作者：深圳51TESTING

WebDriver 进阶

欢迎阅读WebDriver进阶讲义。本篇讲义将会重点介绍Selenium WebDriver API的重点使用方法，以及使用模块化和参数化进行自动化测试的设计。

WebDriver API 进阶使用

元素定位

从之前的讲义和学习中，我们知道，WebDriver API的调用以及自动化测试，务必从页面元素的定位开始，那么回顾之前的内容，WebDriver提供了一系列的定位符以便使用元素定位方法。常见的定位符有以下几种：

- id
- name
- class name
- tag
- link text
- partial link text
- xpath
- css selector

那么我们以下的操作将会基于上述的定位符进行定位操作。

对于元素的定位，WebDriver API可以通过定位简单的元素和一组元素来操作。在这里，我们需要告诉Selenium如何去找元素，以至于他可以充分的模拟用户行为，或者通过查看元素的属性和状态，以便我们执行一系列的检查。

在Selenium2中，WebDriver提供了多种多样的`findElement(by.)`方法在一个网页里面查找元素。这些方法通过提供过滤标准来定位元素。当然WebDriver也提供了同样多种多样的`findElements(by.)`的方式去定位多个元素。

Selenium2提供的8个`findElement(by.)`方法去定位元素。在这里我们来具体查看每个方法的详细使用方式。下面的表格将会列出这些具体的方法：

方法Method	描述Description	参数Argument	示例Example
`id`	该方法通过ID的属性值去定位查找单个元素	id: 需要被查找的元素的ID	`findElement(by.id("search"))`
`name`	该方法通过name的属性值去定位查找单个元素	name: 需要被查找的元素的名称	`findElement(by.name("q"))`
`class name`	该方法通过class的名称值去定位查找单个元素	class_name: 需要被查找的元素的类名	`findElement(by.className("input-text"))`
`tag_name`	该方法通过tag的名称值去定位查找单个元素	tag: 需要被查找的元素的标签名称	`findElement(by.tagName("input"))`
`link_text`	该方法通过链接文字去定位查找单个元素	link_text: 需要被查找的元素的链接文字	`findElement(by.linkText("Log In"))`
`partial_link_text`	该方法通过部分链接文字去定位查找单个元素	link_text: 需要被查找的元素的部分链接文字	`findElement(by.partialLinkText("Long"))`
`xpath`	该方法通过XPath的值去定位查找单个元素	xpath: 需要被查找的元素的xpath	`findElement(by.xpath("//*[@id='xx']/a"))`
`css_selector`	该方法通过CSS选择器去定位查找单个元素	css_selector: 需要被查找的元素的ID	`findElement(by.cssSelector("#search"))`

接下来的列表将会详细展示`find_elements_by`的方法集合。这些方法依据匹配的具体标准返回一系列的元素。

方法Method	描述Description	参数Argument	示例Example
	该方法通过ID的属性		

`id`	该方法通过ID的属性值去定位查找多个元素	id: 需要被查找的元素的ID	`findElements(by.id("search"))`
`name`	该方法通过name的属性值去定位查找多个元素	name: 需要被查找的元素的名称	`findElements(by.name("q"))`
`class_name`	该方法通过class的名称值去定位查找多个元素	class_name: 需要被查找的元素的类名	`findElements(by.className("input-text"))`
`tag_name`	该方法通过tag的名称值去定位查找多个元素	tag: 需要被查找的元素的标签名称	`findElements(by.tagName("input"))`
`link_text`	该方法通过链接文字去定位查找多个元素	link_text: 需要被查找的元素的链接文字	`findElements(by.linkText("Log In"))`
`partial_link_text`	该方法通过部分链接文字去定位查找多个元素	link_text: 需要被查找的元素的部分链接文字	`findElements(by.partialLinkText("Long"))`
`xpath`	该方法通过XPath的值去定位查找多个元素	xpath: 需要被查找的元素的xpath	`findElements(by.xpath("//div[contains(@class,'list')]"))`
`css_selector`	该方法通过CSS选择器去定位查找多个元素	css_selector: 需要被查找的元素的ID	`findElements(by.cssSelector(".input_class"))`

