软件测试复习题

## 1.什么是软件测试？

使用人工或自动手段，来运行或测试某个系统的过程。其目的在于检验它是否满足规定的 需求或弄清预期结果与实际结果之间的差别。

## 2.软件测试目的?

测试的最终目的是确保最终交给用户的产品的功能符合用户的需求，把尽可能多的问题在 产品交给用户之前发现并改正。

**1**、提高软件的质量

软件测试的首要目的就是让用户对产品有更好的体验，保证软件的高质量。

**2**、保证软件的安全

软件测试的第二大目的就是保证软件的安全，有一些软件是经过数据加密的， 比如各大银 行系统的 APP。涉及到资金的支出和存入，对软件的安全性要求是特别高的。现在很火的 区块链，其实它的本质就是保障安全，通过 Hash 算法等一系列的安全算法来保障资金流动 的安全。

**3**、降低软件开发成本

软件测试的另外一个目的就是降低软件的开发成本。比如迪士尼的一款狮子王的软件，预 期本应是好评如潮，也能通过这款软件获益不匪。但因为在很多系统上都无法使用，所以 造成了大量的用户投诉和下线、卸载等。对成本造成了非常大的损失。如果当时那款软件 能够在不同的系统上进行测试， 肯定会大大降低成本。

**4**、降低因软件缺陷带来的商业风险

除了降低开发成本，还可以降低因软件缺陷带来的商业风险。如果软件的缺陷过多，也会 降低合作企业的信任度，从而造成商业上的一些损失和风险。进行测试则会降低商业风险。

**5**、为用户接受软件提供有力的依据

测试是在用户使用软件之前提早介入，进行软件质量的保障。只有经过了软件测试这一环 节，才能把软件交付到用户手中。公司的人员才对软件的信心更高，不至于把软件交付出 去之后，时常担心用户的投诉或产品有致命的风险。

**6**、树立用户对软件的信心

显然，用户拿到经过测试的软件和没经过测试的软件，对质量的信心是不一样的。所以测 试可以树立用户对软件的信心。

## 3.黑盒测试、白盒测试、单元测试、集成测试、

系统测试、验收测试的区别与联系？

黑盒测试：把测试对象当成一个黑盒子，测试人员完全不考虑逻辑结构和内部特性， 只依 据程式的需求说明书 来检查程式的功能是否满足它的功能说明。

白盒测试：把测试对象当成一个透明的盒子，允许测试人员利用程序内部逻辑结构及相关 信息，设计或选择测 试用例，对程式所有逻辑路径进行测试。

单元测试： 白盒测试的一种，对软件设计中的单元模块进行测试。

集成测试：通过测试单元模块组装成系统或者子系统，再进行测试，重点 测试不同模块的 接口部分

系统测试： 在所有都考虑的情况下， 对系统进行测试。

验收测试： 第三方进行的确认软件满足需求的测试

## 4 黑盒测试和白盒测试常用的测试方法有哪些，

举个例子？

一、白盒测试

白盒测试是对程序内部的逻辑结构进行检查， 从中获取测试数据。

白盒测试主要包含如下测试方法：

**1.**语句覆盖

语句覆盖要求必须编写足够多的测试用例，使得每一个可执行的语句都至少被执行一次， 语句覆盖常常被称为“最弱的覆盖”，因为它只考虑了可执行语句，但是无法测试隐式分支 （比如：if 判断中没有写出来的 else），也没有考虑各个分支的组合。

**2.**判定（分支）覆盖

判定覆盖要求每个判断都必须有“true”和“false”两个结果，并且每条语句都至少被执 行一次。判定覆盖弥补了语句覆盖的缺点，即隐式分支可以被测试到，这样， 判定覆盖将 所有的判断的每种可能的结果的至少执行一次。但是判定覆盖仍然存在问题， 判定覆盖只 强调了判断的结果，而忽略了判断的具体条件，所以依然可能遗漏部分的测试路径。

**3.**条件覆盖

条件覆盖要求编写足够多的测试用例以确保将每个判断中的每个条件的所有可能的结果都 至少执行一次， 弥补了判定覆盖的缺点。但是却有可能无法让每条语句都被执行到。

**4.**判定**/**条件覆盖

判定/条件覆盖综合了判定覆盖和条件覆盖的优缺点，判定/条件覆盖要求设计出足够多的 测试用例，将每个判断中的每个条件的所有可能结果至少执行一次，将每个判断的所有可 能的结果至少执行一次。但是仍然存在缺点，判定/条件覆盖没有考虑条件的组合情况，有 些特定的条件会屏蔽掉其他的条件。

**5.**条件组合覆盖

条件组合覆盖能解决判定/条件覆盖中条件组合的问题，多重条件覆盖要求编写足够多的测 试用例，将每个判定中的所有可能的条件结果的组合至少执行一次

二、黑盒测试

黑盒测试将程序看成是一个黑盒子，不涉及程序内部的逻辑结构， 而是通过规格说明对程 序功能进行检查，从中获取测试数据。

黑盒测试主要包含如下测试方法：

**1.** 等价类划分

由于对程序是不可能进行穷举输入测试的，所以我们可以去寻找最小子集以覆盖尽可能多 的输入情况。等价类划分要求每个测试用例都要尽可能多的体现不同的输入情况，以减少 测试用例的数量，同时需要将输入范围进行划分，划分为有限数量的等价类。使用等价类 划分有两个步骤：

确定等价类：确定等价类是选取每一个输入条件并将其划分成两个或者更多的组。我们通 常是确定两个等价类：有效等价类和无效等价类。有效等价类代表有效输入， 无效等价类 代表无效输入。

生成测试用例：使用等价类来生成测试用例，可以为每个等价类设置一个编号以方便记录。 在编写测试用例时，需要尽可能多的去覆盖还没有覆盖过的有效等价类，直到所有的有效 等价类都被覆盖。而对于无效等价类，则需要为每一个无效等价类都编写一个测试用例，

因为如果一个测试用例覆盖了多个无效等价类，那么有些无效等价类的错误可能会被屏蔽掉。

**2.** 边界值分析

经验证明，考虑了边界条件的测试用例与其他没有考虑边界条件的测试用例相比，具有更 高的测试回报率。边界值分析其实是对等价类划分进行了补充，因为在等价类划分中，并 没有考虑到边界值， 然而有很多错误都会出现在边界值附近。

边界值的概念可以想象成高等数学中极限，输入值就在边界值的附近，但是和极限不同的 是，输入值可以等于边界值本身。例如： 规定密码为 1 到 6 为纯数字组成， 那么它有一个 有效等价类为： 1 到 6 个数字， 两个无效等价类为：0 个数字， 6 个以上数字。这里的边界 值为 1 和 6，根据边界值分析， 我们可以选取 0 ，1 ，2 ，5 ，6 ，7 个数字来设置测试用例。

综上： 边界值分析考察正处于等价类划分边界或者边界附近的状态。

**3.** 因果图分析

由于等价类划分和边界值分析都是着重考虑输入条件的罗列，但是却没有考虑输入条件的 组合，这是等价类划分和边界值分析的缺点。而因果图弥补了这一缺点，因果图有助于用 一个系统的方法选择出高效的测试用例集，它还有一个额外的好处，就是可以指出规格说 明的不完整性和不明确之处。

因果图是一种形式语言，有点像简化版的逻辑电路， 我们可以将规格说明转化为可视化的 因果图，然后通过系统的方法的出测试用例集。

**4.** 错误猜测

错误猜测是指在测试人员接到具体程序后，利用直觉和经验猜测出错的可能类型，然后编 写测试用例来暴露这些错误。

错误猜测的基本思想是列举出可能犯的错误或错误易发情况的清单，然后依据清单来编写 测试用例。

## 5.简述黑盒测试和白盒测试的优缺点？

※ 黑盒测试的优点有：

1. 比较简单， 不需要了解程序内部的代码及实现；

2. 与软件的内部实现无关；

3. 从用户角度出发，能很容易的知道用户会用到哪些功能，会遇到哪些问题； 4. 基于软件开发文档， 所以也能知道软件实现了文档中的哪些功能；

5. 在做软件自动化测试时较为方便。

※ 黑盒测试的缺点有：

7. 不可能覆盖所有的代码，覆盖率较低， 大概只能达到总代码量的 30%；

8. 自动化测试的复用性较低。

※ 白盒测试的优点有：

9. 帮助软件测试人员增大代码的覆盖率， 提高代码的质量，发现代码中隐藏的问题。

※ 白盒测试的缺点有：

10. 程序运行会有很多不同的路径，不可能测试所有的运行路径；测试基于代码，只能测试 开发人员做的对不

对，而不能知道设计的正确与否，可能会漏掉一些功能需求；系统庞大时，测试开销会非 常大。

## 6.Beta 测试和 Alpha 测试有什么区别？

大型通用软件，在正式发布前， 通常需要执行 Alpha 和 Beta 测试，目的是从实际终端用户 的使用角度，对软 件的功能和性能进行测试， 以发现可能只有最终用户才能发现的错误。

