1、简介

HttpUnit 是基于JUnit构建的一个开源测试框架，专门针对Web应用的测试，解决使用JUnit框架无法对远程Web内容进行测试的弊端。

2、工作原理

HttpUnit通过模拟浏览器的行为，包括提交表单（form）、处理页面框架（frames）、基本的http验证、cookies及页面跳转（redirects）处理等。通过HttpUnit提供的功能，用户可以方便的和服务器端进行信息的交互，将返回的网页内容作为普通文本、XML Dom对象或者是作为链接、页面框架、图像、表单、表格等的集合进行处理，然后使用JUnit框架进行测试，还可以导向一个新的页面，然后进行新页面的处理，这个功能使你可以处理一组在一个操作链中的页面。

3、特征

对于一般Web测试工具使用记录、回放的功能来说，这些测试工具的缺陷就是当页面设计被修改以后，这些被记录的行为就不能再重用了，每当页面改变一次，就需要重新录制一次才能正常重放。例如，若页面上有个元素开始的设计是采用单选框，此时这些工具记录的就是你的单项选择动作，一旦设计发生了变化，比如说改成了下拉选择，这时候，以前录制的测试过程就无效了，必须要重新录制。

而HttpUnit因为关注的是这些控件的内容，而不管页面的表现形式（layout），所以不管表现形式如何变化，都不影响已确定的测试的可重用性。

4、HttpUnit 工作原理

4.1 如何使用httpunit处理页面的内容

WebConversation类是HttpUnit框架中最重要的类，WebConversation可以被看作是“浏览器”。用户首先创建一个请求（WebRequest），然后让WebConversation返回响应（WebResponse）。如下：

```
//新创建一个“浏览器”对象
WebConversation wc = new WebConversation();
// WebRequest类，用于模仿客户的“请求”，通过它可以向服务器发送信息。
```

```
WebRequest req = new GetMethodWebRequest( http://www.sqalab.com
);
// WebResponse类，用于模仿浏览器获取服务器端的响应信息。
WebResponse resp = wc.getResponse ( req );
```

4.2 获取指定页面的内容

4.2.1 通过 getText 直接获取页面的所有内容

```
// 建立一个“浏览器”实例
WebConversation wc = new WebConversation();
// 将指定URL的请求传给wc，然后获取相应的响应
WebResponse wr = wc.getResponse( "http://www.sqalab.com" );
// 用wc的getText方法获取相应的全部内容
System.out.println( wr.getText() );
```

4.2.2 增加参数通过Get方法访问页面

```
// 建立一个WebConversation实例
WebConversation wc = new WebConversation();
// 向指定的URL发出请求
WebRequest req = new GetMethodWebRequest(
"http://www.sqalab.com/sea       rch" );
// 给请求加上参数
req.setParameter("keyword"，"httpunit");
// 获取响应对象
WebResponse resp = wc.getResponse( req );
// 用getText方法获取相应的全部内容
System.out.println( resp.getText() );
```

4.2.3 增加参数通过Post方法访问页面

```
//建立一个WebConversation实例
WebConversation wc = new WebConversation();
//向指定的URL发出请求
WebRequest req = new PostMethodWebRequest(
"http://www.sqalab.com/se       arch" );
//给请求加上参数
req.setParameter("keyword"，"httpunit");
//获取响应对象
WebResponse resp = wc.getResponse( req );
```

```
//用getText方法获取相应的全部内容
//用System.out.println将获取的内容打印在控制台上
System.out.println( resp.getText() );
```

4.3 处理页面的链接（links）

模拟用户点击请求页面中的某一个链接，然后获得它指向文件的内容。比如在页面index.html中有一个链接 "应用 HttpUnit 进行Web测试 "，它显示的内容是这篇文章的内容，它链向的页面是 http://www.sqalab.com/article/html/article_59.html.

```
// 建立一个WebConversation实例
WebConversation wc = new WebConversation();
// 获取响应对象
WebResponse resp = wc.getResponse(
"http://www.sqalab.com/index.html" );
// 获得页面链接对象
WebLink link = resp.getLinkWith( "应用 HttpUnit 进行Web测试 " );
// 模拟用户单击事件
link.click();
// 获得当前的响应对象
WebResponse nextLink = wc.getCurrentPage();
// 用getText方法获取相应的全部内容，并打印
System.out.println( nextLink.getText() );
```

4.4 处理页面的表格（table）

表格是用来控制页面显示的对象，在HttpUnit中使用数组来处理页面中的多个表格，可以用 resp.getTables() 方法获取页面所有的表格对象。将它们依次出现在页面中的顺序保存在一个数组里。

```
// 创建一个WebConversation对象
WebConversation wc = new WebConversation();
// 设置HTTP代理服务器地址和端口
wc.setProxyServer( "proxy.pvgl.sap.corp",  8080 );
// 新建一个URL请求对象req
WebRequest req = new
GetMethodWebRequest("http://httpunit.sourceforge.
net/doc/cookbook.html");
```

```
// 发出一个请求req，并取得它相对应的响应resp
WebResponse resp = wc.getResponse(req);
// 获得响应的页面中的 Table
WebTable[] tables = resp.getTables();
// 取出第一个 table
WebTable table = tables[0];
// 从 2 * 2 的 table 取出cell里的值
for ( int i=0 ; i<3 ; i++ ) {
for ( int j=0 ; j<2 ; j++ )
System.out.println(table.getCellAsText(i，j).trim());
}
```

4.5 处理页面的表单（form）

表单是用来接受用户输入，也可以向用户显示用户已输入信息（如需要用户修改数据时，通常会显示他以前输入过的信息），在HttpUnit中使用数组来处理页面中的多个表单，你可以用resp.getForms()方法获取页面所有的表单对象。他们依照出现在页面中的顺序保存在一个数组里面。

```
// 建立一个WebConversation实例
WebConversation wc = new WebConversation();
// 获取响应对象
WebResponse resp = wc.getResponse(
"http://www.sqalab.com/article/html/a        rticle_59.html" );
// 获得对应的表单对象
WebForm webForm = resp.getForms()[0];
// 获得表单中所有控件的名字
String[] pNames = webForm.getParameterNames();
int i = 0;
int m = pNames.length;
// 循环显示表单中所有控件的内容
while(i<m){
System.out.println("第"+(i+1)+"个控件的名字是"+pNames[i]+
"，里面的内容是"+webForm.getParameterValue(pNames[i]));
++i;
}
```