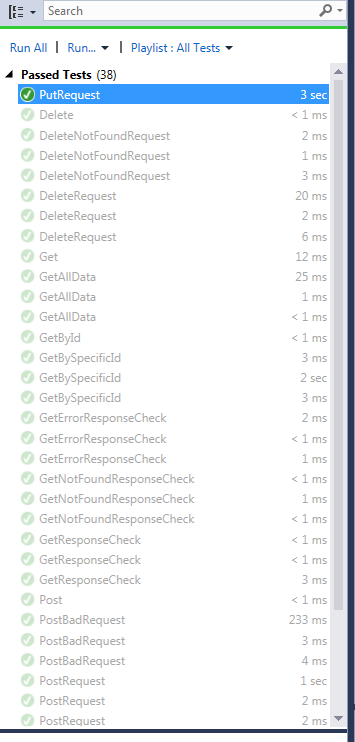
Assignment 27

Test Project for MVC Web API Project

Visual Studio default test suite used to do the validation for the project. Code slightly modified in order to do the disconnected testing.

Output



Code Snippet

MVC Controllers

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

using WebAPIDemo.Models;

namespace WebAPIDemo.Controllers

{

public class ItemController : ApiController

{

//IItemRepository \_repository;

//public ItemController(IItemRepository repository)

//{

// \_repository = repository;

//}

PODbEntities1 context = new PODbEntities1();

public IEnumerable<ITEM> Get()

{

context.Configuration.ProxyCreationEnabled = false;

return context.ITEMs.ToList();

}

[HttpGet]

public HttpResponseMessage Get(string ItemNumber)

{

context.Configuration.ProxyCreationEnabled = false;

ITEM i = context.ITEMs.Where(x => x.ITCODE == ItemNumber).FirstOrDefault();

if (i != null)

{

return Request.CreateResponse(HttpStatusCode.OK, i);

}

else

{

return Request.CreateErrorResponse(HttpStatusCode.NotFound, "Item Number : " + ItemNumber + " Not Found");

}

}

public HttpResponseMessage Post([FromBody] ITEM i)

{

try

{

context.Configuration.ProxyCreationEnabled = false;

context.ITEMs.Add(i);

context.SaveChanges();

HttpResponseMessage Message = Request.CreateResponse(HttpStatusCode.Created, i);

// Message.Headers.Location = new Uri(Request.RequestUri + i.ITCODE);

return Message;

}

catch (Exception ex)

{

return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);

}

}

public HttpResponseMessage Put([FromBody] ITEM i)

{

try

{

context.Configuration.ProxyCreationEnabled = false;

ITEM value = context.ITEMs.Where(x => x.ITCODE == i.ITCODE).FirstOrDefault();

value.ITDESC = i.ITDESC;

value.ITRATE = i.ITRATE;

context.SaveChanges();

HttpResponseMessage Message = Request.CreateResponse(HttpStatusCode.OK, value);

// Message.Headers.Location = new Uri(Request.RequestUri + value.ITCODE);

return Message;

}

catch (Exception ex)

{

return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);

}

}

public HttpResponseMessage Delete(string ItemCode)

{

HttpResponseMessage httpResponse = null;

try

{

context.Configuration.ProxyCreationEnabled = false;

ITEM I = context.ITEMs.Where(x => x.ITCODE == ItemCode).FirstOrDefault();

if (I == null)

{

httpResponse = Request.CreateErrorResponse(HttpStatusCode.NotFound, "Item with Code " + ItemCode + " Not found");

}

else

{

context.Entry(I).State = System.Data.Entity.EntityState.Deleted;

context.SaveChanges();

httpResponse = Request.CreateResponse(HttpStatusCode.OK);

}

return httpResponse;

}

catch (Exception ex)

{

return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);

}

}

protected override void Dispose(bool disposing)

{

context.Dispose();

base.Dispose(disposing);

