Table of Contents

UI自动化测试课程

序号	章节	知识点
1	UI自动化测试介绍	1. UI自动化测试
2	Web自动化测试基础	1. Web自动化测试框架 2. 环境搭建 3. 元素定位和元素操作 4. 鼠标和键盘操作 5. 元素等待 6. HTML特殊元素处理 7. 验证码处理
3	移动自动化测试基础	1. 移动自动化测试框架 2. ADB调试工具 3. UIAutomatorViewer工具 4. 元素定位和元素操作 5. 滑动和拖拽事件 6. 高级手势TouchAction 7. 手机操作
4	PyTest框架	1. PyTest基本使用 2. PyTest常用插件 3. PyTest高级用法
5	PO模式	1. 方法封装 2. PO模式介绍 3. PO模式实战
6	数据驱动	1. 数据驱动介绍 2. 数据驱动实战
7	日志收集	1. 日志相关概念 2. 日志的基本方法 3. 日志的高级方法
8	黑马头条项目实战	1. 自动化测试流程 2. 项目实战演练

课程目标

1. 掌握使用Selenium实现Web自动化测试的流程和方法，并且能够完成自动化测试脚本的编写。
2. 掌握使用Appium实现移动自动化测试的流程和方法，并且能够完成自动化测试脚本的编写。
3. 掌握如何通过PyTest管理用例脚本，并使用Allure生成HTML测试报告。
4. 掌握使用PO模式来设计自动化测试代码的架构。
5. 掌握使用数据驱动来实现自动化测试代码和测试数据的分离。
6. 掌握使用logging来实现日志的收集。

项目实战

目标

1. 熟悉自动化测试的流程
2. 能够对一个项目实现自动化测试
3. 熟练使用selenium常用的API
4. 熟练使用appium常用的API
5. 能够把PyTest应用到项目中
6. 能够把PO模式应用到项目中
7. 能够把数据驱动应用到项目中
8. 能够把日志收集功能应用到项目中

自动化测试流程

目标

1. 熟悉自动化测试的流程

1. 自动化测试的流程

1. 需求分析
2. 挑选适合做自动化测试的功能
3. 设计测试用例
4. 搭建自动化测试环境 [可选]
5. 设计自动化测试项目的架构 [可选]
6. 编写代码
7. 执行测试用例
8. 生成测试报告并分析结果

项目介绍

目标

1. 熟悉项目需求
2. 了解项目架构
3. 熟悉待测功能模块

---

1. 黑马头条项目简介

1.1 项目背景

作为一个IT教育机构，拥有自己开发且实际运营的产品，将开发和运营的技术作为授课的内容，对于学员而言学到的都是一手的真实案例和实际经验，知识内容也可以细化深入。 而且一个产品就可以涵盖公司多个学科的技术，衍生的课程价值辐射多个学科。这可以作为公司的一个核心竞争力。