Alpha 测试是由一个用户在开发环境下进行的测试， 也可以是公司内部的用户在模拟实际操 作环境下进行的受控测试，Alpha 测试不能由程序员或测试员完成。Alpha 测试发现的错误， 可以在测试现场立刻反馈给开发人员，

由开发人员及时分析和处理。目的是评价软件产品的功能、可使用性、可靠性、性能和支 持。尤其注重产品的界面 和特色。Alpha 测试可以从软件产品编码结束之后开始，或在模 块（子系统）测试完成后开始，也可以在确认测试 过程中产品达到一定的稳定和可靠程度 之后再开始。有关的手册（草稿）等应该在 Alpha 测试前准备好。

Beta 测试是软件的多个用户在一个或多个用户的实际使用环境下进行的测试。开发者通常 不在测试现场，Beta 测试不能由程序员或测试员完成。因而，Beta 测试是在开发者无法控 制的环境下进行的软件现场应用。

在 Beta 测 试中， 由用户记下遇到的所有问题， 包括真实的以及主管认定的， 定期向开发者 报告，开发者在综合用户的报告后， 做出修改，最后将软件产品交付给全体用户使用。 Beta 测试着重于产品的支持性， 包括文档、客户培训和支持产品的生产能力。

只有当 Alpha 测试达到一定的可靠程度后，才能开始 Beta 测试。由于 Beta 测试的主要目标 是测试可 支持性， 所以 Beta 测试应该尽可能由主持产品发行的人员来管理。

## 7 讲一下你们的测试流程？

步骤一：需求分析

（阅读需求，理解需求，分析需求点，参与需求评审会议，评审需求是否合理）跟产品和 开发， 测试一起参加

步骤二：制定测试计划

（测试范围、进度安排（时间进度、流程进度）、人力、物力的分配、对于整体测试策略的 制定、对于风险评估与规避措施要有一个制定）

步骤三：设计测试用例、编写测试用例、准备测试数据

（通过参考需求文档（原型图）、概要设计、详细设计等文档，来编写测试用例，用例编写 完成之后会进行评审。）

步骤四：实施测试（执行测试用例）

步骤五：提交 **bug**，提交缺陷报告

步骤六：生成测试总结和报告 （给出测试报告， 并确认是否可以上线。）

步骤七：上线的维护

产品发布上线后，关注是否正常运行，要进行常规的维护性测试

## 8 回归测试是什么测试方法？

回归测试是指修改了旧代码后，重新进行测试以确认修改没有引入新的错误或导致其他代 码产生错误的一种测试方法。

1、回归测试是指重复以前的全部或部分的相同功能测试

2、新加入测试的模块，可能对其他模块产生副作用，因此要进行某些程度的回归测试

3、回归测试的重心， 是以关键性模块为核心

## 9 讲一下测试用例需要包含的要素？

测试用例八要素：用例编号，测试项目， 测试标题， 重要级别，预置条件，测试输入，操 作步骤，预期结果

（1） 用例编号（规则：由字符和数字组成的字符串，具有唯一性，易识别性） （2） 测试项目（或者测试模块）

（3） 测试标题（体现测试出发点关注点以及测试用例期盼的测试结果）

（4） 重要级别、优先级别（重要级别一般分为高中低 ）

（5） 预置条件： 测试用例在执行时需要满足一些前提条件， 环境的设置

（6） 测试输入（测试执行中需要加工的外部信息，避免用描述性语言，要具体，根据测 试用例具体情况，有手工输入， 文件，数据库记录）

（7） 操作步骤： 执行当前用例需要经过的操作步骤，需要明确的给出每一个步骤的描述 （8） 预期结果： 需要判断测试对象是否正常工作

## 10 测试计划的内容都包括什么？

1.测试背景。在进行测试项目前，先要清楚这个软件项目的背景介绍，还有项目设计了哪 些参与人员，例如软硬件项目负责人等， 参与人员的介绍及联系方式都需要记录清楚。

2**.**测试依据。主要包括了软件需求文档， 软件规格书，软件设计文档， UI 原型图， 接口文

档及其他参考产品等。

3.测试资源，在测试计划上详细写出每一项需求，例如测试设备需求，测试人员需求，测 试环境需求等其他需求。

4**.**测试策略。在测试计划上详细写出采取的测试方法，搭建哪些测试环境，采取哪些测试 工具以及测试管理工具， 对人员进行培训等。

5.测试范围，描述本次测试活动的测试范围

6.测试排期。主要包括了测试需求分析，测试用例编写，测试实施，根据项目计划，测试 分成哪些测试阶段（如单元测试、集成测试、系统测试阶段，α、β测试阶段等），每个阶 段的工作重点以及投入资源等。

7.风险估计和应急计划

8.测试失败**/**通过的标准 (开始标准:代码编译通过，软件可以争取安装运行，实现功能与产 品设计出入，冒烟测试通过)

## 11 测试报告包括哪些内容 ？

测试结论（是否达到发布标准， 是否可发布）

已知风险、未知风险

测试时间， 测试人员（测试起止时间）

测试环境， 测试设备（用到哪些测试数据，客户端环境， 浏览器）

需求大纲（当前这个版本，包含哪些需求点）

Bug 数据分析（从多个维度分析：bug 等级分布，遗留 bug 分析，bug 类型分布。模块 bug 分布， bug 激活次数分析）

测试总结（从测试角度， 对版本存在的问题， 提出建议）

## 12. 测试结束的标准？

（1）模块测试用例执行完毕，覆盖了全部软件需求

（2）缺陷收敛趋势符合质量要求

（3）缺陷修复率达到产品设计人员的需求

（4）达到预先的缺陷度量原则（缺陷密度值达到客户的要求）

## 13 如何保证测试的覆盖率？

1）将不同的需求来源(包括隐式需求）划分成一个个需求点，针对每一点需求进行测试点 分析

2）要保证测试用例能够全面覆盖测试需求，测试场景覆盖需求中的业务逻辑

3）当测试用例设计完成后，要组织测试用例的评审，这样可以吸取别人的意见，减少遗漏， 补全测试用例

5）测试用例编写完成后， 就是测试执行要 100%

6）在测试执行过程中，要继续对测试用例补充完善，确保提高测试覆盖率

7）在整个测试过程中，需求都是不可能不变的，所以要及时的更新测试需求、测试用例

8）测试准备的数据符合实际业务，具备比较好的合理性

9）进行交叉测试，可以多个人， 不同测试考虑点

## 14 测试中有哪些风险？（如何把控风险）

1，测试，需求理解上面有偏差

2，测试人员水平不够，测试人员覆盖点不全

3，测试人员时间不够，导致测试不完成

4，测试环境上面不足，导致测试点不能完全测试完成

## 15 需求评审有哪些人参与？主要做些什么？

需求评审目的：

1、了解需求的动机、目标、方案、排期等，为开发设计、测试设计做准备 2、降低需求设计本身的不完整、不一致、不准确等出现的可能性

3、通过评审尽可能降低团队成员理解的不一致性

4、提早考虑排期风险、实现风险

主要内容：

需求评审通常是由产品经理主持，通过讲解产品需求文档，让相关人员了解具体需求，并 提出疑问， 进行沟通的过程。统一大家对产品需求的理解，为后续“如何做”打好基础。

主要参加人员：

项目经理、产品经理、前后端开发、测试，可能还有 UI/设计

## 16 测试如何为需求评审做准备？

1. 需求评审前，提前进行需求熟悉阶段，逐一分析需求点，做好准备，相关需求疑问点 列好清单， 带着问题去参会。

2. 产品宣讲时期，就算过程有问题，不要试探打断产品的宣讲，一是节约时间， 二是不 礼貌，等产品将一个模块宣讲完毕，开始带着你的问题，分析给项目成员听，并提出 改进建议

3. 当需求有问题确认需要修正，或需进一步跟业务确认再做定夺的，做好标记，并提醒 产品， 做好会议记录

4. 宣讲时期如果开发进入技术凯旋（数据到底是前端自己去拿还是后端传给前端），在适 当时期提醒产品，进行控场，有效的控制时间，接着宣讲其他模块业务流

5.需求评审完后，项目群推送需求评审会议记录点，周知各位目前的版本的疑问点以及修 改点， 并提醒产品进行下次需求确认会议时间,这个会议也可以由测试主持，

## 17 需求不明确，通过哪些方式解决？

需求来源方，做确认

需求评审、用例评审、通过流程，和定下来的产品文档、测试用例等来明确 跟产品、研发沟通， 讨论不明确的需求

记录需求变更， 更新测试用例， 做好记录

## 18 用例评审有哪些人参与？主要做些什么？

用例评审目的：

1 、 用例设计的结构安排是否清晰、合理， 是否利于高效对需求进行覆盖。 2 、 优先级安排是否合理。

3 、 是否覆盖测试需求上的所有功能点。

4 、 用例是否具有很好可执行性。例如用例的前提条件、执行步骤、输入数据和期待结 果是否清晰、正确； 期待结果是否有明显的验证方法。

5 、 是否已经删除了冗余的用例。

6 、 是否包含充分的负面测试用例。充分的定义，如果在这里使用 2&8 法则，那就是 4 倍于正面用例的数量，毕竟一个健壮的软件，其中 80%的代码都是在“保护”20%的功能 实现。

7 、 是否从用户层面来设计用户使用场景和使用流程的测试用例。

8 、 是否简洁，复用性强。例如， 可将重复度高的步骤或过程抽取出来定义为一些可复 用标准步骤。

用例评审人员

1. 这里会分为多个级别进行评审，有时候需求小， 是只有测试内部评审。

2. 正式的用例评审，测试预约时间组织开发，产品/ 测试经理， 测试组其他小伙伴进行测 试点评审。评审会议完成后，整理会议纪要， 邮件知会到与会人。

测试根据评审结果修正测试点文档， 再次邮件知会到与会人， 细节问题会再次进行谈论。

## 19 敏捷测试是什么？

敏捷测试(Agile testing)

测试的一种, 主张尽早开始测试， 重点关注持续迭代地测试新开发的功能.

