**测试工程师**

测试工程师的工作本质上是一个由表及里的工作，如果你还是每次工作都处在和终端用户使用行为几乎一致的流程上，那么只能说明你还不算一名合格的测试工程师。

**测试工程师的几个方面:**测试基础理论,数据库学习,Linux使用,接口测试,性能测试,自动化测试,其它.

1. 能迅速掌握被测软件的业务功能与内部架构，并在此基础上运用各种测试方法，尽可能多地发现潜在缺陷，并能够在已知缺陷的基础上进一步发现相关的连带缺陷。
2. 需要全面的计算机基础知识，还需了解互联网的基础架构、安全攻击、软件性能、用户体验和常见缺陷等知识.
3. 能够使用常见的测试框架或者工具，需要具有一定的自动化测试脚本的开发能力，这可以把你从大量重复的工作中解放出来，然后你才能有时间去做更有意思的工作.
4. 合格的测试工程师关注的是纯粹的测试，而优秀的测试工程师关注更多的是软件整体的质量，需要根据业务风险以及影响来制定测试策略，有效控制测试的时间和成本，并且能够对测试框架以及工具做出适合项目需求的选型.
5. 理解自动化脚本的分层设计、页面对象模型以及业务流程模型，并且能够把这些设计应用到你的测试框架里.
6. 面对大量测试用例的执行，无论是 GUI 还是 API，都需要一套高效的能够支持高并发的测试执行基础架构；再比如，面对测试过程中的大量差异性数据要求，需要统一的测试数据准备平台；再比如，为了可以更方便地和持续集成与发布系统（CI/CD）以解耦的形式做集成，需要统一发起测试执行的接口, 对一些前沿的测试方法和技术有自己的理解，并能够在恰当的时候、因地制宜地把它们应用到实际项目中.

**测试基础理论:**软件开发流程及模型, 软件测试的定义、分类、测试的流程、测试用例编写方法以及缺陷（bug）的管理及生命周期。web端和app的常见测试点和测试方法, 会编写常见的测试文档，如测试计划、软件测试报告等。

**软件测试:** 在规定条件下对软件系统进行审核、运行、评估，检验软件系统是否满足规定需求或者找出预期结果与实际结果之间的差别。为软件产品的质量和评价提供依据。

**软件测试的目的:** 确保软件完成了它所承诺或公布的功能, 确保软件满足性能和效率的要求。确保软件是健壮的、适应用户的环境。提早预防、尽早发现、及时跟踪软件缺陷，满足产品发布需求。

**软件测试如何进行:** 通过手工和自动化方式，利用各种测试工具和管理工具等手段、更早、更快、更多的发现缺陷，并确保这些缺陷得以修复。

**软件测试对象:** 软件程序。与程序匹配的文档。支撑软件运行的配置数据。

**软件测试原则:** 有计划的尽早测试，解决问题的成本越小。测试并不能保证软件100%的没有问题。测试工作的本质都应追溯到用户需求。测试的规模由小而大，从单元测试到系统测试。Good-enough原则，穷举测试是不可能实现的。二八定律：80%的缺陷集中在20%的模块中，经常出错的模块修改后出现错误的概率也会更高。

**软件生命周期:** 从可行性研究到需求分析、软件设计、编码、测试软件发布维护的全部过程。

**软件开发常见模型:** **瀑布模型:** 强调产品定义，各步骤是分离的，前一阶段完成后才能开始后一阶段。缺点：无法回溯，测试在最后运行，惧怕需求变更。

**螺旋模型:**总体思想是一开始不必详细定义所有细节。从小开始，定义重要功能，努力实现这些功能，接受客户反馈，然后进入下一阶段。重复上述过程，直至得到最终产品。缺点：需要经常风险分析。

**敏捷开发:** 敏捷开发以用户的需求进化为核心，采用迭代、循序渐进的方法进行软件开发。在敏捷开发中，软件项目在构建初期被切分成多个子项目，各个子项目的成果都经过测试，具备可视、可集成和可运行使用的特征。

**V模型:**软件开发瀑布模型的变种，把测试过程作为在需求分析、概要设计、详细设计及编码之后的一个阶段，从左到右，描述了基本的开发过程和测试行为。

**W模型/双V模型:** 测试伴随着整个软件开发周期，而且测试的对象不仅仅是程序，需求、设计等开发输出的文档同样要测试。有利于尽早地发现问题。

**H模型:** H模型将测试活动从开发流程完全独立出来，使测试流程形成一个完全独立的流程，只要测试条件成熟了，测试准备活动完成了，测试执行活动就可以进行了。

**敏捷开发过程:** 产品经理收集需求，形成产品待办列表.项目团队参与迭代(sprint)计划会，形成迭代任务.

开发团队进行具体的开发任务，每日站会。开发团队进行成果演示（迭代评审会），产品经理，市场/高层，项目经理参与.评审通过，产品上线。评审不通过，再对产品做修改。开发团队与项目经理进行迭代回顾会，分析好的地方和需要改进的点。

**软件测试分类: 按测试阶段划分,** **单元测试：**完成对最小的软件设计单元模块的验证工作。对代码风格和规则、程序设计和结构、业务逻辑等进行静态测试。白盒测试技术，开发人员自己执行。

**集成测试:**通过测试的单元模块组装成子系统，然后再进行的测试，主要测试内容是接口。集成测试大部分是接口测试和交互测试。

**系统测试:** 将整个软件系统全部集成好之后作为一个整体进行的测试。主要包括一下测试方向：功能测试： 对产品的各功能进行验证，以检查是否满足需求的要求。 性能测试：通过自动化测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能指标进行测试。 安全测试：检查系统对非法入侵的防范能力。 兼容测试： 测试系统在不同的软硬件环境下是否能够正常的运行。

**用户验收/确认测试:** **Alpha测试,**用户在开发环境下进行的测试，Alpha测试是在一个受控的环境中进行的。**Beta测试:**由软件的最终用户在一个或多个用户场所来进行的，开发者通常不在现场，用户记录测试中遇到的问题并报告给开发者。

**对代码的可见划分：黑盒测试：**又称功能测试或数据驱动测试，是针对软件的功能需求/实现进行测试，通过测试来检测每个功能是否符合需求，不考虑程序内部的逻辑结构。**常见的黑盒测试方法：**等价类划分,边界值分析.因果图,错误推测,场景法.

**白盒测试：**白盒测试也称结构测试或逻辑驱动测试，必须知道软件内部工作过程，通过测试来检测软件内部是否按照需求、设计正常运行.白盒测试方法：逻辑覆盖（语句覆盖、分支覆盖、条件覆盖、判定-条件覆盖、条件组合覆盖）、循环覆盖（简单循环、嵌套循环、串接循环）。

**灰盒测试：**灰盒测试介于黑盒测试与白盒测试之间。

**其他测试分类：回归测试：**对软件进行修改之后进行的测试，目的是检验对软件进行的修改是否正确。一是所做的修改达到了预定的目的，也就是确认测试，二是还要保证不影响软件的其他功能的正确性。

**冒烟测试：**在软件中，“冒烟测试”是指测试版本的主要功能，如果能通过测试，才继续进行接下来的其它全功能测试。对一个新版本进行大规模测试之前，先验证一下软件的基本功能是否实现，是否具备可测性。目的是确认软件基本功能正常，可以进行后续的正式测试工作。

**软件测试流程:**

产品需求,需求评审,开发人员写开发计划,测试人员写测试计划与排期,编写测试用例,测试用例评审.开发代码自测,提交测试且代建测试环境,测试人员测试问题反馈开发,开发修复后测试通过,编写测试报告,项目上线.

**测试人员介入流程：**

需求评审,分析需求与编写测试用例,搭建环境与配置数据,执行测试用例,发现bug且提交bug,回归测试,bug追踪与管理,测试报告.

**测试计划:** 软件测试工作正式实施以前，对测试资源、测试时间、测试风险、测试策略、测试范围等方面的分析和规划，保证有序有效的实施测试工作。不同公司根据项目内容和管理方式会有自己专有的模板，内容如下:

1.测试范围与目的. 2.测试方法. 3.测试环境与测试辅助工具的描述.4.测试启动准则与结束准则.5.已递交的文档(<<测试计划>>,<<测试用例>><<测试报告>>)与其预计递交时间. 6.测试人员的组织.7.任务表(测试任务,开始时间,结束时间,执行人员),8.可能存在的问题与对策.

**测试方案:** 测试需求的细化，明确测试测试策略，测试用例的设计方法，测试环境的规划，自动化测试框架的设计，测试工具的设计和选择，测试脚本和测试数据的设计。

**测试计划强调“做什么”。测试方案强调“怎么做”。测试计划属于管理文件，而测试计划是技术文档。**

**测试用例（Test Case）:** 对特定软件的具体功能模块的测试描述，由一组测试输入、执行条件以及预期结果等组成，以便测试某个程序路径或核实是否满足某个特定需求。

**测试用例包括：**用例编号、测试标题、重要级别、预置条件、测试输入、操作步骤、预期结果、实际结果与是否通过.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 项目名称 | 平台系统 | 用例执行人 |  | | | | | | |
| 功能模块名 | 注册登录 | 程序版本 |  | | | | | | |
| 用例设计人 | 梁渡 | 编制时间 |  | | | | | | |
| 功能特性 | 注册账号登录 | 特殊规程说明 |  | | | | | | |
| 测试目的 | 注册登录正常 | | | | | | | | |
| 参考信息 |  | | | | | | | | |
| 用例编号 | 测试对象 | 检查点 | 预置条件 | 用例说明/操作步骤 | 优先级 | 预期结果 | 测试结果/(通过/不通过) | 缺陷编号 | 备注 |
| 注册页面\_001 |  |  |  |  |  |  |  |  |  |

**测试方法:等价类划分法,边界值分析法, 错误推测法,判定表, ,随机测试法.(日常常用前三种)**

**等价类划分：**把所有可能的输入数据划分成若干子集,然后从每一个子集中选取少数具有代表性的数据作为测试数据，就可以用少量代表性的测试数据。分为:有效等价类和无效等价类有效等价类:是指对于程序的规格说明来说是合理的,有意义的输入数据构成的集合。无效等价类：无意义的，不符合需求规定的集合。

如何用等价类划分设计用例？一般有如下几个步骤：

1、划分等价类和非等价类并编号。2、设计组合方式和可能性。3、根据组合选择数据生成测试用例。

例如注册功能：用户名要求：6到10位字符首字母必须是字母或数字，不能为空和汉字。密码要求：6到10位字符，不能为空和汉字。确认密码：与密码一致。划分等价类和非等价类：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 输入条件 | 有效等价类 | 编号 | 无效等价类 | 编号 |
| 用户名 | 6-10位 | 1 | 小于6位 | 8 |
| 首字母为字母 | 2 | 大于10位 | 9 |
| 首字母为数字 | 3 | 首字母不是数字或字母 | 10 |
| 密码 | 6-10位 | 5 | 少于6位 | 12 |
| 特殊字符及字符组合 | 6 | 大于12位 | 13 |
| 确认密码 | 与密码相同 | 7 | 与密码字符不同 | 15 |
| 大小写不同 | 16 |

