Mocking HttpContext, HttpResponse, HttpRequest, HttpSessionState etc. in ASP.Net

Posted on July 9, 2011 by Mohit Thakral

While developing web application using ASP.Net. If one wants to follow TDD, there are a lot challenges that one has to face. One of the challenge is to mock various web based objects like HttpContext, HttpRequest etc.. In this post I will talk about mocking following web based objects

- HttpContext
- HttpRequest
- HttpResponse
- HttpSessionState (Page's Session Property's Type)
- HttpApplicationState (Application Property's Type)

To ease the unit testing of the components dependent, on above mentioned web based objects. Microsoft released System.Web.Abstractions.dll in .Net 3.5(in .Net 4.0 this dll was merged into System.Web.dll) all these classes were still under System.Web namespace. We will be using those classes only.

What Microsoft did was pretty simple. On the place of modifying existing classes. They created a "wrapper class" that inherits from an "abstract base class". Wrapper classes have only one constructor to which one has to pass an object of the real class. For example HttpContext is the real class for that they created HttpContextBase that is the "abstract base class" and HttpContextWrapper is the "wrapper class" which has only one constructor that accepts the "real class" i.e. "HttpContext".

This way all of the components that are dependent on "real class" should be made dependent on "abstract base class". So, that while unit testing one creates a mock of the "abstract base class" and in real-time execution an object of the wrapper class is passed, that take the real class.

For HttpContext, HttpRequest, HttpResponse,HttpSessionState and HttpApplicationState following are the corresponding classes.

Class to be Mocked	Wrapper Class	Abstract Class
HttpContext	HttpContextWrapper	HttpContextBase
HttpResponse	HttpResponseWrapper	HttpResponseBase
HttpRequest	HttpRequestWrapper	HttpResponseBase
HttpSessionState	HttpSessionStateWrapper	HttpSessionStateBase
HttpApplicationState	HttpApplicationStateWrapper	HttpApplicationStateBase

For the sample below. We we will test a class that needs all four of these.The only thing is that class will use HttpContext to access all these classes.

Following is how a class might look like when it is using these objects without worrying about unit testing.

```
namespace TestWebApp.Web
{
    public class CreditCardProcessor
    {

        public void Save()
        {
            // Do some operation in database
            var currentContext = HttpContext.Current;
            currentContext.Application["UserCreditCardSaveCount"] =
                (int)(currentContext.Application["UserCreditCardSaveCount"]) + 1;
            currentContext.Session["EnteredCreditInfo"] = "Y";
            currentContext.Response.Redirect(currentContext.Request.QueryString["rUrl"]);
        }

    }
}
```

After changing context class to abstract base class, code will look like following, Enabling the injection context object. To use this class you have to add reference to System.Web.Abstractions as stated earlier no need to change "using". Since classes are still in System.Web namespace.

```
using System.Web;
using System.Web.SessionState;
```

Search

Search

Follow Us!

Not Found

The resource could not be found.

Subscribe

Your email:
Enter email address..

Subscribe Unsubscribe

Categories

Administration
ASP.Net
C#
MySQL
PHP
Random
Uncategorized
Unit Testing
Web
Windows Phone 7

Tags

ASP.Net Moq UnitTesting ASP.NET MVC ASP.Net MVC2 Blunder C# CSS debug IIS 7.5 JS Minifier MugBug mysql php ec2 mysql shell Performance ASP.Net php phpstrom Power Shell Windows 7 WP7 Learn-WP7-Series YUI Compressor

```
namespace TestWebApp.Web
{
    public class CreditCardProcessor
    {
        private HttpContextBase context;


        public CreditCardProcessor(HttpContextBase httpContext) {
            this.context = httpContext;
        }

        public void Save() {
            // Do some operation in database

            context.Application["UserCreditCardSaveCount"] =
                (int)(context.Application["UserCreditCardSaveCount"]) + 1;
            context.Session["EnteredCreditInfo"] = "Y";
            context.Response.Redirect(context.Request.QueryString["rUrl"]);
        }

    }
}
```

