# 测试

目录

[测试 1](#_Toc3628)

[一， 测试基础： 1](#_Toc7454)

[1） 测试分类： 1](#_Toc11446)

[3） 测试主流技能： 2](#_Toc793)

[4） 常见测试分类： 2](#_Toc284)

[1， 按测试阶段划分： 2](#_Toc4482)

[2， 按代码可见度划分： 2](#_Toc22321)

[5） 模型： 3](#_Toc7921)

[1， 软件质量模型： 3](#_Toc17656)

[2， 测试模型（W模型）： 3](#_Toc17409)

[6） 测试流程（测试需求分析）： 4](#_Toc30905)

[二， 手工测试： 6](#_Toc30091)

[三， 接口测试（Postman）： 6](#_Toc23763)

[四， 网络代理工具Fiddler： 7](#_Toc16822)

[五， 性能测试： 8](#_Toc26378)

[六， 自动化测试： 8](#_Toc20099)

手工测试；接口测试；自动化测试；性能测试

# 测试基础：

## 测试分类：

软件测试：

性能测试：有很多用户同时登录该网页；

兼容性测试：不同浏览器是否可以打开该网页；

安全测试：黑客入侵

网络测试：

稳定性测试：

功能测试：

1. 测试工具：
2. selenium IDE：自动化测试
3. Postman：接口测试
4. Fiddler应用：网络代理工具，前后端交互应用

Jmeter：测试接口，性能

## 测试主流技能：

功能测试，自动化测试，接口测试，性能测试

## 常见测试分类：

### 按测试阶段划分：



内测：公司内部进行测试

公测：让玩家来进行测试

### 按代码可见度划分：

黑盒测试：看不见源代码，主要对程序功能进行测试（比如测试微信）（系统测试，自动化测试）

灰盒测试：看见部分代码，主要对程序进行接口进行测试。（集成测试）

白盒测试：看见全部代码，主要对程序源代码进行测试。（单元测试）

补充：

测试策略：

冒烟测试：大规模执行测试之前，针对程序主功能进行验证，保证程序具备可测性。（首先需要进行冒烟测试，然后再进行大规模的测试）

## 模型：

### 软件质量模型：

质量模型提供测试设计的不同角度和验证方向。（当执行测试的时候，需要思考到的方面）

（功能性，性能效率，兼容性，易用性，可靠性，信息安全，可维护性，可移植性）

重点：

功能（根据需求文档，测试功能是否满足需求），

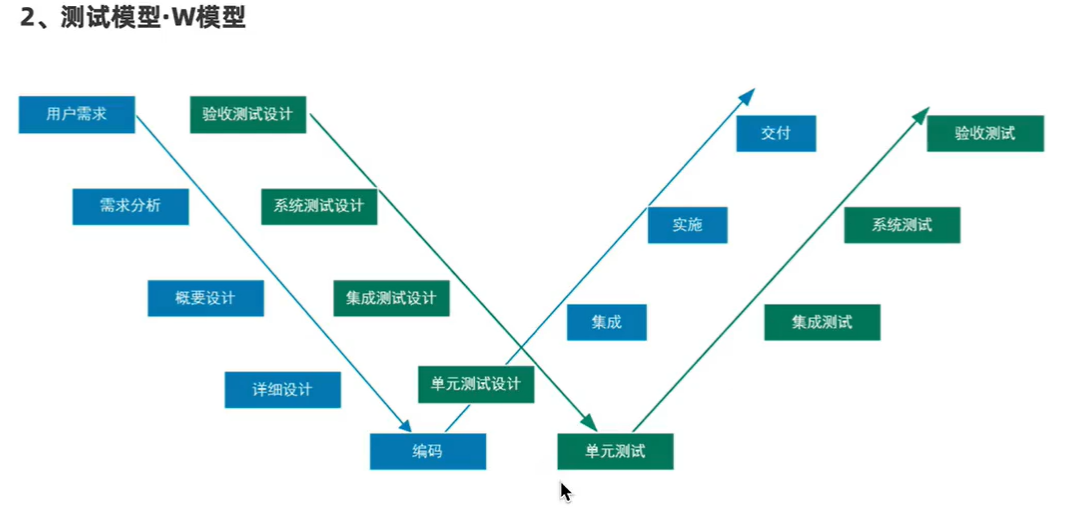
兼容（软件与主流硬件和软件兼容），

性能（配置和网络是否能够支持大量人数同时访问该系统/APP），

易用（便于使用，查看功能，登录时间，用户体验感），

安全

### 测试模型（W模型）：



W模型：

优点：

测试伴随整个产品开发周期，测试对象有程序，需求，设计文档；

较早介入测试，发现问题，降低修复成本。