敏捷的测试团队还要保证整个软件开发过程是正确的是符合用户需求的

遵循

1、强调从客户的角度，即从使用系统的用户角度， 来测试系统

2、重点关注持续迭代地测试新开发的功能，而不再强调传统测试过程中严格的测试阶段

3、建议尽早开始测试，一旦系统某个层面可测，比如提供了模块功能， 就要开始模块层面 的单元测试，同时随着测试深入，持续进行回归测试保证之前测试过内容的正确性

## 20 敏捷测试的特点

敏捷测试的特点

既然敏捷测试属于一种新的测试实践， 那么到底它有什么的特点呢?我用“四个更”来 归纳：

更强的协作：敏捷开发人员和测试人员工作得更加紧密，喜欢更直接的沟通方式而不 是通过邮件文档这种一来一回反反复复的沟通模式;

更短的周期：需求验证或测试的时间不再是按月来计算，而是按天甚至按小时计算。 用户验收测试在每个 sprint 的结尾都会进行;

更灵活的计划：敏捷测试也需要拥抱变化， 测试计划不再是一成不变的文档，而会根 据业务价值交付的顺序进行灵活的调整;

更高效的自动化：相比传统测试， 自动化在敏捷测试中扮演了极其重要的角色。它是 实现快速交付确保质量的一种非常有效的手段

## 21 敏捷测试的作用

缩短价值交付周期

通过采用敏捷测试这种模式，可以契合整个敏捷开发周期，使得整个敏捷开发按照相 同而快速的迭代速率和周期交付，让最终用户尽快获取到业务价值;

更早发现测试风险

敏捷测试使得测试人员尽早开始进行测试，尽早的发现系统缺陷或存在的问题，避免 所有的问题都堆积在最后的测试阶段形成“Big-bang”的结果， 降低整体系统风险;

强调质量属于大家

质量是构建出来的，而不是测出来的。敏捷测试一直强调质量属于每一个人的责任， 除了测试之外， 开发、产品经理等都有义务对自己的交付件质量负责，这样才能确保项目 的整体质量;

化繁为简节省成本

敏捷测试没有要求需要详细的测试计划和测试文档，也没有定义繁复的测试流程及缺 陷流程，这种轻量级的管理模式为测试人员减少不必要的负担，节省了工作量及成本。

## 22.提交 Bug 需要注意哪些问题？

1、不要急着提交，先做一下复现，进行证实， 如果需要的话，也可以使用不用的版本测试 对比一下

2、简单明了的概括 bug 标题，清晰的描述 bug 重现步骤， 分析 bug 和预期正确结果，附加 bug 的截图

或者日志。描述 bug 的时候。

3、在不能确认该情况是否为 bug 的时候，可以请教其他人。

4、提交完 bug 以后， 后面还要跟进 bug。

|-1. Bug 标题要简明扼要的表述清楚

|-2. Bug 的级别（严重级别，优先级别）

|-3. 内容:操作步骤（一定要描述清楚，以便开发可以复现）,预期结果，实际结果 |-4. 最好， 把截图， 日志相关的信息一并的提交（方便开发定位）

|-5. 测试环境（在 google,华为手机上）

|-6. 属于哪个产品， 模块的

## 23.如果一个缺陷被提交后，开发人员认为不是

问题，怎么处理？

a)首先， 将问题提交到缺陷管理库里面进行记录。

b)然后，要获取判断的依据和标准：

v.根据需求说明书、产品说明、设计文档等，确认实际结果是否与计划有不一致的地方， 提供缺陷是否确 认的直接依据；

vi.如果没有文档依据，可以根据类似软件的一般特性来说明是否存在不一致的地方，来确 认是否是缺陷；

vii.根据用户的一般使用习惯，来确认是否是缺陷；

viii.与设计人员、开发人员和客户代表等相关人员探讨，确认是否是缺陷；

## 24.Bug 的生命周期有哪些？

New：新建的 Bug，未经评审需要决定是否指派给开发人员进行修改。

Open：确认是 bug 后，并且认为需要进行修改，指派给相应的开发人员。

Fixed：开发人员修改后标识为修改， 需测试人员回归测试验证。

Delay：确认为暂时不需要修改后续需要修改的 bug

Closed：修改的 bug 经测试人员进行回归测试通过验证， 关闭 bug。

Reopen：经验证后发现 bug 未修改或未修改完， 则重新打开， 让开发人员重新修改

## 25.线上反馈 bug，你是如何处理？

原则： 所有线上问题， 记录在案。 至于具体形式可 bug 系统，可 Excel 。

至于如何收集线上反馈，几个建议：

1. 测试对接客户

2. 运营对接客户，测试对接运营

3. 客服对接客户

4. 外部反馈收集系统，每日分析。

关于线上问题， 常见有以下几类： 1. bug 2. 操作不当 3. 需求 4. 外部原因。 需定期出 bug 分 析报表，同步全公司，特别是相关人。

具体处理措施

1 、 重现问题， 跟开发、项目负责人，讨论应急方案。比如回滚上一版本，回滚后再在线 上环境测试一遍，

2、了解用户的使用场景（使用的浏览器版本、操作系统、问题产生的时间、问题产生的 前后时间用户都操作了什么）

3、同步给开发， 并查找用户提供的时间点的线上日志

4、同步在测试环境进行复现，若无法复现则在详细的咨询下用户，尽可能复现；若复现则 把原因反馈给开发

5、开发修复后验证上线、并由客服人员告知问题已解决， 需表示一定的歉意

6、将反馈的 bug 入库归档，分析 bug 产生的原因， 如人为漏测、修改的其他问题引发的新 问题

7、分析原因，是漏测导致还是什么原因？ 完善测试用例

## 26 如果让你一个人接手测试新项目， 你会怎样着

手准备？

熟悉新项目

对于 APP 要尽量多去使用，先从视觉上熟识每一个页面， 每一个操作选项：

理解核心逻辑流程 ，这个应用的核心功能是什么？ 是否包含注册流程？ 怎么成为会员？

阅读需求文档 ，理解这个项目的现在和过去：这个项目的初心是什么？现在做到了什么地 步？未来方向是什么？

查看设计文档 ，看图识物，视觉效果可以清楚地知道这个应用有哪些页面，结合第一步可 以提升对应用的整体认识。

查看接口文档 ，看过了应用的颜值，是时候深入了解它的后端设计，框架组成，对于后续 定位排查问题很有帮助。

了解数据库 ，输入每一个表的用途， 最好牢记关键表的每个参数

设计测试用例，对于应用的前后端都有一定了解后， 可以开始设计测试用例， 把学到的内 容整理输出，进而查漏补缺

测试练习，学的再多不实践也没用。直接上手测试是加深理解的最有效办法。测试过程记 录问题，然后逐个确认。

编写测试报告，执行完一轮整体测试后，需要把测试过程中遇到的问题记录下来。包括 bug、确认的问题、问题定位排查方法、测试资源，化零为整， 将单个方法按照逻辑形成合 集生成一套方案，以便后续提升测试效率。

控制项目节奏，保障项目按预期进度开始和结束测试。

## 27 开发和产品压榨测试时间如何处理？

1 、沟通确认是否可以延期，以争取测试时间。

2 、排优先级。按照重要性和风险排个优先级， 优先测重要的和风险大的特性、功能点。

3 、裁剪需求。部分不重要的特性是否可以裁剪，不进行测试， 此方法要和 2 结合一起开展， 并且要和相关利益人进行沟通确认。

4 、降低全部或部分特性**\**功能点的质量要求。也就是降低测试的强度。与 3 一样， 此方法 也要和 2 结合一起开展，并且要和相关利益人进行沟通确认。

5 、加人啦！这个选择要慎重啦， 加个老手还稍微好一点， 但也会增加沟通成本。要是加个 对这个项目和这个特性不熟的人，那考虑到培训成本和沟通成本，可能进度会比原来还慢。 总之， 中途加人要慎之又慎。