1.2 产品定位

一款汇集科技资讯、技术文章和问答交流的用户移动终端产品。 用户通过该产品，可以获取最新的科技资讯，发表或学习技术文章，讨论交流技术问题。

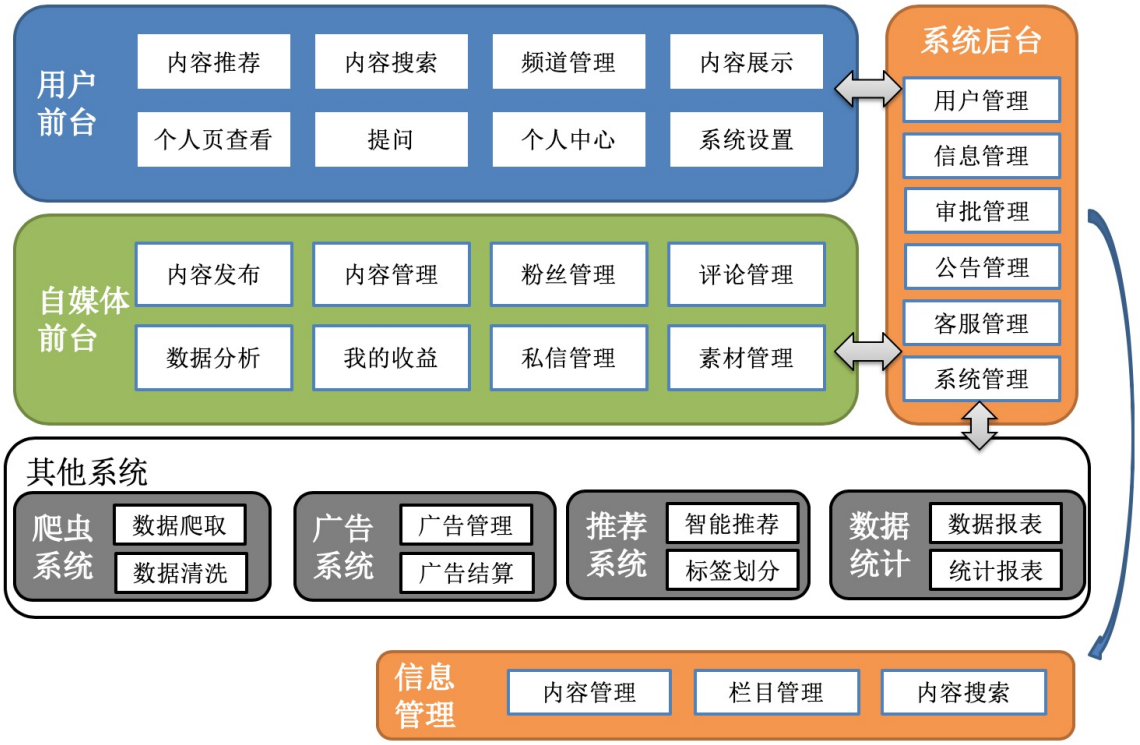
1.3 项目目标

1. 研发并上线运营头条产品
2. 从实际的产品技术中孵化产品经理、Python 人工智能、Python 数据分析、Python Web、测试、运维等课程案例
3. 构建公司自己的数据仓库和算法模型

---

2. 产品功能架构

> 自媒体：又称“个人媒体”，是指私人化、平民化、自主化的传播者， 以现代化、电子化的手段，向不特定的大多数或者特定的单个人传递信息的新媒体的总称。 自媒体平台包括：博客、微博、微信、抖音、百度贴吧、论坛/BBS等网络社区。