组合方式：用户名正确、密码正确、确认密码正确，用户名无效、密码正确、确认密码正确

用户名正确、密码无效、确认密码正确，用户名正确、密码正确、确认密码无效

设计一个新的测试用例,使其尽可能多地覆盖尚未被覆盖地有效等价类,重复这步.直到所有的有效等价类都被覆盖为止

**边界值分析：**一般大量的错误都是发生在输入或输出范围的边界上,而不是发生在输入输出范围的内部。

应当选取正好等于,刚刚大于或刚刚小于边界的值作为测试数据,而不是选取等价类中的典型值或任意值作为测试数据。一般要取边界点上的上点、内点和离点。上点：边界上的点。内点：区间内的点离点：离边界值最近且与上点不属于同一等价类的点。对于小数不用考虑离点。如：（0，100]，上点：0、100  内点：50 离点：0,101。

**错误推测法：**基于经验和直觉推测程序中所有可能存在的各种错误, 从而有针对性的设计测试用例的方法。错误推测方法的基本思想: 列举出程序中所有可能有的错误和容易发生错误的特殊情况,根据他们选择测试用例。例如，在单元测试时曾列出的许多在模块中常见的错误。以前产品测试中曾经发现的错误等,这些就是经验的总结。还有输入数据和输出数据为0的情况，这些都是容易发生错误的情况。

**判定表：**分析和表述若干输入条件下，被测对象对这些输入作出相应的一种表格。在遇到复杂业务逻辑时可以用该表理清业务逻辑关系。

条件桩：需求规格说明书所定义的被测对象的所有输入。条件项：针对条件桩所有可能的输入数据的真假值。

动作桩：针对条件，被测对象所采取的操作。动作项：针对条件项的各种取值，被测对象响应的动作规则：任何一个条件组合的特定取值及要执行的相应操作。在判定表中贯穿条件项和动作项的一列就是一条规则。

1、确定规则个数，假如有n个条件.每个条件有两个取值（0,1）,故有2^n种规则。2、列出所有的条件桩和动作桩。3、填入条件项。4、填入动作项，等到初始判定表。5、简化，合并相似规则（相同动作）。

**随机测试法**：随意测试，不考虑任何测试用例和需求，完全站在一个用户的角度对产品进行使用。

适用于：所有之前设定的用例已经执行完毕。海量的条件组合没有办法意义遍历的时候。

**用户登录的测试用例**

**基于等价类划分和边界值分析方法，测试用例包括：**

输入已注册的用户名和正确的密码，验证是否登录成功；

输入已注册的用户名和不正确的密码，验证是否登录失败，并且提示信息正确；

输入未注册的用户名和任意密码，验证是否登录失败，并且提示信息正确；

用户名和密码两者都为空，验证是否登录失败，并且提示信息正确；

用户名和密码两者之一为空，验证是否登录失败，并且提示信息正确；

如果登录功能启用验证码功能，在用户名和密码正确的前提下，输入正确的验证码，验证是否登录成功；

如果登录功能启用验证码功能，在用户名和密码正确的前提下，输入错误的验证码，验证是否登录失败，提示信息正确。

用户名和密码是否大小写敏感；

页面上的密码框是否加密显示；

后台系统创建的用户第一次登录成功时，是否提示修改密码；

忘记用户名和忘记密码的功能是否可用；

前端页面是否根据设计要求限制用户名和密码长度；

如果登录功能需要验证码，点击验证码图片是否可以更换验证码，更换后的验证码是否可用；

刷新页面是否会刷新验证码；

如果验证码具有时效性，需要分别验证时效内和时效外验证码的有效性；

用户登录成功但是会话超时后，继续操作是否会重定向到用户登录界面；

不同级别的用户，比如管理员用户和普通用户，登录系统后的权限是否正确；

页面默认焦点是否定位在用户名的输入框中；

快捷键 Tab 和 Enter 等，是否可以正常使用。-**-以上用例都是直接针对“用户登录”功能的功能性进行验证和测试的。**

**安全性测试用例包括：**

用户密码后台存储是否加密；

用户密码在网络传输过程中是否加密；

密码是否具有有效期，密码有效期到期后，是否提示需要修改密码；

不登录的情况下，在浏览器中直接输入登录后的 URL 地址，验证是否会重新定向到用户登录界面；

密码输入框是否不支持复制和粘贴；

密码输入框内输入的密码是否都可以在页面源码模式下被查看；

用户名和密码的输入框中分别输入典型的“SQL 注入攻击”字符串，验证系统的返回页面；

用户名和密码的输入框中分别输入典型的“XSS 跨站脚本攻击”字符串，验证系统行为是否被篡改；

连续多次登录失败情况下，系统是否会阻止后续的尝试以应对暴力破解；

同一用户在同一终端的多种浏览器上登录，验证登录功能的互斥性是否符合设计预期；

同一用户先后在多台终端的浏览器上登录，验证登录是否具有互斥性。

**性能压力测试用例包括:**

单用户登录的响应时间是否小于 3 秒；

单用户登录时，后台请求数量是否过多；

高并发场景下用户登录的响应时间是否小于 5 秒；

高并发场景下服务端的监控指标是否符合预期；

高集合点并发场景下，是否存在资源死锁和不合理的资源等待；

长时间大量用户连续登录和登出，服务器端是否存在内存泄漏。

兼容性测试用例包括：

不同浏览器下，验证登录页面的显示以及功能正确性；

相同浏览器的不同版本下，验证登录页面的显示以及功能正确性；

不同移动设备终端的不同浏览器下，验证登录页面的显示以及功能正确性；

不同分辨率的界面下，验证登录页面的显示以及功能正确性。

**在绝大多数的软件工程实践中，测试由于受限于时间成本和经济成本，是不可能去穷尽所有可能的组合的，而是采用基于风险驱动的模式，有所侧重地选择测试范围和设计测试用例，以寻求缺陷风险和研发成本之间的平衡。**

**这些问题：**

**1. 测试用例设计看似简单，却总是不够完备；**

**2. 软件的复杂度和更新迭代速度超乎想象，你需要寻找更高效的学习路径；**

**3. 自动化测试已成主流，但你的团队还停留在手工测试，困惑于怎么借助自动化测试工具提升团队效率；**

**4. 大量的测试框架和工具，让你不知所措，不知如何选择最适合自己项目的框架和工具；**

**5. 由 Google 等公司主导的“去 QE，开发自己测试”的模式，让你不清楚自己未来职业该如何规划；。**

**专栏共五大模块，系统阐述了测试工程师必须具备的核心测试理念和技能。**

**• 软件测试基础篇，将系统地讲解软件测试的基本概念、主流的测试方法和测试理念，为后续技术知识的学习打好基础。**

**• 软件测试技术篇，将包括 GUI 自动化测试、API 自动化测试、代码级测试和性能测试技术，通过理论结合实例的方式，为你讲述这四种测试技术的核心知识、设计思想，以及最佳实践。**

**• 测试架构篇，将讲述测试数据准备与测试基础架构的内容，既有解决测试数据准备痛点的方案，又有大型互联网企业测试基础架构设计的最佳实践。**

**• 测试新技术篇，将选取当下比较热门的探索式测试、测试驱动开发、精准测试、渗透测试技术和基于模型的测试，分别讲述它们解决的问题和具体的实现方法。**

**• 测试人员的互联网架构核心知识篇，将从高性能、高可用、伸缩性和可扩展性四个维度对大型网站架构进行深度剖析，弥补测试工程师相较与开发工程师以及测试架构师之间的鸿沟。**

**o 不仅需要从业务本身出发来对软件进行手工测试验证，还需要掌握完整的自动化测试开发技术来设计自动化测试用例.**

**o 作为测试工程师，你就必须掌握设计开发测试基础架构的关键技术。你必须系统性地思考如何才能将测试数据的准备工具化，服务化，最终实现平台化。**

**初级测试工程师：必须具有快速学习的能力，能迅速掌握被测软件的业务功能与内部架构，并在此基础上运用各种测试方法，尽可能多地发现潜在缺陷，并能够在已知缺陷的基础上进一步发现相关的连带缺陷。从知识体系上看，你需要有比开发人员更全面的计算机基础知识，还需要了解互联网的基础架构、安全攻击、软件性能、用户体验和常见缺陷等知识。从测试技术上看，你需要能够使用常见的测试框架或者工具，需要具有一定的自动化测试脚本的开发能力，这可以把你从大量重复的工作中解放出来，然后你才能有时间去做更有意思的工作。**

**中级：合格的测试工程师关注的是纯粹的测试，而优秀的测试工程师关注更多的是软件整体的质量，需要根据业务风险以及影响来制定测试策略，有效控制测试的时间和成本，并且能够对测试框架以及工具做出适合项目需求的选型. 其次，优秀的测试工程师不仅可以娴熟地运用各类测试工具，还非常清楚这些测试工具背后的实现原理，以及多个同类测试工具各自的优缺点和适用场景在遇到问题时，你还需要能够通过二次开发解决工具和框架层面的问题，对于没有合适可用工具的场景，可以自行设计开发一些小工具来更好地展开测试工作。这个阶段，你很有可能会接触到一些代码级的测试，这就要求你具有一定的开发背景，并能够很好地理解代码级的测试技术, 要求你理解自动化脚本的分层设计、页面对象模型以及业务流程模型，并且能够把这些设计应用到你的测试框架里。,**