6 、加班。这个是个很无奈的选择。

## 28 时间有限，测试人员应该先测什么后测什

么？

迅速找出重要程序问题：

先测试变更部分，再测试没有变化的部分；

先测试核心功能，再测试辅助功能；

先测试能力，再测试可靠性（先能用，再检查不同条件的表现）；

先测试常见情况，再测试少见情况；

先测试常见威胁，再测试罕见威胁（压力和错误情况）；

先测试影响大的，再测试影响小的（出现失效情况时，产生大量破坏的产品部件）；

先测试最需要的，再测试没有要求的（对团队其他人有重要意义的任何部分的任何问 题)。

## 29 版本紧急发布， 时间不够，怎么做？

1.上报风险，列举紧急发布版本的风险，以便上级考虑（直属领导和项目的利益相关人员） 2.确定当前版本核心业务和主要功能，需要和项目负责人 ，产品 ，业务方明确所能接受的 最低测试程度

3.首先保证正常流程能通，确保模块基本流程和重要功能用例覆盖

将不影响主要功能和用户非首要关注的部分只做功能验证，用户体验可能考虑后续版本优 化

4.适当精简测试用例集， 用测试点代替详细的测试用例，

5.建议开发做一定的单元测试

6.保存出现 bug 的截图和 log，以便开发定位问题，减少定位时间

7.加班不能少，或者寻求人力资源调整

## 30 避免出现测试时间不够的情况，可以采取哪

些措施？

1.需求确认阶段多与项目负责人沟通，快速掌握原型和需求，计算出完整测试所需时间， 需求评审时积极主动提出测试所需时间， 合理正确更多的时间进行测试，减少测试时间被 挤压或不被重视；

2.重新审视需求，缩小测试范围，减少工作量，告知测试负责人和项目负责人，在相关测 试报告中体现出因时间压缩漏测的部分功能点可能带来的风险；

3.识别任务优先级和重要性；优先完成核心模块和重点任务，保证业务流程正确无误，注 意模块间的连通性；

4.降低非关键模块的测试强度，根据个人的测试经验去关注容易出问题的功能点，提高测 试准确度， 保证工作量不会太难看；

5.在需求评审后尽快编写出相关的测试大纲设计和具体的测试用例，在测试过程中再根据

实际情况稍作调整， 分离测试过程， 保证测试有序进行；

6.提缺陷时尽量附上截图、日志信息、接口传参和返回值等等信息，文字描述清晰，有必 要的直接录视频上传到缺陷管理系统。否则开发看不明白，经常追着你问重现步骤，也是 很影响工作效率的；

7.在测试小组内， 进行员工激励，提高工作热情；

后期改进：

1.1 识别风险，增加风险应对策略，预留风险应对时间；

1.2.优化项目流程，加强控制，避免意外情况的发生；

1.3.引入技术和工具， 比如自动化测试，提高测试效率，缩短测试时间；

1.4.加强项目组内沟通，提高沟通效率，缩短因沟通引起的延迟时间；

## 31 怎样避免发版后，线上出 BUG ？

1. 代码层面控制， 避免有问题的代码产出（责任人：研发 Leader）

2. Git 代码分支控制，所有合并到主分支的代码，必须 Code review 通过才允许合并 。

3. 代码版本 Owner，每个版本的最终代码质量，有一个最终把关人

4. 测试用例， 覆盖度（责任人： 业务测试工程师）

5. 回归测试

6. 接口自动化测试

7. 版本质量 Owner，每个版本， 每天业务线， 有一个对最终质量负责的业务测试 。

8. 线上监控

## 32 如果让你来做测试管理，你会怎么做？

**1.** 管人

a. 招聘新人，找到合适的人，新人是需要做入职培训。让新人快速融入到团队中去， 和开 发和测试打成一片。

b. 与团队成员多做有效沟通，了解每个人真正诉求，根据诉求和公司的要求做一个管理上 的平衡。

c. 设定 KPI，合理的激励， 奖惩分明。

d. 团建，增进了解。

**2.** 理事

a. 做好日常项目管理工作，监控项目的方方面面， 保证公司的项目开发和质量要求。

a1. 团队成员往往以测试组的名字接不同的测试项目，需要全局把握项目的进展， 比如说：

需求是否了解足够充分， 测试用例设计的是否足够充分， 项目执行过程中是否有什么风险

b. 根据每个人的诉求和特点，合理的安排每个人的工作（喜欢挑战的，就给一些挑战性的 工作； 不喜欢挑战的， 就给一些日常性的工作）。

## 33 优秀测试工程师身上所具备的素质&能力

1 、 工作提前做计划，有条不紊的进行， 主动推进项目进展， 日清日结。

2、积极主动跟进产品的每一个迭代，从参与需求评审，到预估测试时间，关注开发进度， 提测后的日常 bug 管理，回归测试， 产品上线前后各事项跟进，到线上用户问题反馈跟进 等等。

3 、 推进产品如期保质上线，上线后，除已知非严重遗留问题外， 无其他严重问题。

4 、 产品上线后，要多使用我们自己的产品，毕竟测试环境和线上环境不一样，用户使用 场景更复杂更随机

5 、 利用空余时间，想办法提高工作技能，学习业内先进技术，提高工作效率。

6 、 日常工作中， 多跟开发，产品， 运营交流沟通，有助于更好的完成测试工作。

7、组内成员多沟通，不要小看每个人，多学他人的优点。 最后，非常重要的一点是责任 心，一般有责任心的人自我驱动力都很强，工作自然出色。大家不妨看看自己周围，这样 的员工大家都很喜欢。

8 、 踏实细心和积极主动

9、代码能力，积极学习各项新技术

10、持续不断的自我提高和总结能力

11、与他人良好交流的能力

12 、 强烈的责任感

## 34 测试用例设计的方法， 你用到过哪些？举例

子说明一下，等价类， 边界值等方法是怎么用的？

**### 1.**等价类

等价类：把输入（特殊情况下才考虑输出）划分为若干个等价类， 从每个等价类中选取一 个测试用例来进行测试，如果这个测试用例通过，那就说明这个测试用例所代表的等价类 测试通过。为了解决测试用例太多， 输入没有办法穷举的情况。

（**1**）有效等价类：根据需求规格说明，有意义的输入的数据集合，成为有效等价类 （**2**）无效等价类：根据需求说明，不符合需求的

**==**等价类举例子**==**

注册邮箱的前缀需求：

6~18 个字符、字符类型

根据长度划分等价类：

有效等价类：6~18 位

无效等价类：小于 6 位、大于 18 位

根据字符类型划分等价类：

有效等价类：A~Z 、a ~z、大小写混用

无效等价类：汉字、特殊字符、标点符号、空格、A~Z 或者 a ~ z 和其它字符混用

**### 2.**边界值

边界值:边界值分析法就是对输入或者输出的边界值进行测试的一种黑盒测试方法,通常边界 值分析法是作为对等价类划分方法的补充,这种情况下,其测试用例来自等价类的边界

**\*\***还是以注册网易邮箱前缀为例： **\*\***

当用户名长度为 6 、18 个字符时，测试用例是否还可以通过

或者当用户名长度为 5 、7 、17 、19 个字符时， 测试又会出现什么情况

**### 3.**因果图

因果图： 是一种简化的逻辑图（恒等、与、或、非）。能直观的表明程序输入条件(原因)和 输出动作(结果)之间的相互关系。

恒等： 如果原因为真,那么结果必为真

与：只有当原因都为真的时候结果才为真

或：只要有一个原因为真,那么结果就为真

非：只有原因为假,结果才为真

用因果图设计测试用例的步骤：

（1）分析所有的输入、输出

（2）找出输入输出之间的逻辑关系

（3）根据输入输出之间的关系画因果图

（4）根据因果图画判定表

（5）根据判定表设计测试用例

**==** 因果图设计测试用例举例**==**

淘宝双十一活动，订单已提交， 订单合计金额大于 300 元或有红包， 则进优惠。

（**1**）分析所有的输入、输出

输入： 金额大于 300；金额小于等于 300；有红包；没有红包； 订单已提交； 订单没有提交

输出： 有优惠； 没有优惠

（**2**）找出输入输出之间的逻辑关系：

订单提交， 金额大于 300，有红包 ---------- 有优惠

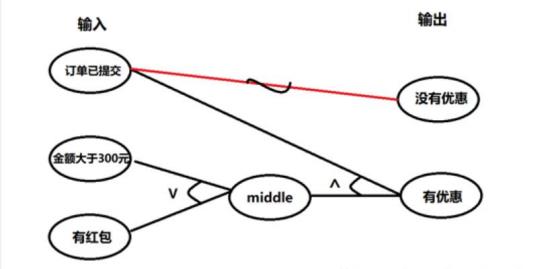
订单提交， 金额大于 300，没有红包---------有优惠

订单提交， 金额小于等于 300，有红包-------有优惠

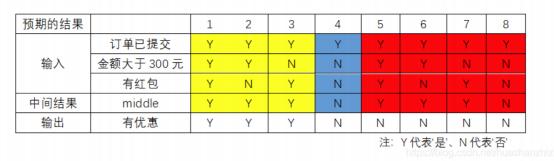
订单提交， 金额小于等于 300，没有红包-----没有优惠

订单未提交-----没有优惠

（3）根据输入输出之间的关系画因果图:



