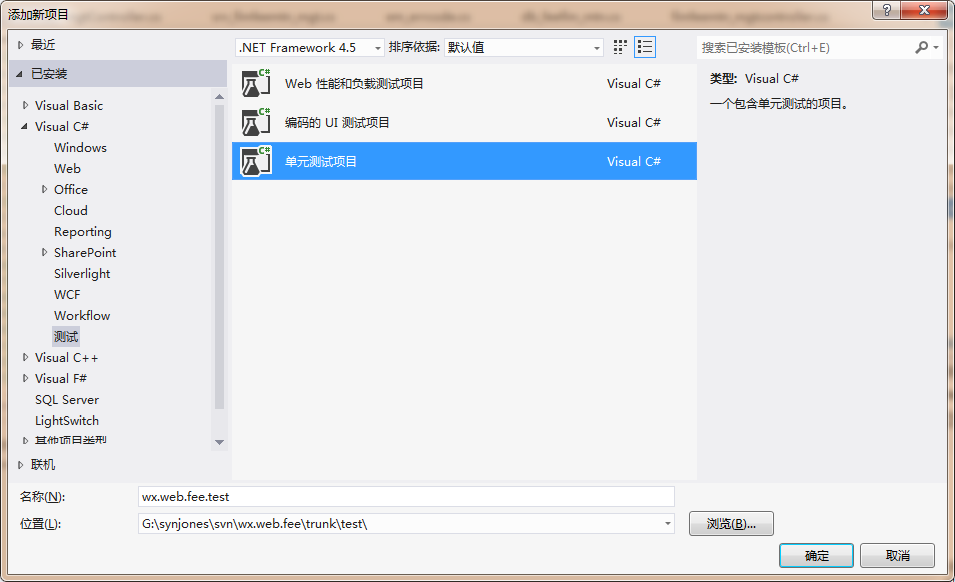
**1.前言**

　现在这个项目已经有阶段性的模块完成了，所以就想着对这些模块进行单元测试，以保证项目的代码的质量。首先虽然标题是mvc+webapi 实质上我只是对mvc进行的测试。用的vs的unit test generator（2015自带）.至于其它版本的安装和介绍在这不做详细介绍。还有其实我也任务单元测试就是个鸡肋。第一它浪费时间，第二他也不能完全 保证我的代码完全没有问题，第三我的测试并不能算到的绩效当中去。但是不写它也总感觉不舒服，我是把它当做对我一个阶段代码的梳理来对待。

**2.单元测试建立**

**2.1新建单元测试项目**

****

此项目我是把它当作模块插件来处理。

**2.2.添加单元测试类**

[TestClass]

public class mtnfim

{

　　　　//实例化测试mvc Controller

mgtController mtn\_fim = new mgtController();

[TestMethod]

public void IndexTest()

{

　　　　　　//调用类中方法

var ret = mtn\_fim.Index() as ViewResult;

int errcode = testresult.action\_result(ret).errcode;

Assert.AreEqual(0, errcode);

}

}

**2.3.代码解释如下：**

**2.3.1Assert类的使用**

　　　　Assert.AreEqual()      测试指定的值是否相等，如果相等，则测试通过；

　　　　AreSame()                  用于验证指定的两个对象变量是指向相同的对象，否则认为是错误

　　　　AreNotSame()             用于验证指定的两个对象变量是指向不同的对象，否则认为是错误

　　　　Assert.IsTrue()         测试指定的条件是否为True，如果为True，则测试通过；

　　　　Assert.IsFalse()           测试指定的条件是否为False，如果为False，则测试通过；

　　　　Assert.IsNull()             测试指定的对象是否为空引用，如果为空，则测试通过；

　　　　Assert.IsNotNull()        测试指定的对象是否为非空，如果不为空，则测试通过；

**2.3.2.返回类型的处理**

　　　　　　　我项目当中mvc Controller的返回类型有两个JsonResult。ActionrResult.两种，所以针对于assert类的使用显然我要对mvc的返回的对象进行处理。

jsonresult的处理方式：

通过nugt下载ExposedObject把  对象转为dynami然后对其对象进行操作。代码如下。

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

var jsonret = mtn\_fim.add\_feeitem("测试", "cs");

var ret = Exposed.From(jsonret.Data);

int errcode = ret.errcode;

Assert.AreEqual(6002, errcode);

ActionrResult的处理：

相对与jsonresult直接操作返回对象即可，对此我写了一个方法。

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

public static class testresult

{

public static result<object> action\_result(ViewResult ret)

{

return ((wx.web.result<object>)(((System.Web.Mvc.ViewResultBase)((ret))).Model));

}

public static result apicitroller\_result(System.Web.Http.IHttpActionResult action)

{

var rs = ((System.Web.Http.Results.OkNegotiatedContentResult<wx.web.result>)

(action)).Content;

return rs;

}

}

[复制代码](javascript:void(0);)

使用如下：

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

var ret = mtn\_fim.Index() as ViewResult;

int errcode = testresult.action\_result(ret).errcode;

Assert.AreEqual(0, errcode);

 3.实现之外的事：

　　3.1.数据库初始化

           以上已经初步实现了单元测试的基本实现，但是考虑到单元测试重复测试，我用ef code first动态的数据库创建和需要测试的数据进行处理.

