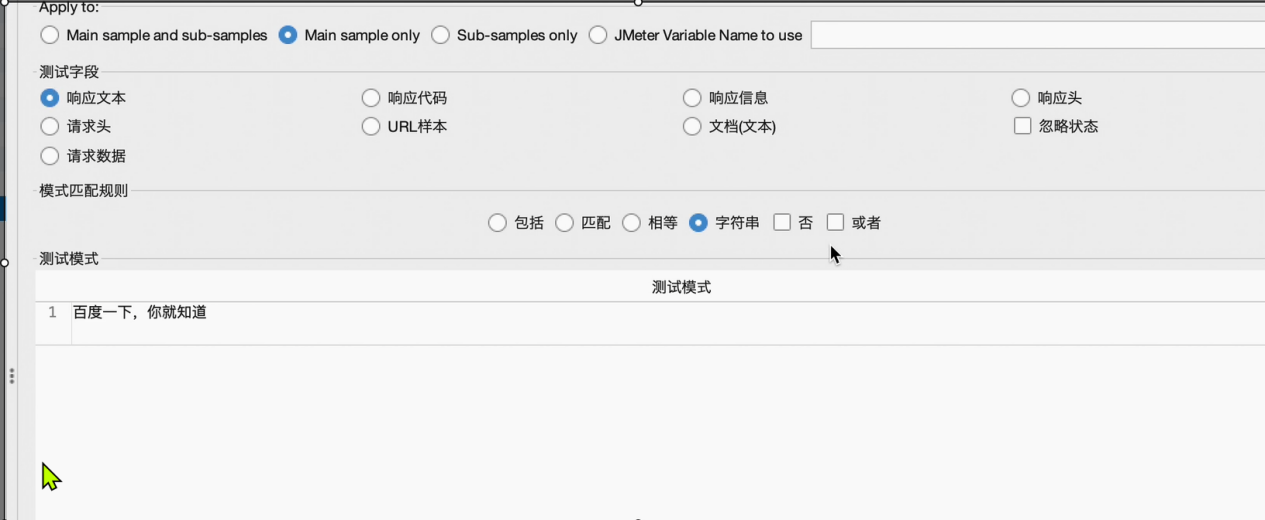
**3. 特殊配置**

* **延迟创建线程（Delayed Thread Creation）**
* 仅在需要时创建线程（减少资源占用）
* **独立运行每个线程（Independent Thread）**
* 确保线程按顺序执行（默认勾选）

**二、断言（Assertions）**

**1. 响应断言（Response Assertion）**

* **断言范围**
* **响应文本**：验证返回内容（如包含“百度一下”）
* **响应代码**：验证HTTP状态码（如200表示成功）
* **响应头**：检查特定Header（如Content-Type）



* **匹配规则**
* **包含（Contains）**：模糊匹配
* **匹配（Equals）**：精确匹配
* **自定义失败消息**：断言失败时返回提示（如“测试失败！”）

**2. 实际应用示例**

* **场景**：验证淘宝接口返回状态码是否为200
* 添加断言 → 选择“响应代码” → 输入200 → 运行测试

**三、聚合报告（Aggregate Report）**

**1. 关键指标解析**

* **样本（Samples）**
* 总请求数（如线程数100 × 循环次数1 = 100个样本）
* **平均值（Average）**
* 所有请求的平均响应时间（单位：毫秒）
* **中位数（Median）**
* 50%请求的响应时间低于该值
* **90%/95%/99% Line**
* 对应百分比请求的响应时间上限（如90%请求在500ms内完成）
* **异常率（Error %）**
* 失败请求占比（需重点关注）
* **吞吐量（Throughput）**
* 每秒处理请求数（评估系统负载能力）

**2. 性能分析技巧**

* **异常率 >0%**：检查接口稳定性或网络问题
* **99% Line过高**：优化慢查询或缓存策略

**四、HTTP请求默认值（HTTP Request Defaults）**

**1. 作用**

* 统一设置公共参数（如协议、域名、端口），避免重复配置

**2. 配置示例**

* **协议**：HTTPS
* **服务器名称**：api.weixin.qq.com
* **路径**：留空（后续每个请求单独填写）

**五、定时器（Timers）**

**1. 固定定时器（Constant Timer）**

* **作用**：模拟用户固定思考时间（如每次请求间隔3秒）
* **场景**：模拟用户浏览页面后的点击行为

**2. 随机定时器（Uniform Random Timer）**

* **作用**：在指定范围内随机延迟（如0-3秒）
* **参数**
* **随机延迟最大值（Random Delay Maximum）**：3000ms
* **固定延迟偏移量（Constant Delay Offset）**：1000ms
* **总延迟范围**：1000ms ~ 4000ms
* **场景**：模拟真实用户操作差异（如手速快慢）

**六、测试流程示例**

**1. 淘宝搜索接口测试**

1. **线程组配置**

* 线程数：10
* 循环次数：5

1. **添加定时器**

* 随机定时器：最大延迟3000ms

1. **添加断言**

* 验证响应文本包含“商品”

