一些话

面试：电话/视频/面试，增加成功率

北京/上海/深圳(纸醉金迷)/广州（白云区）/西安/杭州/成都(大侠)/天津(落户)/哈尔滨的冬天

此处不了爷，自有留爷处

Python开发/爬虫/运维自动化/数据分析/测试

运维自动化（经验/学历） > 数据分析(经验/学历) > 爬虫 > python开发(面试难度高/学历要求高/经验要求也高) > 测试(Python开发经验打底/)

什么样的身份适合测试?

1. 学历，大专起
2. 工作经验，咱们有开发经验，加分项
3. 测试，外包公司多

应该要牢记:

1. 数据库
2. linux
3. docker&k8s 加分项
4. nosql：redis，es，MongoDB，消息队列

1. 大家一起助攻
2. 面试完成后，在群里自己把面试情况说一下
3. 面试心态不能蹦

测试基础

参考：https://www.cnblogs.com/Neeo/articles/10669652.html

什么是测试？**通过手工或者相关工具，对`被测对象`进行测试操作。从而验证实际与预期结果是否存在差异。**

被测对象：

1. 代码
2. 文档
3. 用户手册

测试的主要工作：

- 检查代码，评审开发文档。
- 进行测试设计、写作测试文档、测试计划、测试方案、测试用例等等。
- 执行测试、发现软件缺陷，提交缺陷报告，并追踪缺陷修复的过程。

测试用例的定义

测试用例（Test Case）是为特定的目的而设计的一组测试输入、执行条件和预期的结果，以便测试是否满足某个特定需求，通过大量的测试用例来检验软件的运行效果，它是指导测试工作进行的重要依据。

一个测试用例的组成：

- 用例编号：产品名字-测试阶段
- 测试项目：对应一个功能模块
- 测试标题：直接对测试点进行细化得出
- 重要级别：高/中/低
- 预置条件：需要满足一些前提条件，否则用例无法执行
- 测试输入：需要加工的输入信息，根据具体情况设计
- 操作步骤：明确给出每个步骤的描述，执行人员根据该步骤执行工作
- 预期结果：根据预期输出对比实际结果，判断被测对象是否符合需求。
- 实际结果：根据实际结果，填写报告。(可写可不写)

版本控制

熟练掌握git

测试工具

1. 禅道
2. selenium
3. jmeter
4. 抓包工具：fiddler，charles(mac)
5. postman
6. appium
7. Jenkins

开发/测试模型

参考：https://www.cnblogs.com/Neeo/articles/11795967.html

开发模型：

1. 瀑布模型，阶段划分清晰
2. 快速原型模型，首先建立模型，让用户提供意见，修改原型，最会依此展开开发
3. 敏捷开发模型，小步快跑节奏，开发节奏
4. 螺旋模型，唯独增加了风险控制，参考：https://www.cnblogs.com/Neeo/articles/10579119.html

测试模型：

1. v模型，跟瀑布模型保持一致
2. w模型，弥补v模型缺点

测试分类

按照程序执行与不执行来分：

1. 静态测试，不执行程序，测试文档静态的代码
2. 动态测试，执行程序，观察程序的外在表现，是否和符合软件需求说明书中的要求。

按照手工和自动化化来分：

1. 手工测试，
2. 自动化测试，通过计算机来完成人手工达不到的要求的测试，如并发，适合重复性，枯燥的测试场景。

按照测试环节分类：

1. 单元测试，开发一般在做，单元测试的粒度最小，一般是函数，类
2. 黑盒测试，不关注程序的内部构造，只是通过输入数据来查看程序的输出结果是否符合预期。
3. 百盒测试，关注程序的内在表现。代码，逻辑上是否存在问题。
4. 灰盒测试，介于黑盒和白盒之间的测试。
5. 集成测试，组装测试，主要用来测试模块与模块之间的接口，同时还要测试一些主要业务功能。集成测试（也叫组装测试，联合测试）是单元测试的逻辑扩展。它最简单的形式是：把两个已经测试过的单元组合成一个组件，测试它们之间的接口。从这一层意义上讲，组件是指多个单元的集成聚合。在现实方案中，许多单元组合成组件，而这些组件又聚合为程序的更大部分。方法是测试片段的组合，并最终扩展成进程，将模块与其他组的模块一起测试。最后，将构成进程的所有模块一起测试。此外，如果程序由多个进程组成，应该成对测试它们，而不是同时测试所有集成测试进程。
6. 系统测试，粒度最大，一般由独立测试小组采用黑盒方式来测试，主要测试系统是否符合`需求规格说明书`。
7. 验收测试，用户来做，测试人员充当用户。

https://img2018.cnblogs.com/blog/1168165/201911/1168165-20191105004226647-1736627202.png

黑盒测试常见的测试方法

参考：
https://www.cnblogs.com/Neeo/articles/11795983.html#%E5%B8%B8%E8%A7%81%E7%9A%84%E9%BB%91%E7%9B%92%E6%B5%8B%E8%AF%95%E6%96%B9%E6%B3%95

在编写黑盒测试用例的时候，我们可以有以下几种形式来完成：

- 等价类划分法（5星）
- 边界值法（5星）
- 场景法（5星）
- 因果图法
- 判定表法
- 正交（排列）法
- 测试大纲法

我们至少要了解每种方法的使用场景（在哪用？）和使用步骤（怎么用？）。

除此之外，我们编写测试用例，都有哪些可以参考的呢？

- 需求文档
- 被测系统（已经开发出来的被测系统），一边对照程序，一边编写用例，很多企业都采用这种方式，如果只对照需求文档可能仅能完成测试的30~40%。
- 开发（设计）文档，当然，有可能也拿不到，比如说开发和测试分属不同的公司，人家开发就不一定提供文档。
- 积极与开发、产品、客户进行沟通

等价类划分法

常用在输入的场景中，将用户所有的可能的输入划分为了若干份（或称为不同的子集），然后从每个子集中**选取少数具有代表性的数据作为测试用例**，这种测试用例我们称为**等价类（类别）划分法。**

等价类划分分为（基本概念）：

- 有效等价类，指符合《需求规格说明书》，输入合理的数据集合。
- 无效等价类，指不符合《需求规格说明书》，输入不合理的数据集合。

等价类思考步骤：

- 首先确定有效等价类和无效等价类
- 有效等价类就是一目了然的题目条件（比如在0~20之间测试），要考虑到两端的极值（边界值）和中间值。
- 无效等价类先划分与条件相反的情况（比如不在0~20范围内），再去找特殊情况，如中英文，符号、空格、空等。

边界值

什么是边界值?