}

}

}

POMasterController

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

using WebAPIDemo.Models;

namespace WebAPIDemo.Controllers

{

public class POMasterController : ApiController

{

public IEnumerable<POInformation> Get()

{

using (PODbEntities1 PE = new PODbEntities1())

{

PE.Configuration.ProxyCreationEnabled = false;

IEnumerable<POMASTER> pomasters = PE.POMASTERs.ToList();

IEnumerable<PODETAIL> podetails = PE.PODETAILs.ToList();

IEnumerable<ITEM> items = PE.ITEMs.ToList();

IEnumerable<SUPPLIER> suppliers = PE.SUPPLIERs.ToList();

List<POInformation> poinformation = new List<POInformation>();

var value = from p in pomasters

join q in podetails on p.PONO equals q.PONO

join i in items on q.ITCODE equals i.ITCODE

join s in suppliers on p.SUPLNO equals s.SUPLNO

select new { p.PONO, p.PODATE, p.SUPLNO, s.SUPLNAME, q.ITCODE, i.ITDESC, q.QTY };

foreach (var v in value)

{

POInformation poi = new POInformation();

poi.PONO = v.PONO;

poi.ITDESC = v.ITDESC;

poi.QTY = v.QTY;

poi.SUPLNAME = v.SUPLNAME;

poi.SUPLNO = v.SUPLNO;

poi.ITCODE = v.ITCODE;

poi.PODATE = v.PODATE;

poinformation.Add(poi);

}

return poinformation;

}

}

[HttpGet]

public HttpResponseMessage Get(string PoNumber)

{

using (PODbEntities1 PE = new PODbEntities1())

{

PE.Configuration.ProxyCreationEnabled = false;

IEnumerable<POMASTER> pomasters = PE.POMASTERs.ToList();

IEnumerable<PODETAIL> podetails = PE.PODETAILs.ToList();

IEnumerable<ITEM> items = PE.ITEMs.ToList();

IEnumerable<SUPPLIER> suppliers = PE.SUPPLIERs.ToList();

List<POInformation> poinformation = new List<POInformation>();

var value = from p in pomasters

join q in podetails on p.PONO equals q.PONO

join i in items on q.ITCODE equals i.ITCODE

join s in suppliers on p.SUPLNO equals s.SUPLNO

where p.PONO == PoNumber

select new { p.PONO, p.PODATE, p.SUPLNO, s.SUPLNAME, q.ITCODE, i.ITDESC, q.QTY };

foreach (var v in value)

{

POInformation poi = new POInformation();

poi.PONO = v.PONO;

poi.ITDESC = v.ITDESC;

poi.QTY = v.QTY;

poi.SUPLNAME = v.SUPLNAME;

poi.SUPLNO = v.SUPLNO;

poi.ITCODE = v.ITCODE;

poi.PODATE = v.PODATE;

poinformation.Add(poi);

}

if (poinformation.Count > 0)

{

return Request.CreateResponse(HttpStatusCode.OK, poinformation);

}

else

{

return Request.CreateErrorResponse(HttpStatusCode.NotFound, "POMASTER Number : " + PoNumber + " Not Found");

}

}

}

public HttpResponseMessage Post([FromBody] List<POInformation> i)

{

try

{

using (PODbEntities1 PE = new PODbEntities1())

{

PE.Configuration.ProxyCreationEnabled = false;

// convert PO Information to PO Master and PO Details

foreach (POInformation pi in i.GroupBy(x => x.PONO).Select(x => x.FirstOrDefault()))

{

POMASTER po = new POMASTER();

po.PONO = pi.PONO;

po.SUPLNO = pi.SUPLNO;

po.PODATE = DateTime.Now;

PE.POMASTERs.Add(po);

}

foreach (POInformation pi in i)

{

PODETAIL pod = new PODETAIL();

pod.ITCODE = pi.ITCODE;

pod.QTY = pi.QTY;

pod.PONO = pi.PONO;

PE.PODETAILs.Add(pod);