产品主要分为三个前端子产品：

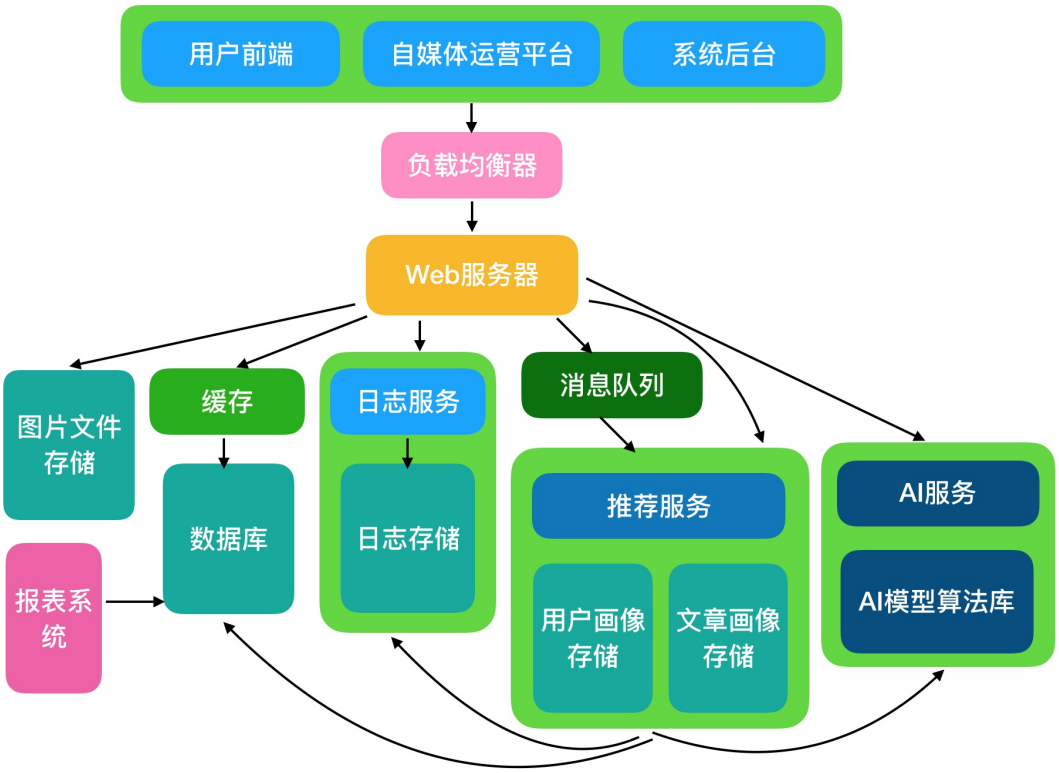
1. 用户端
    - APP，用户可以查看资讯、文章内容，进行问答讨论交流
2. 自媒体运营平台
    - PC网站，自媒体用户可以管理文章、评论，查看分析粉丝数据
3. 系统后台
    - PC网站，内部运营管理系统

产品后端系统功能可分为以下几个部分：

1. 推荐系统部分
    - 负责为用户个性化推荐资讯和文章
2. 人工智能部分
    - 机器自动审核文章、文章画像提取、机器学习推荐算法等
3. 日志系统部分
    - 收集保存用户行为数据和系统运行状态数据
4. 爬虫部分
    - 爬取网站资讯文章数据，作为产品启动的初期数据来源