依据ID查找

请查看如下HTML的代码，以便实现通过ID的属性值去定义一个查找文本框的查找：

```
<input id="search" type="text" name="q" value=""
       class="input-text" maxlength="128" autocomplete="off"/>
```

根据上述代码，这里我们使用`findElement(By.id())`的方法去查找搜索框并且检查它的最大长度`maxlength`属性。我们通过传递ID的属性值作为参数去查找，参考如下的代码示例：

```
void testSearchTextFieldMaxLength(){
    // get the search textbox
    WebElement searchField = driver.findElement(By.id("search"))
    // check maxlength attribute is set to 128
    assertEqual("128", searchField.getAttribute("maxlength"))
}
```

如果使用`findElement(By.id())`方法，将会返回所有的具有相同ID属性值的一系列元素。

依据名称name查找

这里还是根据上述ID查找的HTML代码，使用`findElement(By.id())`的方法进行查找。参考如下的代码示例：

```
// get the search textbox
searchField = self.driver.findElement(By.name("q"))
```

同样，如果使用`findElements(By.name())`方法，将会返回所有的具有相同name属性值的一系列元素。

依据class name查找

除了上述的ID和name的方式查找，我们还可以使用class name的方式进行查找和定位。

> 事实上，通过ID，name或者类名class name查找元素是最提倡推荐的和最快的方式。当然Selenium2 WebDriver也提供了一些其他的方式，在上述三类方式条件不足，查找无效的时候，可以通过这些其他方式来查找。这些方式将会在后续的内容中讲述。

请查看如下的HTML代码，通过改代码进行练习和理解.

```
<button type="submit" title="Search" class="button">
  <span><span>Search</span></span>
</button>
```

根据上述代码，使用`findElement(By.cssName())`方法去定位元素。

```
void testSearchButtonEnabled(){
    // get Search button
    searchButton = self.driver.findElement(By.cssName("button"))
    // check Search button is enabled
    assertTrue(searchButton.isEnabled())
}
```

同样的如果使用`findElements(By.cssName())`方法去定位元素，将会返回所有的具有相同name属性值的一系列元素。

依据标签名tag name查找

利用标签的方法类似于利用类名等方法进行查找。我们可以轻松的查找出一系列的具有相同标签名的元素。例如我们可以通过查找表中的`<tr>`来获取行数。

下面有一个HTML的示例，这里在无序列表中使用了`<img>`标签。

```
<ul class="promos">
    <li>
        <a href="http://demo.magentocommerce.com/home-decor.html">
            <img src="/media/wysiwyg/homepage-three-column-promo-
        01B.png" alt="Physical &amp; Virtual Gift Cards">
        </a>
    </li>
    <li>
        <a href="http://demo.magentocommerce.com/vip.html">
            <img src="/media/wysiwyg/homepage-three-column-promo-
        02.png" alt="Shop Private Sales - Members Only">
        </a>
    </li>
    <li>
        <a href="http://demo.magentocommerce.com/accessories/
        bags-luggage.html">
            <img src="/media/wysiwyg/homepage-three-columnpromo-
        03.png" alt="Travel Gear for Every Occasion">
        </a>
    </li>
</ul>
```

这里面我们使用`findElements(By.tag())`的方式去获取全部的图片，在此之前，我们将会使用`findElement(By.tag())`去获取到指定的`<ul>`。

具体代码如下：

```
void testCountOfPromoBannersImages(){
    // get promo banner list
    bannerList = driver.findElement(By.className("promos"))
    // get images from the banner_list
    banners = bannerList.findElements(By.tagName("img"))
    // check there are 20 tags displayed on the page
    assertEqual(20, banners.length)
}
```

依据链接文字link Text查找

链接文字查找通常比较简单。使用`findElement(By.linkText())`请查看以下示例

```
<a href="#header-account" class="skip-link skip-account">
    <span class="icon"></span>
    <span class="label">Account</span>
</a>
```

测试代码如下：

```
void testMyAccountLinkIsDisplayed(){
    // get the Account link
    accountLink =
    driver.findElement(By.linkText("ACCOUNT"))
    // check My Account link is displayed/visible in
    // the Home page footer
    self.assertTrue(accountLink.isDisplayed)
}
```