}

PE.SaveChanges();

HttpResponseMessage Message = Request.CreateResponse(HttpStatusCode.Created, i);

// Message.Headers.Location = new Uri(Request.RequestUri + i.First().PONO);

return Message;

}

}

catch (Exception ex)

{

return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);

}

}

public HttpResponseMessage Put([FromBody] List<POInformation> i)

{

try

{

using (PODbEntities1 PE = new PODbEntities1())

{

PE.Configuration.ProxyCreationEnabled = false;

// convert PO Information to PO Master and PO Details

foreach (POInformation pi in i.GroupBy(x => x.PONO).Select(x => x.FirstOrDefault()))

{

POMASTER po = PE.POMASTERs.Where(x => x.PONO == pi.PONO).FirstOrDefault();

po.SUPLNO = pi.SUPLNO;

po.PODATE = DateTime.Now;

PE.SaveChanges();

}

foreach (POInformation pi in i)

{

PODETAIL pod = PE.PODETAILs.Where(x => x.PONO == pi.PONO && pi.ITCODE == x.ITCODE).FirstOrDefault();

pod.QTY = pi.QTY;

PE.SaveChanges();

}

PE.SaveChanges();

HttpResponseMessage Message = Request.CreateResponse(HttpStatusCode.OK, i);

// Message.Headers.Location = new Uri(Request.RequestUri + i.First().PONO);

return Message;

}

}

catch (Exception ex)

{

return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);

}

}

public HttpResponseMessage Delete(string POMASTERCode)

{

HttpResponseMessage httpResponse = null;

try

{

using (PODbEntities1 PE = new PODbEntities1())

{

PE.Configuration.ProxyCreationEnabled = false;

IEnumerable<PODETAIL> p = PE.PODETAILs.Where(x => x.PONO == POMASTERCode);

foreach (PODETAIL value in p)

{

PE.Entry(value).State = System.Data.Entity.EntityState.Deleted;

}

POMASTER I = PE.POMASTERs.Where(x => x.PONO == POMASTERCode).FirstOrDefault();

if (I == null)

{

httpResponse = Request.CreateErrorResponse(HttpStatusCode.NotFound, "POMASTER with Code " + POMASTERCode + " Not found");

}

else

{

PE.Entry(I).State = System.Data.Entity.EntityState.Deleted;

PE.SaveChanges();

httpResponse = Request.CreateResponse(HttpStatusCode.OK);

}

}

return httpResponse;

}

catch (Exception ex)

{

return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);

}

}

}

}

Supplier Controller

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

using WebAPIDemo.Models;

namespace WebAPIDemo.Controllers

{

public class SupplierController : ApiController

{

public IEnumerable<SUPPLIER> Get()

{

using (PODbEntities1 PE = new PODbEntities1())

{

PE.Configuration.ProxyCreationEnabled = false;

return PE.SUPPLIERs.ToList();

}

}

[HttpGet]

public HttpResponseMessage Get(string SUPPLIERNumber)

{

using (PODbEntities1 PE = new PODbEntities1())

{

PE.Configuration.ProxyCreationEnabled = false;

SUPPLIER i = PE.SUPPLIERs.Where(x => x.SUPLNO == SUPPLIERNumber).FirstOrDefault();

if (i != null)

{

return Request.CreateResponse(HttpStatusCode.OK, i);

}

else

{

return Request.CreateErrorResponse(HttpStatusCode.NotFound, "SUPPLIER Number : " + SUPPLIERNumber + " Not Found");

}

}

}

public HttpResponseMessage Post([FromBody] SUPPLIER i)

{

try

{

using (PODbEntities1 PE = new PODbEntities1())

{

PE.Configuration.ProxyCreationEnabled = false;

PE.SUPPLIERs.Add(i);

PE.SaveChanges();

HttpResponseMessage Message = Request.CreateResponse(HttpStatusCode.Created, i);