---

3. 产品技术架构



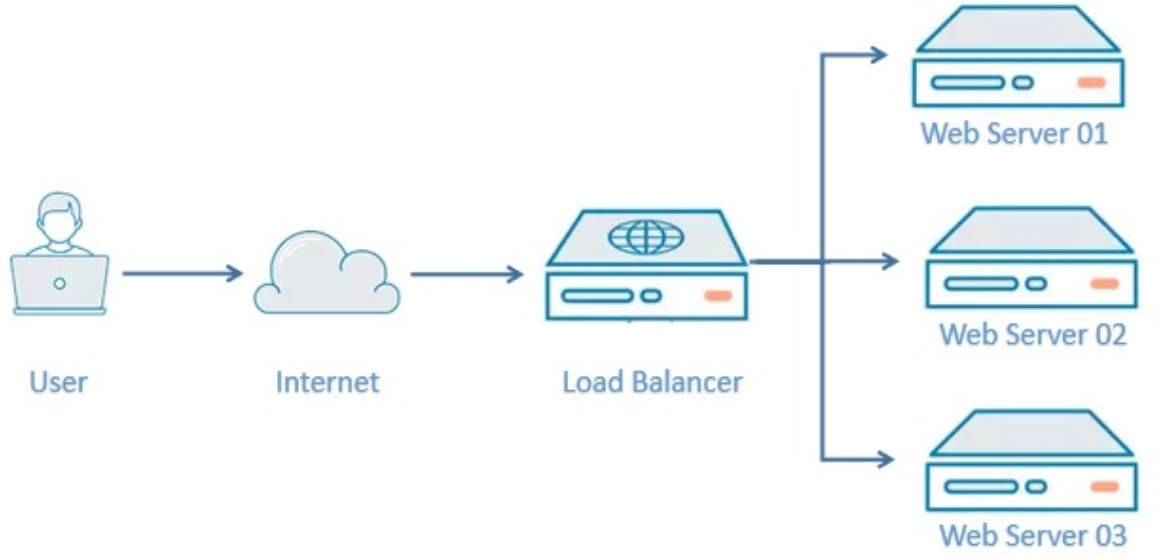
3.1 负载均衡

负载均衡(Load Balance)可以将工作任务分摊到多个处理单元，从而提高并发处理能力。

负载均衡建立在现有网络结构之上，使用它可以实现扩展网络设备的带宽、增加吞吐量、加强网络数据处理能力、提高网络的灵活性和可用性。

不使用负载均衡

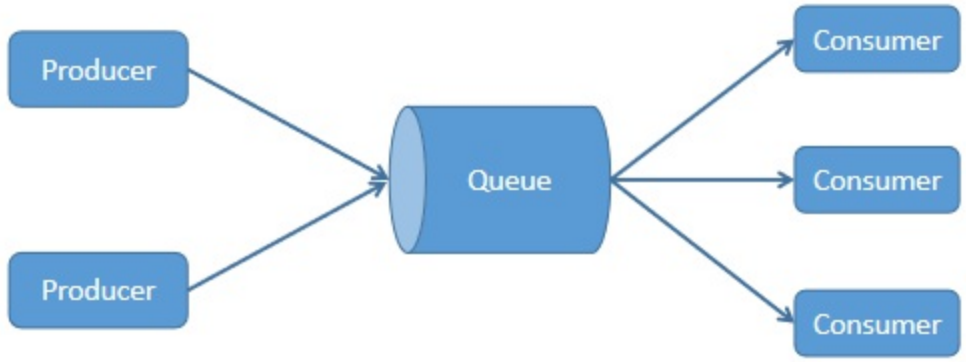
使用负载均衡



3.2 消息队列

消息队列(Message Queue 简称MQ)：是在消息传输过程中保存消息的容器。
消息队列中间件是分布式系统中重要的组件，主要解决应用解耦、异步消息、流量削峰等问题，实现高性能、高可用、可伸缩和最终一致性架构。

目前使用较多的消息队列有：Kafka、ActiveMQ、RabbitMQ、ZeroMQ、MetaMQ、RocketMQ



消息队列应用场景

- 异步处理
  - 将业务逻辑处理由串行方式变成并行方式(好友推荐、新闻推荐)
- 应用解耦
  - 订单系统-->库存系统
  - 发送短信验证码
- 流量削峰
  - 秒杀、抢购活动(一般会因为用户访问量过大，导致流量暴增、应用挂掉)
- 日志处理
  - 将消息队列用在日志处理中，解决大量日志传输的问题

4. 测试功能模块

文章发布流程：

- 新建文章（自媒体端）
- 审核文章（后台管理系统）
- 查看文章（APP端）

用例设计

目标

1. 掌握如何编写自动化测试用例文档

1. 编写自动化测试用例的原则

1. 自动化测试用例一般只实现核心业务流程或者重复执行率较高的功能。
2. 自动化测试用例的选择一般以“正向”逻辑的验证为主。
3. 不是所有手工用例都可以使用自动化测试来执行。
4. 尽量减少多个用例脚本之间的依赖。
5. 自动化测试用例执行完毕之后，一般需要回归原点。

2. 编写测试用例

自媒体

ID	模块	优先级	测试标题	预置条件	步骤描述	测试数据	预期结果	测试结果
001	登录	P0	登录成功	1.打开登录页面	1. 输入用户名 2. 输入验证码 3. 点击登录按钮	1.用户名：13012345678 2.验证码：246810	1.登录成功，页面顶部显示用户名	
002	内容管理	P0	发布文章	1.用户成功登录 2.进入后台管理页面	1.点击‘发布文章’菜单 2.输入标题 3.输入内容 4.选择封面 5.选择频道 6.点击发表按钮	1.标题：test-20190101-001 2.内容：xxx 3.封面：自动 4.频道：数据库	1.提示：文章添加成功	