**测试架构师：配置一套高效的能够支持高并发的测试执行基础架构，准备统一的测试数据准备平台，统一发起测试执行的接口，必须对一些前沿的测试方法和技术有自己的理解，并能够在恰当的时候、因地制宜地把它们应用到实际项目中。**

**“好的”测试用例一定是一个完备的集合，它能够覆盖所有等价类以及各种边界值，而跟能否发现缺陷无关**

**手机APP**

**APP测试要点：**功能测试、兼容测试、流量测试、耗电量测试、性能测试、安全测试、网络测试、稳定性测试等。

**APP功能测试：主**要是依据需求规格和产品说明来验证各项功能，需要关注软件在正常和异常场景下的运行情况。

**UI测试:** (1）界面（菜单、结构、窗口、按钮）等是否满足需求，文字，图片，是否美观统一。(2）程序界面和操作是否友好、易用、易理解。

**安装卸载:** 验证App是否能正确安装、运行、卸载以及操作过程和操作前后对系统资源的使用情况。

**安装:**

1）软件安装后是否能够正常运行，安装目录和文件是否正常建立。

2）在不同系统版本和手机品牌下安装。

3）安装向导UI及功能是否正常。

4）安装过程中取消，下次安装是否正常。

5）安装过程来电，短信，通知，结束后是否继续安装。

6）是否支持覆盖安装。

7）安装空间不足时是否有相应提示。

8）安装后没有生成多余的目录结构和文件。

9）软件安装过程中关机重启，断电，断网的处理机制是否符合需求。

**卸载:**

1）直接卸载app是否有提示。

2）卸载后是否删除相应的安装目录。

3）卸载是否支持取消功能，单击取消后，是否正常可用。

4）卸载过程中死机，断电，重启等，手机恢复后能否正常卸载。

**登录运行:**

**登录:**

1）用户名和密码错误、漏填时，界面有提示信息。

2）自动登录时间失效后，启动app进入登录界面。

3）密码更改后，登录是否正常。

4）用户主动退出登录后，下次启动APP时，应该进入登录界面。

5）切换账号登录，检验登录的信息是否做到及时更新。

6）对于多个端都进行操作时，确保数据库操作无误，且每个端可以及时看到数据的更新。

7）登录界面切换到后台，或其他界面，或者锁屏是否影响登录过程。

8）ios与android设备登录同一个账号，用户数据是否同步。

9）已经登录的账号，再次登录是否强制退出。

10）登录超时的处理是否符合需求。

**运行:**

1)APP安装完成后，是否可以正常打开，是否有加载图示等。

2）APP的运行速度正常，切换是否流畅。

3）用户登录状态太久，sessionId会过期，会出现“虽然是登录状态，系统会提示用户没有登录。

**切换测试:**切换场景包括：app切换到后台、多个app之间切换。

1）app切换到或其他app或者系统界面，再回到app，是否停留在上一次操作的界面。

2）app切换到后台或其他app或者系统界面，再回到app，app是否正常使用。

3）当app使用过程中有电话进来中断后再切换到app，功能状态是否正常。

**消息推送:**设置开关打开状态下，消息推送是否可正常接收（应用启用中和应用关闭时都应该可以收到）

1）推送默认状态，一般默认开关应该是打开状态。

2）推送设置开关，存在“打开”，“关闭”选项。

3）开关打开时，可以收到消息推送，且点击可查看。

4）设置开关关闭时，客户端接收不到消息推送。

5）用户设置了免打扰的时间内，用户接收不到推送。在非免打扰时间段内，用户能正常收到推送。

6）检查推送消息内容与用户账号是否符合。

**升级更新:**

1）当app有更新版本时，手机端有更新提示。

2）当app版本为非强制升级版时，可以取消更新，旧版本能正常使用。用户在下次启动app时，仍出现更新提示。

3）当app版本为强制升级版时，给出强制更新后用户取消更新时，退出客户端。下次启动app时，仍出现强制升级提示。

4）当app有新版本时，直接更新检查是否能正常更新。

5）更新后，检查更app功能是否是新版本。

**中断测试**:app使用过程中突然来电、短信弹出、闹钟、QQ聊天信息、微信、低电量等提示时能否正常使用。

1）当app使用过程中有电话进来中断后再切换到app，功能状态是否正常。

2）当杀掉app进城后，再开启app，app能否正常启动。

3）出现必须处理的提示框后，切换到后台，再切换回来，检查提示框是否还存在，有时候会出现应用自动跳过提示框的缺陷。

**网络测试:**目前手机手机接入的网络主要分为2G、3G、4G、wifi。

1) 无网络时，有切换网络的操作或者提示。

2）网络间切换app断网有相应提示，重新联网后正常使用。

3) 在网络信号不好时，检查功能状态是否正常，确保不因提交数据失败而造成crash。

4) 在网络信号不好时，检查数据是否会一直处于提交中的状态，有无超时限制。如遇数据交换失败时要给予提示。

5) 弱网络下操作是否有提示。

**兼容性测试:** 1）操作系统版本的兼容性（Android各个版本，ios各个版本）2）不同手机品牌的兼容性。

3）手机分辨率兼容性4）网络的兼容性：2G\3G\4G\WIFI,弱网下、断网时5）app跨版本的兼容性。

6）与其他app的兼容性。

**权限测试:** 当权限没有开启时，或友好提示是否允许设置，当允许开启时，跳转到设置界面。

1）有限制允许接入网络提示或选项。

2）有限制允许读写通讯录、用户数据提示或选项。

3）有限制允许相机提示或选项。

4）有限制允许录音功能提示或选项。

5）有限制允许定位功能提示或选项。

**其他手机端特性测试:**1）关机、待机后app能否正常使用。2）手机解锁屏幕后进入进入app是否正常。

3）app在清空数据或强制退后还能正常运行。4） 长时间开机app开启情况下是否会出现异常情况。

5）app运行时关机重启。6）app运行时充电。

**Web端**

**WEB功能测试：** UI及易用性测试、表单测试、cookies测试、链接测试、兼容性测试。

**UI及易用性测试：**

　1）各个页面的样式风格是否美观统一，如图片大小、颜色是否统一，页面、文字、图片是否居中等。

　2）各个页面的标题和描述是否正确，有无错别字，字体大小、颜色是否正确统一，文字描述准确，无歧义。

　3）页面布局统一，美观，间距合理。

　4）操作是否符合人们的常规习惯（有没有把相似的功能的控件放在一起，方便操作）。

　5）提示信息是否正确，鼠标停留到上面是否正常显示提示。

　6）调整分辨率验证页面格式是否错位现象。

　7）窗口的最大化、最小化是否能正确切换。

8）执行风险操作时，有确认、删除等提示。

9）快捷键检查：是否支持常用快捷键，如Ctrl+C Ctrl+V Backspace等。

**cookies测试：**cookies是否正常工作。刷新操作是否影响cookies。cookies是否按预定时间保存。

**表单测试：**表单测试主要是验证对数据的增删改查修改是否正常实现，以及验证码功是否可用。

（1）、注册、登陆、输入信息提交等操作是否正常。

（2）、用户填写的信息是否合理，是否在需求规定的范围内，对于一起日期时间地点等选择是否合理；

（3）、检验默认值的正确性；

（4）、如表单只能接受指定的某些值，测试时跳过这些字符，看系统是否会报错。

（5）、短信验证码、邮箱验证码、字符验证码、图片验证码功能是否正常。

**链接测试：**

（1）、测试所有链接是否按指示的那样确实链接到了该链接的页面；

（2）、测试所链接的页面是否存在；

（3）、保证Web应用系统上没有孤立的页面(所谓孤立页面是指没有链接指向该页面，只有知道正确的URL地址才能访问)。

**兼容性测试：**

1）平台兼容性：不同硬件平台（PC、手机、平板等），不同操作系统（linux、windows、macOS、android、ios等）。

2）浏览器兼容性（IE、360、搜狗、chrome、火狐等）。

**测试报告**

**项目概述：**XXXX平台是一款为政府企事业单位提供音视频业务管理平台，XXXXXXXXX,为政府企事业单位应急处突、活动安保、抗险救灾等提供有力保障。

**测试目标：**

**测试背景：**本次测试需求来源为，目的为了实现XXXXX，验证同时在线1000人在线下。

**接口性能指标：**每秒事务数（TPS）（寻找系统支持最大TPS） 响应时间小于2s 服务器CPU：平均利用率小于70%，瞬时峰值不高于80% 服务器内存：平均利用率小于70% 磁盘IO：平均利用率小于70%

并发用户数 事务处理成功率>=99%，若有任何错误，需给出错误具体原因； 带宽利用率：根据业务需求给出测试结果，无具体指标要求；

**测试范围：**接口范围，业务场景

**测试环境：**硬件配置与软件配置，测试拓扑

**测试策略：**测试策略，负载测试，稳定性测试

**测试数据：**性能指标数据，测试图表

**测试结论及风险分析：**结果分析，风险分析及建议，测试结论

**接口测试**

接口测试：又称API测试，测试系统组件间接口，属于灰盒测试。通过脚本或者工具，模拟客户端对服务端接口进行调用。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。从接口层测试能更早的发现问题，从而提高测试效率，降低修复成本。

http/https协议学习,常见请求方法GET/Post,cookie和session学习,接口的基本概念,接口文档,接口测试用例的编写,接口测试工具postman,jmeter,SoupUI.

**分类：**模块之间的接口：如果订单支付时，需要去用户资金模块查询余额等。系统对外部系统提供的接口,内部子系统的服务之间的接口.

**场景：**web网站的API接口：如微信接口、新浪微博接口等。公司内部系统API：订单接口需要调用用户资金接口查询余额。APP，客户端软件等。

**流程：**分析接口文档和需求文档，编写接口测试计划与测试用例，接口测试执行，输出接口测试报告。

**工具：**postman、jmeter、soupUI、java+httpclient、robotframework+httplibrary等。

**接口测试与自动化测试的区别？**答:自动化测试如app自动化测试、web自动化测试：都是模拟人类的行为去测试，底层都是通过接口去和服务器进行交互,接口测试可以在底层模拟人类的行为去进行测试。

**接口测试与性能测试？**答: 自动化测试的模拟行为，测试效率较慢，接口测试可以直接和服务器进行快速交互，对接口进行性能、压力测试。

**接口测试的价值？**提升测试效率：底层的一个bug能够引发上层的8个左右bug，而且底层的bug很容易引起全网的宕机，接口测试能够提供系统复杂度上升情况下的低成本高效率的解决方案。能快速定位bug.

**接口文档:**

|  |  |
| --- | --- |
| 功能描述 | 处理查询某一时间段的文件请求 |
| 服务协议 | Http+Json |
| 服务地址 | http;// |
| 服务类型 | POST |

**接口测试用例：**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 模块 | 测试点 | url | Header | Method | Body | 预期结果 | 实际结果 |
| Dll\_01 | 登录 | 正常登录 | <http://ip:port:api/login> | Content-Type=application/json | Post | Data:{“userName”:小明} | 1. 返回字段正确 2. 返回的字段内容正确 | Pass |
| Dll\_02 | 登录 | 登录名为空 | <http://ip:port:api/login> | Content-Type=application/json | Post | Username为空 | 提示用户名不为空 | pass |

**完整http的请求过程:**

浏览器(客户端)发起TCP连接请求,TCP三次握手建立连接,客户端将HTTP请求数据包发送给服务端,服务器端响应http请求,浏览器得到html代码,浏览器解析html代码,并请求html代码中的资源,渲染后呈现给用户.