// Message.Headers.Location = new Uri(Request.RequestUri + i.SUPLNO);

return Message;

}

}

catch (Exception ex)

{

return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);

}

}

public HttpResponseMessage Put([FromBody] SUPPLIER i)

{

try

{

using (PODbEntities1 PE = new PODbEntities1())

{

PE.Configuration.ProxyCreationEnabled = false;

SUPPLIER value = PE.SUPPLIERs.Where(x => x.SUPLNO == i.SUPLNO).FirstOrDefault();

value.SUPLNAME = i.SUPLNAME;

value.SUPLADDR = i.SUPLADDR;

PE.SaveChanges();

HttpResponseMessage Message = Request.CreateResponse(HttpStatusCode.OK, value);

// Message.Headers.Location = new Uri(Request.RequestUri + value.SUPLNO);

return Message;

}

}

catch (Exception ex)

{

return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);

}

}

public HttpResponseMessage Delete(string SUPPLIERCode)

{

HttpResponseMessage httpResponse = null;

try

{

using (PODbEntities1 PE = new PODbEntities1())

{

PE.Configuration.ProxyCreationEnabled = false;

SUPPLIER I = PE.SUPPLIERs.Where(x => x.SUPLNO == SUPPLIERCode).FirstOrDefault();

if (I == null)

{

httpResponse = Request.CreateErrorResponse(HttpStatusCode.NotFound, "SUPPLIER with Code " + SUPPLIERCode + " Not found");

}

else

{

PE.Entry(I).State = System.Data.Entity.EntityState.Deleted;

PE.SaveChanges();

httpResponse = Request.CreateResponse(HttpStatusCode.OK);

}

}

return httpResponse;

}

catch (Exception ex)

{

return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);

}

}

}

}

POInformation

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace WebAPIDemo.Models

{

public class POInformation

{

public string PONO { get; set; }

public string ITCODE { get; set; }

public Nullable<int> QTY { get; set; }

public DateTime PODATE { get; set; }

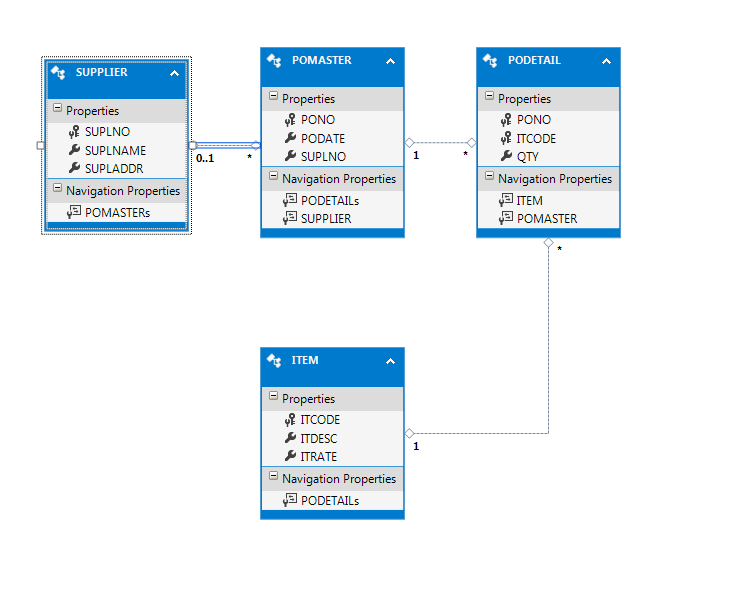
public string SUPLNO { get; set; }

public string SUPLNAME { get; set; }

public string ITDESC { get; set; }

}

}



Test Project

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net.Http;

using System.Text;

using System.Web.Http;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using WebAPIDemo;

using WebAPIDemo.Controllers;

using Moq;

using WebAPIDemo.Models;

using System.Net;

namespace WebAPIDemo.Tests.Controllers

{

[TestClass]

public class ItemControllerTest

{

[TestMethod]

public void GetBySpecificId()