1. **运行测试**

查看聚合报告分析性能瓶颈

**8-4 JMeter提取器及控制器**

**一、同步定时器**

**1. 同步定时器概述**

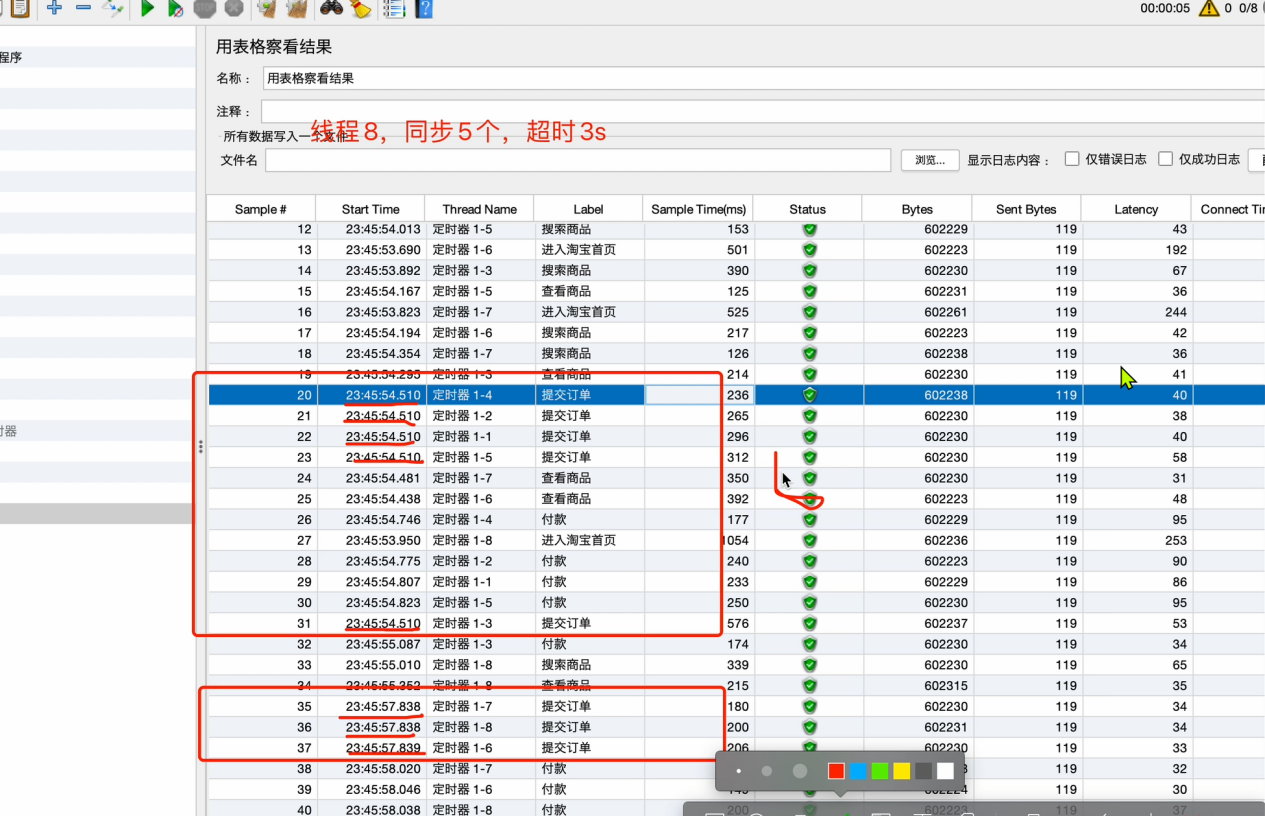
同步定时器（Synchronizing Timer）用于模拟多个用户同时执行某个操作，类似于水坝蓄水到一定高度后开闸放水。它在性能测试中非常重要，特别是在模拟高并发场景（如双11零点抢购）时。

**2. 同步定时器参数**

* **模拟用户组的数量**：指定同时运行的线程数。例如，设置为5，则同时运行5个线程。
* **超时时间**：
* 设置为0：表示无限等待，直到满足指定的用户组数量。
* 大于0：表示等待超时时间，如果超时时间内未满足指定的用户组数量，则剩余线程直接运行。

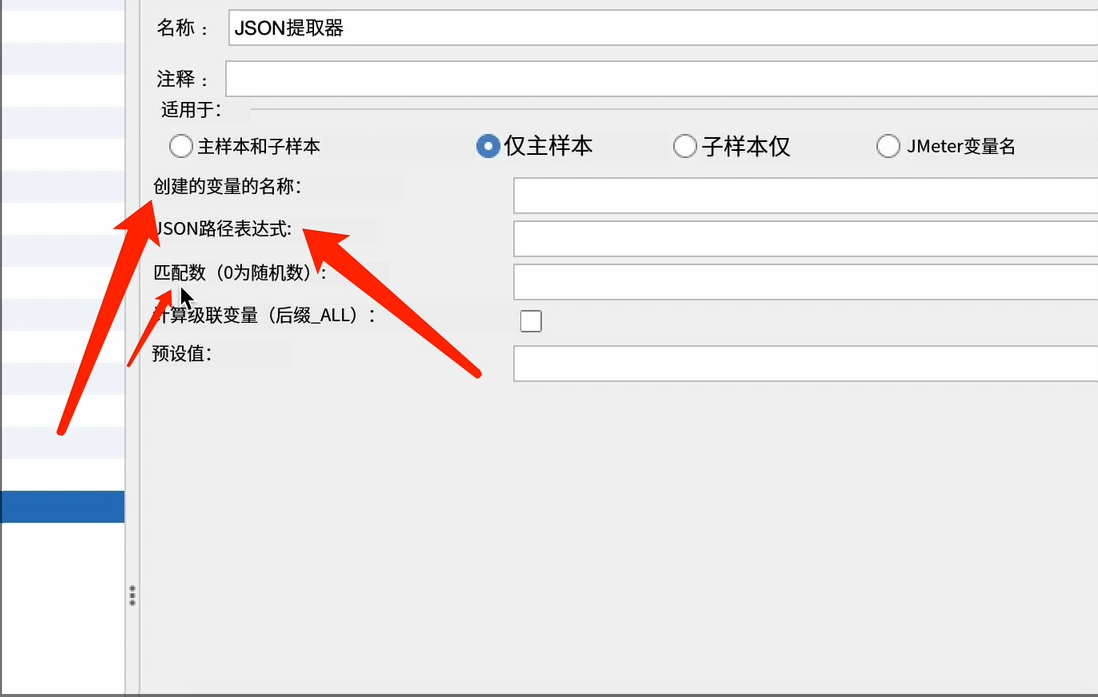
**3. 同步定时器示例**

* **场景设置**：
* 用户组数量：5
* 超时时间：3000毫秒（3秒）
* 线程组数量：8



* **操作步骤**：

1. 在JMeter中添加同步定时器。
2. 设置模拟用户组的数量为5。
3. 设置超时时间为3000毫秒。
4. 设置线程组数量为8。
5. 运行测试，观察结果。