**http中存在如下问题：**请求信息明文传输，容易被窃听截取。数据的完整性未校验，容易被篡改,没有验证对方身份，存在冒充危险.

**HTTPS 协议：** HTTP+SSL/TLS，通过 SSL证书来验证服务器的身份，并为浏览器和服务器之间的通信进行加密。

**SSL**：安全套接字层1994年为 Netscape 所研发，SSL 协议位于 TCP/IP 协议与各种应用层协议之间，为数据通讯提供安全支持。

**TLS：**传输层安全其前身是 SSL，它最初的几个版本（SSL 1.0、SSL 2.0、SSL 3.0）由网景公司开发，1999年从 3.1 开始被 IETF 标准化并改名.

**REST:**表现层状态转移是HTTP协议（1.0版和1.1版）的主要设计者Roy Fielding提出的的一种架构风格。REST 规范中有如下几个核心：REST中一切实体都被抽象成资源，资源通过URI来指定。使用HTTP1.1协议定义的通用动词GET/POST/PUT/DELETE，来实现资源的增删改查操作。服务端和客户端之间用JSON，XML传输文本，或者用JPG，WebP传输图片。

**RESTful API:**RESTful是一种接口设计的架构风格，它不是需要严格执行的标准，而只是提供了一组可用设计原则，通过约定的规范，便于开发、测试理解沟通。越来越多的互联网公司，已经要求接口设计必须按照这种规范执行。

这样的API格式，可以达到如下目标：通过URI，就知道需要操作什么资源,通过http请求方法，就可以知道对资源进行何种操作。通过http 返回状态码，就知道操作的结果。

RESTful API格式如下: http://服务器地址:端口号[/项目名称/版本]/资源集合[/单个资源]

**浏览器在使用HTTPS传输数据的流程:**

首先客户端通过URL访问服务器建立SSL连接。 服务端收到客户端请求后，会将网站支持的证书信息（证书中包含公钥）传送一份给客户端。客户端的服务器开始协商SSL连接的安全等级，也就是信息加密的等级。

客户端的浏览器根据双方同意的安全等级，建立会话密钥，然后利用网站的公钥将会话密钥加密，并传送给网站。

服务器利用自己的私钥解密出会话密钥。服务器利用会话密钥加密与客户端之间的通信。

**HTTPS的缺点:**HTTPS协议多次握手，导致页面的加载时间延长近50%；HTTPS连接缓存不如HTTP高效，会增加数据开销和功耗；申请SSL证书需要钱，功能越强大的证书费用越高。SSL涉及到的安全算法会消耗 CPU 资源，对服务器资源消耗较大。

**HTTPS和HTTP的区别：**https协议需要到CA申请证书，一般免费证书较少，使用需要一定费用。

http是超文本传输协议，信息是明文传输，https则是具有安全性的ssl加密传输协议。

http和https使用连接方式不同，默认端口也不一样，http是80，https是443。

HTTPS协议是SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，比http协议安全。

**Cookie的原理:** W3C组织提出的保存在客户端本地，方便下次访问服务端时直接放到请求报文头中，如登录的cookie，下次访问同一网站时，用户不必再次输入用户名和密码就已经直接登录，所有的主流浏览器如IE、Netscape、Firefox、Opera等都支持Cookie。

**session原理**Session保存在服务器上用来记录用户状态的机制。客户端浏览器访问服务器的时候，服务器把客户端信息以某种形式记录在服务器上。这就是Session。Session在用户第一次访问服务器的时候自动创建。客户端浏览器再次访问时只需要从该Session中查找该客户的状态就可以了。一般客户端和服务端通过一个SessionID来进行沟通，为了防止不同的用户之间出现冲突和重复，SessionID一般是一个32或者48个字节的随机字符串。

**cookie 和session 的区别：**存放位置不同：cookie数据存放在客户端（浏览器）；session数据放在服务器端，一般存储在内存中，但是SessionID存储在客户端cookie中。cookie由浏览器存储在本地，安全有风险，不宜存储敏感信息，如密码等。session会在一定时间内保存在服务器上，访问较多时，影响服务器性能。

**接口的常见类型：**

**webService接口：**它是简单对象访问协议（soap）通过http传输，SOAP=RPC+HTTP+XML，即采用HTTP作为通信协议，RPC（Remote Procedure Call Protocol 远程过程调用协议）作为一致性的调用途径，XML作为数据传送的格式，从而允许服务提供者和服务客户经过防火墙在Internet上进行通信交互，请求报文和返回报文都是xml格式，常使用测试调用工具有SoapUI、jmeter、loadrunner等。

**http api接口：**基于http协议，一般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，请求方法最常见的为get和post方法，常见的测试调用工具有postman、RESTClient、jmeter、loadrunner等。

**Fiddler：**Fiddler被称为抓包神器，它是一个http协议调试代理工具，它能够记录互联网之间的http协议通信，可以设置断点，查看所有“进出”Fiddler的数据（如cookie,html,js,css等文件）。Fiddler 要比其他的网络调试器要更加简单，操作便捷，格式显示清晰，这个工具也可以进行接口测试，但是不能写断言，只能用composer构造HTTP请求，需要人工确认返回结果的正确性。

**Postman：**是谷歌开发的一款接口测试软件，它使用简单，功能强大，能够发送任何类型的HTTP请求 (GET, HEAD, POST, PUT..)，可以进行文件上传、响应验证、变量管理、环境参数管理，还可以批量管理和运行用例，并支持用例导出、导入，可以进行接口的自动化测试。缺点是自动化断言不够强大，不能和jenkins、代码管理库进行持续集成测试。

**soapUI：**是一个开源测试工具，通过soap/http来检查、调用，实现Web Service的接口和性能测试。该工具既可作为一个单独的测试软件使用，也可利用插件集成到Eclipse，maven2.X，Netbeans 和intellij中使用。soapUI pro是soapUI的商业非开源版本，实现的功能较开源的soapUI更多，国外公司使用较多。

**Jmeter：**是一款100%纯Java编写的免费开源工具，主要用来做性能测试，但也可以做接口测试，配合后置处理器与断言，可以满足大部分的接口测试场景，JMeter提供了BeanShell编程能力，可以写出比较灵活的测试脚本，通过jmeter+ant+jenkins可以实现接口和性能自动化测试。

**代码实现**:java代码实现接口测试：java可以通过httpclient.jar包，利用JUnit或者TestNG这样的单元测试工具进行用例编写，结合jenkins实现接口持续集成测试。

**python代码实现接口测试：**python可以利用强大的第三方库Requests实现接口用例编写，结合unittest单元测试框架，生成测试报告，也可以和jenkins做持续集成测试。

**Postman的使用：**最常用的请求方法有GET，POST，PUT和DELETE，它们的区别如下：GET请求：用于查询获取数据。POST请求：创建或更新数据，PUT请求：更新数据，DELETE请求：删除数据，HTTP请求包含四个部分：URL，方法，header头部和body主体。

了解接口测试都包含哪些测试活动，如何整理接口测试需要的输入条件。合成接口测试技术和业务测试知识，让你形成接口测试思维。

会通过案例的方式教你如何从流水账式的接口测试脚本，一步步抽象出属于自己的接口测试框架。随着课程的深入，你会建立建立一套自己的接口测试框架。

在上一个模块的基础上，继续从场景切入，教你学习如何测试一个完全陌生的协议接口，以及如何在测试微服务接口时，用 Mock 梳理混乱的调用关系。当然，这些能力最后都会叠加到你的测试框架中。不断提升它的测试能力。测试框架是你强大的武器，框架数据层的封装则是你的弹药库。所以最后我会教你完成外部依赖解耦的方法和思路，保证你能顺利完成测试任务.

**接口测试的思维开始，教你把业务测试的思维和接口测试的技术结合到一起，合成接口测试的思维，最终让你拥有接口测试的能力，这种能力既包含了工具的使用、代码的编写同时也包含用例的设计.**

项目的影响来说，接口测试直接测试后端服务，更加接近服务器上运行的代码程序，也更能发现影响范围广泛的 Bug。

通过接口自动化完成接口回归测试，让自己的工作更轻松、更高效；

通过持续集成平台调用接口自动化测试，为流水线提供质量保障方法和手段，赋能研发。

接口测试必要性，再讲讲什么是接口、什么是接口测试.

测试三环节:单元测试,接口测试,界面测试.

单元测试并非测试工程师的本职工作，它属于开发工程师的工作范筹.但开发很少做单元测试,故采用智能化框架补充单元测试工作或是加大接口测试,增大接口测试的测试深度和测试广度，往下逐渐覆盖一些公共接口的单元测试内容，往上则逐渐覆盖应该由 UI 层保障的业务逻辑测试.

接口测试更容易和其他自动化系统相结合；相对于界面测试，接口测试可以更早开始，也可测试一些界面测试无法测试的范围，使“测试更早的投入”这句话变成现实；接口测试还可以保障系统的健壮性，使得被测系统更健壮。

接口测试其实就是模拟调用方，比如 Client 端，通过接口通信来检测被测接口的正确性和容错性。

**理想的提测项目:**

**包含前期参与的产品需求、原型设计，由产品经理来提供的；**产品需求。它描述系统的业务逻辑，通过这个文档，你才能知道怎么来设计测试用例；原型设计。它会更加直观地告诉你系统的使用逻辑，这对测试用例的设计、对系统的前期认知都是有辅助作用的。

**包含后端接口文档、代码单元测试脚本，由开发工程师提来供的。**接口文档。它详细地描述了后端接口的访问方式和参数说明，使用这个输入项才能开展接口测试用例的设计、测试脚本的准备和测试数据的构建。单元测试脚本。它是保障提测质量的必要环节，是研发工程师自测的一个有效手段，可以保障提测项目的提测质量。

**现实的提测项目:**一个项目没有接口文档.

拿到一个 SUT 环境的时候，首先就要进行接口测试，因为单元测试不是由测试工程师来完成的，而是由开发工程师编写、并由持续集成系统自动完成执行的。但开发没有给有价值的文档,要开始接口测试,可从工具辅助,分析问题,询问解惑三个步骤完成.