边界是指对于输入等价类和输出等价类而言，稍高于边界值和稍低于边界值的一些特定情况，边界值分析法常应用于黑盒测试中。边界值也可以称为极值。

根据以往的经验来看：**大量的错误是发生在输入或者输出范围的边界上，而不是再输入范围的内部。**所以，为了保证测试质量，我们需要重点关注测试边界。

边界值通常跟等价类一起使用。

因果图/判定表(重点理解)

等价类和边界值都是考虑输入输出的问题，而没有太关注，多组输入和输出之间的组合以及制约关系。而因果图和判定表恰好解决了这个问题。

因果图（Cause-Effect Graph）法是一种**利用图解法分析输入的各种组合情况，从而设计测试用例的方法，它适用于检查程序输入条件的各种组合情况。**
因果图法的特点

- 考虑输入条件的相互制约及组合关系
- 考虑输出条件对输入条件的依赖关系

因果图法产生的背景

- 等价类划分法和边界值分析方法都是着重的考虑输入条件，但是极少的考虑输入条件的各种组合、输入条件之间的相互制约关系，这样虽然各种输入条件可能出错的情况已经测试到了，但是多个输入条件组合起来可能出错的情况却很难发现。
- 如果在测试的时候，必须考虑输入条件的组合情况，那么可能的组合情况将非常的多，导致测试任务繁重。因此，我们必须考虑一种适合于描述多种条件的组合，相应产生多个动作的形式来进行测试用例的设计，这就用到了因果图，也就是设计逻辑模型。

因果图的核心

- 因果图法适用于输入条件较多的情况，测试所有输入条件的排列组合。因果图所谓的因就是输入，而果就是输出。
    - 因果图之`因`，即输入条件。
    - 因果图之`果`，即输出结果。

因果图法要注意考虑的要点

- 所有输入/输出条件的相互制约关系及组合关系。
- 输出结果对输入条件的依赖关系，也就是什么样的输入组合会产生怎样的输出结果，即`因果关系`。

因果图法基本步骤

利用因果图导出测试用例需要经过以下几个步骤：

- 找出所有的原因（条件/输入），原因即是输入条件或输入条件的等价类。
- 找出所有的结果，结果即输出条件。
- 明确所有输入条件之间的制约关系及组合关系。比如，哪些条件可以组合到一起，哪些条件不能组合到一起。
- 明确所有输出条件之间的制约关系以及组合关系。比如，哪些输出结果可以同时输出，哪些输出结果不能同时输出。
- 找出什么样的输入条件组合会产生哪种输出结果。
- 把因果图转换成判定表/决策表。
- 为判定表/决策表中的每一列表示的情况设计测试用例。

因果图只是一种辅助工具，我们通过因果图分析出一个表，这个表就是判定表，然后通过判定表编写测试用例。但是有的时候，画因果图非常麻烦，影响测试效率，所以，我们很多时候都是直接写判定表，进而编写测试用例。

判定表的组成

- 条件桩：问题的所有条件，即所有的条件组合。
- 动作桩：问题的所有输出，即所有的输出组合。
- 条件项：针对条件桩的取值。
- 动作项：条件项的各种取值情况下的输出结果。

判定表制作步骤

- 列出所有的条件桩和动作桩。
- 填入条件项与动作项，得到初始判定表。
- 简化判定表（合并相似规则（相同动作））。

场景法

场景法就是模拟用户操作软件时的场景，主要用于测试系统的业务流程。

- 当接到一个测试任务时，我们并不关注某个控制的边界值，等价类是否满足要求。而是要关注它的主要功能和业务流程是否正确实现，这就需要使用场景法来完成测试。
- 当业务流程没问题时，即软件的主要功能没有问题时，我们再重新从边界值、等价类方面对软件进行测试。

在冒烟测试时也主要采用场景法进行测试。

用例场景定义

在场景法中有两个重要的概念：

- 基本流：按照正确的业务流程来实现一条操作路径，即模拟正确的操作流程。
- 备选流：导致程序出现错误的操作流程，即模拟错误的操作流程。

用例场景使用来描述流经用例路径的过程，这个过程从开始到结束遍历用例中所有的基本流和备选流。

用例场景产生的背景

现在的软件几乎都是由事件触发来控制流程的，事件触发时的情景便形成了场景，而同一事件因不同的触发顺序和处理结果形成事件流。

将这种在软件设计方面的思想引入到软件测试中，生动的描绘出时间触发时的情景，有利于测试设计者测试用例，同时测试用例也更容易得到理解和执行。

- 在使用场景法设计测试用例时，需要覆盖系统用例中的主**成功场景**和**扩展场景**。并且需要适当补充各种正反面的测试用例和考虑出异常场景的情形。
- 当使用场景测试软件没有问题时，可以再使用边界值、等价类划分法等测试方法对软件进行更加细致、完整的测试。

流程分析法

流程分析法主要是针对测试场景类型，属于流程测试场景的测试项下的一个测试子项进行设计。是从白盒测试设计方法中的路径覆盖分析法借鉴过来的一种方法。

- 在白盒测试中，路径就是指函数代码的某个分支组合，路径覆盖法需要构造足够的用例，以覆盖函数的所有代码路径。
- 在黑盒测试中，若将软件系统的某个流程看成路径的话，则可以针对该路径使用路径分析法设计测试用例。

优点

- 降低了测试用例的设计难度，只要搞清楚各种流程，就可以设计出高质量的测试用例来，而不需要太多测试方面的经验。
- 在测试时间较为紧迫的情况下，可以有的放矢的选择测试用例，而不用完全根据经验来取舍。

流程分析法的实施步骤

- 详细了解需求。
- 根据需求说明或界面原型，找出业务流程的各个页面以及各个页面之间的流转关系。
- 画出业务流程，由产品经理使用Axure软件制作。
- 写用例，覆盖所有的路径分支。

错误推测法

错误推测法是指利用**直觉**和**经验**猜测出出错的可能类型，有针对性的列举出程序中所有可能的错误和容易发生错误的地方。它是骨灰级测试大佬喜欢使用的一种测试用例设计方法。

基本思想

基本思想是列举出可能犯的错误或错误易发生的清单，然后根据清单编写测试用例；这种方法很大程度上是凭经验进行的，即凭人们对过去所作测试结果的分析，对所揭示缺陷的规律性作直觉的推测来发现缺陷。

采用错误推测法，最重要的是要思考和分析测试对象的各个方面，多参考以前发现的Bug的相关数据、总结的经验，个人多考虑异常的情况、反面的情况、特殊的输入，以一个攻击者的态度对待程序，才能够设计出比较完善的测试用例。

正交测试法

正交测试源于正交实验设计方法。

正交试验设计方法是一种研究多因素多水平的实验设计方法，它根据正交性从全面实验中挑选出部分有代表性的点进行试验，这些有代表性的点具备了**均匀分散，齐整可比**的特点。