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

public void Initialize(FeeDbContext context)

{

List<organization> orgs = new List<organization>() {

new organization(){oid=1, name="新中新集团1", state=true,reg\_time=DateTime.Now},

new organization(){oid=2, name="山东大学", state=true,reg\_time=DateTime.Now},

new organization(){oid=3, name="浙江大学", state=true,reg\_time=DateTime.Now},

new organization(){oid=4, name="同济大学", state=true,reg\_time=DateTime.Now}

};

List<fee\_item> fim = new List<fee\_item>() {

new fee\_item(){name="电费",code="pow",type= feeItemType.Normal,State=true,Icon="img/jf\_dianf.png",sort=0,appid=0,reg\_time=DateTime.Now},

new fee\_item(){name="网费",code="net",type= feeItemType.Normal,State=true,Icon="img/jf\_wangf.png",sort=1,appid=1,reg\_time=DateTime.Now},

new fee\_item(){name="四六级",code="cet",type= feeItemType.Normal,State=true,Icon="img/jf\_siliuj.png",sort=2,appid=2,reg\_time=DateTime.Now},

new fee\_item(){name="报名费",code="pow",type= feeItemType.Normal,State=true,Icon="img/jf\_baomingf.png",sort=3,appid=3,reg\_time=DateTime.Now},

new fee\_item(){name="保险费",code="pow",type= feeItemType.Normal,State=false,Icon="img/jf\_baox.png",sort=4,appid=4,reg\_time=DateTime.Now},

new fee\_item(){name="住宿费",code="pow",type= feeItemType.Normal,State=true,Icon="img/jf\_dianf.png",sort=5,appid=5,reg\_time=DateTime.Now}

};

List<school\_power> con\_power = new List<school\_power>() {

new school\_power(){oid=1,power\_code="ykt",power\_id="1",reg\_time=DateTime.Now},

new school\_power(){oid=1,power\_code="sims",power\_id="2",reg\_time=DateTime.Now}

};

//depts.ForEach(o => context.dept.Add(o));

orgs.ForEach(o => context.organization.Add(o));

fim.ForEach(o => context.fee\_item.Add(o));

con\_power.ForEach(o => context.control\_power.Add(o));

context.SaveChanges();

}

[复制代码](javascript:void(0);)

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

public void initial()

{

try

{

var context = new FeeDbContext();

if (context.Database.Exists())

{

context.Database.Delete();

}

new List<IDataInitializer<FeeDbContext>>() {

new DataInit4dept()

}.Setup<FeeDbContext>(context);

}

catch (DbEntityValidationException ex)

{

StringBuilder error = new StringBuilder();

foreach (var item in ex.EntityValidationErrors)

{

foreach (var item2 in item.ValidationErrors)

{

error.Append(string.Format("{0}:{1}\r\n", item2.PropertyName, item2.ErrorMessage));

}

}

Console.WriteLine("数据库初始化报错：" + error);

throw ex;

}

catch (Exception e)

{

Console.WriteLine("数据库初始化报错：" + e.Message);

throw e;

}

}

[复制代码](javascript:void(0);)

　　　　ef code first写好之后就又到单元测试的部分了，所以要想每次执行单元测试内容的时候都执行一下以上内用就要用到单元测试的附属属性：

**[ClassInitialize()]  在运行类的第一个测试前先运行代码**

**[ClassCleanup()]  在运行完类中的所有测试后再运行代码**

**[TestInitialize()]   在运行每个测试前先运行代码**

**[TestCleanup()]   在运行完每个测试后运行代码**

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

[ClassInitialize]

public static void initialdb(TestContext context)

{

//初始化数据库

new DataBuilder().initial();