{

// Set up Prerequisites

var controller = new ItemController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Get("ITM1");

// Assert the result

ITEM item;

Assert.IsTrue(response.TryGetContentValue<ITEM>(out item));

Assert.AreEqual("Lux", item.ITDESC);

}

[TestMethod]

public void GetAllData()

{

// Set up Prerequisites

var controller = new ItemController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Get();

// Assert the result

Assert.IsTrue(response.Count() > 0);

}

[TestMethod]

public void GetErrorResponseCheck()

{

// Set up Prerequisites

var controller = new ItemController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Get("ITM9");

// Assert the result

Assert.IsTrue(!response.IsSuccessStatusCode);

}

[TestMethod]

public void GetResponseCheck()

{

// Set up Prerequisites

var controller = new ItemController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Get("ITM1");

// Assert the result

Assert.IsTrue(response.IsSuccessStatusCode);

}

[TestMethod]

public void GetNotFoundResponseCheck()

{

// Set up Prerequisites

var controller = new ItemController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Get("ITM112");

// Assert the result

Assert.IsTrue(response.StatusCode == HttpStatusCode.NotFound);

}

[TestMethod]

public void PostBadRequest()

{

// Set up Prerequisites

var controller = new ItemController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

ITEM i=new ITEM();

i.ITCODE="ITM8";

i.ITDESC="500ML Bottle";

i.ITRATE=60;

i.PODETAILs = null;

// Act on Test

var response = controller.Post(new ITEM { ITCODE = "ITM8", ITDESC = "500ML Bottle", ITRATE= 78 });

// Assert the result

Assert.IsTrue(response.StatusCode == HttpStatusCode.BadRequest);

// Assert.AreEqual(i.ITRATE, item.ITRATE);

}

[TestMethod]

public void PostRequest()

{

// Set up Prerequisites

var controller = new ItemController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Post(new ITEM { ITCODE = "ITM9", ITDESC = "1L Bottle", ITRATE = 138 });

// Assert the result

Assert.IsTrue(response.StatusCode == HttpStatusCode.Created);

// Assert.AreEqual(i.ITRATE, item.ITRATE);

}

[TestMethod]

public void PutRequest()

{

// Set up Prerequisites

var controller = new ItemController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Put(new ITEM { ITCODE = "ITM9", ITDESC = "1Ltr Bottle", ITRATE = 148 });

// Assert the result

Assert.IsTrue(response.StatusCode == HttpStatusCode.OK);

}

[TestMethod]

public void PutBadRequest()

{

// Set up Prerequisites

var controller = new ItemController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Put(new ITEM { ITCODE = "ITM10", ITDESC = "1Ltr Bottle Out of Bound Exception", ITRATE = 148 });

// Assert the result

Assert.IsTrue(response.StatusCode == HttpStatusCode.BadRequest);

}

[TestMethod]

public void DeleteNotFoundRequest()

{

// Set up Prerequisites

var controller = new ItemController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Delete ( "IT09" );

// Assert the result

Assert.IsTrue(response.StatusCode == HttpStatusCode.NotFound);

}

[TestMethod]

public void DeleteRequest()

{

// Set up Prerequisites

var controller = new ItemController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Delete("ITM9");

// Assert the result

Assert.IsTrue(response.StatusCode == HttpStatusCode.OK);

}

}

}

PO Master Test Class

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Text;

using System.Threading.Tasks;

using System.Web.Http;

using WebAPIDemo.Controllers;

using WebAPIDemo.Models;

namespace WebAPIDemo.Tests.Controllers

{

[TestClass]

public class POMasterControllerTest

{

[TestMethod]

public void GetBySpecificId()

{

// Set up Prerequisites

var controller = new POMasterController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Get("PO1");

// Assert the result

Assert.IsNotNull(response.Content);

}

[TestMethod]

public void GetAllData()