正交试验设计方法一般使用已经造好了的正交表格来安排实验并进行数据分析。

正交测试法与正交实验设计方法类似，也使用了已经造好的正交表格来生成测试用例。它简单易行，应用性较好。

正交表测试法的适用范围

正交表测试法适用于输入条件相互独立，并且需要对输入的**各种组合**进行测试的场合。

正交表的正交性

- 整齐可比性：在同一张正交表中，每个因素的每个水平出现的次数是完全相同的。由于在实验中每个因素的每个水平与其它因素的每个水平参与实验的几率是完全相同的，这就保证了在各水平中最大程度的排除了其他因素水平的干扰。因而能有效的进行比较和作出展望。
- 均衡分散性：在同一张正交表中，任意两列（两个因素）的水平搭配（横向形成的数字对）是完全相同的。这就保证了实验条件均衡的分散在因素水平的完全组合中，因而具有很强的代表性。

正交测试用例设计步骤

- 根据所测程序中控件的个数（因素）以及每个控件的取值个数（水平），选取一个合适的正交排列表。
- 将控件及其取值列举出来，并对其进行编号。
- 将控件及其取值映射到正交排列表中。
    - 把正交排列表中的ABCD（因子）分别替换成4个控件。
    - 把每列中的1,2,3（状态）分别换成这个控件的3个取值（水平），排列顺序要按照表中给出的顺序。
- 根据映射好的正交排列表编写测试用例。

混合正交表

它解决了正交表中所有的因素和水平都是一致的缺陷，说白了就用allparis这个工具来生成测试用例。

重点需要理解的：**因果图和判定表及正交表。**

软件的质量管理

参考：https://www.cnblogs.com/Neeo/articles/11809478.html

iso9000系列的八大原则：

原则	内容	ISO9001标准条款	
一	以顾客为中心	组织依存与其顾客，因此，组织应理解顾客当前和未来的需求，满足顾客要求争取超越顾客期望。	0.1、5.2、7.2.1、7.2.3、7.3、7.5.3、7.5.4、8.2.1
二	领导作用	领导者将本组织的宗旨、方向和内部环境统一起来，并创造使用员工能够充分参与实现组织目标的环境。	5.1、5.3、5.4.1、5.4.2、5.5.2、5.5.3、5.6、6.1
三	全员参与	各级人员是组织之本，只有他们的充分参与，才能使他们的才干为组织带来最大的收益。	5.1、5.3、6.2、7.5.4
四	过程方法	将相关的资源和活动作为过程进行管理，可以更高效的得到期望的结果。	0.3、5、6、7、8（标准的每一条款都涉及过程）
五	管理的系统方法	针对设定的目标，识别、理解并管理一个由相互关联的过程所组织成的体系，有助于提高组织的有效性和效率	4.1、7.1、8.2.2
六	持续改进	持续改进是组织的一个永恒的目标	5.2、5.6、7.5、8.2.2、8.5.1、8.5.3
	基于事实的决策方法	对数据和信息的逻辑分析或者直觉判断是有效决策的基础。	7.5.2、7.5.5、7.6、8.2.3、8.3、8.4、8.5.2、8.5.3
八	互利的供方关系	通过互利的关系，增强组织及其供方创造价值的能力	7.4、8.3

八项质量管理原则的意义

是质量管理的理论基础。

用高度概括、易于理解的语言所表述的质量管理的最基本、最通用的一般性规律。

为组织建立质量管理体系提供了理论依据。

是组织的领导者有效地实施质量管理工作必须遵循的原则。

cmm（能力成熟度模型）

软件流程能力：遵循标准的软件流程，有多大可能达到预计的结果。软件流程能力提供一种有效的手段，可以预计软件组织承担某个项目最有可能结果是什么样子。

软件流程性能：遵循标准的软件流程，真正达到的结果是怎么样的，换而言之，软件流程能力是表示期望的结果，而软件流程性能表述的是软件表达的实际结果。

软件流程成熟度：指一个特定的流程，在多大程度上，被明白无误的定义、管理、衡量和控制，以及软件表达的效果是怎么样的。一个软件组织的软件流程成熟度是预示着它的软件流程能力有多大的发展潜力，这不仅指它的软件流程的丰富性、完备性，并且代表软件流程要做到一致。

CMM能力成熟度是一个阶梯式的模型。

CMM级别考核依据

过程能力等级	特点	关键过程域
1.初始级	软件过程是无序的，有时甚至是混乱的，对过程几乎没有定义，成功取决于个人努力，管理是反应式的。	
2.可重复级	建立了基本的项目管理过程来跟踪费用、进度和功能特性。制定了必要的过程纪律，能重复早先类似应用项目取得的成功。	a.**需求管理** b.软件项目计划 c.软件项目跟踪和监督 d.软件子合同管理 e.软件质量保证 f.**软件配置管理**
3.已定义级	已将软件管理和工程两方面的过程文档化、标准化、并综合成该组织的标准软件过程。所有项目均使用经批准、裁剪的标准软件过程来开发和维护软件。	a.组织过程定义 b.组织过程焦点 c.培训大纲 d.集成软件管理 e.软件产品工程 f.组际协调 g.**同行评审**
4.已管理级	收集对软件过程和产品质量的详细度量，对软件过程和产品都有定量的理解与控制。	a.定量的过程管理 b.软件质量管理
5.优化级	过程的量化反馈和先进的新思想、新技术促使过程不断改进	a.缺陷预防 b.技术变更管理 c.过程变更管理

划定流程成熟度的依据就是该级别的KPA 。

第一级别没有KPA，在CMM中，共有18个KPA分布在4个级别中。

什么是KPA呢？KPA（Key Process Area）是关键过程域。

如果你的公司想申请CMM3级，那么评定组织会首先评定你公司的二级6个KAP是否达到目标，如果达到则再评定3级的KPA，达到则是3级，否则还是2级。

每个KPA都设定了 2~4 个目标，我们称为KG（Key Goal）关键目标。这些关键目标是通过关键实践（key Practice）来完成的。

通过关键实践达到关键目标，关键目标达到，那么你的关键过程域也就达到了，这些关键过程域都达到了，你公司也就达到了某个级别。

但是，你也不要忽略非关键过程域，而是关键过程域和非关键因素相结合来评定。比如第2级别中的缺少某些工程活动，比如说软件测试活动。

如果你对这些理解起来比较困难，相对抽象，那么你可以参考医院的评定级别来考虑。比如医院如何被评选为三级甲等的，要达到哪些硬性指标，或者其他的指标，这跟CMM很相似。

那么这些我们都需要重点了解什么呢？如上表加粗部分，2级的需求管理、配置管理，3级的同行评审。

CMM:1级特点

一个软件公司成立之后，默认的就是CMM1级。

那么初始级的软件公司是处于什么样的状况呢?