Now, following is one sample of the test. Following test uses nunit testing framework and Moq mocking framework.

```
using Moq;
using NUnit.Framework;
using TestWebApp.Web;
using System.Web;

namespace MockSampleWebApp.Tests
{
    [TestFixture]
    public class CreditCardProcessorTests
    {
        [Test]
        public void Can_Save()
        {
            //SetUp
            var mockContext = new Mock();
            var mockRequest = new Mock();
            var mockResponse = new Mock();
            var mockSessionState = new Mock();
            var mockApplication = new Mock();
            var processor = new CreditCardProcessor(mockContext.Object);

            mockRequest.SetupGet(req => req.QueryString).Returns(HttpUtility.ParseQueryString("?rUrl=BlahBlah"));

            mockContext.Setup(ctxt => ctxt.Request).Returns(mockRequest.Object);
            mockContext.Setup(ctxt => ctxt.Response).Returns(mockResponse.Object);
            mockContext.Setup(ctxt => ctxt.Session).Returns(mockSessionState.Object);
            mockContext.Setup(ctxt => ctxt.Application).Returns(mockApplication.Object);

            mockApplication.Setup(app => app["UserCreditCardSaveCount"]).Returns(2);
            mockApplication.SetupSet(app => app["UserCreditCardSaveCount"]=3).Verifiable();
            mockSessionState.SetupSet(sstat => sstat["EnteredCreditInfo"]="Y").Verifiable();
            mockResponse.Setup(resp => resp.Redirect("BlahBlah"));


            processor.Save();

            mockContext.VerifyAll();
            mockSessionState.VerifyAll();
            mockRequest.VerifyAll();
            mockResponse.VerifyAll();


        }
    }
}
```

And finally the production usage would be as following.

```
 using System;
 using System.Collections.Generic;
 using System.Linq;
 using System.Web;
 using System.Web.UI;
 using System.Web.UI.WebControls;
 using TestWebApp.Web;

 namespace MockSampleWebApp
 {
     public partial class CreditCardSave : System.Web.UI.Page
     {
         protected void Page_Load(object sender, EventArgs e) {

         }

         protected void Save_Clicked(object sender, EventArgs e)
         {
             var ccProcessor=new CreditCardProcessor(new HttpContextWrapper(HttpContext.Current));
             ccProcessor.Save();
         }
     }
 }
```

Please click on following image to download complete code.

MockSam ..1

This entry was posted in ASP.Net, C#, Unit Testing and tagged ASP.Net Moq UnitTesting. Bookmark the permalink.

← Learn Windows Phone 7 Development , Day-3 : Hello! World – Deep Dive 2

Learn Windows Phone 7 Development , Day-4 : Hello! World – Deep Dive 3 →

4 Responses to *Mocking HttpContext, HttpResponse, HttpRequest, HttpSessionState etc. in ASP.Net*

yawar says:
June 20, 2012 at 6:01 am

Wonderfull! great work Mohit Thakral

Riddhi M says:
August 6, 2012 at 12:49 am

Thanks.. Really helpful.

Jaspreet Singh says:
September 18, 2012 at 10:02 am

Hi,
Thanks this post was really very helpful for me. Can you please provide me with some more demo's or POC's related to session mocking or if you have some useful link then please share.

jarofkla says:
May 23, 2013 at 9:38 am

Thanks for the great walkthrough. I tried to understand/implement other suggestions on how to mock http objects, but this guidance and sample ultimately got me a working unit test. Great job.

Leave a Reply

Your email address will not be published. Required fields are marked *

Name *

Email *

Website

Check here to Subscribe to notifications for new posts

You may use these HTML tags and attributes:
<a href="" title=""> <abbr title=""> <acronym title=""> <b>
<blockquote cite=""> <cite> <code> <del datetime=""> <i>
<q cite=""> <strike>

Post Comment

Please note: JavaScript is required to post comments.

Proudly powered by WordPress JustCSS v1.2 by Pross