1. 根据因果图画判定表：



注意： 红色部分， 订单已提交应该都是 N

（5）根据判定表设计测试用例:

订单提交， 金额大于 300，有红包 ----------有优惠

订单提交， 金额大于 300，没有红包---------有优惠

订单提交， 金额小于等于 300，有红包-------有优惠

订单提交， 金额小于等于 300，没有红包-----没有优惠

订单未提交，金额大于 300，有红包---------没有优惠

订单未提交，金额大于 300 ，没有红包-------没有优惠

订单未提交，金额小于等于 300，有红包-----没有优惠

订单未提交，金额小于等于 300，没有红包----没有优惠

**### 4.**正交试验法

正交试验法：研究多因素水平的一种测试方法。根据正交性，从输入组合当中选取最优的 组合进行测试，分析结果。通过这些最优的组合得出来的实验结果来分析这个是实验结果。 解决因果图法因测试用例过多而导致的不足。

因素： 测试用例考虑到的变量

水平： 测试用例考虑到的变量的取值

正交表的构成：

行数： 正交表中的行的个数.用 N 表示。

因素数：正交表中列的个数.用 C 表示。

水平数：任何单个因素能够取得的值的最大个数.用 T 表示。

正交表的表示形式： L = 行数(水平数\* 因素数) L = N ( TC )

正交表的两个性质：

每一列中各数字出现的次数都一样多。

任何两列所构成的各有序数对出现的次数都一样多。

正交法设计测试用例的步骤：

1、确定所有的输入（变量）

2、确定每个变量的取值

3、选择一个合适的正交表

4、把变量的值映射到表中

5、写测试用例， 正交表的每一行就是一个测试用例

6、加上你认为可能出现且没有在表中出现的用例组合

**==**正交设计法案例**==**

以注册一个账号为例。

因素： 姓名、邮箱、密码、确认密码、验证码

水平： 填写、不填写

（1）确定所有的输入（变量）：

因素数：5

水平数：2

（2）确定每个变量的取值:

正交表的列 = 因素数 = 5

正交表的行 = （水平数 - 1）\* 因素数 + 1 = 6

（3 、4）选择一个合适的正交表 + 把变量的值映射到表中:

（5）写测试用例:

①姓名填写， 邮箱不填写，密码填写， 确认密码不填写， 验证码填写

②姓名不填写， 邮箱填写， 密码不填写，确认密码填写， 验证码不填写 ③姓名不填写， 邮箱不填写，密码填写， 确认密码不填写，验证码不填写 ④姓名填写， 邮箱填写， 密码不填写，确认密码不填写， 验证码填写

⑤姓名填写， 邮箱填写， 密码填写， 确认密码填写， 验证码不填写

⑥姓名填写， 邮箱不填写，密码不填写， 确认密码填写， 验证码填写

（6）加上你认为可能出现且没有在表中出现的用例组合:

想到就写

**### 5.**场景设计法

场景设计法：把使用场景中的每一个功能点提出来， 考虑功能点可能不同的情况，根据这些情况去设计测试用例。

**==**场景设计法案例**==**

以 ATM 取款为例

ATM 取款正常流程：

插卡----输入密码----输入金额----取钱----退卡

异常情况

:

（1）插卡：卡插反了、卡消磁了、插入其他卡（公交卡） 卡丢失、卡注销、停留时间过长、 卡被吞

（2）输入密码： 密码连续三次输错，账户被锁定； 密码连续输错，第三次输入正确；忘记 密码。手机重置密码

（3）金额大于银行卡余额；ATM 机本身月不足； 输入的金额低于 ATM 机要求的金额； 输 入零钱（ATM 不允许）；超过每日最大金额

（4）长时间未取（看 ATM 机器的处理）；遗忘了部分钱没有取

（5）其他： ATM 网络异常，断电了， 机器故障

**### 6.**错误猜测法

错误猜测法：根据测试人员的知识，经验直觉去判断哪一个模块会出现问题， 专门针对这 个模块进行测试用例的编写。作为一种补充的设计测试用例的方法，适用于经验丰富的开 发人员。

**==**错误猜测法案例**==**

以注册为例

1、校验中特殊字符空格的处理?

2、密码校验中的大小写？

3、姓名中的特殊字符？

4、密码发送是否明文

## 35 白盒测试的用例设计方法有哪些?

白盒测试的目的：

通过检查软件内部的逻辑结构，对软件中的逻辑路径进行覆盖测试；在程序不同地方设置

检查点，检查程序的状态，以确定实际运行状态与预期状态是否一致。

**\*\*** 白盒测试的方法：大致分为静态方法和动态方法两大类。 **\*\***

**\*\*A.** 静态分析： **\*\***

是一种不执行程序而进行测试的技术。静态分析的主要目的是检查软件的表示和描述 是否一致,没有冲突或者没有歧义。

**\*\*B.** 动态分析： **\*\***

当软件系统在模拟或真实的环境中执行前、过程中和执行后,对其行为分析。它显示了 一个系统在检查状态下是否正确。 \*\*在动态分析技术中,最重要的技术是路径和分支测试。 \*\*下面要介绍的六种覆盖测试方法属于动态分析方法。 \*\*

**\*\***六种覆盖标准发现错误的能力呈由弱到强的变化**\*\***

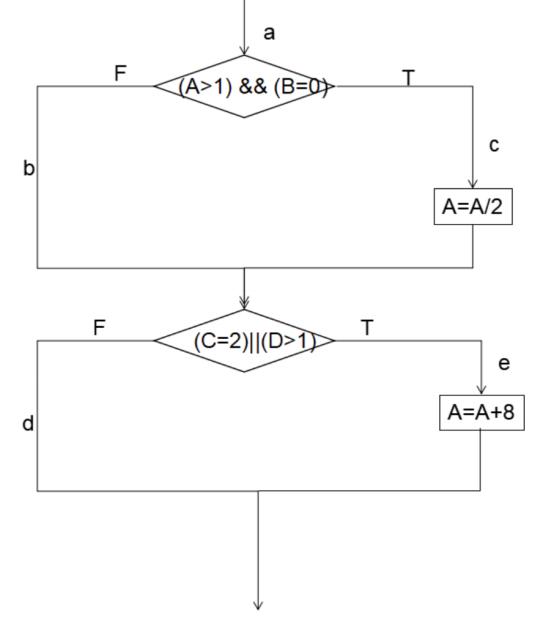
语句覆盖：每条语句至少执行一次。

判定覆盖：每个判定的每个分支至少执行一次。 （包含语句覆盖， 每个判断 T 、F 各一次） 条件覆盖：每个判定的每个条件应取到各种可能的值。 （包含语句覆盖， 每个条件 T 、F 各 一次）

判定**/**条件覆盖：同时满足判定覆盖和条件覆盖。

条件组合覆盖：每个判定中各条件的每一种组合至少出现一次。

路径覆盖：使程序中每一条可能的路径至少执行一次



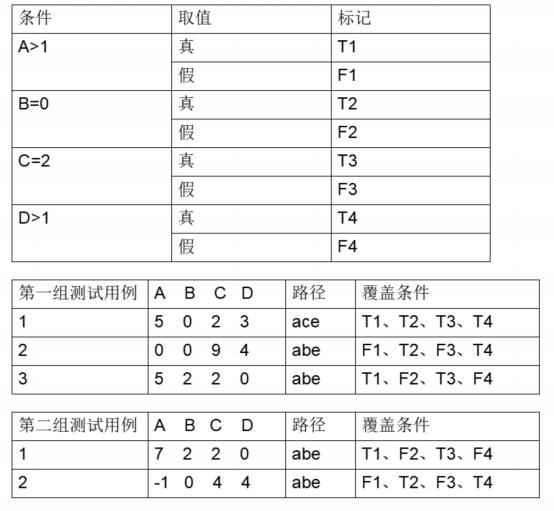
语句覆盖法：

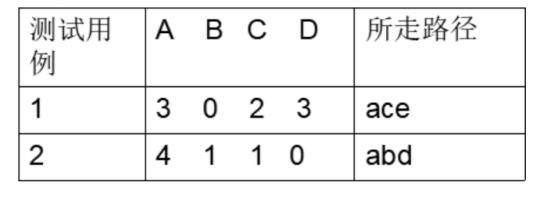
语句覆盖法是设计若干个测试用例，然后运行程序， 使程序的每个可执行语句至少执行一 次



