**14、如果一个元素无法定位，你一般会考虑哪些方面的原因？**

1.页面加载元素过慢，加等待时间

2.页面有frame框架页，需要先跳转入frame框架再定位

3.可能该元素是动态元素，定位方式要优化，可以使用部分元素定位或通过父节点或兄弟节点定位。

4.可能识别了元素，但是不能操作，比如元素不可用，不可写等。需要使用js先把前置的操作完成，

**16、遇到frame框架页面怎么处理？**

先用driver.switch\_to.frame()跳转进去frame，

然后再操作页面元素，

操作完后使用driver.swith\_to.default\_content()跳转出来

**17、遇到alert弹出窗如何处理？**

使用driver.switch\_to.alert()方法先跳转到alert弹出窗口

br.switch\_to.alert.accept()点击确定按钮，

br.switch\_to.alert. dismiss ()点击取消按钮，

br.switch\_to.alert. text()获得弹出窗口的文本。

**18、如何处理多窗口？**

1.我们在点击链接前使用driver.current\_window\_handle获得当前窗口句柄。

2.再点击链接。点击后通过driver.window\_handles获得所有窗口的句柄，

3.然后再循环找到新窗口的句柄，然后再通过driver.switch\_to.window()方法跳转到新的窗口。

19、怎么验证元素是enable/disabled/checked状态？

定位元素后：分别通过isEnabled()，isSelected()，isDisplayed()三个方法进行判断。

20、 如何处理下拉菜单？

在Selenium中有一个叫Select的类，这个类支持对下拉菜单进行操作。使用方法如下：

1.定位元素

2.把定位的元素转化成Select对象。

sel = Select(定位的元素对象)

3.通过下标或者值或者文本选中下拉框。  
sel.select\_by\_index(index);  
sel.select\_by\_value(value);  
sel.select\_by\_visible\_text(text);

21、在日历这种web 表单你是如何处理的?

首先要分析当前网页试用日历插件的前端代码，看看能不能通过元素定位，点击日期实现，如果不能，可能需要借助javascript。还有些日历控件一个文本输入框，可以直接sendKeys()方法来实现传入一个时间的数据。

**1、你会封装自动化测试框架吗？**

先在.py文件中调通要封装的测试脚本。避免封装后分不清是脚本还是框架使用问题

1.1创建pytest文件。以test开头

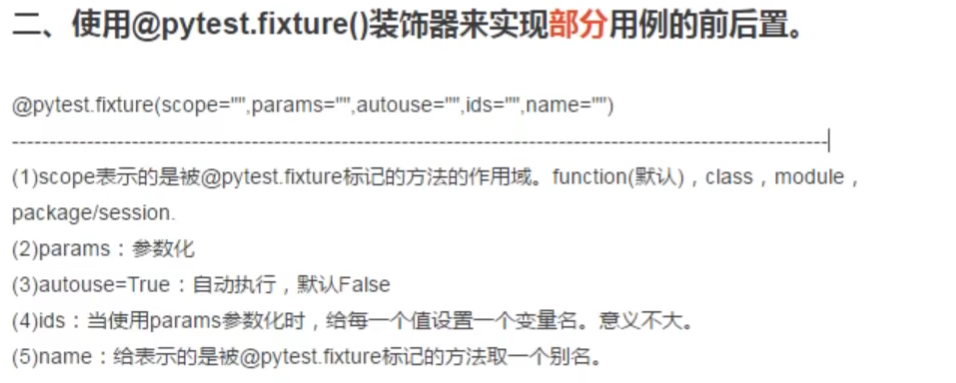
1.2创建测试类，类以Test开头，记得T大写

1.3创建测试方法，以test开头，执行顺序按照啊斯康码执行

Pytest脚本优化

2.1不同的测试方法，有相同冗余的代码前后置封装

**@pytest.fixture()**



2.2数据驱动技术:

**@pytest.mark.parametrize**("参数1"，“参数2”,([‘参数1’],['参数2']))

**3、请描述一下自动化测试流程？**

1.编写自动化测试计划

2.设计自动化测试用例

3.编写自动化测试框架和脚本

4.调试并维护脚本

**6、自动化测试发现BUG多吗？**

不多，因为之前项目组是把已经测试通过的基本功能再进行自动化脚本编写和在后续版本执行自动化测试，它主要是保证已经测试通过的功能在新版本更新后没有问题。

**7、你觉得自动化测试的价值在哪里？你们公司为什么要做自动化测试？**

引用自动化测试之后，能代替大量繁琐的回归测试工作，把业务测试人员解放出来，既而让业务测试人员把精力集中在复杂的业务功能模块上，自动化测试一般是对稳定下来的功能进行自动化，保证不会因为产品的更新导致之前稳定下来的功能出现BUG

**8、自动化测试有误报过bug吗？产生误报怎么办？**

有误报过，有时候自动化测试报告中显示发现了bug,实际去通过手工测试去确认又不存在该bug。

误报原因一般是：

1.元素定位不稳定，需要尽量提高脚本的稳定性；

2.开发更新了页面但是测试没有及时更新维护!