缺点：

实施复杂，难度大，对需求阶段和设计阶段的测试设计要求较高。

## 测试流程（测试需求分析）：

需求分析，计划编写，用例设计，用例执行，缺陷管理，测试报告

1. 需求分析：

首先需要完全了解需求文档，记录下不明白的地方。（确定各部门对需求理解一致；站在不同角度对需求进行查漏补缺）

1. 计划编写：

测什么，谁来测（人员进度安排），怎么测（测试策略，工具）

整个测试团队都是围绕测试计划进行的。

1. 测试用例（用例设计）：

验证项目是否符合需求的操作文档，执行测试的文档（用户使用的案例）

作用：防止漏测，实施测试标准

考虑点：质量模型（功能，性能，兼容，易用，安全）

用例设计编写格式：

用例编号（项目+模块+编号），标题（预期结果+操作步骤），模块/项目，前置条件，优先级（p0-p4，p0最高），测试步骤，测试数据，预期结果，实际结果，测试是否成功

补充：alt+回车：在当前单元格内换行

用例的实现步骤：

1. 明确需求；
2. 确定有效和无效等价类
3. 确定数据，范围
4. 提取数据编写测试用例

### 用例的设计方法：

#### 等价类划分法：解决穷举：

1. 步骤：明确需求-->确定有效和无效等价类-->提取数据编写.用例
2. 案例：

案例1：6-10为自然数

示例：

有效：在所有有效数据集合中取一条数据（8位自然数）

无效：在所有无效数据集合中取一条数据（4位自然数；8位非自然数）

参考点：参数，类型，长度，规则

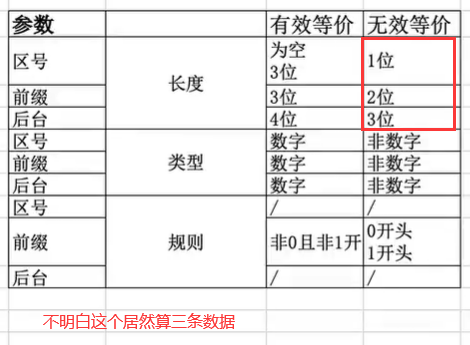
案例2：

要求：1，区号：空或者是三位数；2，前缀码：非“0”且非“1”开头的三位数字；3，后缀码：四位数字。

技巧：1，正向：并集（共有），参数不同条件进行组合（有效正向数据最多的条件，最多就有该条件数量的组合）；2，逆向：验证一条数据的时候，其他数据必须符合正确的条件。

（正向两条用例，逆向八条用例，自己尝试一下）

CTRL+M：合并多个单元格



1. 适用场景

需要大量测试数据输入，但是没法穷举的地方。（比如输入框，下拉列表，单选复选框这种页面级的输入框类测试）

#### 边界值分析法：限定边界值：

1. 边界范围节点：上点，离点，内点（最多七条）
2. 例题：标题长度大于0，小于等于30个字符

有效：字符大于0，小于等于30（有效范围值：1，29，30）

无效：0，大于三十（0，31）

优化：开内闭外

补充：边界类可以覆盖等价类的

#### 判定表法：多条件依赖关系：

1. 定义：一种以表格形式表达多条件逻辑判断的工具。（条件桩，动作桩，条件项，动作项）
2. 案例：

需求：

1. 如果金额大于500元，又未过期，则发出批准单和提货单；
2. 如果金额大于500元，又过期了，则不发批准单和提货单；
3. 如果金额小于等于500元，则不论是否过期都发出批准单和提货单；
4. 在过期的情况下，不论金额大小，还需要发出通知单；

