Table of Contents

Web自动化测试课程

序号	章节	知识点
1	第一章 Web自动化入门	1. 认识自动化及自动化测试 2. 自动化测试工具选择 3. 环境搭建
2	第二章 Selenium-API操作	1. 元素定位方式 2. 元素和浏览器的操作方法 3. 鼠标和键盘操作 4. 元素等待 5. HTML特殊元素处理 6. 窗口截图 7. 验证码处理
3	第三章 UnitTest框架	1. UnitTest基本使用 2. UnitTest断言 3. 参数化 4. 生成HTML测试报告
4	第四章 PO模式	1. 方法封装 2. PO模式介绍 3. PO模式实战
5	第五章 数据驱动	1. JSON读写 2. 数据驱动介绍 3. 数据驱动实战
6	第六章 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
7	第七章 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进行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. 掌握如何通过UnitTest管理用例脚本，并生成HTML测试报告。
3. 掌握使用PO模式来设计自动化测试代码的架构。
4. 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。
5. 掌握使用logging来实现日志的收集。

第4章-PO模式

目标

1. 深入理解方法封装的思想
2. 能够使用方法封装的思想对代码进行优化
3. 深入理解PO模式的思想
4. 熟练掌握PO模式的分层思想

PO模式学习思路

采用版本迭代的方式来学习，便于对不同版本的优缺点进行对比和理解。

- V1：不使用任何设计模式和单元测试框架
- V2：使用UnitTest管理用例
- V3：使用方法封装的思想，对代码进行优化
- V4：采用PO模式的分层思想对代码进行拆分
- V5：对PO分层之后的代码继续优化
- V6：PO模式深入封装，把共同操作提取封装到父类中，子类直接调用父类的方法

无模式

目标

1. 熟悉web自动化测试代码编写的基本流程
2. 掌握如何使用UnitTest管理测试脚本

---

1. 案例说明

对TPshop项目的登录模块进行自动化测试。

> 提示：登录模块包含了很多测试用例，比如：账号不存在、密码错误、验证码错误、登录成功等等。
> 为了节省时间我们只选取几个有代表性的用例来演示。

1.1 选择的测试用例

- 账号不存在
  1. 点击首页的‘登录’链接，进入登录页面
  2. 输入一个不存在的用户名
  3. 输入密码
  4. 输入验证码
  5. 点击登录按钮
  6. 获取错误提示信息
- 密码错误
  1. 点击首页的‘登录’链接，进入登录页面
  2. 输入用户名
  3. 输入一个错误的密码
  4. 输入验证码
  5. 点击登录按钮
  6. 获取错误提示信息

---

2. V1版本

不使用任何设计模式和单元测试框架。
每个文件里编写一个用例，完全的面向过程的编程方式。

2.1 存在的问题

- 一条测试用例对应一个文件，用例较多时不方便管理维护
- 代码高度冗余

2.2 示例代码

登录功能-账号不存在

```
from selenium import webdriver

# 创建浏览器驱动对象，并完成初始化操作
driver = webdriver.Firefox()
driver.maximize_window()
driver.implicitly_wait(10)
driver.get("http://localhost")

"""
登录功能-账号不存在
"""

# 点击首页的‘登录’链接，进入登录页面
driver.find_element_by_link_text("登录").click()

# 输入用户名
driver.find_element_by_id("username").send_keys("13099999999")

# 输入密码
driver.find_element_by_id("password").send_keys("123456")

# 输入验证码
driver.find_element_by_id("verify_code").send_keys("8888")

# 点击‘登录’按钮
driver.find_element_by_name("sbtbutton").click()

# 获取提示信息
msg = driver.find_element_by_class_name("layui-layer-content").text
print("msg=", msg)

# 关闭驱动对象
driver.quit()
```

登录功能-密码错误

```
from selenium import webdriver
```

```
# 创建浏览器驱动对象，并完成初始化操作
driver = webdriver.Firefox()
driver.maximize_window()
driver.implicitly_wait(10)
driver.get("http://localhost")

"""
登录功能-密码错误
"""

# 点击首页的‘登录’链接，进入登录页面
driver.find_element_by_link_text("登录").click()

# 输入用户名
driver.find_element_by_id("username").send_keys("13012345678")

# 输入密码
driver.find_element_by_id("password").send_keys("error")

# 输入验证码
driver.find_element_by_id("verify_code").send_keys("8888")

# 点击‘登录’按钮
driver.find_element_by_name("sbtbutton").click()

# 获取提示信息
msg = driver.find_element_by_class_name("layui-layer-content").text
print("msg=", msg)

# 关闭驱动对象
driver.quit()
```