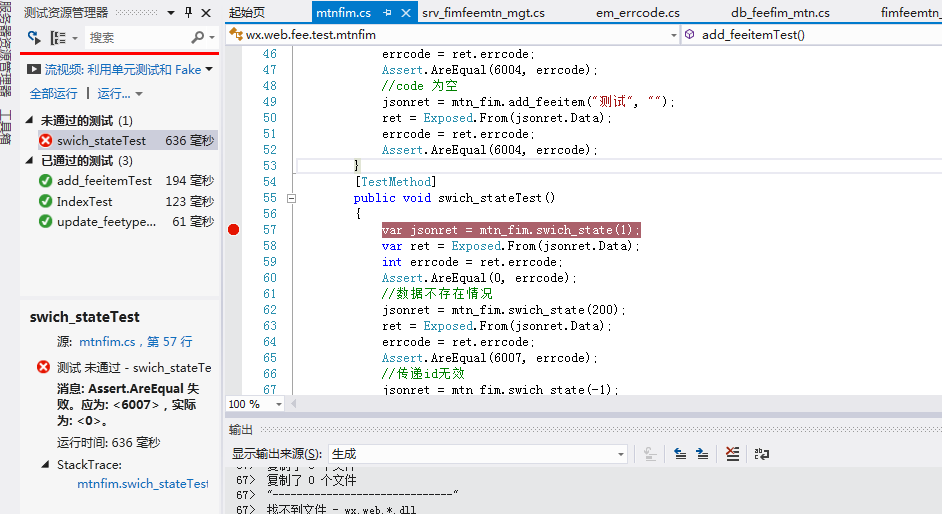
//加载webapi bin文件

apiconfig\_factory.Instance.Load\_Apis();

}

[复制代码](javascript:void(0);)

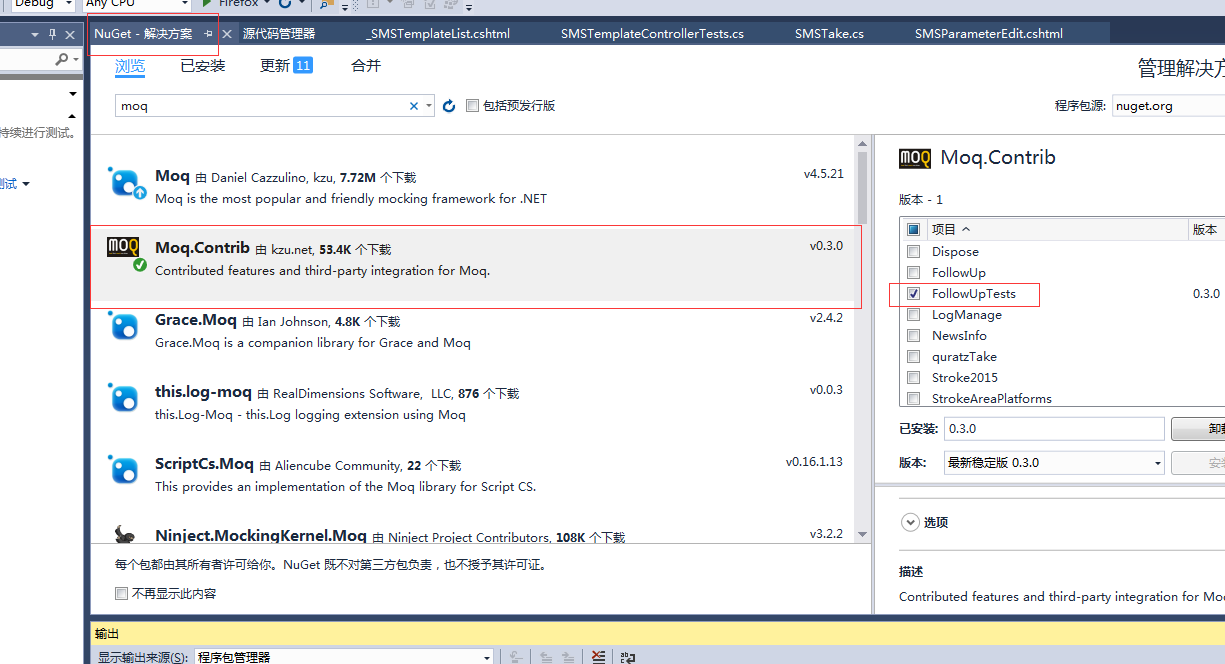
运行测试结果如下：



直到这单元测试的实现也就算结束了，但是对于单元测试还有很多东西没有往上写，比如测试模型，覆盖率等等，对于这个项目的测试也不仅限于单元测试，后边的性能测试和ui测试都会抽时间进行总结。

**3.使用Moq模拟AspnetMvc中的Request.Form**

**3.1nuget下载安装moq**

****

**3.2.实例**

1、比如Request.Form["MingCheng"]为control中需要模拟的内容

2、在单元测试中调用的代码

[复制代码](javascript:void(0);)

var formvalues = new NameValueCollection { { "ST\_MingCheng", "" } };//可以选用KeyValuePair

var request = new NameValueCollection { { "pageIndex", "" } };

var mockHttpContext = new Mock<HttpContextBase>();

mockHttpContext.Setup(c => c.Request.QueryString).Returns(request);

mockHttpContext.Setup(c => c.Request.Form).Returns(formvalues);

Mock<SMSTemplateController> mockEnterprise = new Mock<SMSTemplateController>();

var context = new ControllerContext(mockHttpContext.Object, new RouteData(), stc);

stc.ControllerContext = context; //设置control的context熟悉

var result = stc.\_SMSTemplatelist(); //调用action方法

[复制代码](javascript:void(0);)