后台管理系统

ID	模块	优先级	测试标题	预置条件	步骤描述	测试数据	预期结果	测试结果
001	登录	P0	登录成功	1.打开登录页面	1. 输入用户名 2. 输入密码 3. 点击登录按钮	1.用户名：testid 2.密码：testpwd123	1.登录成功，跳转到后台管理页面	
002	信息管理	P0	审核文章	1.用户成功登录 2.进入后台管理页面	1.点击‘信息管理’->‘内容审核’菜单 2.输入搜索的文章名称 3.选择文章状态 4.点击查询按钮 5.点击通过按钮 6.点击提示框的确定按钮	1.文章名称：test-20190101-001 2.选择状态：待审核	1.提示：文章审核成功	

APP

ID	模块	优先级	测试标题	预置条件	步骤描述	测试数据	预期结果	测试结果
001	文章	P0	查找文章	1.用户成功登录 2.进入首页	1. 切换到指定的频道下 2. 滑动查找指定文章	1.频道：数据库 2.标题：test-20190101-001	1.查找到指定文章	
002	频道管理	P0	清空频道	1.用户成功登录 2.进入首页	1.点击选项按钮 2.点击编辑 3.删除所有频道 4.点击完成		1.我的频道数据为空	
003	频道管理	P0	添加频道	1.用户成功登录 2.进入首页	1.点击选项按钮 2.点击要添加的频道	1.添加的频道：python	1.我的频道中存在已添加的频道	
004	频道管理	P0	检查频道	1.用户成功登录 2.进入首页	1.点击选项按钮 2.获取我的频道数据 3.返回首页校验频道数据		1.我的频道中的数据和首页显示的数据一致	

项目搭建

目标

```
1. 掌握如何进行自动化测试框架的搭建
```

---

1. 初始化项目

1.1 新建项目

项目名称：uiAutoTestHmtt

1.2 创建目录结构

```
uiAutoTestHmtt     # 项目名称
├── base           # 封装PO基类
├── page           # 封装PO页面对象
├── script         # 定义测试用例脚本
├── data           # 存放测试数据
├── report         # 存放生成的测试报告
├── log            # 存放日志文件
├── screenshot     # 存放截图
├── config.py      # 定义项目的配置信息
├── utils.py       # 定义工具类
└── pytest.ini     # pytest配置文件
```

1.3 安装依赖包

- 安装 `selenium` 包
- 安装 `Appium-Python-Client` 包
- 安装 `pytest` 包
- 安装 `pytest-ordering` 包
- 安装 `allure-pytest` 包

---

2. 初始化代码

- 封装驱动工具类
- 封装PO基类，定义 `BasePage` 和 `BaseHandle`
- pytest.ini

```
[pytest]
addopts = -s
python_files = test*.py
python_classes = Test*
python_functions = test_*
```

```
testpaths = ./script
```

编写代码

目标

1. 掌握如何采用PO模式的分层思想对页面进行封装
2. 掌握如何使用pytest管理项目中的测试用例

---

1. 抽取PO

根据用例分析待测功能，提取页面对象

1.1 定义页面对象文件

- 自媒体
  - 登录页：login_page.py
  - 后台主页：home_page.py
  - 发布文章页：publish_page.py
- 后台管理页面
  - 登录页：login_page.py
  - 后台主页：home_page.py
  - 文章审核：article_audit_page.py
- APP
  - 首页：index_page.py
  - 频道管理页：channel_page.py

1.2 PO分层封装

分别编写对象库层、操作层、业务层的代码

---

2. 编写测试脚本

2.1 定义测试脚本文件

- 自媒体
  - 登录模块：test_login.py
  - 发布文章模块：test_publish.py
- 后台管理页面
  - 登录模块：test_login.py
  - 文章审核模块：test_article_audit.py
- APP
  - 文章模块：test_article.py
  - 频道管理模块：test_channel.py