- 一般不能提供开发和维护软件的稳定环境，缺乏健全的管理实践，不适当的规划和反应式的驱动体系会降低良好的软件工程实践所带来的效益。
- 在危急时刻，项目一般抛弃预定的规程，恢复到仅做编码和测试，项目的成功完全依赖于有一个杰出的经理及一个有经验的、战斗力强的软件团队，但当他们离开项目后，他们能使过程稳定的影响也随之消失。
- 等级1组织的过程能力是不可以预测的，过程是无序的。进度、预算、功能性和产品质量一般是不可预测的，实施情况依赖于个人的技能、知识和动机。

总的来说，全靠英雄主义来救场了。

CMM：2级特点

再来看可重复级的特点:

- 已建立管理软件项目的方针和实施这些方针的规程，基于在类似项目上的经验对新项目进行策划和管理，达到等级2目的是使软件项目的**有效**管理过程制度化，这使得组织能重复在以前类似项目上的成功实践。
- 项目已设置基本的软件管理和控制。
- 过程能力课概括为**有纪律**的，因为软件项目的策划和跟踪是稳定的， 能重复以前的成功，由于遵循切实可行的计划，项目过程处于项目管理系统的有效控制之下。

总的来说，根据以往的经验，总结下来，应用到新项目中，降低对人的依赖，并且项目是可控的。

但这仍然是不完善的， 因为这一级别只是项目级的，而不是组织级的。

CMM：3级特点

CMM2级只能针对项目的成功，会使下一个类似项目成功，但仍然是有局限性的，那么我们来寻求更高层次的成功，比如说组织级的成功，也就是已定义级。

- 全组织的开发和维护软件的标准过程已文档化，包括软件工程过程和软件管理过程，而且这些过程被集成为一个有机的整体，称为**组织的标准软件过程。**
- 组织中有一个专门组织的软件过程活动的组，例如软件工程过程组（SEPG，Software Engineering Process Group）。它负责整个组织的流程活动，明确组织各个角色以及角色职责，当然还要制定并实施全组织的培训计划。
- 项目根据其特征剪裁组织的标准软件过程，建立**项目定义软件过程。**
- 过程能力可概括为**标准的和一致性的。**在所建立的产品线内，成本、进度和功能性均受控制、对软件质量进行跟踪，整个组织范围内对已定义过程中的活动、角色和职责有共同的理解。

CMM：4级特点

再来看CMM4级，已管理的特点:

- 组织对软件产品的过程都设置定量的质量目标，对所有项目都测量其重要的软件过程活动的生产率和质量。利用全组织的软件过程数据库收集和分析从项目定义软件过程中得到的数据，软件过程均已配备有妥善定义的和一致的度量。
- 项目通过将其过程实施的变化限制在定量的可接受的范围之内，从而实现对其产品和过程的控制，开发新应用领域的软件所带来的风险是已知的，并得到精心的管理。
- 过程能力可概括为**可预测的**，因为过程是已测量的并在可测的范围内运行，组织能定量地预测过程和产品质量方面的趋势，软件产品具有可预测的高质量。

三级简单来说，就是说这个人漂亮不漂亮，下了定义，但4级可以通过调整一些量让这个人变得更漂亮，也就是纠正偏差（纠偏）。

4级也就是有了更多的度量指标，来调控整个项目的过程，所以软件的质量是可预测的，它强调的是量化管理。

那么我们根据度量指标来纠偏，以致于软件质量有更大的进步，所以，我们来看优化级。

CMM：5级特点

优化级的特点是：

- 整个组织集中精力进行不断的过程改进。为了预防缺陷出现，组织有办法识别出过程的弱点并预先予以加强。利用有关软件过程有效性的数据，识别出最佳技术创新，推广到整个组织。
- 所有软件项目组都分析缺陷，确定期原因，并且认真评价软件过程，以防止已知类型的缺陷再次出现，同时将经验教训告知其它项目。
- 过程能力的基本特征是**不断改进**，不断改善期项目的过程性能，为此，即采用现有过程中增量式前进的办法，也采用借助新技术、新方法进行革新的办法。

第5级强调的是流程的持续改进思想。

再回过头来看软件组织的流程，从无到有，从杂乱无章到某一个相同项目的成功重复，再到一致标准性到定量的发展，再到持续的优化。

所以，`1~4`级着重创建，而第5级，重点在优化、持续改进。

处于第5级的软件组织已经具备了自我改进的基础架构，因为它经历了前面的经验积累、技术储备，有了相当陈厚的沉淀。

六西格玛：

这里的西格玛指的是统计学的偏差，表示数据的离散程度。那六西格玛就是六倍的西格玛，也就是六倍的标准偏差。

六西格玛管理法

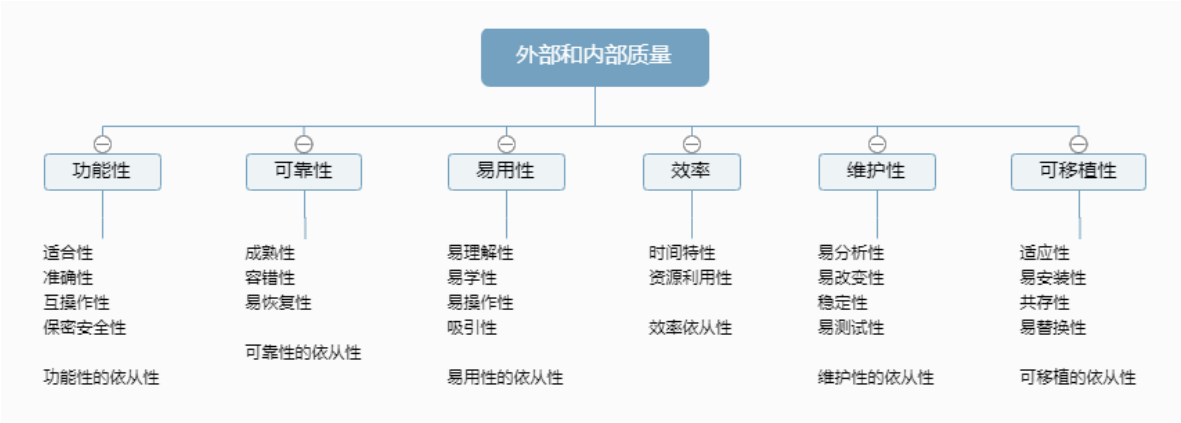
- 六西格玛管理法是以质量作为主线，以客户需求为中心，利用对事实和数据的分析，改进提升一个组织的业务流程能力，从而增强企业竞争力，是一套灵活，综合性的管理方法体系。
- 六西格玛要求企业完全从外部客户角度，而不是从自己的角度，来看待企业内部的各种流程。
- 利用客户的要求来建立标准，设立产品与服务的标准与规格，并以此来评估企业流程的有效性与合理性。
- 它通过提高企业流程的绩效来提高产品服务的质量和提升企业的整体竞争力。
- 通过贯彻实施来整合塑造一流的企业文化。