* **结果分析**：
* 如果8个线程中能同时满足5个线程的条件，则这5个线程会同时运行。
* 剩余的3个线程会等待3秒，如果3秒内未满足5个线程的条件，则这3个线程直接运行。

**4. 同步定时器与水坝类比**

* **水坝蓄水**：水坝蓄水到一定高度后开闸放水。
* **同步定时器**：线程数达到指定数量后同时运行，类似于开闸放水。如果等待超时时间后仍未满足条件，则直接运行剩余线程，类似于水坝必须放水以避免下游干旱。

**二、JSON 提取器**

**1. JSON 提取器概述**

JSON 提取器用于从JSON格式的响应中提取特定字段的值，并将其存储为变量，以便在后续请求中使用。这在需要多次请求并传递动态参数（如token）的场景中非常有用。

**2. JSON 提取器参数**

* **变量名称**：为提取的字段值指定的变量名。
* **JSON表达式**：使用$.key格式指定要提取的字段。例如，$.token表示提取响应中的token字段。
* **匹配数**：
* 0：随机匹配。
* 1：匹配第一个。
* -1：匹配所有。
* **默认值**：如果未匹配到字段，则使用默认值。

**3. JSON 提取器示例**

* **场景设置**：
* 请求：获取token。
* 响应：JSON格式，包含token字段。
* **操作步骤**：

1. 在获取token的请求后添加JSON提取器。
2. 设置变量名称为token。
3. 设置JSON表达式为$.token。
4. 设置匹配数为1。
5. 设置默认值为NOT\_FOUND。

* **使用提取的变量**：
* 在后续请求中，使用${token}引用提取的token值。

**4. 调试取样器**

* **作用**：用于验证提取的变量是否正确。
* **操作步骤**：

1. 添加调试取样器。
2. 运行测试，查看调试取样器的响应数据，确认提取的变量值是否正确。

**三、CSV 数据文件设置**

**1. CSV 数据文件设置概述**

CSV 数据文件设置用于从CSV文件中读取数据，并将其作为变量传递给测试计划中的请求。这在需要模拟多个用户使用不同数据进行测试的场景中非常有用。

**2. CSV 数据文件设置参数**

* **文件名**：指定CSV文件的路径。
* **文件编码**：指定文件的编码格式，如UTF-8。
* **变量名称**：为CSV文件中的每一列指定变量名。
* **分隔符**：指定CSV文件中的分隔符，默认为英文逗号。
* **忽略首行**：如果CSV文件包含标题行，则设置为true以忽略首行。
* **遇到文件结束符后的行为**：
* True：循环读取文件。
* False：停止线程。
* **线程共享模式**：
* 所有线程：CSV文件适用于整个测试计划。
* 当前线程组：CSV文件仅适用于当前线程组。
* 当前线程：CSV文件仅适用于当前线程。

**3. CSV 数据文件设置示例**

* **场景设置**：
* CSV文件内容：

|  |
| --- |
| Plain Text name,pwd aaa111,password1 bbb222,password2 ccc333,password3 ddd444,password4 |