依据部分链接文字partial text查找

这里依旧使用上述的列子进行代码编写：

```
void test_account_links(){
    // get the all the links with Account text in it
    accountLinks = self.driver.findElements(By.partialLinkText("ACCOUNT"))
    // check Account and My Account link is
    displayed/visible in the Home page footer
    assertTrue(2, accountLinks.length)
}
```

依据XPath进行查找

XPath是一种在XML文档中搜索和定位节点node的一种查询语言。所有的主流Web浏览器都支持XPath。Selenium2可以用强大的XPath在页面中查找元素。

常用的XPath的方法有`starts-with()`，`contains()`和`ends-with()`等

> 若想要了解更多关于XPath的内容，请查看http://www.w3schools.com/XPath/

如下有一段HTML代码，其中里面的`<img>`没有使用ID，name或者类属性，所以我们无法使用之前的方法。亚这里我们可以通过`<img>`的`alt`属性，定位到指定的tag。

```
<ul class="promos">
    <li>
        <a href="http://demo.magentocommerce.com/home-decor.html">
            <img src="/media/wysiwyg/homepage-three-column-promo-
        01B.png" alt="Physical &amp;amp; Virtual Gift Cards">
        </a>
    </li>
    <li>
        <a href="http://demo.magentocommerce.com/vip.html">
            <img src="/media/wysiwyg/homepage-three-column-promo-
        02.png" alt="Shop Private Sales - Members Only">
        </a>
    </li>
    <li>
        <a href="http://demo.magentocommerce.com/accessories/
        bags-luggage.html">
            <img src="/media/wysiwyg/homepage-three-columnpromo-
        03.png" alt="Travel Gear for Every Occasion">
        </a>
    </li>
</ul>
```

具体代码如下：

```
void testVipPromo(){
    // get vip promo image
    vipPromo = driver.findElement(By.xpath("//img[@alt='Shop Private Sales - Members Only']"))
    // check vip promo logo is displayed on home page
    assertTrue(vipPromo.isDisplayed)
    // click on vip promo images to open the page
    vipPromo.click()
    // check page title
    assertEqual("VIP", driver.title)
}
```

当然，如果使用`find_elements_by_xpath()`的方法，将会返回所有匹配了XPath查询的元素。

依据CSS选择器进行查找

CSS是一种设计师用来描绘HTML文档的视觉的层叠样式表。一般来说CSS用来定位多种多样的风格，同时可以用来是同样的标签使用同样的风格等。类似于XPath，Selenium2也可以使用CSS选择器来定位元素。

请查看如下的HTML文档。

```
<div class="minicart-wrapper">
    <p class="block-subtitle">Recently added item(s)
        <a class="close skip-link-close" href="#" title="Close">×</a>
    </p>
    <p class="empty">You have no items in your shopping cart.
    </p>
</div>
```

我们来创建一个测试，验证这些消息是否正确。

```
void testShoppingCartStatus(){
    // check content of My Shopping Cart block on Home page
    // get the Shopping cart icon and click to open the
    // Shopping Cart section
    shoppingCartIcon = driver.findElement(By.cssSelector("div.header-minicartspan.icon")
    shoppingCartIcon.click()
    // get the shopping cart status
    shoppingCartStatus = driver.findElement(By.cssSelector("p.empty")).text
    self.assertEqual("You have no items in your shopping cart.",
    shoppingCartStatus)
    // close the shopping cart section
    closeButton = driver.findElement(By.cssSelector("div.minicart-wrappera.close")
    closeButton.click()
}
```

鼠标事件

Web测试中，有关鼠标的操作，不只是单击，有时候还要做右击、双击、拖动等操作。这些操作包含在ActionChains类中。

常用的鼠标方法：

- contextClick() //右击
- douchClick() //双击
- dragAndDrop() //拖拽
- moveToElement() //鼠标停在一个元素上
- clickAndHold() // 按下鼠标左键在一个元素上