六西格玛模式的本质是一个全面管理概念，而不仅仅是质量提高手段

何为六西格玛（6 sigma）

6个西格玛流程能力等于百万个样本中，仅有3.4个缺陷。

软件的质量模型



需求管理

参考：https://www.cnblogs.com/Neeo/p/11809525.html

仔细参考文档进行研究。

缺陷管理

参考：https://www.cnblogs.com/Neeo/articles/11796009.html

要了解：

1. 缺陷的分类有哪些
2. 缺陷的优先级和严重性

软件缺陷的严重性参考：

严重等级	描述
5-Critical	系统瘫痪、异常退出、死循环、严重的计算错误等
4-VeryHigh	频繁的死机，系统大部分功能不可用
3-High	a.功能点没有实现，或不符合用户需求 b.数据丢失
2-Medium	a.影响一个相对独立的功能 b.仅仅在特定条件上发生 c.与产品需求定义不一致 d.断断续续的出现问题
1-Low	表面性错误（如错别字）

软件缺陷的优先级参考：

优先级别	描述
5-Urgent	最高优先级。在这个错误影响下，系统几乎不可用。
4-VeryHigh	高优先级。错误对这套系统的能力产生严重的影响。
3-High	中优先级。如果这个错误存在与系统中，会制约开发和测试的活动的进行，如果先前没有修复它，那么需要在发布前修复它。
2-Medium	低优先级。不会延迟发布，但是会在以后修正这个错误。
1-Low	最低优先级。时间和资源允许时修正。

缺陷的状态：

编号	缺陷状态	描述
1	提交（submited）	已提交的缺陷
2	打开（open）	确认“提交缺陷”，等待处理
3	拒绝（rejected）	拒绝“提交缺陷”，不需要修复或不是缺陷、重复缺陷、无法重现
4	修复（resolved）	缺陷被修复
5	关闭（closed）	确认缺陷已修复，将其关闭
6	推迟（later）	可以在以后解决，但要确定修复日期或版本

缺陷的修改说明：

现在，我们再来聊聊缺陷修改中碰到的一些情况。
开发人员拒绝修改的缺陷

- 程序员无法重现或者现象难以捕捉
- 没有明确的报告以说明重现缺陷的步骤
- 程序员无法读懂的缺陷报告
- 用户很少使用或者不符合用户使用习惯的操作出错
- 由不受信任的测试人员提出

不是所有缺陷都会修改

- 市场的压力使得产品最终发行有时间限制
- 测试人员错误理解或者不正确操作引出的缺陷(FAQ)
- 错误的修改影响的模块较多，带来的风险较大(遗留)
- 修改性价比太低
- 缺陷报告中提出的问题很难重现

缺陷管理工具

- JIRA：JIRA是集项目计划、任务分配、需求管理、缺陷跟踪于一体的软件。它基于Java架构的管理系统，被广泛应用于缺陷跟踪、客户服务、需求收集、流程审批、任务跟踪、项目跟踪和敏捷管理等工作领域。

JIRA创建的问题类型包括New Feature（新功能）、Bug（缺陷）、Task（任务）和Improvement（改进）四种，还可以自定义，所以它也是一个过程管理系统。同时融合了项目管理、任务管理和缺陷管理。JIRA功能强大，可配合着一些组件及工具一起使用，如： Confluence 用于 wiki 管理需求，JIRA管理任务、进度和 Bug 。

JIRA设计以项目为主线，产品、测试结合管理，通过issues控制管理。因此它的核心诉求还是围绕issue展开的，以issue驱动管理、分工、以及团队协作，进而实现项目的规划、建设，终完成产品开发。

- 禅道：禅道项目管理软件（简称：禅道）集产品管理、项目管理、质量管理、文档管理、组织管理和事务管理于一体，是一款功能完备的项目管理软件，完美地覆盖了项目管理的核心流程。

  禅道的主要管理思想基于国际流行的敏捷项目管理方式—Scrum。Scrum是一种注重实效的敏捷项目管理方式，它规定了核心的管理框架，但具体的细节还需要团队自行扩充。禅道在遵循其管理方式基础上，又融入了国内研发现状的很多需求，比如bug管理，测试用例管理，发布管理，文档管理等。因此禅道不仅仅是一款scrum敏捷项目管理工具，更是一款完备的项目管理软件。基于scrum，又不局限于scrum。

  禅道最大的特色是创造性的将产品、项目、测试这三者的概念明确分开，互相配合，又互相制约。通过需求、任务、bug来进行交相互动，最终通过项目拿到合格的产品。

目前，禅道（市场占有率50%+）和JIRA（市场占有率30%+，剩下就其他的）用的人较多。禅道和JIRA功能有很多的通用性，所以，学会一门另一门就会了。

单元测试框架unittest

在unittest中，讲了：

1. 测试用例，必须是以类的形式，类必须继承`unittest.TestCase`，用例名必须是以test开头的
2. setUp/teardown,用例执行前，执行后要做的事情
3. setupClass/teardownClass，在类中，所有的用例执行前后要做的事情。
4. 用例的执行结果，`.`通过（pass），`F`表示失败（failed），`E`表示运行错误（errors）。
5. 测试套件Suite
    1. 手动，手动的实例化用例对象，手动的创建测试套件，手动的将测试用例添加测试套件中，然后交给执行去执行。

```
import unittest

class MyCase(unittest.TestCase):

    def test_case_01(self):
        self.assertEqual(1, 1)

if __name__ == '__main__':

    case = MyCase(methodName='test_case_01')
    suite = unittest.TestSuite()
    suite.addTest(case)
    print(suite.countTestCases())
    unittest.TextTestRunner(verbosity=2).run(suite)
```

    2. 半自动，手动的生成测试套件Suite对象，在示例suite对象的时候，将用例也添加到套件中，然后交给执行器去执行。

```
import unittest

class MyCase(unittest.TestCase):

    def test_case_01(self):
        self.assertEqual(1, 1)

if __name__ == '__main__':

    suite = unittest.makeSuite(testCaseClass=MyCase, prefix='test')  # 有的时候，makeSuite可能不会自动的提示，会飘黄，但是不要怕，写就完了。
    unittest.TextTestRunner(verbosity=2).run(suite)
```

3. 全自动，unitest.main

```
import unittest

class MyCase(unittest.TestCase):

    def test_case_01(self):
        self.assertEqual(1, 1)

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

6. verbosity，控制用例的输出详细程度：
    1. `verbosity=0`，静默模式，输出测试的简单结果
    2. `verbosity=1`，默认模式，输出测试一般结果
    3. `verbosity=2`，详细结果，输出测试的详细结果
7. 跳过
    1. unittest.skip(reason), reason:跳过的原因
    2. unittest.skpilf(condition, reason), condition跳过的条件，reason是跳过原因。
8. unittest.TestLoader.discover