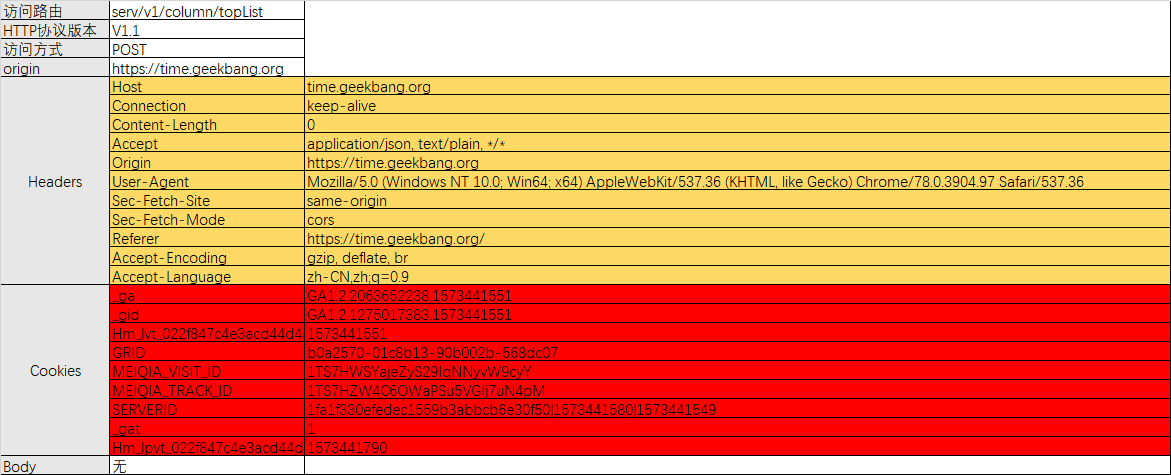
具体的工作模式：借助一些工具的辅助来完成接口分析；通过工具截获一些接口信息；通过分析接口的访问方式、参数等信息整理出一些问题，和研发工程师沟通这些问题，将一些不知道的参数含义、参数取值范围等问题问清楚。

**使用工具Fiddler:**是一个 HTTP 的调试代理， HTTP 协议的抓包工具，运行时会在本地建立一个代理服务，默认地址为 127.0.0.1:8888。当浏览器和被访问的服务之间发生交互，Request（请求）和 Response（响应）都会经由 Fiddler 代理，可截获全部的访问信息流。

先打开Fiddler(需配置抓取HTTPS的网址),然后启动浏览器访问被测项目地址,如<https://time.geekbang.org>,点击Fiddler的Inspectors的标签页,查看Request请求的内容和Response请求的内容.注意一下几个参数:

HOST，它表示指定访问的服务器域名；Connection 的值为 keep-alive，这表示需要持久连接；Accept，它表示客户端可以接受的内容类型为 application/json, text/plain, User-Agent，它说明请求是从什么浏览器发出去的；Sec-Fetch-Site 和 Sec-Fetch-Mode，它们是 JS 中对跨域的一些设置；Accept-Encoding 设置为 gzip、deflate、br，这表示可以支持的 Web 服务器返回内容压缩编码类型；Accept-Language，它表示接受的语言。

其中的 Cookie 的内容，特别重要，因为 Cookie 中传递的参数很多都是用来确认用户身份、鉴定角色权限等需要的参数。通过上面的分析绘制如下的表格.



在表格中，被标注为白色背景的部分，是这次访问的基本信息；被标注为黄色背景的部分，是访问的头信息，同时也是我们已知的内容；被标注为红色背景的部分，就是 Cookie 信息，是我们未知的内容。同时你可以看到，本次消息访问的 body 是空的，是没有内容的。

再看请求的Response信息，这些返回值包含了很多参数，也需要关注这些参数，很多时候，一个接口的返回值会是另外一个接口的入参，也就是我常说的串联业务逻辑上下文的参数。

拿着这张接口信息表，进入了第三步，询问解惑。去搞懂这些参数是干什么，起什么作用，怎样算搞懂，判断如下：

1. 你要搞清楚每一个参数的含义，也就是这个参数对应的实际自然语言的名字，通过记录每一个参数的中文语义，也会让更你容易记住这个函数是干什么的。同时，你也要知道这个参数的赋值是从哪里来的，是从其他页面的返回值中得到的？还是 JS 生成的？如果是其他页面或者接口返回的，那么，是哪一个接口返回的哪个字段？这样，当你开始做接口测试的时候，你就知道去哪里拿到这个参数的赋值了。如果是另一个接口的返回字段，那么，你还需要维护一份返回该参数接口的接口信息文档，以便于自己下一次创建对应的参数，如果不可以创建，那么你就要知道这个参数的生成规则，也要知道如何手动构造它。
2. 参数的作用域：参数的作用域指的是这个参数在这个接口中是做什么用的，它在哪一个访问周期里是一直存在的，它是否导致了业务逻辑分支等。比如说，这个参数是用来验证用户权限吗？它的验证算法是什么？之所以要搞清楚这些内容，是为了你在做接口测试的时候，可以设计更小的参数组合来覆盖更多的业务逻辑，这是测试用例去除冗余的一个很好的方法。
3. 返回值的含义：针对上面一大串的返回 JSON，你要搞清楚在返回值中，每一个 JSON 的 Key 所对应的含义，这样，当你需要和这个接口产生交互的时候，就可以快速地拿到对应参数的含义，完成业务逻辑上下文的参数串联了。

以上借助工具、通过分析问题明确未知参数，也通过询问解决未知参数的中文含义、作用域，以及对应返回参数的中文含义，现在，即使面对没有接口文档的提测项目，也能收集明确的、足够的信息。接下来进行业务逻辑的接口测试。

多个接口之间并不是随意组合的，而是按照业务逻辑、通过数据传递来完成的。要想使用接口测试完成业务逻辑，要制作一个流程中所有接口的接口信息表，同时理清每一个流程的数据流程，数据流程驱动业务流处理，才能开始业务逻辑的接口测试。

问题1：公司项目的单接口、串联接口都已经在做 但是好多时候接口已经测试完 然后开发改了部分接口，导致接口再次自动化回归的时候就又有问题了 这样很被动，请问下如何解决此类问题？

答：变更频繁这个问题目前只能在项目内部解决，服务端工程师应该设计接口，所有相关人评审后就不能变更，要变更需要有人承担成本。这个可以通过引入敏捷模式来进行规避，可以关注一下对应内容。

**如何通过一个实际方法落地接口测试呢？这里面就涉及到怎么做单接口的接口测试，怎么完成业务逻辑接口测试，以及用什么手段来完成接口测试等问题。接下来我会为你详细解答这些问题。**

工具使用Postman（HTTP协议客户端工具）

明确被测系统：xxx。先正确的跑遍流程 “正确”指的是“正确接受合法 Request 入参，正确拒绝非法 Request 入参”。

在postman中选择请求类型，输入请求地址，在body中的raw输入Request构造参数点击发送，查看Response内容。

然后设计测试用例，设计用例偏向于单元测试，即走通流程的用例。接下来是持续集成，使用工具Newman等，将postman的业务逻辑接口测试脚本导出后，push到本地的git仓库中，持续集成平台就可以通过pull对应的接口测试脚本然后通过newman执行，完成持续集成平台的赋能。

**“在接口测试中，我们通过单个接口测试完成了全部异常状态的覆盖；而在业务流程中，我们更需要关心业务流和数据流的关系，并不需要再过度关心如何用业务流的方法覆盖更多的代码逻辑异常，这也是分层测试中为什么在单元测试和界面测试之间要加入一层接口测试的主要原因之一”。**

**开发自己的测试框架：**

在win10环境配好python,先pip install requests,然后敲代码设计用例，见相关文件。再新建文件夹,新建一个common.py(见下图),然后一个个设计测试用例,模板使用unittest.



**测试框架支持RESTful风格接口：**

RESTful风格接口使用JSON格式进行数据交换.使框架支持可以借助python的相关支持库或自己封装.

**测试平台：**用工具和框架结合搭建平台.

**使用post工具设计脚本(在code里面,可读性差要改写),放到自己的框架中,使用github保存.**

**测试未知协议：**

接口测试最快速的方法不是去看协议的说明文档，而是直接去看开发实现的客户端代码.

**性能测试**

**性能测试不仅包括测试，还包括分析和调优。**

**第一个模块是性能测试基础篇。我想在这个模块里澄清一些性能测试的基础概念，讲解一些关键部分。但并不是对概念的简单描述，而是根据实际项目，告诉你真正具有指导价值的性能测试概念是什么，并解析这些概念在实际操作中的指导性作用。**

**在第二个模块中，我将通过性能测试工具的实际操作实例，对应性能测试的前后逻辑关系。在这一部分中，我会重点给你讲解，为什么要使用某些工具的某些功能，以便确保工具的使用及结果是为性能测试需求指标和性能分析报告而服务的，而不是浮于表面的“炫技”。**

**在第三个模块中，我将通过操作系统、应用服务器、数据库、缓存服务器、Java、C++ 等监控工具的使用和分析方法，告诉你它们产生的数据在性能分析过程中该如何判断，为测试报告及性能分析提供有效的历史数据。**

**最后一个模块是对前三个模块的凝练，我会讲解不同实际操作场景中的性能测试分析过程，比如实际的瓶颈判断的过程是怎样的，怎么分析出根本的原因，如何提出具体的解决方案，最后的实施效果又是怎样的。**

性能测试的指标、测试分类、测试设计有很深刻的理解。还要学习系统业务和架构相关知识，这样才能更好的设计性能场景，分析出系统的性能瓶颈。性能测试常用的工具有jmeter和loadrunner。

性能测试的意义，常见性能指标的理解,性能测试的分类.性能测试的流程包括需求分析,场景设计,脚本编写,执行资源监控,性能调优,回归测试. jmeter和loadrunner的使用,性能测试报告输出.