2.2 编写测试脚本

使用pytest管理测试脚本

---

3. 执行测试脚本

1. 使用pytest执行测试脚本
2. 调试代码

完善代码

目标

1. 掌握如何把数据驱动应用到项目中
2. 能够把日志收集功能应用到项目中
3. 掌握如何使用pytest生成测试报告

---

1. 数据驱动

1.1 定义数据文件

1. 定义存放测试数据的目录，目录名称：data
2. 分模块定义数据文件
    - 自媒体
        - 登录模块：login.json
        - 发布文章模块：publish.json
    - 后台管理页面
        - 登录模块：login.json
        - 文章审核模块：article_audit.json
    - APP
        - 文章模块：article.json
        - 频道管理模块：channel.json
3. 根据业务编写用例数据

1.2 测试数据参数化

修改测试脚本，使用`@pytest.mark.parametrize`实现参数化

---

2. 日志收集

使用logging模块实现日志的收集

示例代码

```
import logging.handlers
import os

# 工程目录
BASE_DIR = os.path.dirname(os.path.abspath(__file__))


def init_log_config():
    """
    初始化日志配置
```

```
    """
    logger = logging.getLogger()
    logger.setLevel(logging.INFO)

    # 日志输出格式
    fmt = "%(asctime)s %(levelname)s [%(filename)s(%(funcName)s:%(lineno)d)] - %(message)s"
    formatter = logging.Formatter(fmt)

    # 输出到控制台
    sh = logging.StreamHandler()
    sh.setFormatter(formatter)
    logger.addHandler(sh)

    # 输出到文件，每日一个文件
    log_path = os.path.join(BASE_DIR, "log", "hmtt.log")
    fh = logging.handlers.TimedRotatingFileHandler(log_path, when='MIDNIGHT', interval=1,
                                                   backupCount=3, encoding="UTF-8")
    fh.setFormatter(formatter)
    logger.addHandler(fh)
```

Allure测试报告

目标

```
1. 能够将项目生成 allure 报告
2. 能够在 allure 报告上添加测试步骤
3. 能够在 allure 报告上添加测试描述
4. 能够在 allure 报告上添加严重级别
```

1. Allure 的简介和使用

应用场景

我们自动化的结果一定是通过一个报告来进行体现。Allure 是一个独立的报告插件，生成美观易读的报告，目前支持 Java、PHP、Ruby、Python、Scala、C# 这些语言。生成的报告无论是帮我们定位问题，还是发送给领导看，都能快速上手。

帮助文档

> https://docs.qameta.io/allure/

步骤概述

最终我们会生成一个 html 格式的报告，中间我们需要操作两步来进行。

1. 生成测试结果文件(json文件)
2. 将测试结果文件转成 html

1.1 生成测试结果文件

安装

```
pip install allure-pytest
```

使用步骤

1. 将 pytest 配置文件中的命令行参数加上如下代码

   ```
   --alluredir report
   ```

2. 编写好测试脚本后，正常的在命令行中运行 pytest 即可

   ```
   [pytest]
   # 添加行参数
   addopts = -s --alluredir report
   # 文件搜索路径
   testpaths = ./scripts
   # 文件名称
   python_files = test_*.py
   # 类名称
   python_classes = Test*
   ```

```
# 方法名称
python_functions = test_*
```

3. 程序运行结束后，会在项目的report目录中生成一些json文件

1.2 将测试结果文件转成 **html**

安装

1. https://bintray.com/qameta/generic/allure2 下载 allure-2.6.0.zip
2. 解压缩到一个目录（不经常动的目录）
3. 将压缩包内的 bin 目录配置到 path 系统环境变量
4. 右键我的电脑 - 属性 - 高级设置 - 环境变量 - 找到系统环境变量的path项 - 增加 allure到bin 目录
5. 在命令行中敲 allure 命令，如果提示有这个命令，即为成功