```
discover = unittest.TestLoader().discover(
    start_dir=base_dir,    # 该参必传
    pattern='test*.py',    # 保持默认即可。
    top_level_dir=None
    )
unittest.TextTestRunner(verbosity=2).run(discover)
```

通过`TestLoader()`实例化对象，然后通过实例化对象调用discover方法，discover根据给定目录，递归找到子目录下的所有符合规则的测试模块，然后交给TestSuit生成用例集suite。完事交给TextTestRunner执行用例。
该discover方法接收三个参数：

- start_dir：要测试的模块名或者测试用例的目录。
- pattern="test*.py"：表示用例文件名的匹配原则，默认匹配以`test`开头的文件名，星号表示后续的多个字符。
- top_level_dir=None：测试模块的顶层目录，如果没有顶层目录，默认为None。

注意！！！意！！

- discover对给定的目录是有要求的，它只**识别Python的包，也就是目录内有`__init__.py`文件的才算是Python的包，只要是要读取的目录，都必须是包。**
- 关于start_dir和top_level_dir的几种情况：
  - start_dir目录可以单独指定，这个时候，让top_level_dir保持默认（None）即可。
  - start_dir == top_level_dir， start_dir目录与top_level_dir目录一致，discover寻找start_dir指定目录内的符合规则的模块。
  - start_dir < top_level_dir，start_dir目录是top_level_dir目录的子目录。discover寻找start_dir指定目录内的符合规则的模块。
  - start_dir > top_level_dir，start_dir目录如果大于top_level_dir目录，等待你的是报错`AssertionError: Path must be within the project`。说你指定的路径（start_dir）必须位于项目内（top_level_dir）。

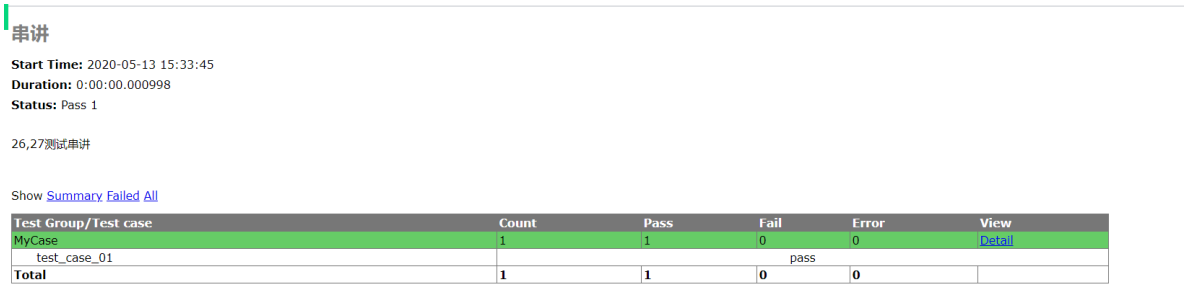
9. 测试报告，有下面两个插件可用，并且该插件在网上有各种改版的版本。在pypi的官网下不到，下载参考：https://www.cnblogs.com/Neeo/articles/7942613.html
   1. HTMLTestRunner
   2. BSTestRunner

```
import unittest
from HTMLTestRunner import HTMLTestRunner

class MyCase(unittest.TestCase):

    def test_case_01(self):
        self._testMethodName = '登录'
        self._testMethodDoc = '登录接口的描述'
        self.assertEqual(1, 1)

if __name__ == '__main__':
    # 构造 suite
    suite = unittest.makeSuite(testCaseClass=MyCase)
    f = open('./report.html', 'wb')
    HTMLTestRunner(
        title='串讲',
        description='26,27测试串讲',
        stream=f,
        verbosity=2
    ).run(suite)
```



串讲

Start Time: 2020-05-13 15:33:45
Duration: 0:00:00.000998
Status: Pass 1

26,27测试串讲

Show Summary Failed All

Test Group/Test case	Count	Pass	Fail	Error	View
MyCase	1	1	0	0	Detail
test_case_01			pass		
Total	**1**	**1**	**0**	**0**	

问题1：如何在测试报告中展示用例的名称和描述信息使用`self._testMethodName`和`self._testMethodDoc`

问题2：如何获取当前执行中，共执行了多少个用例？有多少通过的？有多少失败的？有多少错误的？

```
if __name__ == '__main__':
    # 构造 suite
    suite = unittest.makeSuite(testCaseClass=MyCase)
```

```
    f = open('./report.html', 'wb')
    result = HTMLTestRunner(
        title='串讲',
        description='26,27测试串讲',
        stream=f,
        verbosity=2
    ).run(suite)

    print(result.__dict__['testsRun'])
    print(result.__dict__['error_count'])
    print(result.__dict__['failure_count'])
    print(result.__dict__['success_count'])
```

10. 常用的断言：

```
self.assertEqual()
self.assertTrue()
self.assertFalse()
```

单元测试：pytest

参考：https://www.cnblogs.com/Neeo/articles/11832655.html

pytest可以执行unittest风格的测试用例,无须修改unittest用例的任何代码,有较好的兼容性。 pytest插件丰富,比如flask插件,可用于用例出错重跑;还有xdist插件,可用于设备并行执行。

1. pytest要下载

```
pip install pytest
```

2. pytest中，用例支持函数和类两种写法
3. setup和teardown有：
    1. 模块级别的：setup_module/teardown_module
    2. 函数级别的：setup_function/teardown_function
    3. 类级别的：setup_class/teardown_class
    4. 类中方法级别的：setup_method/teardown_method
4. 配置文件ini，在项目的根目录下新建一个`pytest.ini`文件。

```
[pytest]
addopts = -s -v
testpaths = ./scripts
python_files = test_*.py
python_classes = Test*
python_functions = test_*
```

5. 跳过用例
    1. skip
    2. skipif

```
import pytest

@pytest.mark.skip(condition='我就是要跳过这个用例啦')
def test_case_01():
    assert 1

@pytest.mark.skipif(condition=1 < 2, reason='如果条件为true就跳过用例')
def test_case_02():
    assert 1
```

6. 标记预期失败，如果我们事先知道测试函数会执行失败，但又不想直接跳过，而是希望显示的提示。

   Pytest 使用 `pytest.mark.xfail` 实现预见错误功能：：

   ```
   xfail(condiition, reason, [raises=None, run=True, strict=False])
   ```

   需要掌握的必传参数的是：

   - condition，预期失败的条件，当条件为真的时候，预期失败。
   - reason，失败的原因。

   那么关于预期失败的几种情况需要了解一下：

   - 预期失败，但实际结果却执行成功。
   - 预期失败，实际结果也执行执行失败。

7. **参数化**，pytest身为强大的测试单元测试框架，那么同样支持DDT数据驱动测试的概念。也就是当对一个测试函数进行测试时，通常会给函数传递多组参数。比如测试账号登陆，我们需要模拟各种千奇百怪的账号密码。

   当然，我们可以把这些参数写在测试函数内部进行遍历。不过虽然参数众多，但仍然是一个测试，当某组参数导致断言失败，测试也就终止了。

   通过异常捕获，我们可以保证程所有参数完整执行，但要分析测试结果就需要做不少额外的工作。

   在 pytest 中，我们有更好的解决方法，就是参数化测试，即每组参数都独立执行一次测试。使用的工具就是 `pytest.mark.parametrize(argnames, argvalues)`。

   - argnames表示参数名。
   - argvalues表示列表形式的参数值。

   使用就是以装饰器的形式使用。

8. 固件（Fixture）是一些函数，pytest 会在执行测试函数之前（或之后）加载运行它们，也称测试夹具。

   我们可以利用固件做任何事情，其中最常见的可能就是数据库的初始连接和最后关闭操作。

   Pytest 使用 `pytest.fixture()` 定义固件，下面是最简单的固件，访问主页前必须先登录：

   ```
   import pytest