9、自动化测试过程中，你遇到了哪些问题，是如何解决的？

1.频繁地变更页面，经常要修改页面对象类里面的代码

2.自动化测试偶尔出现过误报

3.自动化测试结果出现覆盖的情况：Jenkins根据时间建立文件夹

4.自动化测试代码维护比较麻烦

5.自动化测试进行数据库对比数据

10、在上一家公司做自动化测试用的什么框架？

可以说出以下自己擅长的一种：

1.python+selenium+unittest+htmltestrunner

2.python+selenium+pytest+allure

3. robotframework+Selenium2Library

**12、在selenium自动化测试中，你一般完成什么类型的测试？自动化覆盖率？**

主要是冒烟测试和回归测试。回归测试主要写一些功能稳定的场景，通过自动化手段去实现，节约测试时间。因为自动化测试用例也是在不断的更新和迭代，没有刻意去统计，大概在30%-40%左右！

**22、举例说明一下你遇到过那些异常**

常见的selenium异常有这些：

NoSuchElementException：没有该元素异常  
TimeoutException ： 超时异常

ElementNotVisibleException ：元素不可见异常  
NoSuchAttributeException ：没有这样属性异常  
NoSuchFrameException ：没有该frame异常

**23、关闭浏览器中quit和close的区别**

简单来说，两个都可以实现退出浏览器，close是关闭你当前的tab页面，而quit是关闭全部浏览器tab页面，并退出浏览器。知道这两个区别，我们就知道quit一般用在结束测试之前的操作，close用在执行用例过程中关闭某一个页面的操作。

**24、在Selenium中如何实现截图，如何实现用例执行失败才截图**

在Selenium中提供了一个get\_screenshot\_as\_file()的方法来截图的，一般结合try/except捕获异常时使用，进行错误截图。

**25、如何实现文件上传？**

定位元素后，直接使用send\_keys()方法设置就行，参数为需要上传的文件的路径。

26、自动化中有哪三类等待？他们有什么特点？

1.线程等待（强制等待）如time.sleep(2)：线程强制休眠2秒钟，2秒过后，再执行后续的代码。建议少用。

2.implicitlyWait（隐式等待）

from selenium.webdriver.support import expected\_conditions as ec # 提供等待条件的

from selenium.webdriver.support.ui import WebDriverWait # 提供等待类

from selenium.webdriver.common.by import By # 提供元素寻找方法的

from selenium import webdriver

driver = webdriver.Chrome()

"""

隐式等待：设置一个超时时间，在这个时间内，不断的寻找元素（每隔0.5s找一次）

若超时仍未找到则抛出timeout异常，若在时间内找到了，则继续往下执行代码

**优点：**

**使用简单，一次添加终生有效**

**一般在创建 driver 之后设置**

**设置以后，在其之后执行的元素寻找都有效**

**缺点：**

**无法指定元素进行等待**

**需要等页面全部加载完成，相对而言浪费时间**

"""

# driver.implicitly\_wait(10)

driver.get("http://127.0.0.1:8088/")

# 找到用户名输入框，并输入用户名

driver.find\_element\_by\_name("username").send\_keys("libai")

# 找到密码输入框，并输入密码

driver.find\_element\_by\_name("password").send\_keys("opmsopms123")

# 找到登录按钮，并点击

driver.find\_element\_by\_tag\_name("button").click()

显示等待：设置一个超时时间和一个元素查找条件，在这个时间内不断寻找元素

若超时仍未找到则抛出timeout异常，若在时间内找到了，则继续往下执行代码

**优点：**

**不需要等待页面全部加载完成，相对节省时间**

**可以指定需要等待的元素，对于某些加载特别缓慢的元素，可以为其增加时间**

**缺点：**

**使用相对复杂，代码多**

**每次等待都需要再写一遍代码**

"""

ele = WebDriverWait(driver, 20, 0.5).until(

ec.visibility\_of\_element\_located(

(By.CSS\_SELECTOR,

"body > section > div.left-side.sticky-left-side > div.left-side-inner > ul > li:nth-child(1) > a > span")

)

)

ele.click()

**当隐式等待与显示等待同时出现，取时间更多的一个**

比如隐式等5秒，显示等7秒，则最长等待时间为 7 秒

"""

28、**什么是PO模式，为什么要使用它**

PO是Page Object 模式的简称，它是一种设计思想，意思是，把一个页面，当做一个对象，页面的元素和元素之间操作方法就是页面对象的属性和行为，PO模式一般使用三层架构，分别为：基础封装层BasePage，PO页面对象层，TestCase测试用例层。