典型的两个业务方向，一个是音频业务，一个是视频业务。尤其视频，已经是目前甚至是未来信息交互的主要形式，而视频业务对于带宽的要求会更高，相同并发下对CPU、内存的消耗也更多。

**线程:**线程是程序中的一个执行流，每个线程都有自己的专有寄存器(栈指针、程序计数器等)，但代码区是共享的，

即不同的线程可以执行同样的函数。一般在性能测试过程中，我们认为一个线程就是一个虚拟用户。

**多线程:** 多线程是指程序中包含多个执行流，即在一个程序中可以同时运行多个不同的线程来执行不同的任务，

也就是说允许单个程序创建多个并行执行的线程来完成各自的任务。一个进程是由一或多个线程组成，进程只负责资源的调度和分配，线程才是程序真正的执行单元，负责代码的执行。

**多线程的好处：**可以提高CPU的利用率。在多线程程序中，一个线程必须等待的时候，CPU可以运行其它的线程而不是等待，这样就大大提高了程序的效率。并发用户数：在同一时刻与服务器进行交互的在线用户数量。

**性能测试:** 性能测试是通过自动化测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能测试。

**性能测试分类:**

**性能指标:**

响应时间：从客户端发起的一个请求开始，到客户端接收到从服务器端返回的响应结束，这个过程所耗费时间。

平均响应时间（average）:每个请求request的平均响应时间。

吞吐量TPS（Transaction Per Second）:每秒钟系统能够处理的交易或事务的数量。

并发数：即并发用户数，指系统可以同时承载的正常使用系统功能的用户数量。

每秒点击次数HPS（Hits Per Second）：指在一秒钟的时间内用系统处理的事务量。它一般和TPS成正比关系，是B/S系统中非常重要的性能指标之一。

**事务（Transaction）:** 通俗理解就是用户某一步或几步操作的集合。一般，事务的定义都需要跟业务相关，比如用户一次登录过程，使用淘宝时的支付过程等，需要理解事务不等于请求。

**事务响应时间:**完成整个事务所用的时间，这个性能指标是我们在业务层面重点关注的一个指标。如跨行转账业务。

**TPS（Transaction Per Second）:**每秒事务数，它是吞吐量的常用量化指标，系统吞吐量是最能直接体现软件系统负载承受能力的指标。

**TPS需要理解的几个点：**(利用jmeter测试的指标为Throughput，在聚合报告中可以查看到该指标。)

QPS（Query Per Second）：每秒查询数。HPS/RPS（Request Per Second）：每秒HTTP请求数。QPS（TPS）= 并发数/平均响应时间.如果是单接口请求，QPS有时也说TPS，指的就是系统的统统量，每秒请求/事务数。

**一个原则，一个系统的性能其实是由TPS决定，跟并发用户数没有多大关系。一般在同样的TPS下，可以用不同并发用户数去压测得到，只是得到响应时间不同。**

**响应时间公式：**响应时间=请求传输时间+Server处理时间+响应传输时间+前端解析渲染时间。

一般测试的是服务器级别的响应时间，不包括前端解析渲染时间。2/5/8 原则：2秒之内用户觉得很快， 5秒之内用户觉得还可以，8秒之外用户觉得系统慢，无法忍受。

**并发用户数:**在同一时刻与服务器进行了交互的在线用户数量。并发用户数是指对Server产生压力的用户数。

**在线用户数:**当一个用户登录到一个网站或应用后，无论是否进行发送请求操作，这些用户都称为在线用户。

**两者的关系：**一般并发用户数是包含在在线用户数中的。并发用户数计算：一般需求采集人员会将线上的并发用户数根据日志或工具分析统计出。

**资源利用率:**指的是对不同系统资源的使用程度，例如服务器的CPU利用率，磁盘利用率等。资源利用率是分析系统性能指标进而改善性能的主要依据。

**并发场景策略设计**:多个用户同时在同一系统进行同一操作，比如同时购买商品。多个用户同时在同一系统进行不同操作，比如有的在浏览查看商品，有的在购买商品。

**Jmeter**

**Jmeter：**apache组织使用纯java开发的一个开源免费测试工具，它可以实现接口和性能测试。用于对软件做压力测试，它最初被设计用于Web应用测试，但后来扩展到其他测试领域。它可以用于测试静态和动态资源，例如静态文件、Java 小服务程序、CGI 脚本、 Java 对象、数据库、FTP 服务器， 等等。Jmeter能够对应用程序做功能/回归测试，通过创建带有断言的脚本来验证你的程序返回期望的结果。

**JDK安装与安装配置:**

官网下载解压http://jmeter.apache.org/ ,配置环境变量: 在系统变量框，点击“新建”，建立一个变量：JMETER\_HOME,值为你解压的jmeter安装路径。配置classpath量%JMETER\_HOME% %JMETER\_HOME%\lib\ext\ApacheJMeter\_core.jar;%JMETER\_HOME%\lib\jorphan.jar;%JMETER\_HOME%\lib/logkit-2.0.jar; 确定即可.

**文件目录：**

**bin目录：**主要存放可执行文件，主程序的jar包，配置文件，日志文件等。

**常用文件：**jmeter.bat:windows下的应用启动文件。jmeter.sh:linux下启动文件。jmeter.log：应用运行的日志文件。jmeter.properties：系统配置文件，经常需要根据需要修改。

**说明：**jmeter.bat/jmeter.sh：启动脚本中，可以对Jmeter的启动参数JVM堆内存大小进行配置，默认为512M,建议修改成1024M。\* setHEAP=Xms1024m-Xmx1024m\*

docs目录：存放Jmeter官方API文档，便于进行二次开发。

extras目录：扩展插件目录，常用的是 Jmeter与ant集成的文件。

lib\ext目录:该目录存放的是Jmeter的插件或者扩展组件。Jmeter会自动在lib和ext下寻找需要的类。

一般扩展的依赖包，会打成jar包放入ext目录。

printtable\_docs目录：该目录存放的是官方用户手册。

**组件介绍:**

**线程组:**用来模拟用户，所以也叫Users。一个线程组模块可以包含多个线程，每个线程代表一个用户，这样可以模拟高并发下的请求，并根据网站的响应信息来判断网站的相关性能。

**线程数：**虚拟用户数，即并发数，一个线程表示一个虚拟用户；

**Ramp-Up Period：**所有线程启动的时间，单位s。通过这个参数可以设置每个线程间的启动间隔；

**循环次数：**测试循环的次数，如果勾选了“永远”，那么所有线程会一直发送请求，直到手动停止运行脚本；

**监听器（Listener）：**是用来监听系统资源的元件。它是用来对测试结果数据进行处理和可视化展示的一系列元件。 图形结果、查看结果树、聚合报告、用表格察看结果都是我们经常用到的元件。

**聚合报告(aggregate report):**对于每个请求，它统计响应信息并提供请求数，平均值，最大，最小值，错误率，大约吞吐量(以请求数/秒为单位)和以kb/秒为单位的吞吐量.

**测试计划(Test Plan)：**使用 Jmeter 进行测试的起点，它是其它 Jmeter 测试元件的容器。

**线程组(Thread Group)：**代表一定数量的并发用户，它可以用来模拟并发用户发送请求。

**采样器(sampler)：**定义实际的请求内容，被线程组包含，比如HTTP请求、java请求等。

**监听器(Listener)** ：响应结果的显示，统计等。

**逻辑控制器(Logic Controller)** ：帮助用户控制Jmeter的测试逻辑及执行顺序。

**断言(Assertions) ：**用来检查从服务器获得的响应内容是否符合预期。

**配置元件(Config Element) ：**初始化默认值和变量，以便采样器使用。

**前置处理器(Pre Processors)和后置处理器(Post Processors)** ：请求前后的处理操作。

**定时器(Timer)：**为采样器设置等待时长。

**每个接口请求会显示独立的一行数据，每行有13个字段，含义分别如下：**

Label：请求名或者请求标签，每个JMeter 的组件（例如 HTTP Request）都有一个 “名称” 属性，这里显示的就是“名称”属性的值。

**#Samples：**表示你这次测试中一共发出了多少个请求，如果模拟10个用户，每个用户迭代10次，那么这里显示就是100了。

Average：平均响应时间，默认情况下是单个 Request 的平均响应时间，当使用了 Transaction Controller 时，也可以以Transaction 为单位显示平均响应时间。

Median：中位数，也就是所有请求响应时间中排在中间的那个响应时间点，就是50% Line，参考90% Line解释。

90% Line：90%用户请求响应时间，如果把某次任务的所有请求的响应时间按从小到大排序，它是指排在90％处那个点的请求的响应时间，也就是说有90%的请求的响应时间小于等于这个响应时间。后面的50/60/70/80/90/95/99 % Line 也是类似的含义。

95% Line：95%用户请求响应时间，请参考90% Line解释。 99% Line：99%用户请求响应时间，请参考90% Line解释。Min：最小响应时间，所有请求样本中的最小响应时间。 Max：最大响应时间，所有请求样本中的最大响应时间。

Error%：本次测试中“出现错误的请求数/请求的总数”百分比值。

Throughput：吞吐量，服务器一定时间范围内处理的请求数。在本报告中它的含义其实其实是吞吐率，表示每秒完成的请求数（Request per Second），此指标代表服务器的处理能力，比如支付宝峰值处理量达到12万笔/秒。

Received KB/sec：每秒从服务器端接收到的数据量。 Sent KB/sec：每秒从客户端发送到服务器端的数据量。

**jmeter操作数据库：**

接口测试中对数据进行查询，传入数据作为请求参数，或者断言时查看数据是否正确写入数据库。接口或者性能测试，通过接口并发请求，批量实现数据的构造，数据恢复操作。快速实现对数据库的增删改查。

添加数据库驱动:将对应的数据库驱动jar包放入jmeter的lib目录下，以mysql驱动为例，重启jmeter生效。

添加JDBC Connection Configuration（数据库配置连接池）添加路径：测试计划->添加->配置元件->JDBC Connection Configuration. 添加线程组及JDBC Request. 查看执行sql语句响应数据

**Jmeter接口测试之参数关联:**

**关联:**请求之间通过传递参数建立联系。一般需要将一个请求的响应参数，作为另一个请求的入参。比如登录后的操作，第一步实现登录请求，然后将请求返回的token提取出来保存到一个变量中，后续请求作为入参使用。

**Top命令**

**top命令：**Linux下常用的性能分析工具，能够实时监控系统的运行状态，主要显示系统中各个进程的资源使用情况，并根据cpu及内存等进行排序，类似于windows系统中的任务管理器，可以实时查看系统的各种资源状态。

首先前五行显示的系统资源的总体统计信息：

**第一行：类似于update命令，**他可以快速简洁的获取系统的负载，依次显示现在时间，多少个用户，系统1分钟、5分钟、15分钟的平均负载。up xx days, HH:MM：系统当前时间和系统已运行时间， 当前登录用户数，系统平均负载可以运行任务的平均数量（3个数值分别统计最近1，5，15分钟的系统平均负载）系统平均负载：单核CPU情况下，0.00 表示没有任何负荷，1.00表示刚好满负荷，超过1侧表示超负荷，理想值是0.7。

**第二行：Tasks 任务（进程）**系统现在共有的进程数，运行中进程数，休眠中的进程数，停止状态的进程数，僵尸状态进程数。

**第三行：cpu状态信息，具体属性说明如下：**

0.2% us：用户空间占用CPU的百分比

0.3% sy：内核空间占用CPU的百分比

0.0% ni：改变过优先级的进程占用CPU的百分比

99.6% id：空闲CPU百分比，反映一个系统cpu的闲忙程度，越大越空闲。

0.0% wa：IO等待占用CPU的百分比

0.0% hi：硬中断（Hardware IRQ）占用CPU的百分比

0.0% si：软中断（Software Interrupts）占用CPU的百分比

0.0% st：Steal time虚拟机被hypervisor偷去的CPU时间

**第四行：内存信息：**

物理内存总量：total 空闲的内存总量：free 使用中的内存总量：used 缓存的内存总量：buffers

**第五行（Swap）：交换分区**

总的交换空间大小；空余空间交换空间大小；已经使用交换空间大小；cached缓冲的交换空间大小

**进程列表说明：**

PID: 进程pid USER: 拉起进程的用户

PR: 该列值加100为进程优先级，若优先级小于100，则该进程为实时(real-time)进程，否则为普通(normal)进程，实时进程的优先级更高，更容易获得cpu调度，以上输出结果中，java进程优先级为120，是普通进程，had进程优先级为2，为实时进程，migration 进程的优先级RT对应于0，为最高优先级