   @pytest.fixture()
   def login():
       print('登录....')

   def test_index(login):
       print('主页....')
   ```

9. 常用插件：

1. pytest-html
2. allure，重点掌握
    1. 首先是allure模块生成json数据，然后allure插件读取json数据来生成allure报告，但是，要注意，json数据不会自动清空，你要根据你的使用情况来手动的清除json数据。
3. pytest-ordering
4. pytest-rerunfailures

moco框架

用来做假数据的，通过配置文件（config.json）来快速的配置一个接口，用于开发没有完成真正的接口的时候，应急来用，不耽误当前的测试或者其他工作。

安装和使用：https://www.cnblogs.com/Neeo/articles/11514251.html

postman

Postman是一款非常流行的HTTP/HTTPS接口测试工具，入门简单，功能强大，不但可以进行接口手动测试，还可以非常方便的进行自动化测试。支持参数化、断言、用例设计、测试报告等功能。

安装：官网下载，https://www.getpostman.com/downloads/

重点：

1. 环境管理
    1. 自定义环境，需要手动的选中。
    2. 全局环境，作用于整个postman环境
    3. 集合环境，作用于当前集合
    4. postman内置的动态变量，比如时间戳，UUID，

    普通的环境使用`{{变量名}}`，内置动态变量使用`{{g变量名}}`
2. 断言，不仅仅能用来断言，还能做请求之后要做的事情。
    1. 状态码断言
    2. json断言
    3. set和get变量
3. 特殊接口
    1. webservice接口，以xml做数据交互的HTTP请求
    2. cookies处理，postman默认的，如果请求的响应中，有cookies返回，就以域名的形式保存该cookies到cookies管理器中；后续有相同域名的请求，就自动的携带。
    3. token，你需要说动的将接口返回的token值，保存到全局变量中，在有需要该token的接口，发送之前，将token值手动的携带
    4. 签名接口，都是通过在请求之前把相关数据处理好，然后在请求中携带
4. 集合自动化，在postman中，可以将同一个项目中的接口封装到一个集合中，然后可以对集合中的接口做自动化执行。在自动化执行中，使用数据驱动（读取CSV文件），还可以做集合的公共断言
5. 命令行执行：使用newman插件对postman脚本在终端执行并且生成测试报告
    1. newman是nodejs为postman终端执行开发的，依赖nodejs环境（nodejs版本不要太低）

2. 使用nodejs的npm下载newman，但是npm默认的服务器在国外，下载容器失败，我们需要为npm配置国内的镜像——cnpm淘宝镜像。

参考：https://www.cnblogs.com/Neeo/articles/12186502.html

requests

使用前先下载：

```
pip install requests
```

使用：

1. 关于请求的的类型：
    1. get
    2. post
    3. ....
2. 关于请求的请求参数：
    1. headers
    2. params，get请求结合
    3. data，post请求
    4. json
    5. cookies，强调，如果请求中有需要携带cookies，不要放在headers头里，而是单独的使用cookies参数携带。

```
import requests

# 登录
response =
requests.post(url='http://www.neeo.cc:6002/pinter/bank/api/login', data=
{"userName": "admin", "password": 1234})
print(response.cookies.get_dict())

# 查询请求依赖登录的cookies
response = requests.get(url='http://www.neeo.cc:6002/pinter/bank/api/query',
params={"userName": "admin"}, cookies=response.cookies.get_dict())
print(response.json())

"""
requests处理cookies和postman处理cookies的区别：
    - postman将cookies添加到headers头中携带
    - requests需要使用cookies参数携带，而不是放在headers头中
"""
```

    6. file，文件对象

```
import requests

file = {"file": open(r'D:\video\20200514串讲\note\_两年测试经验--自动化测试工程师--王xx.doc', 'rb')}

response = requests.post(url='http://www.neeo.cc:6001/post', files=file)
print(response.json())
```

7. timeout

3. 关于响应的方法都有哪些?
    1. response.json
    2. response.status_code
    3. response_encoding

```
import requests
response = requests.get(url='https://www.autohome.com.cn/news/')
print(response.encoding)
response.encoding = "GBK"    # 处理乱码
print(response.text)
```

    4. response.text
    5. response.content
    6. response.cookies
        1. response.cookies.get_dict()
        2. response.cookies.items()
    7. response.iter_content(chunk_size),以块的形式下载，chunk_size是块的大小

跟requests搭配的是bs4模块，用前需要下载，然后导入：

```
"""
pip install Beautifulsoup4
"""
from bs4 import Beautifulsoup

soup = Beautifulsoup(response.text, 'html.parser')
soup.find
soup.find_all
soup.get
soup.text
```

接口自动化框架

逻辑：

1. 读取Excel表格，
2. 使用pytest的参数化来获取Excel表格中的每一行（一行是一个用例）
3. 使用requests发请求，获取请求结果
4. 将请求的结果和预期值做断言
5. 使用allure来生成测试报告
6. (可选的操作)，添加日志功能
7. (使用SMTP)发送邮件测试报告

未完成的事情是，没有处理用例之间的数据依赖，比如查询余额接口依赖登录接口的cookies值。那如何处理cookies呢?

第一种方式:

1. 首先，让被依赖的请求先发送，然后，如果该请求有cookies返回，就以域名的形式保存到本地
2. 当有向同一个域名发请求的时候，自动的带上该cookies

第二种方式:

1. 首先，让被依赖的请求先发送，然后，将该请求的请求参数和响应参数都临时保存到用例对象中
2. 自己实现规则`${依赖的用例编号>依赖参数>依赖参数中的那个字段}$`
3. 使用re来匹配规则，取值后替换到原来的数据中。

补充两个模块

deepdiff：https://www.cnblogs.com/Neeo/articles/12785493.html

jsonpath-rw: https://www.cnblogs.com/Neeo/articles/12787888.html

还可以参考使用：beyond compare