案例：

1. 输入的第一列字符必须是A或B
2. 第二列字符必须是一个数字
3. 如果第一列字符不正确，则给出信息L
4. 如果第二列字符不正确，则给出信息M
5. 如果两列字符输入正确，则修改文件成功

通过判断：条件，动作（结果），来展示用例所有可能性并编写用例计划

1. 使用场景：

有多个输入条件，多个输出结果，并且存在一定的关系。判定表一般适用于条件组合数量较少的情况。

如果项目中多条件组合大于四个依赖，可以使用正交表，因果图来实现（一般不会用到）

1. 场景法：项目业务
2. 流程图：根据需求，解决业务用例问题。（使用工具：[ProcessOn - 免费在线作图，思维导图，流程图，实时协作](https://processon.com/)）
3. 流程图优先于单功能（平时测试都是单个功能进行测试，容易忽略多个功能的组合测试）
4. 案例：

ATM机取款流程图（一般做冒烟测试用例）

小程序的业务用例

1. 错误推断法：

使用场景：时间紧任务量大时使用（没有时间写测试用例）

验证码：正确，为空，错误，过期

1. 用例执行：
2. 缺陷管理：
3. 缺陷：
4. 定义：软件在使用过程中的所有问题，也成为bug（预期结果与实际结果不一致，为缺陷）
5. 判定标准：功能（少，多，错误，隐形功能）；使用感
6. 扩展：IT行业常见岗位：前端，后端，测试，UI，运维（管理服务器，相当于安装软件项目的电脑），产品（梳理需求，设置原型），运营（上线运行策划）
7. 缺陷出现原因：需求，设计，编码，运行中的某个阶段出现问题
8. 缺陷提交要素：

缺陷报告编号：缺陷的唯一性标志

严重程度：严重（S1：主功能）；一般（S2：次要功能）；微小（S3：易用性，界面）；建议（S4：简易性问题）

缺陷优先级：

Bug类型：代码错误；页面（UI）错误；兼容性错误；设计缺陷；性能问题；架构；数据；易用性

缺陷状态：new（新建）；open（打开）；closed（关闭）；postponed（延期）

1. 缺陷管理流程：
2. 流程：提交缺陷-->分派缺陷-->判断缺陷是否重复(否)，是否bug(是)，是否推迟处理(否)-->处理缺陷-->回归测试-->判断验证是否通过-->关闭缺陷
3. 作用：管理用例，管理缺陷
4. 使用Excel进行缺陷管理：
5. 禅道工具管理缺陷：
6. 测试报告：

# 手工测试：

1. 测试用例设计：
2. Linux常用命令：
3. 数据库操作：
4. 项目实战：
5. 使用缺陷工具：

禅道：

缺陷（质量）管理工具

测试用例（凭直觉；参考文档；方法套路），测试执行（web，UI测试；APP测试；接口测试，postman），测试报告；

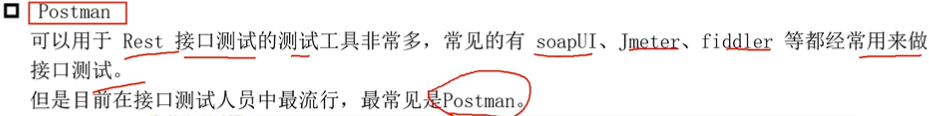
整理知识体系；

# 接口测试（Postman）：

接口测试步骤：

1. 拿到接口的URL地址
2. 查看接口是用什么方法发送（get/post/put…）
3. 添加请求头，请求参数
4. 发送查看返回结果，校验返回结果是否正确