键盘事件

键盘操作经常处理的如下：

代码	描述
`sendKeys(Keys.BACK_SPACE)`	删除键(BackSpace)
`sendKeys(Keys.SPACE)`	空格键(Space)
`sendKeys(Keys.TAB)`	制表键(Tab)
`sendKeys(Keys.ESCAPE)`	回退键(Esc)
`sendKeys(Keys.ENTER)`	回车键(Enter)
`sendKeys(Keys.CONTROL,'a')`	全选（Ctrl+A）
`sendKeys(Keys.CONTROL,'c')`	复制（Ctrl+C）

模块化与类库

线性测试

至此之前，我们介绍的测试脚本，尽管使用了unittest测试框架，但是测试是按照指定的线路进行的，是线性的测试，完全遵循了脚本的执行顺序。

测试脚本1

```
driver = FirefoxDriver()
driver.get("http://wwww.xxx.com")
driver.findElement(By.id("tbUserName")).sendKeys("username")
driver.findElement(By.id(("tbPassword")).sendKeys("123456")
driver.findElement(By.id(("btnLogin")).click()
#执行具体用例操作
......
driver.quit ()
```

如上图，其实登录的模块可以共用。

模块化

模块话是自动化测试的第一个延伸和基础。需要对自动化重复编写的脚本进行重构（refactor），将重复的脚本抽取出来，放到指定的代码文件中，作为共用的功能模块。

测试脚本1：RanzhiCommon.java

```
/**
* 实际上的登录方法
*
* @param account：用户名
* @param password：密码
*/
public void logIn(String account, String password) {
    WebDriver driver = this.driver;
    // 输入用户名
    WebElement accountElement = driver.findElement(By.id("account"));
    accountElement.clear();
    accountElement.sendKeys(account);

    // 输入密码
    WebElement passwordElement = driver.findElement(By.id("password"));
    passwordElement.clear();
    passwordElement.sendKeys(password);

    // 点击登录按钮
    driver.findElement(By.id("submit")).click();

    try {
        Thread.sleep(2000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

另一份文件 RanzhiCommon.java

```
/**
* 登出系统
*/
public void logOut() {
    WebDriver driver = this.driver;
    driver.findElement(By.id("start")).click();
    driver.findElement(By.linkText("退出")).click();
    try {
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

}
```

自动化的测试：代码如下

```
package com.hello;

import com.hello.library.RanzhiCommon;
import org.junit.After;
import org.junit.Assert;
import org.junit.Before;
import org.junit.Test;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

/**
 * Created by Linty on 9/11/2016.
 * 使用模块化的测试用例
 */
public class RanzhiTestCase02 {

    // 成员变量
    private WebDriver driver;
    private String baseUrl;
    private RanzhiCommon common;

    @Before
    public void setUp() {
        this.driver = new FirefoxDriver();
        this.baseUrl = "http://172.31.95.220/ranzhi/www";
        this.common = new RanzhiCommon(this.driver, this.baseUrl);
    }

    @After
    public void tearDown() {

        this.driver.quit();
        this.common = null;
    }

    @Test
    public void testLoginWithModule() {
        WebDriver driver = this.driver;
        RanzhiCommon common = this.common;
        // 步骤一：打开页面
        common.openWebPage("/");
        Assert.assertEquals("登录页面打开错误",
                this.baseUrl + "/sys/index.php?m=user&f=login&referer=L3JhbnpoaS93d3cvc3lzL2luZGV4LnBocA==",
                driver.getCurrentUrl());

        // 步骤二：切换语言
        String actualLanguage = common.changeChinese();

        Assert.assertEquals("系统语言切换失败", "简体", actualLanguage);

        // 步骤三：进行登录
        common.logIn("admin", "123456");
        Assert.assertEquals("登录页面登录跳转失败",
                this.baseUrl + "/sys/index.php?m=index&f=index",
                driver.getCurrentUrl());

        // 步骤四：退出系统
        common.logOut();
        Assert.assertEquals("登录页面退出跳转失败",
                this.baseUrl + "/sys/index.php?m=user&f=login",
                driver.getCurrentUrl());
    }
}
```

参数化驱动

数据驱动

如果说模块化是自动化测试的第一步，那么数据驱动是自动化的第二步，从本意上来讲。数据改变更新驱动自动化的执行。从而引起测试结果的改变。其实类似于参数化。

示例代码