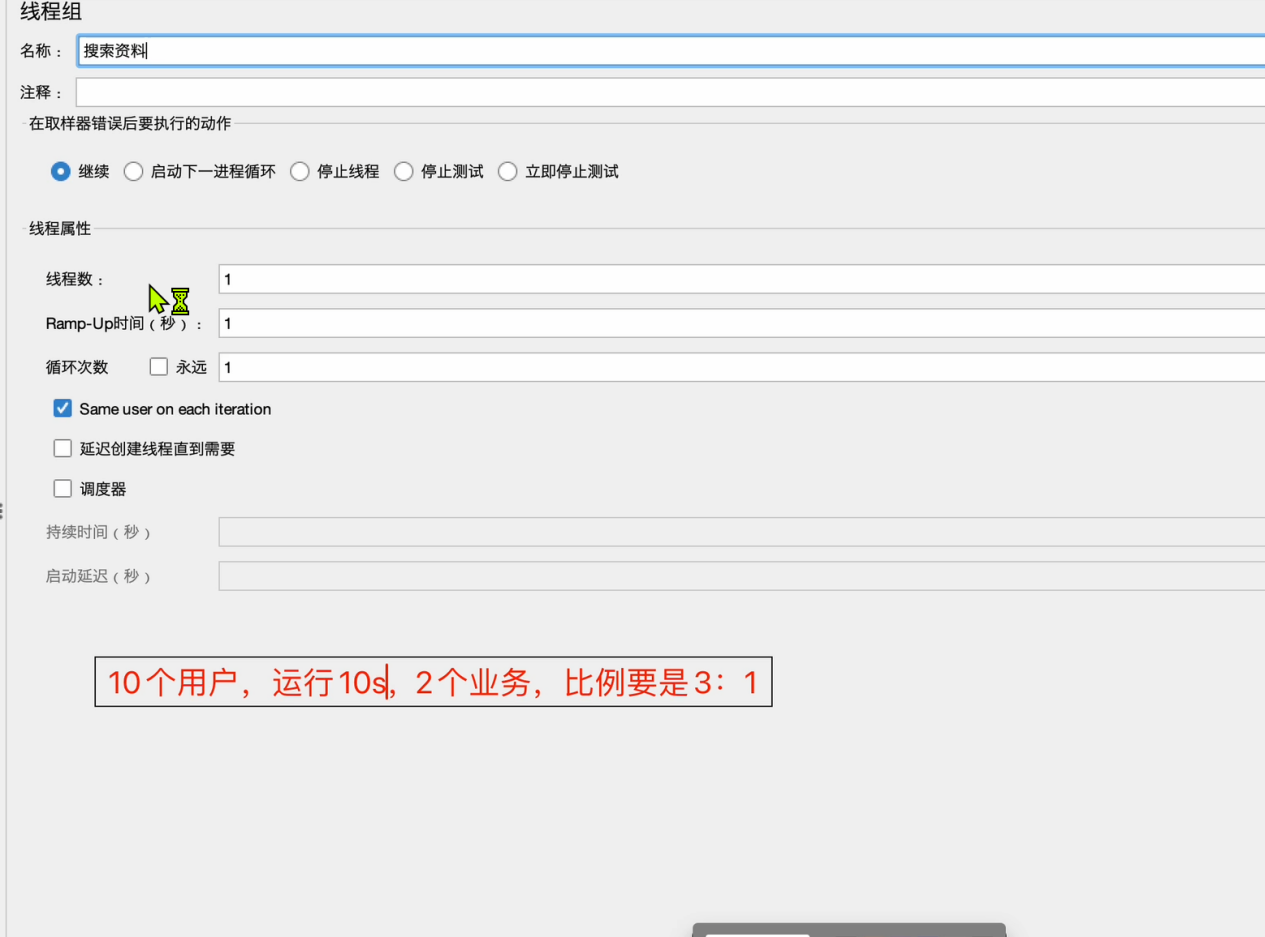
* 请求：登录请求，需要用户名和密码。
* **操作步骤**：

1. 添加CSV 数据文件设置。
2. 设置文件名为CSV文件的路径。
3. 设置文件编码为UTF-8。
4. 设置变量名称为name和pwd。
5. 设置分隔符为英文逗号。
6. 设置忽略首行为false。
7. 设置遇到文件结束符后的行为为true（循环读取）。
8. 设置线程共享模式为所有线程。

* **使用CSV文件中的变量**：
* 在登录请求中，使用${name}和${pwd}引用CSV文件中的用户名和密码。

**4. CSV 数据文件设置与线程组结合**

* **场景设置**：
* 线程组数量：5
* CSV文件数据：4条



* **结果分析**：
* 如果设置为循环读取，则5个线程会依次使用CSV文件中的4条数据，第5个线程会再次使用第1条数据。
* 如果设置为停止线程，则第5个线程会因无数据而报错。

**四、模拟场景示例**

**1. 场景描述**

模拟10个用户运行10秒，其中3个用户执行查看百度请求，7个用户执行搜索资料请求，比例为3:7。

**2. 操作步骤**

* **方法一：使用多个线程组**

1. 创建两个线程组：查看业务和编辑业务。
2. 在查看业务线程组中添加查看百度请求，设置线程数为3，循环次数为无限，持续时间为10秒。
3. 在编辑业务线程组中添加搜索资料请求，设置线程数为7，循环次数为无限，持续时间为10秒。
4. 运行测试，查看聚合报告。

* **方法二：使用IF控制器**

1. 创建一个线程组，设置线程数为10，循环次数为无限，持续时间为10秒。
2. 添加两个请求：查看百度和搜索资料。
3. 在查看百度请求前添加IF控制器，设置条件为${\_\_counter(false)} % 10 <= 2（模拟3:7的比例，这里使用计数器函数）。
4. 在搜索资料请求前添加IF控制器，设置条件为${\_\_counter(false)} % 10 > 2。
5. 运行测试，查看聚合报告。

**3. 结果分析**

* 方法一简单直接，但不够灵活，难以应对更复杂的比例设置。
* 方法二使用IF控制器和计数器函数，更加灵活，可以轻松应对各种比例设置。

**五、JMeter 常用工具总结**

1. **用户自定义变量**：

* 用于定义测试计划中使用的常量变量。
* 在测试计划或线程组中添加用户自定义变量元件。

1. **独立运行每个线程组**：

* 控制线程组是否独立运行。
* 在测试计划中设置。

1. **线程组参数**：

* 线程数：模拟的用户数量。
* 循环次数：每个线程执行的次数。
* 调度器：设置测试的持续时间和启动延迟。

1. **请求参数**：

* 协议：HTTP或HTTPS。
* 服务器名称或IP：目标服务器的地址。
* 端口号：目标服务器的端口。
* 方法：GET或POST。
* 路径：请求的URL路径。
* 参数：GET请求的查询参数或POST请求的消息体数据。