使用步骤

在保证项目中的 report 目录下有 json 文件的时候，执行以下步骤。

1. 进入 report 上级目录执行命令

```
allure generate report/ -o report/html --clean
```

2. report 目录下会生成 html 文件夹，html 下会有一个 index.html ，右键用浏览器打开即可。

1.3 参数和命令详解

应用场景

修改测试结果文件所在的目录名称和 index.html 所在的目录名称。

疑问和解答

1. `addopts = -s --alluredir report` 中的 `--alluredir report` 是什么意思？
   - --alluredir 后面的 report 为测试结果文件输出的目录名
   - 如果希望目录名叫 result 那么可以将命令行参数改为 `--alluredir result`
2. `allure generate report/ -o report/html --clean` 是什么意思？

- report/ 表示测试结果文件所在的目录
- -o 表示 output 输出
- report/html 表示将 index.html 报告生成到哪个文件夹

---

2. Allure 与 pytest 结合

2.1 添加测试步骤

应用场景

一套登录流程需要至少三个步骤，输入用户名，输入密码，点击登录。我们可以通过添加测试步骤，让这些步骤在报告中进行体现

使用方式

例如：在操作层中的方法上加上 `@allure.step(title="测试步骤001")` 装饰器

核心代码

```
# page/login_page.py
class LoginHandle(BaseHandle):
    def __init__(self):
        self.login_page = LoginPage()

    @allure.step(title="输入手机号")
    def input_mobile(self, mobile):
        self.input_text(self.login_page.find_mobile(), mobile)

    @allure.step(title="输入验证码")
    def input_code(self, code):
        self.input_text(self.login_page.find_code(), code)

    @allure.step(title="点击登录按钮")
    def click_login_btn(self):
        self.login_page.find_login_btn().click()
```

结果

2.2 添加图片描述

应用场景

如果我们想将某些操作过后的结果展现在报告上，可以使用添加图片描述的方法。

使用方式 在需要截图的地方添加如下代码：

```
allure.attach(driver.get_screenshot_as_png(), "截图", allure.attachment_type.PNG)
```