```
@Test
public void testAddBatchUserByCsv() {

  // 读取 csv 文件到FilerReader中
  // 用捕获异常的方式 进行文件读取，防止出现 “文件不存在” 的异常
  Reader in = null;
  try {
    in = new FileReader("src/main/resources/team_member.csv");
  } catch (FileNotFoundException e) {
    e.printStackTrace();
  }

  // 读取 csv 到 records中
  Iterable<CSVRecord> records = null;
  try {
    records = CSVFormat.EXCEL.parse(in);
  } catch (IOException e) {
    e.printStackTrace();
  }
  // 遍历 records，循环添加 userToAdd
  for (CSVRecord record : records) {
    RanzhiUser userToAdd = new RanzhiUser(
      record.get(0), record.get(1),
      Integer.parseInt(record.get(3)),
      Integer.parseInt(record.get(4)),
      record.get(2).toCharArray()[0],
      record.get(5),
      record.get(0) + record.get(6)
    );

    baseRanzhiCommon.login("admin", "123456");

    baseDriver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    String expectedMainUrl = baseUrl + "sys/index.php?m=index&f=index";
    Assert.assertEquals("登录成功主页跳转失败", expectedMainUrl, baseDriver.getCurrentUrl());

    baseRanzhiCommon.selectApp(RanzhiApp.Admin);
    baseDriver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    String expectedAdminUrl = baseUrl + "sys/index.php?m=admin&f=index";
    Assert.assertEquals("后台管理主页跳转失败", expectedAdminUrl, baseDriver.getCurrentUrl());

    baseDriver.switchTo().frame("iframe-superadmin");
    baseDriver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

    baseRanzhiCommon.selectSubMenuForAdmin(AdminSubMenu.Organization);


    String expectedOrganizationUrl = baseUrl + "sys/index.php?m=admin&f=index";
    Assert.assertEquals("后台管理组织跳转失败", expectedOrganizationUrl, baseDriver.getCurrentUrl());

    baseRanzhiCommon.clickAddUserButton();
    baseDriver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

    String expectedAddUserUrl = baseUrl + "sys/index.php?m=user&f=create";
    Assert.assertEquals("添加成员主页跳转失败", expectedAddUserUrl, baseDriver.getCurrentUrl());

    baseRanzhiCommon.addNewUser(userToAdd);
    baseDriver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    try {
      sleep(5000);
    } catch (InterruptedException ignored) {
    }
    String expectedUserSavedUrl = baseUrl + "sys/index.php?m=user&f=admin";
    Assert.assertEquals("用户保存跳转失败", expectedUserSavedUrl, baseDriver.getCurrentUrl());
    baseDriver.switchTo().defaultContent();
```

```
    baseRanzhiCommon.logout();
    baseDriver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

    String expectedLogoutUrl = baseUrl + "sys/index.php?m=user&f=login";
    Assert.assertEquals("退出登录页面跳转错误", expectedLogoutUrl, baseDriver.getCurrentUrl());

    baseRanzhiCommon.login(userToAdd.getAccount(), userToAdd.getPassword());

    baseDriver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    String expectedMainUrl2 = baseUrl + "sys/index.php?m=index&f=index";
    Assert.assertEquals("登录成功主页跳转失败", expectedMainUrl2, baseDriver.getCurrentUrl());

    baseRanzhiCommon.logout();
    baseDriver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

    String expectedLogoutUrl2 = baseUrl + "sys/index.php?m=user&f=login";
    Assert.assertEquals("退出登录页面跳转错误", expectedLogoutUrl2, baseDriver.getCurrentUrl());