1. **逻辑控制器**：

* IF控制器：根据条件执行请求。
* 随机控制器：随机执行子请求。
* 事务控制器：将多个请求组合成一个事务。

1. **前置处理器和后置处理器**：

* 前置处理器：在请求发送前执行的操作，如添加HTTP头。
* 后置处理器：在请求发送后执行的操作，如提取响应数据。

1. **断言**：

* 响应断言：检查响应代码和响应文本。
* 大小断言：检查响应数据的大小。
* 持续时间断言：检查响应时间。

1. **定时器**：

* 固定定时器：在请求之间添加固定的延迟。
* 随机定时器：在请求之间添加随机的延迟。
* 同步定时器：模拟多个用户同时执行操作。

1. **监听器**：

* 查看结果树：显示每个请求的详细信息。
* 聚合报告：显示测试的汇总统计信息。
* 图形结果：以图形方式显示测试结果。

1. **CSV 数据文件设置**：

* 从CSV文件中读取数据，并将其作为变量传递给请求。

**六、面试常见问题**

1. **同步定时器的理解**：

* 同步定时器用于模拟多个用户同时执行某个操作，类似于水坝蓄水到一定高度后开闸放水。它在性能测试中非常重要，特别是在模拟高并发场景时。

1. **线程组参数的解释**：

* 线程数：模拟的用户数量。
* 循环次数：每个线程执行的次数。
* 调度器：设置测试的持续时间和启动延迟。

1. **如何模拟多个用户使用不同数据进行测试**：

* 使用CSV 数据文件设置，从CSV文件中读取数据，并将其作为变量传递给请求。

1. **JSON 提取器的作用**：

* JSON 提取器用于从JSON格式的响应中提取特定字段的值，并将其存储为变量，以便在后续请求中使用。

1. **如何判断接口的正确性**：

使用响应断言检查响应文本，确认接口返回的数据是否符合预期。响应代码返回200并不代表数据一定正确。

**8-5 JMeter文件上传及WebSocket**

**一、文件上传接口测试实战**

**1. 接口场景**

以"图片翻译"功能为例，演示如何通过JMeter模拟文件上传操作：

* **接口类型**：POST请求
* **关键参数**：文件路径、文件名称、请求头（Cookie）

**2. 操作步骤**

**（1）创建线程组**

* 右击"测试计划" → "添加" → "线程组"，命名为"文件上传测试"。

**（2）配置HTTP请求**

* 右击线程组 → "添加" → "取样器" → "HTTP请求"，命名为"文件上传接口"。
* **协议**：HTTP（或HTTPS，根据实际接口）
* **服务器名称或IP**：复制接口域名（如api.example.com）
* **路径**：接口路径（如/api/v1/upload）
* **方法**：POST

**（3）设置文件上传参数**

* 在"HTTP请求"的"文件上传"标签页中：
* **文件路径**：选择本地文件（如C:\test\image.png）
* **参数名称**：根据接口文档填写（如file）
* **MIME类型**：自动识别或手动填写（如image/png）

**（4）处理会话（Cookie）**

* 添加"HTTP信息头管理器"（右击线程组 → "添加" → "配置元件" → "HTTP信息头管理器"）：
* 添加键值对：Cookie: 你的Cookie值（从浏览器或前序请求中获取）。

**（5）验证结果**

* 添加"查看结果树"监听器（右击线程组 → "添加" → "监听器" → "查看结果树"）。
* 运行线程组，检查响应是否包含翻译结果或成功标识。

**3. 常见问题**

* **Session过期**：未传递有效Cookie导致，需确保Cookie未失效。
* **文件路径错误**：Windows需使用双反斜杠（C:\\test\\image.png）或正斜杠（C:/test/image.png）。

**二、WebSocket即时通信测试**

**1. WebSocket简介**

* **作用**：解决HTTP无状态、无连接的问题，实现服务端与客户端的双向实时通信（如聊天、实时推送）。
* **协议**：ws://（非加密）或wss://（加密，类似HTTP/HTTPS）。

**2. JMeter中WebSocket配置**

**（1）安装插件**

* 下载JMeter WebSocket插件（[插件地址](https://jmeter-plugins.org/install/Install/)）。
* 将插件文件（jmeter-websocket-samplers-x.x.jar）放入JMeter安装目录的lib/ext文件夹，重启JMeter。

**（2）配置WebSocket取样器**

* 右击线程组 → "添加" → "取样器" → "WebSocket Samplers" → "WebSocket Open Connection"。
* **服务器URL**：输入WebSocket测试地址（如ws://124.224.224.186:8800）。
* **协议**：选择ws或wss。

**（3）发送与接收消息**

* 添加"WebSocket Samplers" → "WebSocket Request"：
* **请求数据**：输入测试消息（如"Hello, WebSocket!"）。
* 添加"WebSocket Samplers" → "WebSocket Close Connection"（可选，用于关闭连接）。

**（4）验证结果**

* 运行线程组，通过"查看结果树"观察服务端返回的实时消息（如"你发送了：Hello, WebSocket!"）。

**3. 测试地址示例**

1. 公共测试地址：ws://echo.websocket.org（发送消息后服务端会原样返回）。
2. **文件上传关键点**：正确设置文件路径、参数名称，并传递有效Cookie。
3. **WebSocket核心**：理解其双向通信特性，掌握插件安装与基础配置。
4. **实际应用**：

* 文件上传常用于图片、视频上传功能测试。

WebSocket用于测试实时通信场景（如直播弹幕、股票行情推送）。

**8-6 JMeter数据库操作**

**一、课程目标**

掌握JMeter通过JDBC连接数据库的核心操作，包括：

1. **JDBC驱动安装与配置**
2. **数据库连接参数设置**
3. **执行SQL查询并验证结果**
4. **理解JDBC在性能测试中的应用场景**



**二、JMeter中配置JDBC连接**

**1. 创建线程组**

* 右击"测试计划" → "添加" → "线程组"，命名为"数据库测试线程组"。

**2. 配置JDBC连接**

* **步骤**：

1. 右击线程组 → "添加" → "配置元件" → "JDBC Connection Configuration"。
2. **关键参数设置**：

* **Variable Name for Created Pool**：连接池变量名（如mysql\_pool）。
* **Database URL**：格式为jdbc:mysql://<主机IP>:<端口>/<数据库名>?useSSL=false&serverTimezone=UTC（示例：jdbc:mysql://127.0.0.1:3306/test?useSSL=false&serverTimezone=UTC）。
* **JDBC Driver Class**：com.mysql.cj.jdbc.Driver（MySQL 8.0+）或com.mysql.jdbc.Driver（旧版本）。
* **Username**：数据库用户名（如root）。
* **Password**：数据库密码（如12345678）。