NI: 进程的nice优先级值，该列中，实时进程的nice值为0，普通进程的nice值范围为-20~19

VIRT: 进程所占虚拟内存大小（默认单位kB） RES: 进程所占物理内存大小（默认单位kB）

SHR: 进程所占共享内存大小（默认单位kB） S: 进程的运行状态 %CPU: 采样周期内进程所占cpu百分比

%MEM: 采样周期内进程所占内存百分比 TIME+: 进程使用的cpu时间总计 COMMAND: 拉起进程的命令

**常用参数说明：**

M：根据驻留内存大小进行排序；

P：根据CPU使用百分比大小进行排序；

T：根据时间/累计时间进行排序；

d：屏幕刷新间隔时间；

p<进程号>：指定进程

**自动化测试**

掌握一门编程语言Python或Java，一定的前端知识。Web自动化测试的selenium环境的搭建，webdriver元素定位，app自动化测试的Appium环境搭建，webdrive元素定位，常见的自动化测试框架，对于接口自动化测试，python实现的话需要掌握requests库和unittest框架。

**框架( Framwork )**：是构成一类特定软件可复用设计的一组相互协作的类。框架规定了你的应用的体系结构。它定义了整体结构，类和对象的分割，各部分的主要责任，类和对象怎么协作，以及控制流程。框架预定义了这些设计参数，以便于应用设计者或实现者能集中精力于应用本身的特定细节。

**自动化测试框架**：是一个工具脚本集合，用于组织和管理自动化基础功能（打开浏览器、鼠标点击、登录等）自动化测试用例、提供测试结果统计报告。自动化测试框架能够提供可靠的测试服务，对于新需求能够快速扩展，能够让测试工程师简单高效的完成自动化测试任务。国内的阿里Macaca,腾讯开源微信小程序自动化测试工具,百度内部的Café框架.测试平台包括阿里MQC,腾讯WeTest,百度MTC,Testin

**测试框架的优点：**代码复用 ，将基础的测试代码封装，从而降低代码的复杂性。提高维护效率，有效组织和管理测试脚本。快速实现项目的自动化测试，不用从0开始，一般测试框架完成后，其他类似程序也可以快速复用。输出各种美观易懂的测试报告。

**自动化测试的目的：**缩短测试周期，节省成本。避免人为出错，提高准确性和可靠性。获取需求覆盖率，代码覆盖率，提供衡量软件质量的指标。

**自动化测试的意义：**手工测试完成后。项目周期长，需求稳定。频繁的回归测试。

**合理的使用UI自动化:**

**单元测试:使用app分层测试策略,增加单测规模,控制UI自动化测试规模.**

**移动app自动化测试:核心集成与验收测试用例交给自动化测试.**

**自动化遍历测试:大部分的基础回归测试交给自动遍历.**

**人工测试:测试团队主负责,整个产品团队参与.**

**典型的自动化测试框架目录：**

config(配置文件目录)：存放所有使用的配置文件，实现配置与代码分离。

common(公共函数)：公共函数、方法以及通用操作的管理。

data(测试数据)：将所有的用例参数化使用的文件放到这里，一般可采用xlsx、csv、xml等格式。实现数据与代码分离。

drivers（驱动目录）：一般存放浏览器驱动，如如Chromedriver等。

logs（日志目录）：日志记录和管理功能，存放运行时日志和错误日志error log等。

testcase（测试用例）：测试用例管理功能，可以分模块编写，建相应的目录。

report（测试报告）：管理和存放程序运行后生成的测试报告，一般可有html报告、excel报告等。

**python主流测试框架：**

**unittest框架：**

python内置的单元测试框架，unittest的设计灵感最初来源于Junit以及其他语言中具有共同特征的单元框架。它能编写组织用例、执行用例，还可输出测试报告等。

**优点**：使用断言判断返回布尔值来判断期望值和实际值的差异。可以构建共同的初始化变量或实例。框架结构可以组织用例批量运行。

**Unittest的几个概念：**test fixture：一个test fixture 表示执行一个或多个测试前的准备工作，以及执行完成后清理工作。例如：创建临时或代理数据库或目录，或者是一个启动服务器进程。

test case 一个test case是一个独立的测试单元，它检查特定输入是否响应特定的输出，unittest提供了一个基本类——TestCase，这个类用于创建一个或多个test case。

test suite：一个test suite是test cases、或test suites或两者的一个集合，用于把想执行tests放在一起。

test runner： 一个test runner 是由两部分成分：合理安排tests的执行、提供给user输出结果。runner可以用一个图形界面、文本、或者是一个特殊的值，代表tests执行的结果。

**Pytest：**

很好的强大的单元测试框架，实际上广泛使用在自动化单元、接口、功能等测试。如果你的项目比较小、复杂度比较低，Pytest是最适合的自动化测试平台。非常多的Python开发者都喜欢它，他们大部分用它来进行单元测试。Pytest最好的特性之一是，它提供了测试用例的详细失败信息，使得开发者可以快速准确地改正问题。它兼容最新版本的Python。它还兼容unittest、doctest和nose，开箱即用。它还有包含更多功能的插件和多样化的现有测试技术和测试用例。官方文档：<https://pytest.org/en/latest/>

**Robot Framework：**

通用的python测试框架，提供一个图形用户界面，方便管理和编辑测试用例，易于上手，生成的报告比较好看，支持关键字和数据等驱动。Robot Framework是独立于操作系统和应用程序，支持跨平台，例如Windows、MacOS和Linux。这个产品是由世界上一些著名的测试人员创建的，拥有关键词驱动方案。它有丰富的工具和库，使得这个框架非常先进和健壮。

**自动化测试**：是把以人为驱动的测试行为转化为机器执行的一种过程。通常，在设计了测试用例并通过评审之后，由测试人员根据测试用例中描述的规程一步步执行测试，得到实际结果与期望结果的比较。自动化测试的主要目标是提高测试效率，降低项目成本，而不是完全取消手工测试。

**自动化测试的意义：**提升效率，减少重复工作：勿容置疑，最大的意义就是提高测试效率，手工测试的最大问题在于，面对快速迭代，无法快速完整的执行冒烟用例。节省人力成本：执行测试脚本可以实现无人值守，不限时间的测试。从而让测试人员可以做更多有意义事情，比如探索性测试等。保证每次测试的完整性和一致性。

发现更多隐藏问题：手工测试无法验证系统的稳定性、可靠性等，需要通过工具等自动化手段，对系统进行压力测试、稳定性测试等。

**Selenium**

web测试框架，丰富的API，支持多种语言编写测试脚本且可在多种浏览器执行测试脚本。

**webdriver：**server -client设计模式设计的。server端即浏览器。当我们的脚本启动浏览器后，等待client发送请求并做出相应；client端即测试代码.

**搭建环境**:本地安装Python,在官网下载,安装时勾选path,安装完成后,在cmd中输入python,查看是否进入.

安装selenium,在cmd中输入pip install selenium

安装Chrome Webdriver(<https://sites.google.com/a/chromium.org/chromedriver/downloads>).

**不同浏览器的driver:**

browser = webdriver.Chrom()

browser = webdriver.Firefox() browser = webdriver.Safari() browser = webdriver.Ie()

**基本模板:**

from selenium import webdriver

browser = webdriver.Chrome(executable\_path="..\..\..\zc\chromedrive\chromedriver.exe")

# executable\_path来指定chromedirver路径

browser.get('https://www.baidu.com')

print(browser.title)

browser.quit()

**八种定位方法:**

**id, name, class name,tag name,link text,partial link text,xpath,css selector**

定位方法find\_element\_by\_xx()依次类推.如: find\_element\_by\_xpath()

**浏览器操作:**

set\_window\_size()方法来设置浏览器的大小 , maximize\_window() 全屏显示.如:browser.set\_window\_size(480,800)

back()和forward()方法来模拟后退和前进按钮, refresh()方法模拟F5刷新页面.

**元素操作:**

Clear()方法,清除文本. Send\_keys(‘value’)方法,模拟键盘输入. Click()方法,单击元素.

submit()方法用于提交表单, (submit()可与 click()方法互换来使用， submit()同样可以提交一个按钮但 submit()的应用范围远不及 click()广泛)

size： 返回元素的尺寸. text： 获取元素的文本. get\_attribute(name)： 获得属性值。is\_displayed()： 设置该元素是否用户可见。

**鼠标事件:**在 WebDriver中鼠标操作的方法封装在 ActionChains 类提供。常用方法如下:

perform()： 执行所有 ActionChains 中存储的行为；move\_to\_element()： 鼠标悬停。

context\_click()： 右击； double\_click()： 双击； drag\_and\_drop()： 拖动；

**如:**

# 引入 ActionChains 类

from selenium.webdriver.common.action\_chains import ActionChains

# 定位到要悬停的元素

cc = browser.find\_element\_by\_xpath('//\*[@id="content"]/div/div[1]/div[2]/ul/li[4]/a')

# 对定位到的元素执行鼠标悬停操作

ActionChains(browser).double\_click(cc).perform()

**键盘事件**: Keys()类提供了键盘上几乎所有按键的方法.

# 引入 Keys 模块

from selenium.webdriver.common.keys import Keys

**#常用键盘方法:**

send\_keys(Keys.BACK\_SPACE) 删除键（BackSpace）

send\_keys(Keys.SPACE) 空格键(Space)

send\_keys(Keys.TAB) 制表键(Tab)

send\_keys(Keys.ESCAPE) 回退键（Esc）

send\_keys(Keys.ENTER) 回车键（Enter）

send\_keys(Keys.CONTROL,'a') 全选（Ctrl+A）

send\_keys(Keys.CONTROL,'c') 复制（Ctrl+C）

send\_keys(Keys.CONTROL,'x') 剪切（Ctrl+X）

send\_keys(Keys.CONTROL,'v') 粘贴（Ctrl+V）

send\_keys(Keys.F1) 键盘 F1

send\_keys(Keys.F12) 键盘 F12

**断言与等待**:

title：用于获得当前页面的标题。 current\_url：用户获得当前页面的URL。 text：获取搜索条目的文本信息。

WebDriver提供两种类型的等待：显式等待和隐式等待。

显式等待:使WebdDriver等待某个条件成立时继续执行，否则在达到最大时长时抛出超时异常（TimeoutException）。

隐式等待:WebDriver提供了implicitly\_wait()方法来实现隐式等待，默认设置为0。

driver ：浏览器驱动。timeout ：最长超时时间，默认以秒为单位。

poll\_frequency ：检测的间隔（步长）时间，默认为0.5S。

ignored\_exceptions ：超时后的异常信息，默认情况下抛NoSuchElementException异常。

WebDriverWait()一般由until()或until\_not()方法配合使用，下面是until()和until\_not()方法的说明。

until(method, message=‘’)