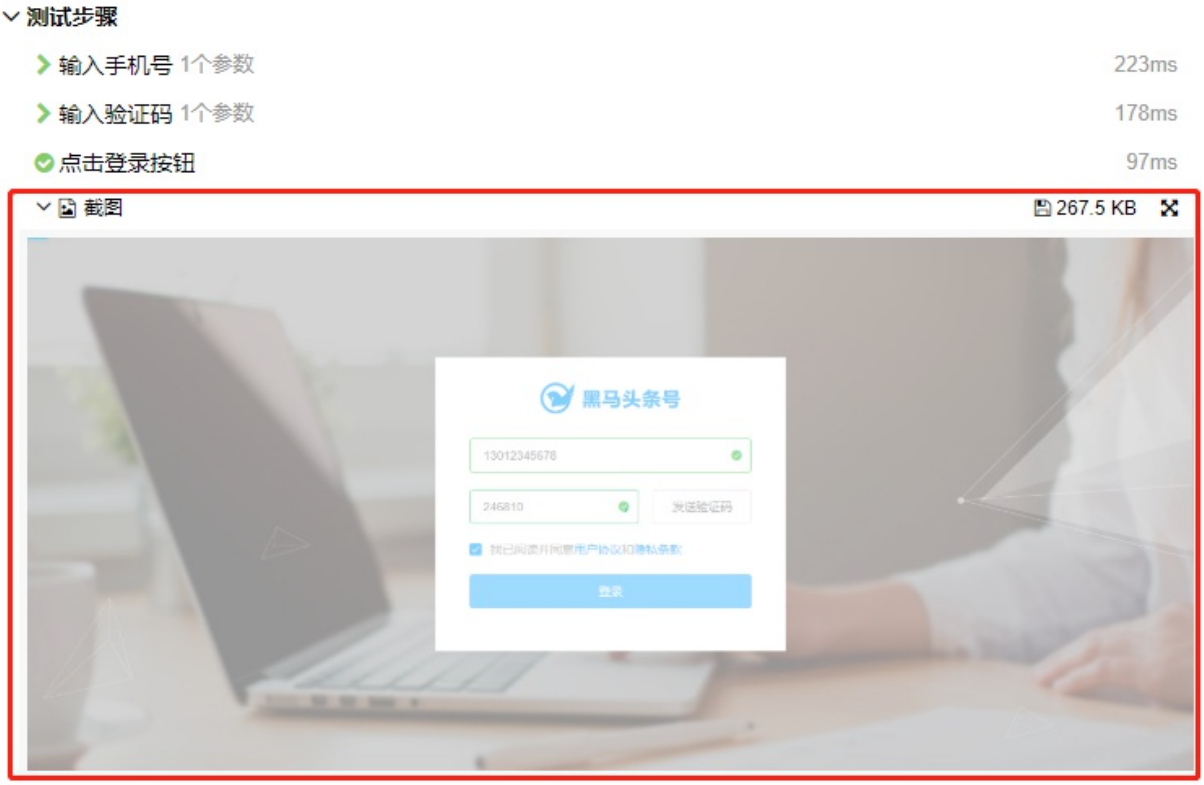
核心代码

```
# script/test_login.py
@pytest.mark.parametrize("mobile,code,username", build_data())
def test_login(self, mobile, code, username):
    logging.info("mobile={} code={} username={}".format(mobile, code, username))
    # 登录
    self.login_proxy.login(mobile, code)

    # 截图
    allure.attach(self.driver.get_screenshot_as_png(), "截图", allure.attachment_type.PNG)

    # 断言
    is_exist = utils.exist_text(DriverUtil.get_mp_driver(), username)
    assert is_exist
```

结果



2.3 添加严重级别

应用场景

在工作中，我们会向开发人员提交很多 bug ，不同的 bug 优先级也应当不同，打开程序就崩溃，和关于软件的页面错了一个字。就这两者而言，崩溃肯定更严重，也更需要开发人员优先修复。那么，我可以将这些 bug 的优先级展示在报告当中。

使用方式

在测试脚本中，增加装饰器 `@allure.severity(allure.severity_level.BLOCKER)`

参数有五个，也对应不同的优先级，只需要将最后一个词替换即可。

1. BLOCKER 最严重
2. CRITICAL 严重
3. NORMAL 普通
4. MINOR 不严重
5. TRIVIAL 最不严重

示例

```
# script/test_login.py
@allure.severity(allure.severity_level.BLOCKER)
@pytest.mark.parametrize("mobile,code,username", build_data())
def test_login(self, mobile, code, username):
    logging.info("mobile={} code={} username={}".format(mobile, code, username))
    # 登录
    self.login_proxy.login(mobile, code)

    # 截图
    allure.attach(self.driver.get_screenshot_as_png(), "截图", allure.attachment_type.PNG)
```

```
    # 断言
    is_exist = utils.exist_text(DriverUtil.get_mp_driver(), username)
    assert is_exist
```

结果

3. 在项目中生成测试报告

使用步骤：

1. 将 pytest 配置文件中的命令行参数加上 `--alluredir=report`
2. 执行测试用例
3. 生成HTML测试报告， `allure generate report/ -o report-html/ --clean`

Jenkins集成

目标

1. 掌握如何在Jenkins中配置并运行测试脚本

---

1. 提交代码至代码服务器

代码管理的常用操作：

- 把代码提交到代码服务器
- 从代码服务器上拉取代码
- 合并代码
- 解决冲突

2. 配置Jenkins

操作步骤：

1. 新建任务
2. 源码管理
3. 构建触发器
4. 添加构建
5. 添加构建后操作-生成测试报告
6. 添加构建后操作-发送邮件