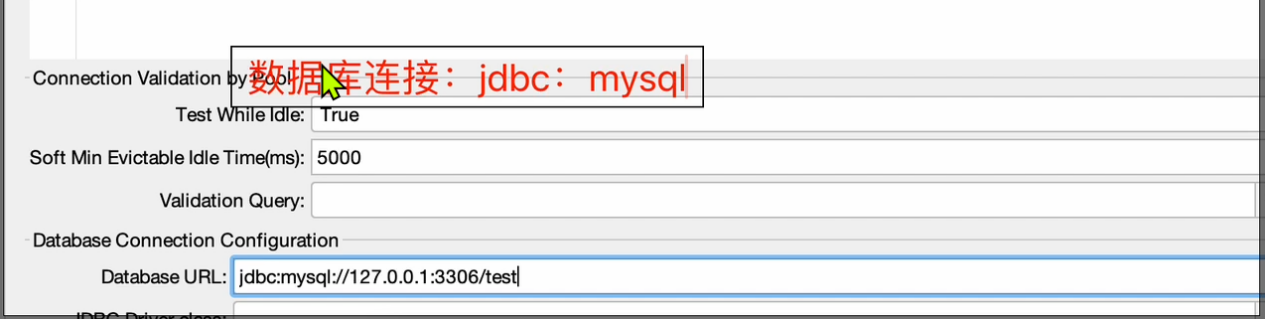
**四、执行SQL查询并验证结果**

**1. 添加JDBC请求**

* **步骤**：

1. 右击线程组 → "添加" → "取样器" → "JDBC Request"。
2. **关键参数设置**：

* **Variable Name**：引用连接池变量名（如mysql\_pool）。
* **Query Type**：选择Select Statement（查询语句）。
* **Query**：输入SQL语句（如SELECT \* FROM test;）。



**2. 验证结果**

* **步骤**：

1. 右击线程组 → "添加" → "监听器" → "查看结果树"。
2. 运行线程组，检查"响应数据"是否包含数据库查询结果（如test表中的王大拿数据）。

**五、常见问题与注意事项**

1. **驱动未找到**：

* 确认JDBC驱动JAR文件已正确放入lib/ext目录，并重启JMeter。

1. **连接失败**：

* 检查数据库URL格式（尤其是IP、端口、数据库名是否正确）。
* 确认数据库服务已启动，且防火墙未阻止连接。

1. **时区问题**：

* 在URL中添加serverTimezone=UTC参数（MySQL 8.0+必备）。

1. **SSL配置**：

* 本地测试可禁用SSL（useSSL=false），生产环境需配置正确SSL证书。

**六、扩展：JDBC在性能测试中的应用**

1. **数据验证**：

* 通过JDBC查询数据库，验证接口操作（如新增、修改）是否正确写入数据。

1. **性能分析**：

* 结合"聚合报告"监听器，分析数据库操作的响应时间、吞吐量等指标。

1. **参数化数据**：

使用JDBC查询结果作为接口请求的参数（如从数据库中动态获取用户ID）。

**8-7 JMeter分布式压测**

**1. 分布式测试的基本概念**

**1.1 什么是分布式测试？**

* **简单类比**：就像一个人抬不动水桶，需要两个人一起抬
* **性能测试角度**：当单台机器无法承受测试压力时，使用多台机器协同工作

**官方解释**：在性能测试中，当并发量较大时，单台机器无法承受，需要多台机器共同执行测试

**1.2 分布式测试的核心目的**

* 分担测试压力
* 实现更高并发的性能测试

模拟更真实的用户场景

**2. 分布式测试的工作流程**

**2.1 角色划分**

* **调度机(Master)**：负责分发测试任务和收集结果

**执行机(Agent/Slave)**：实际执行测试任务的机器

**2.2 环境准备**

1. **准备两台机器**（实际演示中模拟两台）

* 一台作为调度机

1. 一台作为执行机
2. **修改JMeter配置**：

* 调度机配置：
* 修改jmeter.properties文件
* 设置server.rmi.ssl.enable=true（启用加密）
* 配置正确的IP和端口（默认1099）
* 执行机配置：
* 同样修改jmeter.properties文件

确保端口和加密设置与调度机一致

**3. 实际操作步骤**

**3.1 配置调度机**

1. 找到JMeter安装目录下的bin文件夹
2. 修改jmeter.properties文件：

* 查找server.rmi.ssl.enable并设置为true
* 移除行首的#符号激活该配置
* 修改server.rmi.ssl.keystore.file和server.rmi.ssl.truststore.file路径（如需要）

确保server\_port=1099（默认端口）

**3.2 配置执行机**

1. 同样修改执行机的jmeter.properties文件

确保所有加密和端口设置与调度机一致

**3.3 防火墙设置**

* **必须关闭防火墙**或开放1099端口
* Windows系统：
* 控制面板 → Windows Defender防火墙 → 关闭或添加例外
* Mac/Linux系统：

使用iptables或系统设置关闭防火墙

**3.4 启动服务**

1. **在执行机上启动JMeter服务器**：

|  |
| --- |
| Plain Text jmeter-server.bat # Windows ./jmeter-server # Mac/Linux |