调用该方法提供的驱动程序作为一个参数，直到返回值为True。

until\_not(method, message=‘’)

调用该方法提供的驱动程序作为一个参数，直到返回值为False。

**定位一组元素**:WebDriver还提供8种用于定位一组元素的方法。

find\_elements\_by\_id()

find\_elements\_by\_name()

find\_elements\_by\_class\_name()

find\_elements\_by\_tag\_name()

find\_elements\_by\_link\_text()

find\_elements\_by\_partial\_link\_text()

find\_elements\_by\_xpath()

find\_elements\_by\_css\_selector()

**多表单切换与多窗口切换,警告框处理:**

WebDriver只能在一个页面上对元素识别与定位，对于frame/iframe表单内嵌页面上的元素无法直接定位。需要通过switch\_to.frame()方法将当前定位的主体切换为frame/iframe表单的内嵌页面中。

如:browser.switch\_to.frame(‘xx’)

switch\_to.frame() 默认可以直接取表单的id 或name属性。如果iframe没有可用的id和name属性，则可以通过下面的方式进行定位。

#先通过xpth定位到iframe

xf = driver.find\_element\_by\_xpath('//\*[@id="x-URS-iframe"]')

#再将定位对象传给switch\_to.frame()方法

driver.switch\_to.frame(xf)

……

driver.switch\_to.parent\_frame()

除此之外，在进入多级表单的情况下，还可以通过switch\_to.default\_content()跳回最外层的页面。

WebDriver提供了switch\_to.window()方法，可以实现在不同的窗口之间切换。

current\_window\_handle：获得当前窗口句柄。

window\_handles：返回所有窗口的句柄到当前会话。

switch\_to.window()：用于切换到相应的窗口，与上一节的switch\_to.frame()类似，前者用于不同窗口的切换，后者用于不同表单之间的切换。

WebDriver中处理JavaScript所生成的alert、confirm以及prompt十分简单，具体做法是使用 switch\_to.alert 方法定位到 alert/confirm/prompt，然后使用text/accept/dismiss/ send\_keys等方法进行操作。

text：返回 alert/confirm/prompt 中的文字信息。 accept()：接受现有警告框。

dismiss()：解散现有警告框。 send\_keys(keysToSend)：发送文本至警告框。keysToSend：将文本发送至警告框。

**下拉框选择:**

WebDriver提供了Select类来处理下拉框, Select类用于定位select标签。

select\_by\_value() 方法用于定位下接选项中的value值。

send\_keys()方法来实现文件上传

**cookie操作:**

get\_cookies()： 获得所有cookie信息。

get\_cookie(name)： 返回字典的key为“name”的cookie信息。

add\_cookie(cookie\_dict) ： 添加cookie。“cookie\_dict”指字典对象，必须有name 和value 值。

delete\_cookie(name,optionsString)：删除cookie信息。“name”是要删除的cookie的名称，“optionsString”是该cookie的选项，目前支持的选项包括“路径”，“域”。

delete\_all\_cookies()： 删除所有cookie信息。

WebDriver提供了execute\_script()方法来执行JavaScript代码。

WebDriver提供了截图函数get\_screenshot\_as\_file()来截取当前窗口。

**关闭浏览器:**

close() 关闭单个窗口

quit() 关闭所有窗口

**Appium**

**移动自动化测试基础:**

**自动化测试的价值,常见误区,技术体系,UI分层测试体系,Android app的常见概念,adb Android调试桥,测试常用命令**

**Appium框架知识点,技术架构与原理,如何封装定制与改进,Android,ios,手机浏览器,微信小程序.**

**PageObject模式,App的实战,自动遍历测试,兼容性测试在内的验收测试体系,开源的多设备调度,远程交互调试的自动化测试平台,持续集成.**

**待测产品app解析,自动化测试框架,框架原理解析,二次二次封装定制,Android自动化测试,ios自动化测试.**

**微信小程序自动化测试,手机浏览器自动化测试,Page Object设计模式,实战演练环节.**

**App结构分析,自动化测试框架,**

一个自动化测试开源工具，支持 iOS 平台和 Android 平台上的原生应用，web应用和混合应用。

“移动原生应用”是指那些用iOS或者 Android SDK 写的应用（Application简称app）。

“移动web应用”是指使用移动浏览器访问的应用（appium支持iOS上的Safari和Android上的 Chrome）。

“混合应用”是指原生代码封装网页视图——原生代码和 web 内容交互。比如，像 Phonegap，可以帮助开发者使用网页技术开发应用，然后用原生代码封装，这些就是混合应用。重要的是，appium是一个跨平台的工具：它允许测试人员在不同的平台（iOS，Android）使用同一套API来写自动化测试脚本，这样大大增加了iOS和Android测试套件间代码的复用性。

**特点:**

appium是跨平台的，可以用在OSX，Windows以及Linux桌面系统上运行。appium类库封装了标准Selenium客户端类库，为用户提供所有常见的JSON格式selenium命令以及额外的移动设备控制相关的命令，如多点触控手势和屏幕朝向。appium选择了Client/Server的设计模式。只要client能够发送http请求给server，那么的话client用什么语言来实现都是可以的，这就是appium及Selenium(WebDriver)如何做到支持多语言的原因；

**Appium生态工具:**

**Appium Desktop:**内嵌Appium Server和Inspector的综合工具.**Appium Server :**Appium的核心工具,命令行工具.

**Appium Clients:**各种语言的客户端封装库,用于连接appium server.**Appium Crawler**自动遍历工具

**Win10环境搭建:**

1.安装 jdk，配置环境变量.

2.安装 npm install -g appium(Appium Server安装),先安装nvm,再设置切换node和npm的源,在切换npm为淘宝源.

3.安装 android studio 开发工具,配置环境变量.

4.安装npm install -g appium-doctor 查看环境配置有无问题.

5.安装Python与pip3 install Appium-Python-Client 和IDE选VSCode,Python多版本隔离工具选env,测试框架有unittest,pytest,nose.选pytest.

**Appium Desktop安装:**官网下载安装: <https://github.com/appium/appium-desktop/releases>

主要功能:UI分析,录制用例,元素查找测试,Attach已有的session ,云测试

**配置待测应用以启动appium:**以Android Studio的模拟器的youtube为例,配置如下,进行启动.

{

"platformName": "android",

"deviceName": "bluestack",

"appPackage": "com.google.android.youtube",

"appActivity": "com.google.android.apps.youtube.app.WatchWhileActivity",

"autoGrantPermissions": true

}

**基础demo:**

配置要打开的app,连接模拟器,进行控制,进行UI分析,点击顶部的**选择元素**进行元素分析,录制简单测试用例且保存.

点击开始录制,选择代码方式,展开样式代码进行操作,操作完成后,保存代码.

**录制用例并执行:**使用appium-desktop录制用例,安装Python依赖pip install Appium-Python-Client,增加隐式等待增强稳定性,重新运行.

**增加隐式等待:**driver.implicitly\_wait(10)

**隐式等待:**服务端(appium)会在特定超时时间内重试多次寻找控件. **显示等待:**在客服端(用例端)根据更灵活的条件循环等待条件满足.

**在VSCode运行:**$sppium后,直接在IDE运行.

**测试步骤三要素:定位,交互,断言.**

**常用定位手段: id, Accessibility ID, XPath(XML路径语言,用于XML中的节点定位)**

**App DOM:**关键Attribute的clickble,content-desc,resource-id,text,bounds,其中android与ios的DOM属性与节点结构类似,名字和属性的命名不同.

**元素定位符与复用:**findElementByxxx,findElement(by,value),findElement主要用于 Page Object模式.

**常用自动化动作支持:** click,sendkys,swiper,touch action

**手势操作:** Press release longPress, moveTo,tapwait,perform

**使用测试框架管理测试用例:**Java的Unit4,Junit5,TestNG. Python的unittest和pytest

**设备交互API:**模拟电话,短信,横竖屏切换,app处理,键盘,粘贴板,录屏.

**monkey测试**

**monkey测试：**测试软件的稳定性，健壮性的方法，一般可以通过测试过程中打印的日志来发现问题。Monkey是通过命令行来对APP进行测试的工具，允许在模拟器里或真机上。向系统发送伪随机用户时间流实现对应用程序进行压力测试；

**环境搭建：**win10需按照ADB工具, <https://developer.android.com/studio/run/win-usb>,然后再配置变量

https://blog.csdn.net/u013250071/article/details/78416274

**实践：**在手机开发者选项中，勾上USB调试。使用adb命令查看已连接设备：$adb devices

关闭adb的后台进程: adb kill-server 让手机脱离USB线的TCP连接方式: adb tcpip

开启tcp连接方式连接手机: adb connect 收集日志数据,用于后续的分析,比如耗电量: adb bugreport

**发送压测命令：**对随机应用执行100条monkey命令：adb shell monkey 1000

对特定应用进行monkey测试：adb shell monkey -P XXX.apk -v 测试次数

**APP信息:**获取当前界面信息: adb shell dumpsys activity top

获取任务列表: adb shell dumpsys activity activities

获得内存信息: adb shell dumpsys meminfo com.android.settings

获取cpu信息: adb shell dumpsys cpuinfo

获取特定包基本信息: adb shell dumpsys package xx

获取当前activity: adb shell dumpsys activity top

**APP入口:**adb logcat| findsdr -i displayed 或 appt dump badging xx.apk | grep lanunchable-activity 或

apkanalyzer 最新版本的sdk才有

**启动应用:** adb shell am start -n xxx -S 日志查询: adb logcat

当前应用查询 adb logcat | findstr Displayed

**ADB SHELL:**本身Linux的shell,可以调用Android内置命令.

adb shell dumpsys adb shell pm adb shell am adb shell ps adb shell monkey

清理应用缓存:adb shell pm clear 包名

显示常用uiautomator命令: adb shell uiautomator

显示常用输入命令: adb shell input

**招聘**

**自我介绍的的核心逻辑：**

1. 说清楚我是谁。第二，我做过什么，有什么成就。第三，我能给公司带来什么，跟招聘岗位有什么契合点。

您好，我叫XXX，毕业于XXX，在XXX、XXX两家公司工作过，目前担任XXX工程师，负载XXX、XXX产品的开发/测试工作，擅长性能测试、接口测试……，希望能够加入贵公司，担任XXX方向的工作。