09 页面对象（Page Object）模式

本章大纲

9.1 页面对象模式简介

9.2 Paget Object三种实现

页面对象模式简介

- 使用面向对象的设计模式，页面对象模式将测试代码和被测试页面的页面元素及其操作方法进行分离，以此降低页面元素变化对测试代码的影响。
- 每一个测试页面都会被单独定义一个类，类中会定位所有需要参与测试的页面元素对象，并且定义操作每一个页面元素对象的方法。

Page Object 设计模式的优点

- 减少代码的重复
- 提高测试用例的可读性
- 提高测试用例的可维护性

Paget Object实例

- 以百度搜索为例：

声明一个名为SearchPage的类，并且通过定位表达式找到搜索框和搜索按钮“百度一下”，分别定义输入的方法和单击的方法

创建两个类：页面对象类，测试类

PageFactory类封装页面的元素

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.*;

public class SearchPage {

    @FindBy(id="kw")
    public WebElement inputSearch;

    @FindBy(id="su")
    public WebElement btnSearch;

    public SearchPage(WebDriver wd){
        PageFactory.initElements(wd, this);
    }
}
```

PageFactory类封装页面的元素

```
public class SearchPageTest {
    WebDriver driver = null;

    @Test
    public void testBaiduSearch() throws InterruptedException{
        SearchPage loginpage = new SearchPage(driver);
        loginpage.inputSearch.sendKeys("taobao");
        loginpage.btnSearch.click();
        Thread.sleep(5000);
        assertTrue(driver.getPageSource().contains("购物"));
    }
```

使用注解获取页面元素

```
@FindBy(id= "A")
private WebElement A;

@Findbys({
@FindBy(className = "A"),
@FindBy(id = "B") }
)
public WebElement AB;

@FindAll({
@FindBy(id = "A"),
@FindBy(id = "B")
 })
 public List<WebElement> aAndB;
```

PageFactory类封装页面的元素的操作方法

```
public class SearchPage1 {
    WebDriver driver;
    String url="https://www.baidu.com/";
    @FindBy(id="kw")
    public WebElement inputSearch;
    @FindBy(id="su")
    public WebElement btnSearch;
    public SearchPage1(){
        driver = new FirefoxDriver();
        PageFactory.initElements(driver, this);
    }
    public void load(){
        driver.get(url);
    }
    public void quit(){
        driver.quit();
    }
    public void Search(String word){
        this.inputSearch.sendKeys(word);
        this.btnSearch.click();
    }
    public WebDriver getDriver(){
        return driver;
    }
}
```

PageFactory类封装页面的元素的操作方法

```
public class SearchPageTest1 {
    @Test
    public void testBaiduSearch() throws InterruptedException {
        SearchPage1 page = new SearchPage1();
        page.load();
        page.Search("taobao");
        Thread.sleep(5000);
        assertTrue(page.getDriver().getPageSource().contains("购物"));
        page.quit();
    }

}
```

使用LoadableComponent类

- 继承LoadableComponent类可以在页面加载的时候判断是否加载了正确的页面，只需要重写load() 和isLoaded()两个方法。此方式有助于让页面对象的页面访问操作更加健壮。

```java
public class SearchPage2 extends LoadableComponent<SearchPage2> {
    WebDriver driver=null;
    String url = "https://www.baidu.com/";
    String title = "百度一下";
    @FindBy(id = "kw")
    public WebElement inputSearch;
    @FindBy(id = "su")
    public WebElement btnSearch;
    public SearchPage2() {
        driver = new FirefoxDriver();
        PageFactory.initElements(driver, this);
    }
    public void load() {
        driver.get(url);
    }
    @Override
    protected void isLoaded() throws Error {
        assertTrue(driver.getTitle().contains(title));
    }
    public void quit() {
        driver.quit();
    }
    public void Search(String word) {
        this.inputSearch.sendKeys(word);
        this.btnSearch.click();
```

```java
public class SearchPageTest2 {
    @Test
    public void testBaiduSearch() throws InterruptedException {
        SearchPage2 page = new SearchPage2();
        // 继承LoadableComponent类后，只要覆盖了load方法
        // 即使在没有get方法的情况下，也可以进行get方法的调用
        // get方法会默认调用页面对象类中的load
        page.get();
        page.Search("taobao");
        Thread.sleep(5000);
        assertTrue(page.getDriver().getPageSource().contains("购物"));
        page.quit();
    }

}
```

多个Paget Object的自动化测试实例

- 使用正确的用户名错误的密码，判断是否出现“帐号或密码错误”
- 使用正确的用户名正确的密码，判断是否出现“退出”
- 登录后，添加影片信息
- 创建类三个类LoginPage ManageMovie MovieTest