判定覆盖法：

设计若干个测试用例，然后运行程序，使程序中每个判断的取真分支和取假分支至少被执 行 一 次。



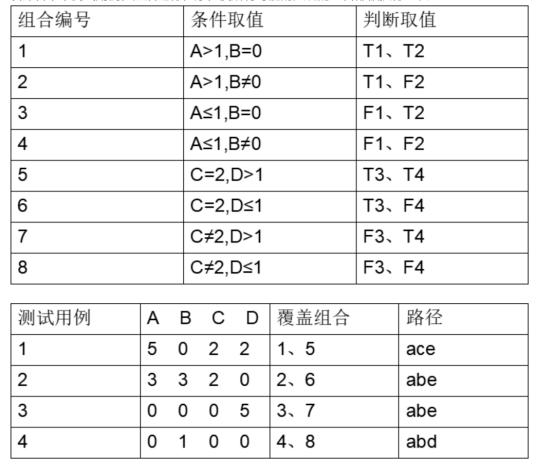
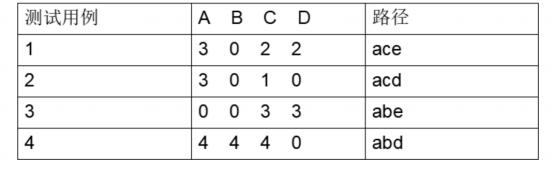


条件覆盖法：

设计若干测试用例，然后执行程序，使程序中每个判断中每个条件的每个可能取值至少满 足一次

判定条件覆盖法：

设计若干测试用例，然后运行程序，使判断中每个条件的可能取值至少出现一次，且每个 判断的判定结果也至少出现一次





条件组合覆盖法：

设计若干个测试用例，然后运行程序，使所有可能的条件的组合都被执行一次

路径覆盖法：

设计若干测试用例然后执行程序，覆盖程序中所有可能的路径

## 36 测试用例的设计考虑哪些方面？

**\*\***界面测试**\*\*** （控件布局是否展示完整、对话框 和 文本框 和 滑动滚轮 和 选择按钮 和 快 捷键的默认值、大小、位置 ）

**\*\***功能测试**\*\*** （接口测试/单元测试）

**\*\*** 易用性测试**\*\*** （UI 组织和布局合理，从一个功能转到另一个功能用户可以轻松转换， 下 一步要什么都应该明显，菜单窗口等都在明显的位置，没有多余的功能，用户如果尝试失 败，可以从帮助系统中获得帮助等等。）

**\*\***性能测试**\*\*** ：压力测试、监控资源消耗、弱网测试

（性能测试：吞吐量：简单来说就是每秒钟完成的事务数或者查询数。通常吞吐量大表明 系统单位时间能处理的请求数越多， 所以通常希望 TPS 越高越好

响应时间： 服务处理一个请求或一个任务的耗时, 或者说一个请求发送出去到接收到响应所 耗费的时间

错误率：一批请求中结果出错的请求所占比例）

（弱网测试：在 2G, 3G, 4G, 5g 网络下测试丢包数和响应时间

或者在突然断网的情况下测试有没有能及时的保存进行保存）

**\*\***兼容性测试**\*\***

（不同手机操作系统

不同的软件版本

不同手机版本

不同的分辨率设备

不同的语言）

**\*\***安全性测试**\*\***

（比如通过抓取数据包来判断是否有关键数据泄露的风险

抓包 包括登录注册模块, 记住密码模块, 文件上传模块, 修改密码模块, 支付模块等等

数据库安全包括身份验证、验证码、会话管理、权限管理、敏感信息传输、安全审计、信 息泄露、输入校验、输出编码、上传下载、异常处理、注释代码等）

## 37 bug 描述的要素有哪些？

（1）bug 标题（问题描述）

（2）bug 测试环境（所属版本， 所属模块）

（3）bug 优先级

（4）bug 类型

（5）可重复性（是否好复现）

（6）操作步骤（通过对什么样的操作，进行了什么 样的步骤）

（7）预期结果

（8）实际结果

## 38 app 测试和 web 测试的区别

单纯从功能测试的层面上来讲的话，APP 测试、 web 测试 在流程和功能测试上是没有区别 的。

**\*\***系统架构**\*\***方面：

web 项目， 一般都是 b/s 架构， 基于浏览器的

app 项目，则是 c/s 的，必须要有客户端， 用户需要安装客户端。

web 测试只要更新了服务器端，客户端就会同步会更新。 App 项目则需要客户端和服务器 都更新。

**\*\***性能**\*\***方面**:**

web 页面主要会关注响应时间

而 app 则还需要关心流量、电量、CPU 、GPU 、Memory 这些。

它们服务端的性能没区别，都是一台服务器。

**\*\***兼容**\*\***方面：

web 是基于浏览器的，所以更倾向于浏览器和电脑硬件， 电脑系统的方向的兼容 app 测试则要看分辨率，屏幕尺寸， 还要看设备系统。

web 测试是基于浏览器的所以不必考虑安装卸载。

而 app 是客户端的， 则必须测试安装、更新、卸载。除了常规的安装、更新、卸载还要考

虑到异常场景。包括安装时的中断、弱网、安装后删除安装文件

此外 APP 还有一些专项测试：如网络、适配性。。。

**\*\*APP** 测试特点**\*\***

（除了按需求说明书外的 功能测试 之外还需要进行如下测试）

1： 适配性测试（也叫兼容性测试， 不同的安卓版本，不同厂商， 不同手机品牌） 2： 不同网络测试 （2G 网络/3G 网络/4G 网络/WIFI 网络）

3： 在线升级测试

4： 中断测试（电话、短中消息打扰）

5： 耗电量测试

6： 弱网测试（信号差，信号屏蔽实验室）

7： 安装卸载 （C/S）

8: 流量测试

## 39 什么是接口？

接口测试主要用于外部系统与系统之间以及内部各个子系统之间的交互点，定义特定的交 互点，然后通过这些交互点来，通过一些特殊的规则也就是协议，来进行数据之间的交互。

**2.** 接口都有哪些类型？

接口一般分为两种：

1.程序内部的接口

2.系统对外的接口

系统对外的接口：比如你要从别的网站或服务器上获取资源或信息，别人肯定不会把数 据库共享给你， 他只能给你提供一个他们写好的方法来获取数据， 你引用他提供的接口就 能使用他写好的方法，从而达到数据共享的目的。

程序内部的接口：方法与方法之间，模块与模块之间的交互，程序内部抛出的接口，比 如 bbs 系统，有登录模块、发帖模块等等，那你要发帖就必须先登录， 那么这两个模块就 得有交互， 它就会抛出一个接口，供内部系统进行调用。

**3.**接口的分类

**1.webservice** 接口

**2.http api** 接口

webService 接口是走 soap 协议通过 http 传输， 请求报文和返回报文都是 xml 格式的，我 们在测试的时候都用通过工具才能进行调用， 测试。

http api 接口是走 http 协议， 通过路径来区分调用的方法， 请求报文都是 key-value 形式

的，返回报文一般都是 json 串， 有 get 和 post 等方法，这也是最常用的两种请求方式。

json 是一种通用的数据类型， 所有的语言都认识它。（json 的本质是字符串，他与其他语 言无关，只是可以经过稍稍加工可以转换成其他语言的数据类型，比如可以转换成 Python 中的字典， key-value 的形式，可以转换成 JavaScript 中的原生对象， 可以转换成java 中的类 对象等。）

接口的本质是什么？

接口你可以简单的理解他就是 URL，工作原理就会说 URL 通过 get 或者 post 请求向服务器 发送一些东西， 然后得到一些相应的返回值， 本质就是数据的传输与接收。

什么是接口测试？

接口测试是测试系统组件间接口的一种测试。接口测试主要用于检测外部系统与系统之间 以及内部各个子系统之间的交互点。测试的重点是要检查数据的交换，传递和控制管理过 程，以及系统间的相互逻辑依赖关系等。简答的说就是通过 URL 像服务器或者其他模块等， 传输我们想传输的数据， 然后看看他们返回的是不是我们预期想要的。

## 40 为什么要做接口测试？

①.越底层发现 **bug**，它的修复成本是越低的。

②.前端随便变，接口测好了，后端不用变，前后端是两拨人开发的。

③.检查系统的安全性、稳定性，前端传参不可信，比如京东购物，前端价格不可能传入-1 元，但是通过接口可以传入-1 元。

④.如今的系统复杂度不断上升，传统的测试方法成本急剧增加且测试效率大幅下降， 接口 测试可以提供这种情况下的解决方案。

⑤. 接口测试相对容易实现自动化持续集成， 且相对 UI 自动化也比较稳定，可以减少人工 回归测试人力成本与时间， 缩短测试周期，支持后端快速发版需求。接口持续集成是为什 么能低成本高收益的根源。