接口就是为了实现目的互补的一种方式。比如我需要吃饭，厨师需要做菜，这两个目的刚好实现互补；比如我需要在不同的地方购买便宜的东西，橙心优选需要将便宜的东西卖给别人，这就有和橙心优选合作的商铺；再比如，我需要一件海外的东西，我们不需要再自己去海外寻找自己想要的东西，而是可以在海外代理那里购买。





# 网络代理工具Fiddler：

数据：由客户端和服务端通过网络接口交互产生的。（客户端和服务器位于不同的机器/网络）

网络监控过程：手机网络 --> Fiddler网络代理 --> 服务器

（这个Fiddler网络代理相当于一个接口；在手机上使用Fiddler，为什么？有什么用？）

使用Fiddler的作用：分析具体的网络请求，得出BUG归属。

客户端和服务器交互是按照接口文档和协议进行发送内容，响应数据的。作为测试人员，我们需要检测接口请求的地址，参数，方法是否正确。（如果是请求不正确，那么就是客户端的问题）。检测服务端返回的数据是否按照接口文档执行（服务端问题）

可以自己添加过滤器，只展示要查看的app与服务器之间的数据。（这样就可以查询问题是在客户端还是服务端，发送请求还是响应内容（状态码；返回格式：json，xml等；具体数据））

客户端和服务端进行交互-->产生数据-->查看问题出现在那一边

学习方法：

学习一个新知识不能只是学习它的工具，而是要知道它的逻辑。即使以后不使用当前工具，也可以根据自己的知识储备很快上手新的工具。（比如网络代理工具）

比如，在测试中，我们需要通过抓包来判断一个bug是客户端导致的还是服务端导致的，这是一个判断这个bug的思路（道）。我们就需要用不同的方式来抓包（术）。

所以，核心思想>执行战术

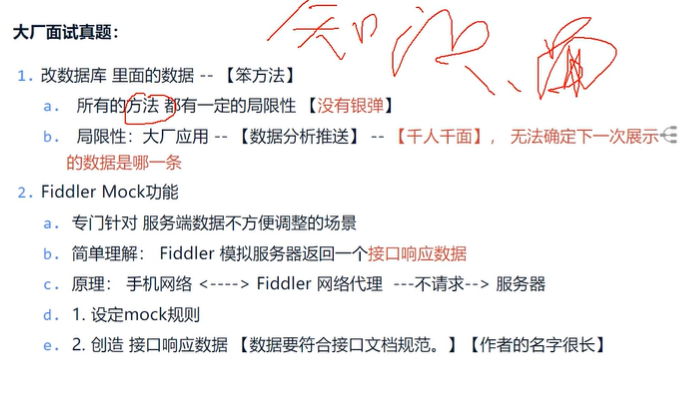
考题：如果碰到这个测试，你会怎么做？？

测试目的：在作者名称很长的情况下，页面展示是否正常

自动转发：

边界值：

需要有知识广度。同一个问题，需要有不同的解决方案





没有银弹：没有一种技术能解决所有问题

# 性能测试：

工具：JMeter，Locust框架

# 自动化测试：

自动化思维：做事情之前，都可以想想哪些事情可以自动化。（重复性较强）

自动化测试借助软件（selenium）完成或者自己写代码（注意：初级的代码不需要自己写）

Web自动化测试

App自动化测试

补充：

GitHub：

Github中文网：[GitHub中文社区 (githubs.cn)](https://www.githubs.cn/)

查找仓库：搜索emacs.d可以查找仓库

Expolore：根据浏览习惯创建仓库列表，并且每天刷新

1. 仓库（repository）管理：
2. 创建仓库：

其中的gitignore和Lience文件在特殊情况下（开源软件发布协议，工具引用他人代码，网页模板开发）可以选择一些模板

1. 添加文件：
2. 提交issue：
3. 提交/接收PR（拉取请求）：
4. git操作：