3. V2版本

使用UnitTest管理用例，并断言用例的执行结果

3.1 引入UnitTest的好处

- 方便组织、管理多个测试用例
- 提供了丰富的断言方法
- 方便生成测试报告
- 减少了代码冗余

3.2 存在的问题

- 代码冗余

3.3 示例代码

```
import unittest
from selenium import webdriver

class TestLogin(unittest.TestCase):
    """
    对登录模块的功能进行测试
    """

    @classmethod
    def setUpClass(cls):
        cls.driver = webdriver.Firefox()
        cls.driver.maximize_window()
        cls.driver.implicitly_wait(10)
        cls.driver.get("http://localhost")

    @classmethod
    def tearDownClass(cls):
        cls.driver.quit()

    def setUp(self):
        # 打开首页
        self.driver.get("http://localhost")

        # 点击首页的‘登录’链接，进入登录页面
        self.driver.find_element_by_link_text("登录").click()

    # 账号不存在
    def test_login_username_is_error(self):
        # 输入用户名
        self.driver.find_element_by_id("username").send_keys("13099999999")

        # 输入密码
        self.driver.find_element_by_id("password").send_keys("123456")

        # 输入验证码
        self.driver.find_element_by_id("verify_code").send_keys("8888")

        # 点击‘登录’
        self.driver.find_element_by_name("sbtbutton").click()

        # 断言提示信息
        msg = self.driver.find_element_by_class_name("layui-layer-content").text
        print("msg=", msg)
        self.assertIn("账号不存在", msg)

    # 密码错误
```

```
    def test_login_password_is_error(self):
        # 输入用户名
        self.driver.find_element_by_id("username").send_keys("13012345678")

        # 输入密码
        self.driver.find_element_by_id("password").send_keys("error")

        # 输入验证码
        self.driver.find_element_by_id("verify_code").send_keys("8888")

        # 点击‘登录’
        self.driver.find_element_by_name("sbtbutton").click()

        # 断言提示信息
        msg = self.driver.find_element_by_class_name("layui-layer-content").text
        print("msg=", msg)
        self.assertIn("密码错误", msg)
```

方法封装

目标

1. 深入理解方法封装的思想
2. 能够使用方法封装的思想对代码进行优化

---

1. 方法封装

方法封装：是将一些有共性的或多次被使用的代码提取到一个方法中，供其他地方调用。

封装的好处：

- 避免代码冗余
- 容易维护
- 隐藏代码实现的细节

目的：用最少的代码实现最多的功能

---

2. V3版本

使用方法封装的思想，对代码进行优化。

- 定义获取驱动对象的工具类
- 封装“获取弹出框的提示消息”

2.1 定义获取驱动对象的工具类

对登录流程的代码进行优化，定义获取驱动对象的工具类

```
# utils.py

class DriverUtil:
    """
    浏览器驱动工具类
    """

    _driver = None

    @classmethod
```

```
    def get_driver(cls):
        """
        获取浏览器驱动对象，并完成初始化设置
        :return: 浏览器驱动对象
        """
        if cls._driver is None:
            cls._driver = webdriver.Firefox()
            cls._driver.maximize_window()
            cls._driver.implicitly_wait(10)
            cls._driver.get("http://localhost")
        return cls._driver

    @classmethod
    def quit_driver(cls):
        """
        关闭浏览器驱动
        """
        if cls._driver:
            cls._driver.quit()
            cls._driver = None
```

2.2 封装“获取弹出框的提示消息”

对登录流程的代码进行优化，封装‘获取弹出框的提示消息’的方法

```
# utils.py

def get_tips_msg():
    """
    获取弹出框的提示消息
    :return: 消息文本内容
    """
    msg = DriverUtil.get_driver().find_element_by_class_name("layui-layer-content").tex
t
    return msg
```

PO模式介绍

目标

1. 深入理解PO模式的思想
2. 熟练掌握PO模式的分层思想

---

1. 存在的问题

在做UI自动化时定位元素特别依赖页面，一旦页面发生变更就不得不跟着去修改定位元素的代码。

举例：假设要对一个元素进行点击操作，而且会经常对该元素进行操作，那么你就可能会编写多处如下代码

```
driver.find_element_by_id("login-btn").click()
```

存在的问题：

- 如果开发人员修改了这个元素的id，这时候你就不得不修改所有对应的代码
- 存在大量冗余代码

> 思考：如何来解决这个问题呢？

2. PO模式

PO是**Page Object**的缩写，PO模式是自动化测试项目开发实践的最佳设计模式之一。
核心思想是通过对界面元素的封装减少冗余代码，同时在后期维护中，若元素定位发生变化， 只需要调整页面元素封装的代码，提高测试用例的可维护性、可读性。