```
from jsonpath_rw import parse, parser   # 我们这里用的是pars， 而不是parser

json_data = {
    "store": {
        "book": [
            {
                "category": "reference",
                "author": "Nigel Rees",
                "title": "Sayings of the Century",
                "price": 8.95
            },
            {
                "category": "fiction",
                "author": "Evelyn Waugh",
                "title": "Sword of Honour",
                "price": 12.99
            },
            {
                "category": "fiction",
                "author": "Herman Melville",
                "title": "Moby Dick",
                "isbn": "0-553-21311-3",
                "price": 8.99
            },
            {
                "category": "fiction",
                "author": "J. R. R. Tolkien",
                "title": "The Lord of the Rings",
                "isbn": "0-395-19395-8",
                "price": 22.99
            }
        ],
        "bicycle": {
            "color": "red",
            "price": 19.95
```

```
            }
        },
        "expensive": 10,
        "category": "xxxxx"
}

rule = "$..category"
match = parse(rule).find(json_data)
# print(match)
for i in match:
    print(i.value)
```

接口自动化平台开发

解决痛点：解决禅道无法执行用例

如何编写面试中的项目：

项目名称：茬苒测试平台 ↵

项目描述：↵

用 Django 框架开发的网站测试平台，具有定时任务，批量导入测试用例，一键批量执行生成测试报告。可视化展示用例信息。↵

责任描述：↵

1. 参与需求评审，查看设计文档的逻辑性，如不合理及时提出；↵
2. 绘制软件组织架构图，了解软件总体架构及功能，根据软件组织架构图书写测试范围列表，根据测试范围列表以及设计文档设计测试用例；↵
3. 从 Excel 中一键批量导入测试用例；↵
4. 一键批量执行测试用例并生成测试报告，支持测试报告的下载和发送执行的邮箱↵
5. 结合禅道记录缺陷、跟踪缺陷，提交测试记录、缺陷报告单；↵
6. 结合 echarts 生成用例的相关可视化图表，展示近一年来用例的创建情况，用例的执行情况和通过情况展示↵
7. 后台使用 requests 和 deepdiff 结合 jsonath-rw 和 unittest 做断言，并生成测试报告，另外记录执行日志，并计算用例的覆盖(用例通过率)率↵
8. 使用 APScheduler/celery 做定时任务，检查项目表中是否有当天结束的项目，如果有，就批量执行该项目下的所有用例并且生成测试报告，并且发送邮件↵

关键字： APP 测试、功能测试、接口测试↵

技术点：django+APScheduler/celery+django 发邮件 + unites+requests+deepdiff+jsonath-rw+echarts↵

禅道：

http://www.neeo.cc:6003/

账号是：admin

密码是：root!1234

django发邮件

参考：https://www.cnblogs.com/Neeo/articles/11199085.html

selenium

selenium IDE

会selenium ide的安装和使用，及导出为pytest脚本。
安装：

- 谷歌插件
- 火狐插件

安装示例，以谷歌为例：

1. 打开谷歌商店，https://chrome.google.com/webstore/category/extensions?hl=zh-CN
2. 搜搜selenium ide，并且添加到扩展程序中即可。

selenium webdriver

下载selenium模块：

```
pip install selenium
```

要为各个浏览器单独配置浏览器驱动，参考：https://www.cnblogs.com/Neeo/articles/10671532.html

webdriver常用的类及参数的导入：

```
from selenium import webdriver  # 浏览器驱动对象
from selenium.webdriver.common.keys import Keys   # 键盘事件
from selenium.webdriver import ActionChains  # 鼠标事件
from selenium.webdriver.support import expected_conditions as EC    # EC 跟wait联用
from selenium.webdriver.support.wait import WebDriverWait   # 显式等待
from selenium.webdriver.common.by import By   # By选择器
# 谷歌无头和火狐无头的参数导入
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.firefox.options import Options
```

基本操作：

```
from selenium import webdriver
from selenium.webdriver.common.by import By   # By选择器
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
driver = webdriver.Chrome()  #
executable_path=r"C:\Python36\Scripts\chromedriver.exe" 你的浏览器驱动的绝对路径
wait = WebDriverWait(driver=driver, timeout=10, poll_frequency=0.5)  # 在最开始就是实例化一个显式等待对象

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

# 标签定位
driver.find_element(By.ID, 'su')
By.ID
By.CLASS_NAME
By.LINK_TEXT
By.PARTIAL_LINK_TEXT
By.TAG_NAME
By.XPATH
By.CSS_SELECTOR

driver.save_screenshot('xxx.png')  # 保存当前可视页面的图片，图片类型必须是 png
driver.switch_to.window()  # 切换窗口
driver.switch_to.alert()  # 从当前文档流切换到alert中
driver.switch_to.frame()  # 从当前的文档流中切换到iframe标签中， iframe标签嵌套在当前文档流中的，但在当前文档流中，定位不到它，只能切换
driver.switch_to.parent_frame()  # 切换到父级iframe中
driver.switch_to.default_content()  # 从alert或iframe中切换到当前的文档流中
# 上述的 switch_to替代了：
driver.switch_to_window()
driver.switch_to_default_content()
driver.switch_to_frame()
driver.switch_to_alert()
driver.switch_to_default_content()


driver.title  # 获取当前窗口的title
driver.current_url  # 获取当前窗口的url
driver.current_window_handle  # 获取当前窗口的窗口对象
driver.page_source  # 获取当前页面的源码
driver.execute_script("alert('xxx000');")   # 执行js代码
driver.minimize_window()  # 窗口对象最小化
driver.maximize_window()  # 窗口最大化

driver.implicitly_wait(time_to_wait=10)  # 隐式等待
# 显式等待
wait.until(EC.visibility_of_element_located((driver.find_element(By.ID, 'su')))).send_keys("xxxxx")  # 出现dom树中，且该element可见
# WebDriverWait(driver=driver).until(EC.presence_of_element_located())  # 只要element出现在dom中即可，无需可见


# 问，拿到一个标签对象之后能干什么？
inp = driver.find_element(By.ID, 'su')
# 获取name属性
```

```
inp.get_attribute('name')
inp.get_property('name')
inp.text
inp.tag_name
inp.value_of_css_property('color')
inp.send_keys()
inp.click()
inp.clear()
inp.find_element()


driver.quit()  # 退出浏览器
driver.close()  # 关闭浏览器的当前窗口对象，如果只有一个窗口对象，等于是退出浏览器
```

css selector 更多参考： https://www.cnblogs.com/Neeo/articles/12362920.html#css-selector

截图相关：https://www.cnblogs.com/Neeo/articles/10672437.html

建议

建议大家都买一个服务器用：

- 阿里
- 腾讯
- 百度
- 华为：https://activity.huaweicloud.com/discount_area_v5/index.html?ggw_hd