为使用JSF FacesContext的方法 编写测试用例—— Mock JSF FacesContext

作者：Hassan

原文链接：https://ld246.com/article/1471917025166

来源网站：链滴

最近在编写单元测试的时候，遇到两个问题。 问题1：因为项目使用了JSF框架，很多方法使用了acesContext这个类，导致在编写这些方法的相关单元测试时，遇到环境问题。 问题2：由于项目与三方服务有交互（使用HTTP，第三方服务测试服务位于公网），而这两个项目存在相互request的需，这意味着当第三方服务需要request本项目的时候，只能通过公网IP来访问。这就面临着无法在本做"第三方服务请求该项目"的相关接口测试。
通过上网查了相关的资料，也找到了一些解决方案。在这里做一个总结。
<h2></>环境：</h2>
JSF 2.0 JUnit 4 - Link Mockito 2.0 - a href="http://site.mockito.org/#intro">Link moco - Link
<h2>简要介绍：</h2>
Mockito：Mockito是一个针对Java的mocking框架。大白话来解释一下它的功能：Mockito可在你需要某个对象的时候，模拟出一个来使用它。或者，当你对某个对象调用某个方法的时候，你希返回的东西，它也可以给你模拟出来。当然，Mockito的功能还不止这些。 moco：可以模拟出一个务器，让你可以事先配置好请求uri和对于的响应，使得在进行与HTTP相关的单元测试的时候，不必手动配置一个Server。你可以在单元测试开始时，启动它，在单元测试结束后，关闭它。
<h2>Mockito使用：</h2>
首先，假设项目中使用了：
<pre class="brush: java">package foo;

import java.util.Map;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;

@ManagedBean
@RequestScoped
public class AlphaBean {
public String incrementFoo() {
Map<String, Object> session = FacesContext.getCurr
ntInstance()
.getExternalContext()
.getSessionMap();
Integer foo = (Integer) session.get("foo");
foo = (foo == null) ? 1 : foo + 1;
session.put("foo", foo);
return null;
}
}</pre>

在执行单元测试的时候，FacesContext.getCurrentInstance()会返回null，所以需要mock一个FcesContext。FacesContext有一个方法：
<pre class="brush: java">protected void javax.faces.context.FacesContext.setCurrentInstance(acesContext context)</pre>
所以，首先创建一个抽象类FacesContextMocker 继承FacesContext类：
<pre class="brush: java">package com.fenxiangz.core;

import javax.faces.context.FacesContext;

import org.mockito.Mockito;
import org.mockito.invocation.InvocationOnMock;
import org.mockito.stubbing.Answer;

public abstract class FacesContextMocker extends FacesContext {
private FacesContextMocker() {
}

private static final Release RELEASE = new Release();

private static class Release implements Answer<
oid> {
@Override
public Void answer(InvocationOnMock invocation) throws Throwable {
setCurrentInstance(null);
return null;
}
}

public static FacesContext mockFacesContext() {
FacesContext context = Mockito.mock(FacesContext.class);
setCurrentInstance(context);
Mockito.doAnswer(RELEASE).when(context).release();
return context;
}
}</pre>

为什么是抽象类？因为FacesContext是一个抽象类，我们不希望实现FacesContext里面的方法所以在这里也定义成抽象类。 这个抽象类有一个静态方法：mockFacesContext()，当我们在写单元试的时候，就可以直接通过FacesContextMocker.mockFacesContext()来获取一个FacesContext。而，通过Mockito配置FacesContext的实例来预先配置一些FacesContext使用的行为。 有了FacesCntextMocker，我们就可以先写一个简单的单元测试代码了。如下：
<pre class="brush: java">package foo.test;

import static org.junit.Assert.assertEquals;
import static org.mockito.Mockito.mock;

```
import static org.mockito.Mockito.when;

import java.util.HashMap;
import java.util.Map;

import javax.faces.context.ExternalContext;
import javax.faces.context.FacesContext;

import org.junit.Test;

import foo.AlphaBean;

public class AlphaBeanTest {
@Test
public void testIncrementFoo() {
FacesContext context = ContextMocker.mockFacesContext();
try {
Map&lt;String, Object&gt; session = new HashMap
lt;String, Object&gt;();
ExternalContext ext = mock(ExternalContext.class);
when(ext.getSessionMap()).thenReturn(session);
when(context.getExternalContext()).thenReturn(ext);


  AlphaBean bean = new AlphaBean();
  bean.incrementFoo();
  assertEquals(1, session.get("foo"));
  bean.incrementFoo();
  assertEquals(2, session.get("foo"));
} finally {
  context.release();
}

}
```
</pre>

这样，就使得即使没有JSF运行环境，也可以正常执行包含FacesContext的逻辑代码。
<h2>moco使用：</h2>
http://github.com/dreamhead/moco/blob/master/moco-doc/usage.md