PO模式可以把一个页面分为三层，对象库层、操作层、业务层。

- 对象库层：封装定位元素的方法。
- 操作层：封装对元素的操作。
- 业务层：将一个或多个操作组合起来完成一个业务功能。比如登录：需要输入帐号、密码、点击登录三个操作。

2.1 引入PO模式的好处

引入PO模式前

- 存在大量冗余代码

- 业务流程不清晰
- 后期维护成本大

引入PO模式后

- 减少冗余代码
- 业务代码和测试代码被分开，降低耦合性
- 维护成本低

PO模式实践

目标

1. 能够采用PO模式的分层思想对页面进行封装

---

1. V4版本

采用PO模式的分层思想对代码进行拆分

1.1 PO分层封装

对登录页面进行分层封装：

- 对象库层：LoginPage
- 操作层：LoginHandle
- 业务层：LoginProxy

调用业务层的方法，编写测试用例：

- 测试用例：TestLogin

1.2 示例代码

```
from po.utils import DriverUtil


class LoginPage:
    """
    对象库层
    """

    def __init__(self):
        self.driver = DriverUtil.get_driver()

        # 用户名输入框
        self.username = None
        # 密码
        self.password = None
        # 验证码输入框
        self.verify_code = None
```

```
        # 登录按钮
        self.login_btn = None
        # 忘记密码
        self.forget_pwd = None

    def find_username(self):
        return self.driver.find_element_by_id("username")

    def find_password(self):
        return self.driver.find_element_by_id("password")

    def find_verify_code(self):
        return self.driver.find_element_by_id("verify_code")

    def find_login_btn(self):
        return self.driver.find_element_by_name("sbtbutton")

    def find_forget_pwd(self):
        return self.driver.find_element_by_partial_link_text("忘记密码")


class LoginHandle:
    """
    操作层
    """

    def __init__(self):
        self.login_page = LoginPage()

    def input_username(self, username):
        self.login_page.find_username().send_keys(username)

    def input_password(self, pwd):
        self.login_page.find_password().send_keys(pwd)

    def input_verify_code(self, code):
        self.login_page.find_verify_code().send_keys(code)

    def click_login_btn(self):
        self.login_page.find_login_btn().click()

    def click_forget_pwd(self):
        self.login_page.find_forget_pwd().click()


class LoginProxy:
    """
    业务层
    """
```

```
    def __init__(self):
        self.login_handle = LoginHandle()

    # 登录
    def login(self, username, password, verify_code):
        # 输入用户名
        self.login_handle.input_username(username)
        # 输入密码
        self.login_handle.input_password(password)
        # 输入验证码
        self.login_handle.input_verify_code(verify_code)
        # 点击登录按钮
        self.login_handle.click_login_btn()

    # 跳转到忘记密码页面
    def to_forget_pwd_page(self):
        # 点击忘记密码
        self.login_handle.click_forget_pwd()
```

```
import unittest

from po import utils
from po.utils import DriverUtil
from po.v4.page.login_page import LoginProxy


class TestLogin(unittest.TestCase):
    """
    对登录模块的功能进行测试
    """

    @classmethod
    def setUpClass(cls):
        cls.driver = DriverUtil.get_driver()
        cls.login_proxy = LoginProxy()

    @classmethod
    def tearDownClass(cls):
        DriverUtil.quit_driver()

    def setUp(self):
        # 打开首页
        self.driver.get("http://localhost")

        # 点击首页的‘登录’链接，进入登录页面
        self.driver.find_element_by_link_text("登录").click()
```

```
    # 账号不存在
    def test_login_username_is_error(self):
        self.login_proxy.login("13099999999", "123456", "8888")

        # 断言提示信息
        msg = utils.get_tips_msg()
        print("msg=", msg)
        self.assertIn("账号不存在", msg)

    # 密码错误
    def test_login_password_is_error(self):
        self.login_proxy.login("13012345678", "123456", "8888")

        # 断言提示信息
        msg = utils.get_tips_msg()
        print("msg=", msg)
        self.assertIn("密码错误", msg)
```