1. **在调度机上启动JMeter**：

正常启动JMeter GUI

**3.5 创建和运行测试计划**

1. 创建测试计划并配置：

* 设置线程数（如60）
* 设置循环次数（如3）

1. 总请求数 = 线程数 × 循环次数 × 请求数
2. **配置远程执行**：

* 在JMeter GUI中：
* 运行 → 远程启动

选择执行机IP（或"远程启动所有"）

**3.6 执行测试**

1. 点击"远程启动"开始测试
2. 观察调度机和执行机的日志输出

测试完成后查看聚合报告

**4. 常见问题解决**

**4.1 连接失败问题**

* **可能原因**：
* 防火墙未关闭
* IP地址配置错误
* 端口号不一致
* RMI加密配置错误
* **解决方案**：
* 检查并关闭防火墙
* 验证IP地址和端口号

统一RMI加密配置

**4.2 启动缓慢问题**

* **可能原因**：
* 网络延迟
* 机器性能不足
* 测试计划过于复杂
* **解决方案**：
* 优化网络环境
* 提升机器配置

简化测试计划

**5. 面试常见问题解答**

**5.1 JMeter分布式原理**

* **回答要点**：
* 通过调度机分发任务到多个执行机
* 每台执行机执行部分测试用例

结果汇总到调度机生成最终报告

**5.2 分布式与单机的区别**

* **回答要点**：
* 单机受限于硬件资源
* 分布式可突破单机限制，实现更高并发

分布式能更好模拟真实用户场景

**5.3 如何配置JMeter分布式**

* **回答要点**：

1. 准备多台测试机
2. 配置调度机和执行机的jmeter.properties
3. 确保网络互通和防火墙设置
4. 启动执行机服务
5. 在调度机上运行测试
6. **6. 最佳实践建议**
7. **执行机数量**：

* 根据测试需求和机器性能决定

1. 通常2-5台执行机较为常见
2. **测试计划设计**：

* 确保测试用例合理分配

1. 避免单个执行机负载过高
2. **结果分析**：

* 关注整体性能指标

1. 分析各执行机的差异
2. **环境一致性**：

* 确保所有执行机环境相同

使用相同的JMeter版本

**8-8 JMeter性能测验**

**1. JMeter运行模式概述**

**1.1 GUI模式**

* **定义**：通过图形界面启动JMeter
* **启动方式**：
* 命令行直接输入jmeter命令
* 双击JMeter安装目录下的.bat文件（Windows）
* **特点**：
* 可视化操作界面
* 适合脚本创建、调试和接口测试

资源消耗较大

**1.2 非GUI模式（命令行模式）**

* **定义**：通过命令行参数启动JMeter，无图形界面
* **启动方式**：
* 命令行输入jmeter -n -t [脚本路径] -l [结果文件路径]
* **特点**：
* 资源消耗小
* 适合正式性能测试

可生成详细测试报告

**2. 非GUI模式核心参数详解**

**2.1 基本参数**

* -n：指定以非GUI模式运行
* -t：指定测试脚本路径（.jmx文件）
* -l：指定结果日志文件路径（.jtl文件）
* -e：生成HTML测试报告

-o：指定测试报告输出目录（必须为空目录）

**2.2 远程执行参数**

* -R：指定远程执行机列表（逗号分隔的IP）

-r：启动所有配置的远程执行机

**2.3 日志相关参数**

* -j：指定JMeter运行日志文件路径

-L：设置日志级别（如ERROR, WARN, INFO, DEBUG）

**3. 非GUI模式操作实战**

**3.1 基础性能测试**

|  |
| --- |
| Bash jmeter -n -t test\_plan.jmx -l result.jtl |

**3.2 生成HTML报告**

|  |
| --- |
| Bash jmeter -n -t test\_plan.jmx -l result.jtl -e -o report |

**3.3 分布式测试配置**

1. **主控机配置**：

|  |
| --- |
| Bash jmeter -n -t test\_plan.jmx -R 192.168.1.101,192.168.1.102 -l result.jtl |

1. **执行机配置**：

* 修改jmeter.properties文件：

|  |
| --- |
| Plain Text server.rmi.ssl.enable=false server\_port=1099 |

* 启动执行机服务：

|  |
| --- |
| Bash jmeter-server.bat # Windows ./jmeter-server # Linux/Mac |

**4. 性能测试关键指标解析**

**4.1 核心指标**

* **并发用户数**：同时向服务器发送请求的用户数量
* **响应时间**：
* 平均响应时间
* 90%响应时间（P90）
* 95%响应时间（P95）
* 99%响应时间（P99）
* **吞吐量（TPS）**：服务器每秒处理的请求数

**错误率**：失败请求占总请求的比例

**4.2 行业基准（2-5-8原则）**

* **2秒**：优秀响应时间（互联网行业）
* **2-5秒**：可接受响应时间
* **5-8秒**：需要优化

**8秒以上**：不可接受

**5. 性能测试报告解读**

**5.1 HTML报告结构**

1. **摘要页**：

* 测试基本信息

1. 关键性能指标概览
2. **图表页**：

* 响应时间趋势图
* TPS趋势图

1. 错误率趋势图
2. **详细指标页**：

* 每个请求的详细统计

响应时间百分比分布

**5.2 关键分析点**

* **系统瓶颈识别**：
* CPU使用率是否持续高于80%
* 内存是否出现频繁GC
* 磁盘I/O是否成为瓶颈
* **优化建议**：
* 数据库查询优化
* 缓存策略调整