⑥. 现在很多系统前后端架构是分离的， 从安全层面来说：

(1)、只依赖前端进行限制已经完全不能满足系统的安全要求（绕过前面实在太容易）， 需

要后端同样进行控制，在这种情况下就需要从接口层面进行验证。

(2)、前后端传输、日志打印等信息是否加密传输也是需要验证的，特别是涉及到用户的 隐私信息， 如身份证，银行卡等。

怎样做接口测试？

由于我们项目前后端调用主要是基于 http 协议的接口，所以测试接口时主要是通过工具或 代 码 模 拟 http 请 求 的 发 送 与 接 收 。 工 具 有 很 多 如 ：postman 、jmeter 、soupUI 、 java+httpclient 、robotframework+httplibrary 等。

也可以用接口自动化来实现，就是用代码实现，框架和 UI 自动化差不多， 发送请求用断言 来判断。

接口测试点是什么？

目的： 测试接口的正确性和稳定性；

原理：模拟客户端向服务器发送请求报文，服务器接收请求报文后对相应的报文做处理 并向客户端返回应答，客户端接收应答的过程；

重点：检查数据的交换， 传递和控制管理过程，还包括处理的次数；

核心：持续集成是接口测试的核心；

优点：为高复杂性的平台带来高效的缺陷监测和质量监督能力，平台越复杂，系统越庞 大，接口测试的效果越明显（提高测试效率， 提升用户体验， 降低研发成本）；

用例设计重点：通常情况下主要测试最外层的两类接口:数据进入系统接口（调用外部系 统的参数为本系统使用） 和数据流出系统接口（验证系统处理后的数据是否正常）；

PS：设计用例时还需要注意外部接口提供给使用这些接口的外部用户什么功能，外部用 户真正需要什么功能；

## 41 接口文档八要素：

封面： 封面最好是本公司规定的封面，有 logo，内容标题，版本号，公司名称，文档产生 日期；

修订历史：表格形式较好些，包括：版本、修订说明、修订日期、修订人、审核时间审 核人等；

接口信息： 接口调用方式，常用的 GET/POST 方式，接口地址；

功能描述： 简洁清晰的描述接口功能，比如： 接口获取的信息不包括哪些；

接口参数说明：每个参数都要和实际中调用的一样，包括大小写；参数的含义言简意赅 的说明，格式， 是 string 还是 int 还是 long 等格式；

说明部分，说明参数值是需要哪里提供，并详细说明参数怎么生成的，例如时间戳，是

哪个时间段的， 参数是否必填， 一些参数是必须要有的， 有些是可选参数等；

返回值说明：

①最好有一个模板返回值，并说明每个返回参数的意义；

②提供一个真实的调用接口，真实的返回值；

调用限制， 安全方面：

加密方式，或者自己公司一个特殊的加密过程，只要双方采用一致的加密算法就可以调 用接口，保证了接口调用的安全性， 比如常见的 md5；

文档维护：文档在维护的时候，如有修改一定要写上修改日期，修改人，对大的修改要 有版本号变更；

## 42 接口测试有哪些测试点?

**\*\***通过性验证**\*\*** ：首先肯定要保证这个接口功能是可用的，也就是正常的通过性测试，按 照接口文档上的参数，正常传入，是否可以返回正确的结果。

**\*\***参数组合**\*\*** ：现在有一个操作商品的接口， 有个字段 type，传 1 的时候代表修改商品， 商品 id、商品名称、价格有一个是必传的，type 传 2 的时候是删除商品，

商品 id 是必传的，这样的，就要测参数组合了，type 传 1 的时候，只传商品名称能不能修 改成功，id、名称、价格都传的时候能不能修改成功。

**\*\***接口安全： **\*\***

1、绕过验证，比如说购买了一个商品，它的价格是 300 元， 那我在提交订单时候， 我把这 个商品的价格改成 3 元，后端有没有做验证， 更狠点，我把钱改成-3，是不是我的余额还 要增加？

2、绕过身份授权，比如说修改商品信息接口， 那必须得是卖家才能修改，那我传一个普通 用户， 能不能修改成功， 我传一个其他的卖家能不能修改成功

3、参数是否加密，比如说我登陆的接口，用户名和密码是不是加密，如果不加密的话，别 人拦截到你的请求， 就能获取到你的信息了， 加密规则是否容易破解。

4、密码安全规则，密码的复杂程度校验

**\*\***异常验证**\*\***：

所谓异常验证，也就是我不按照你接口文档上的要求输入参数，来验证接口对异常情 况的校验。比如说必填的参数不填， 输入整数类型的，传入字符串类型，长度是 10 的， 传 11，总之就是你说怎么来，我就不怎么来，其实也就这三种，

必传非必传、

参数类型、

入参长度。

**\*\***错误码测试**\*\*** ： 通用的错误码与业务错误码是否能够清晰的说明调用问题，错误码是否 能够尽可能的全的覆盖所有的情况

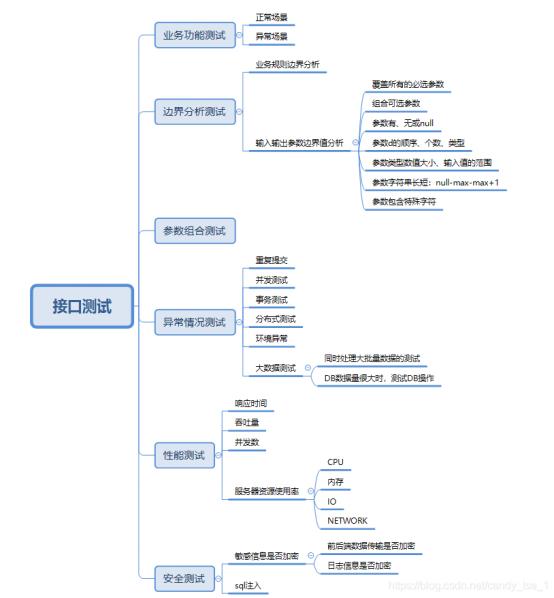
\*\*错误接口请求后，返回值给出友好提示\*\*

\*\*返回值测试\*\* ： 返回值除了内容需要是正确的，还需要类型也是正确的，保证调用方拿 到这些参数能够正确的解析

**\*\***性能测试**\*\***

接口并发情况，如上面提到的： 一个账号， 同时（大于 2 个请求）对最后一个商品下单， 或不同账号，对最后一个商品下单

接口响应时间， 响应时间太长了，肯定需要优化， 一般都是毫秒级别



## 6 你怎么开展接口测试工作？

获取需求文档和接口文档

通过对需求文档分析出接口的业务逻辑要求以及业务边界

通过对接口文档分析出接口的技术指标（接口地址、请求方式、入参、出参） 接口测试用例设计（着重于接口测试数据准备）

使用接口测试工具进行接口测试

接口缺陷管理与跟踪

## 43 接口测试的流程是怎么样的？

接口测试的大概流程

**\*\***熟悉业务需求与接口文档→ 需求评审→ 编写接口测试案例→ 接口用例评审→ 提测后开

始测试→ 提 **bug** 追踪关闭并出具测试报告**\*\***

其实， 和前端测试的流程并无大差， 只是接口测试不仅要看需求文档， 还要看接口文档。

接口测试点。

**\*\***可用性测试**\*\***

根据约定的协议、方法、格式内容， 传输数据到接口经处理后返回期望的结果：

1.接口功能是否正确实现

2.返回值测试-返回值除了内容要正确，类型也要正确，保证调用方能够正确的解析 3.参数值边界值、等价类测试； 错误和异常处理测试

4.输入异常值（空值、特殊字符、超长约定长度等），接口能正确处理， 且按预期响应 5.输入错误的参数，接口能正常处理，按预期响应

6.多输入、少输入参数， 接口能正常处理，按预期响应

7.错误传输数据格式（如 json 格式写成 form 格式） 测试

**\*\***安全性测试，主要指传输数据的安全性： **\*\***

1. 敏感数据（如密码、秘钥）等是否加密传输

2. 返回数据是否含有敏感数据， 如用户密码、完整的用户银行账号信息等 3. 接口是否对传入的数据做安全校验，如身份 ID 加 token 类似校验

4、接口是否防止恶意请求（如大量伪造请求接口致使服务器崩溃）

**\*\***性能测试**\*\*** ，如接口的响应时间、并发处理能力、压测处理情况：

1. 并发请求相同的接口（特别为 POST 请求），接口的处理情况（如插入了相同的记录导致 数据出错， 引发系统故障）

2. 接口响应时长在用户可忍受的范围内

3. 对于请求量大的瓶颈点是否满足当前业务需要

**\*\***接口测试的注意点： **\*\***

很多时候，接口文档不是很规范，我们要按接口文档测试，有问题及时抛出来，要么改代 码要么改接口文档， 不能被开发牵着鼻子走~

2. 接口文档是接口测试的重要依据。没有接口文档， 要找开发或测试经理沟通，要求他们 提供文档。

3. 测试时，优先保证接口功能， 再去验证必输、长度.....好用才是王道， 不能本末倒置~

4. 接口测试案例过多， 建议自动化执行，否则回归测试太耗费人力、时间。会代码最好， 不会的可以用 postman，批量生成测试案例请求~

5. 参数异常的场景一定要测试，保证参数异常时返回的信息符合预期： 易于客户理解、易 于定位问题。

## 44 测接口的关键点有哪些？

功能测试：

接口的功能是否正确实现了

接口是否按照设计文档中来实现（比如 username 参数写为了 user，那么这就不符合，因为 接口文档在整个开发中都需要使用， 所以接口实际的设计要与接口设计文档中保持一致）