  }


}
```

关于参数化驱动，我们可以将数据放到csv中，然后通过读取csv的数据进行自动化测试。同时也可以使用数据库尝试，代码如下：

```
@Test
public void testAddBatchUserByDb() {
  Statement stmt = null;
  ResultSet rs = null;
  try {
    Class.forName("com.mysql.cj.jdbc.Driver").newInstance();
  } catch (InstantiationException e) {
    e.printStackTrace();
  } catch (IllegalAccessException e) {
    e.printStackTrace();
  } catch (ClassNotFoundException e) {
    e.printStackTrace();
  }
  try {
    Connection conn = null;
    try {
      conn = DriverManager.getConnection("jdbc:mysql://localhost/test?" +
                                         "user=root&password=");
    } catch (SQLException e) {
      e.printStackTrace();
    }

    stmt = conn.createStatement();
    String sql = "SELECT \n" +
      "  'account',\n" +
      "  'realname',\n" +
      "  'gender',\n" +
      "  'dept',\n" +
      "  'role',\n" +
      "  'password',\n" +
      "  'email' \n" +
      "FROM\n" +
      "  'test'.'userlist' \n" +
      "LIMIT 0, 1000 ;";
    rs = stmt.executeQuery(sql);

    // or alternatively, if you don't know ahead of time that
    // the query will be a SELECT...

    if (stmt.execute(sql)) {
      rs = stmt.getResultSet();
      System.out.println(rs.next());
    }
    if (rs != null) {
      while (rs.next()) {
        RanzhiUser userToAdd = new RanzhiUser(
```

```
        rs.getString("account"),
        rs.getString("realname"),
        Integer.parseInt(rs.getString("role")),
        Integer.parseInt(rs.getString("dept")),
        rs.getString("gender").toCharArray()[0],
        rs.getString("password"),
        rs.getString("account") + rs.getString("email")
      );

      baseRanzhiCommon.login("admin", "123456");

      baseDriver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
      String expectedMainUrl = baseUrl + "sys/index.php?m=index&f=index";
      Assert.assertEquals("登录成功主页跳转失败", expectedMainUrl, baseDriver.getCurrentUrl());

      baseRanzhiCommon.selectApp(RanzhiApp.Admin);
      baseDriver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
      String expectedAdminUrl = baseUrl + "sys/index.php?m=admin&f=index";
      Assert.assertEquals("后台管理主页跳转失败", expectedAdminUrl, baseDriver.getCurrentUrl());

      baseDriver.switchTo().frame("iframe-superadmin");
      baseDriver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

      baseRanzhiCommon.selectSubMenuForAdmin(AdminSubMenu.Organization);


      String expectedOrganizationUrl = baseUrl + "sys/index.php?m=admin&f=index";
      Assert.assertEquals("后台管理组织跳转失败", expectedOrganizationUrl, baseDriver.getCurrentUrl());

      baseRanzhiCommon.clickAddUserButton();
      baseDriver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

      String expectedAddUserUrl = baseUrl + "sys/index.php?m=user&f=create";
      Assert.assertEquals("添加成员主页跳转失败", expectedAddUserUrl, baseDriver.getCurrentUrl());

      baseRanzhiCommon.addNewUser(userToAdd);
      baseDriver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
      try {
        sleep(5000);
      } catch (InterruptedException ignored) {
      }
      String expectedUserSavedUrl = baseUrl + "sys/index.php?m=user&f=admin";
      Assert.assertEquals("用户保存跳转失败", expectedUserSavedUrl, baseDriver.getCurrentUrl());
      baseDriver.switchTo().defaultContent();

      baseRanzhiCommon.logout();
      baseDriver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

      String expectedLogoutUrl = baseUrl + "sys/index.php?m=user&f=login";
      Assert.assertEquals("退出登录页面跳转错误", expectedLogoutUrl, baseDriver.getCurrentUrl());

      baseRanzhiCommon.login(userToAdd.getAccount(), userToAdd.getPassword());

      baseDriver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
      String expectedMainUrl2 = baseUrl + "sys/index.php?m=index&f=index";
      Assert.assertEquals("登录成功主页跳转失败", expectedMainUrl2, baseDriver.getCurrentUrl());

      baseRanzhiCommon.logout();
      baseDriver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

      String expectedLogoutUrl2 = baseUrl + "sys/index.php?m=user&f=login";
      Assert.assertEquals("退出登录页面跳转错误", expectedLogoutUrl2, baseDriver.getCurrentUrl());

    }
  }
  // Now do something with the ResultSet ....
} catch (SQLException ex) {
  // handle any errors
  System.out.println("SQLException: " + ex.getMessage());
  System.out.println("SQLState: " + ex.getSQLState());
  System.out.println("VendorError: " + ex.getErrorCode());
```