2. V5版本

对PO分层之后的代码继续优化

1. 优化对象库层的代码，抽取元素的定位方式，把定位信息定义在对象的属性中，便于集中管理
2. 优化操作层的代码，针对输入操作应该先清空输入框中的内容再输入新的内容

```
from selenium.webdriver.common.by import By

from po.utils import DriverUtil


class LoginPage:
    """
    对象库层
    """

    def __init__(self):
        self.driver = DriverUtil.get_driver()

        # 用户名
        self.username = (By.ID, "username")
        # 密码
        self.password = (By.ID, "password")
        # 验证码输入框
        self.verify_code = (By.ID, "verify_code")
        # 登录按钮
        self.login_btn = (By.NAME, "sbtbutton")
```

```
        # 忘记密码
        self.forget_pwd = (By.PARTIAL_LINK_TEXT, "忘记密码")

    def find_username(self):
        return self.driver.find_element(self.username[0], self.username[1])

    def find_password(self):
        return self.driver.find_element(self.password[0], self.password[1])

    def find_verify_code(self):
        return self.driver.find_element(self.verify_code[0], self.verify_code[1])

    def find_login_btn(self):
        return self.driver.find_element(self.login_btn[0], self.login_btn[1])

    def find_forget_pwd(self):
        return self.driver.find_element(self.forget_pwd[0], self.forget_pwd[1])


class LoginHandle:
    """
    操作层
    """

    def __init__(self):
        self.login_page = LoginPage()

    def input_username(self, username):
        self.login_page.find_username().clear()
        self.login_page.find_username().send_keys(username)

    def input_password(self, pwd):
        self.login_page.find_password().clear()
        self.login_page.find_password().send_keys(pwd)

    def input_verify_code(self, code):
        self.login_page.find_verify_code().clear()
        self.login_page.find_verify_code().send_keys(code)

    def click_login_btn(self):
        self.login_page.find_login_btn().click()

    def click_forget_pwd(self):
        self.login_page.find_forget_pwd().click()


class LoginProxy:
    """
    业务层
```

```
    """

    def __init__(self):
        self.login_handle = LoginHandle()

    # 登录
    def login(self, username, password, verify_code):
        # 输入用户名
        self.login_handle.input_username(username)
        # 输入密码
        self.login_handle.input_password(password)
        # 输入验证码
        self.login_handle.input_verify_code(verify_code)
        # 点击登录按钮
        self.login_handle.click_login_btn()

    # 跳转到忘记密码页面
    def to_forget_pwd_page(self):
        # 点击忘记密码
        self.login_handle.click_forget_pwd()
```

PO模式深入封装

目标

1. 能够采用继承的思想对PO模式进行深入的封装

---

1. V6版本

把共同操作提取封装到父类中，子类直接调用父类的方法，避免代码冗余

1. 对象库层-基类，把定位元素的方法定义在基类中
2. 操作层-基类，把对元素执行输入操作的方法定义在基类中

1.1 示例代码

```
# base_page.py

from po.utils import DriverUtil

class BasePage:
    """
    基类-对象库层
    """

    def __init__(self):
        self.driver = DriverUtil.get_driver()

    def find_element(self, location):
        return self.driver.find_element(location[0], location[1])


class BaseHandle:
    """
    基类-操作层
    """

    def input_text(self, element, text):
        """
        在输入框里输入文本内容，先清空再输入
        :param element: 要操作的元素
        :param text: 要输入的文本内容
        """
```

```
        element.clear()
        element.send_keys(text)
```

```
from selenium.webdriver.common.by import By

from po.v6.common.base_page import BasePage, BaseHandle


class LoginPage(BasePage):
    """
    对象库层
    """

    def __init__(self):
        super().__init__()

        # 用户名输入框
        self.username = (By.ID, "username")
        # 密码
        self.password = (By.ID, "password")
        # 验证码
        self.verify_code = (By.ID, "verify_code")
        # 登录按钮
        self.login_btn = (By.NAME, "sbtbutton")
        # 忘记密码
        self.forget_pwd = (By.PARTIAL_LINK_TEXT, "忘记密码")

    def find_username(self):
        return self.find_element(self.username)

    def find_password(self):
        return self.find_element(self.password)

    def find_verify_code(self):
        return self.find_element(self.verify_code)

    def find_login_btn(self):
        return self.find_element(self.login_btn)

    def find_forget_pwd(self):
        return self.find_element(self.forget_pwd)


class LoginHandle(BaseHandle):
    """
    操作层
    """
```

```
    def __init__(self):
        self.login_page = LoginPage()

    def input_username(self, username):
        self.input_text(self.login_page.find_username(), username)

    def input_password(self, pwd):
        self.input_text(self.login_page.find_password(), pwd)

    def input_verify_code(self, code):
        self.input_text(self.login_page.find_verify_code(), code)

    def click_login_btn(self):
        self.login_page.find_login_btn().click()

    def click_forget_pwd(self):
        self.login_page.find_forget_pwd().click()


class LoginProxy:
    """
    业务层
    """

    def __init__(self):
        self.login_handle = LoginHandle()

    # 登录
    def login(self, username, password, verify_code):
        # 输入用户名
        self.login_handle.input_username(username)
        # 输入密码
        self.login_handle.input_password(password)
        # 输入验证码
        self.login_handle.input_verify_code(verify_code)
        # 点击登录按钮
        self.login_handle.click_login_btn()

    # 跳转到忘记密码页面
    def to_forget_pwd_page(self):
        # 点击忘记密码
        self.login_handle.click_forget_pwd()
```