**\*\***兼容性测试**\*\*** ： 比如说今天接口进行了调整，但是前端没有进行变更，这时候需要验证 新的接口是否满足旧的调用方式

\***\***错误码测试**\***\*： 通用的错误码与业务错误码是否能够清晰的说明调用问题，错误码是否 能够尽可能的全的覆盖所有的情况

**\*\***返回值测试**\*\*** ： 返回值除了内容需要是正确的，还需要类型也是正确的，保证调用方拿 到这些参数能够正确的解析

参数边界值、等价类测试

**\*\*json** 格式测试**\*\*** ： 通常我们的接口一般设计的都是传递 json 串，那么就需要去测试 如果传递非 json 的情况， 这时候程序会不会正确的处理， 返回相应的 error code

**\*\***默认值测试**\*\*** ： 很多情况一些非必填的参数会有默认值，比如说一个查询的接口，参数 count 为返回查询的结果数量，

默认为 10，那么就应该有一条 case 来测试，当然前置条件是数据库里面必须要存在这样的 数据超过 10 条。

逻辑业务：

是否有依赖业务，比如查看订单，是需要用户首先登录的，所以肯定要保证登录了或有相 应的 cookie

**\*\***业务逻辑测试**\*\*** ： 传递正确的参数，接口对数据库进行查询的操作，需要去验证数据库 查询是否正确， 接口对数据库进行

增删改的操作， 也需要看数据库是否同步进行了这些操作

**\*\***异常测试**\*\***：

异常分为两类， 参数异常和数据异常

\*\*参数异常： \*\*

\*\*关键字参数\*\* ：将参数写为开发语言中的关键字

\*\*参数为空\*\* ：比如去掉了 username 参数

\*\*多或少参数\*\* ：多或者少参数的验证，现在还不确定如果一个接口多了参数如果没有报 错是否是合理的，或者是否需要优化，因为就目前开发给予的答案是，一般不对接口多了 参数的处理

\*\*错误参数\*\* ：比如将 username 参数写为了 user 等看是否能返回相应的 error code 数据异常：

\*\*长度不一致\*\* ：因为数据库中每个字段都设置有字段长度， 填写不符合的长度进行验证 错误数据： 就是将参数的值任意填写，或填写不存在的数值

\*\*异常类型测试\*\* ： 比如 count 参数，这个参数的类型一定是可以转换为 int 类型的， 这时

候我们需要测试如果传的一些不可以转换为 int 类型值来测试代码是否加入判断

**\*\***性能测试**\*\***：

响应时间

吞吐量

并发用户数

占用内存， CPU 等

安全性测试：

**\*\***敏感信息是否加密**\*\***

必要参数是否后端也进行校验（现在很多系统前后端架构是分离的，从安全层面来说，只 依赖前端进行限制已经完全不能满足系统的安全要求（绕过前端太容易了），

需要后端同样进行控制， 在这种情况下就需要从接口层面进行验证）

接口是否防恶意请求（SQL 注入）

cookie：就是将 header 中的 cookie 修改或删除后看是否能返回相应的 error code

header：就是删除或修改 header 中部分参数的值，看是否能返回相应的 error code 唯一识别码：删除修改唯一识别码测试

## 45 怎么做接口测试？

http api 接口

通过工具模拟客户端向服务端发送请求并接受服务器返回的数据来对接口的功能，逻辑业 务，异常， 安全进行测试



功能测试：测试这个接口的功能是否实现，并且测试这个接口是否按照接口文档来进行开 发的（比如说接口文档规定了一些关键字，而开发的时候把关键字改成了其他的关键字， 因为在整个项目周期，并不只有一个开发而是有多个，所以可能因为在开发过程中因为关 键字不一样导致某些开发的功能异常，还有自动化脚本也会发生异常）

逻辑业务，主要指的是一些逻辑业务依赖关系（比如支付宝提交订单的时候要保证你是在 登录的情况下， 如果你没有登录而提交成功了，这就是异常， 可以修改请求的 cookie 来测 试）

异常测试：参数异常：关键字参数（应用其他的关键字替换进行测试）、参数为空、参数多 少（通过添加参数增添个数），参数错误。数据异常：关键字数据（填入的数据用其他的数 据语言的数据替用）、数据长度、数据为空、数据错误。

由于我们项目前后端调用主要是基于 http 协议的接口，所以测试接口时主要是通过工具或 代码模拟 http 请求的发送与接收。工具有很多如： postman 、jmeter、

**\*\***接口测试用例设计思路**\*\***

目的： 测试接口的正确性和稳定性；

原理：模拟客户端向服务器发送请求报文，服务器接收请求报文后对相应的报文做处理并 向客户端返回应答， 客户端接收应答的过程；

重点：检查数据的交换，传递和控制管理过程，还包括处理的次数；

核心：持续集成是接口测试的核心；

优点： 为高复杂性的平台带来高效的缺陷监测和质量监督能力， 平台越复杂，系统越庞大， 接口测试的效果越明显（提高测试效率， 提升用户体验， 降低研发成本）；

用例设计重点：通常情况下主要测试最外层的两类接口:数据进入系统接口（调用外部系统 的参数为本系统使用）和数据流出系统接口（验证系统处理后的数据是否正常）；

PS：设计用例时还需要注意外部接口提供给使用这些接口的外部用户什么功能，外部用户 真正需要什么功能；

\*\*1 输入\*\*

输入参数主要从以下几各方面设计：

**a** 必填项校验

接口文档中有是否必填的说明。参考接口文档即可。

**b** 参数长度校验

参考接口文档即可。

**c** 参数值的有效性校验

如： 身份证号的校验 ，设计的数据虽然符合身份证号的规则， 但是并不是真实有效的身份 证号；这种情况就要看身份证号的校验规则是什么样了，一般都是用的现成的身份证号校 验器，但是有些是自己写的校验算法， 这个本人就遇到过这种问题—校验算法写的不正确； 所以参数有效性的校验就需要结合实际业务场景，判断哪些数据是真实有效的数据， 一定 要确保所有真实有效的数据是可以验证通过的。

**d** 参数组合校验

不同的参数组合可能会存在不同的业务场景；

e 如果参数是枚举值，一定要各种枚举值都要测试， 因为可能不同的枚举走的不同的业务 流程；

f 参数值的默认值的校验

参考接口文档。

g 某些参数具有特定的生成规则，要单独针对生成规则设计用例，一定要保证真实有效的 数据是可以验证通过的。

如身份证号中间几位 19860701，本人就遇到过输入 19861001 这种值校验不正确；

**\*\*2** 接口逻辑**\*\***

接口逻辑我用的设计方法是分支覆盖—>路径覆盖—>场景覆盖, 同样也是要结合实际业务场 景，根本不发生的业务场景就是无效的测试用例。

a 第一步先把业务流程图画出来；

b 依据路程图中的分支分别设计， 不同分支不同的场景，这里就要把异常的场景考虑进去； 如接口超时，接口异常，接口请求成功或失败，成功后怎么处理， 失败后流程是否继续执 行，失败后的数据怎么处理；

以打款接口为例：

打款结果有打款成功或打款失败，成功后怎么处理， 需要回写打款成功状态， 失败后怎么 处理， 也需要回写失败状态，失败后的数据可以操作退回，也可以操作重新出款等等；

c 测试逻辑设计完成后要想一想不同的业务场景怎么去测试，需要哪些人员协助， 如接口超时怎么去测试， 请求重复怎么去测试，请求并发怎么去测试

**\*\*3** 输出**\*\***

输入结果：正常输出和异常输出，常用的方法有错误推断法（列举出程序中可能存在的错 误或者异常，根据他们选择测试用例）

4 以上都完成后，要结合实际的业务场景\*\*去掉冗余的用例\*\* （即实际业务场景不存在的 流程或者输入数据）；

5

5 如果业务流程涉及到状态转换，要单独设计用户—方法：状态转换图；

6 涉及到多个不同金额或者手续费的计算，可能还会用到正交实验法去设计用例；

7 另外，用例设计中还应当包含异常流程中产生的异常数据的处理流程；—通常所说的补 偿机制，这块流程能大大的减轻人工运营的工作量， 当然，这需要在做系统设计的时候就 需要把这部分考虑进去。