```
  } finally {
    // it is a good idea to release
    // resources in a finally{} block
    // in reverse-order of their creation
    // if they are no-longer needed

    if (rs != null) {
      try {
        rs.close();
      } catch (SQLException sqlEx) {
      } // ignore

      rs = null;
    }

    if (stmt != null) {
      try {
        stmt.close();
      } catch (SQLException sqlEx) {
      } // ignore

      stmt = null;
    }
  }
}
```

阶段小结

这里的数据换成了特别的数据，就是关键字。

1. Selenium IDE 的作用
   通过录制页面元素的定位和操作，进行自定义命令的编辑，（Select定位页面元素），导出指定的带单元测试框架的脚本（自动化测试用例），辅助代码编写，以及快速入门。
2. Selenium Java 环境搭建

   使用maven的方式，管理依赖项，进行环境搭建
   新建maven的项目，编辑项目中的 pom.xml，*.jar jar包

   - 依赖项要添加对，包名和版本
   - 电脑联网，需要连接访问maven仓库
3. Selenium WebDriver 的元素定位
   需要注意iFrame

   - 单一元素定位

     id, name, class name(不靠谱), tag name（不靠谱）
     xpath, css selector, link text, partial link text

     - class是以 `.` 开头
     - id 是以`#` 开头
     - css selector：`#langs > button`
     - xpath：`//*[@id="langs"]/button`
     - xpath(绝对路径)：`/html/body/div/div/div[1]/div/div/button`

       解释

```
<html>
<body>
    <div>
        <div>
            <div>
                <div>
                    <div>
                        <button></button>
                    </div>
```

```
                </div>
            </div>
            <div></div>
        </div>
    <div>
</body>
</html>
```

- 定位一系列元素

  id name (不靠谱)
  class name
  css selector

  xpath

4. Selenium 使用 单元测试框架

   Java: junit4
   Python: unittest

项目	Java	Python
方法名	小骆驼 testRanzhiLogin()	类C的命名test_ranzhi_login()
用例的步骤	@Test(方法以test开头)	test_开头
用例的前置条件	@Before(方法 setUp)	重写setUp()
用例的清场方式	@After(方法 tearDown)	重写tearDown()
依赖的处理	添加junit依赖项到pom.xml	import unittest 写一个测试用例类，继承unittest.TestCase

5. Selenium 使用 模块化 用例设计

   1. library文件夹
      RanzhiCommon.java
   2. 面向对象：实例化一个对象（注意构造方法），调用对象的成员，包括普通方法，变量

      Java实体类
      Java枚举类型

      设计模式：自动化测试框架，页面对象模式

      python 面向对象

      dict类型

6. Selenium 使用 数据驱动 进行用例设计和执行

   1. 读文件进行测试（准备的用例数据）
   2. 读数据库进行测试（不推荐）
      - 代码中，注意关闭数据库连接
      - 代码中，注意SQL脚本的效率，
      - 替代方案，导出数据库的数据到文件中，然后读文件，或者把该文件导入本地的数据库，注意类型转换，尤其是日期类型。
   3. 新建一个`CsvUtility.java`，新建一个`DbUtility.java`
7. Selenium 工具的使用汇总 IDEA 和 Git

   `IDEA` `PyCharm`
   打开对的文件夹（作为项目）
   Reformat Code （Ctrl + Alt + L）

   Refactor | rename 重构，修改名字，批量关联的级联修改
   按下ctrl+鼠标左键，导航定位的声明处
   红灯处理，选中错误的行，然后等红灯出来，点右上角的箭头

Java里面 用 Alt + Enter 键
Git的集成：

github.io
coding.net
git.oschina.net
bitbucket.org
code.csdn.net(腾讯的测试工具 apt)

- 在线建仓库
- git clone 下载到本地（需要新建非系统盘的 git文件夹）
- git pull 获取远程变化到本地
- commit 提交到本地
- push 提交到远程 集成在IDEA中 菜单 VCS | Enable ……，里面选择git 提交git 需要先add 导出一份 完全没有版本管理信息的代码。export