代码级优化

**6. 常见问题解决方案**

**6.1 连接远程执行机失败**

* **检查项**：
* 防火墙设置（开放1099端口）
* 网络连通性（ping测试）

RMI配置（jmeter.properties）

**6.2 测试报告数据不完整**

* **解决方案**：
* 确保使用最新版JMeter
* 检查-l参数指定的路径是否有写入权限

避免使用中文路径和文件名

**6.3 内存溢出问题**

* **优化建议**：
* 增加JVM堆内存：

|  |
| --- |
| Bash set HEAP=-Xms2g -Xmx4g jmeter -n -t test\_plan.jmx ... |

1. 优化测试脚本（减少监听器使用）
2. **7. 最佳实践建议**
3. **测试环境准备**：

* 使用与生产环境相同的硬件配置

1. 确保测试数据与生产数据量级相当
2. **脚本设计原则**：

* 每个事务作为一个单独的逻辑控制器
* 合理设置思考时间（Think Time）

1. 参数化与关联处理完善
2. **执行策略**：

* 先进行单节点测试
* 逐步增加并发用户数

1. 监控系统资源使用情况
2. **结果分析**：

* 重点关注P90/P95响应时间
* 结合业务场景解读指标

多次测试取平均值

**8. 面试常见问题解答**

**8.1 JMeter分布式测试原理**

* **回答要点**：
* 通过主控机分发任务到多台执行机
* 使用RMI协议进行通信

统一收集和分析测试结果

**8.2 GUI与非GUI模式区别**

* **回答要点**：
* GUI适合脚本开发和调试
* 非GUI适合正式性能测试

非GUI模式资源消耗更低

**8.3 如何定位性能瓶颈**

* **回答要点**：

1. 检查响应时间是否符合预期
2. 分析错误率变化趋势
3. 监控服务器资源使用情况
4. 检查数据库查询性能

分析应用日志

**9. 实战案例演示**

**9.1 电商系统压力测试**

1. **测试目标**：

* 模拟双11高峰期访问

1. 验证系统承载能力
2. **脚本设计**：

* 包含商品浏览、加入购物车、结算等事务
* 设置合理的思考时间

1. 参数化商品ID和用户信息
2. **执行方案**：

|  |
| --- |
| Bash jmeter -n -t ecommerce\_test.jmx -l result.jtl -e -o report -R 192.168.1.101,192.168.1.102 |

1. **结果分析**：

* 重点查看结算接口的响应时间
* 分析订单创建成功率

检查库存扣减是否准确

**9.2 接口性能基准测试**

1. **测试目标**：

* 获取各接口的基准性能数据

1. 建立性能基线
2. **脚本设计**：

* 每个接口作为一个独立的事务

1. 设置阶梯式并发（100,200,500,1000）
2. **执行方案**：

|  |
| --- |
| Bash jmeter -n -t api\_benchmark.jmx -l api\_result.jtl -e -o api\_report |

1. **结果分析**：

* 制作接口性能对比图表
* 识别性能瓶颈接口

制定优化优先级

**8-9 JMeter性能理论**

**最大并发寻找方法**

* **加大负载尝试**
* 不断加大负载，疯狂尝试找最大并发
* **操作方式**
* 项目中每次生成测试报告展示
* 以聚合报告为例，逐步加大并发
* 初始尝试 200×3 到 600（项目建议一百一百往上加）
* 发现显示时间超 10 秒不满足，停止
* 减小并发至 60，观察聚合报告
* 平均值在三到五秒，认为最大并发与 60 相关（如 6×30 等，示例为 3×6 = 180 人同时点击首页显示时间合适）
* **绘制表格记录测试情况**
* **表格字段**：并发、持续时间、显示时间、错误率、CPU、结果、原因
* **示例记录**
* 100 个并发，持续时间五分钟，显示时间一秒，错误率零，CPU 假设 40%，结果 pass
* 180 个并发，持续时间假设五分钟，显示时间三秒左右，错误率零，CPU 假设 60%，若领导要求显示时间三秒以下合格则 pass
* 600 个并发，持续时间五分钟，显示时间十秒，不满足要求
* 若 180 个并发错误率达 5%，根据条件判断是否 pass（如领导要求错误率条件不满足则不 pass；若显示时间两秒但错误率 4.9%，错误率不满足也不 pass）
* **压力测试做法**
* **逐步增加并发**
* 线程组调整线程数（如从 20 开始）或使用 csv 文件读取数据循环跑
* 逐步增加循环次数，如四五六七八这样增加
* 一直压到崩溃（常见全部报 500，失败率逐步高升）
* **性能测试基准**
* **根据用户量确定并发**
* 根据日活量确定，如日活 1 万，最大不超过 100，做 200 并发差不多
* 若经理未提供数据，根据数据库或向相关人员要日活量数据
* **性能测试理论总结**
* **服务器组成**
* 硬件：CPU、内存、IO 硬盘等
* 软件：如 rs my 等
* **性能测试定义**
* 使用高并发手段模拟和测试高负载下服务器表现
* **性能测试指标**
* 并发、TPS、显示时间、错误率、吞吐量
* **寻找并发方法**
* 加大负载
* **负载测试、压力测试、并发测试概念**
* **负载测试**：如 100 个并发能持续多长时间；以汽车载货为例，载一堆货物能跑多长时间
* **压力测试**：最大压到多少系统崩溃；如食堂容量 100 人，塞 110 人看能否容下
* **并发测试**：测最大并发是多少；如 100 个人一块跑步，最次的人跑多少距离跑不动
* **性能测试注意事项**
* **实际工作情况**
* 模拟以简单形式带入系统测试做法，实际工作类似
* 生成报告、简便做系统测试较简单，但会遇个别难处理情景
* **脚本编写**
* 写脚本较麻烦，不同接口要传入不同数据，需根据实际情况编写
* **检测工具**
* 有第三方检测工具，检测更全面，根据公司使用
* **工具练习**
* 工具要多多练，知道每个工具用途
* **公司性能测试现状**

90%以上工程做技术测试，性能测试多针对某个接口，看压多少是否报错，意义有限