{

// Set up Prerequisites

var controller = new POMasterController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Get();

// Assert the result

Assert.IsTrue(response.Count() > 0);

}

[TestMethod]

public void GetErrorResponseCheck()

{

// Set up Prerequisites

var controller = new POMasterController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Get("uC1");

// Assert the result

Assert.IsTrue(!response.IsSuccessStatusCode);

}

[TestMethod]

public void GetResponseCheck()

{

// Set up Prerequisites

var controller = new POMasterController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Get("PO1 ");

// Assert the result

Assert.IsTrue(response.IsSuccessStatusCode);

}

[TestMethod]

public void GetNotFoundResponseCheck()

{

// Set up Prerequisites

var controller = new POMasterController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Get("SUP1");

// Assert the result

Assert.IsTrue(response.StatusCode == HttpStatusCode.NotFound);

}

[TestMethod]

public void PostBadRequest()

{

// Set up Prerequisites

var controller = new POMasterController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

List<POInformation> poi = new List<POInformation>();

poi.Add(new POInformation { PONO = "PO8", PODATE = DateTime.Now, SUPLNO = "C102", ITCODE = "ITM6", QTY = 2, ITDESC = null, SUPLNAME = null });

var response = controller.Post(poi);

// Assert the result

Assert.IsTrue(response.StatusCode == HttpStatusCode.BadRequest);

// Assert.AreEqual(i.ITRATE, item.ITRATE);

}

[TestMethod]

public void PostRequest()

{

// Set up Prerequisites

var controller = new POMasterController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

List<POInformation> poi = new List<POInformation>();

poi.Add(new POInformation { PONO = "PO8 ", PODATE = DateTime.Now, SUPLNO = "C2", ITCODE = "ITM6", QTY = 2, ITDESC = null, SUPLNAME = null });

var response = controller.Post(poi);

// Assert the result

Assert.IsTrue(response.StatusCode == HttpStatusCode.Created);

// Assert.AreEqual(i.ITRATE, item.ITRATE);

}

[TestMethod]

public void PutRequest()

{

// Set up Prerequisites

var controller = new POMasterController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

List<POInformation> poi = new List<POInformation>();

poi.Add(new POInformation { PONO = "PO8 ", PODATE = DateTime.Now, SUPLNO = "C3", ITCODE = "ITM6", QTY = 2, ITDESC = null, SUPLNAME = null });

var response = controller.Put(poi);

// Assert the result

Assert.IsTrue(response.StatusCode == HttpStatusCode.OK);

}

[TestMethod]

public void PutBadRequest()

{

// Set up Prerequisites

var controller = new POMasterController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

List<POInformation> poi = new List<POInformation>();

poi.Add(new POInformation { PONO = "PO8", PODATE = DateTime.Now, SUPLNO = "C35", ITCODE = "ITM6", QTY = 2, ITDESC = null, SUPLNAME = null });

var response = controller.Put(poi);

// Assert the result

Assert.IsTrue(response.StatusCode == HttpStatusCode.BadRequest);

}

[TestMethod]

public void DeleteNotFoundRequest()

{

// Set up Prerequisites

var controller = new POMasterController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Delete("IT09");

// Assert the result

Assert.IsTrue(response.StatusCode == HttpStatusCode.NotFound);

}

[TestMethod]

public void DeleteRequest()

{

// Set up Prerequisites

var controller = new POMasterController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Delete("PO8");

// Assert the result

Assert.IsTrue(response.StatusCode == HttpStatusCode.OK);

}

}

}

Supplier Test Class

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Text;

using System.Threading.Tasks;

using System.Web.Http;

using WebAPIDemo.Controllers;

using WebAPIDemo.Models;

namespace WebAPIDemo.Tests.Controllers

{

[TestClass]

public class SupplierControllerTest

{

[TestMethod]

public void GetBySpecificId()

{

// Set up Prerequisites

var controller = new SupplierController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Get("C1");

// Assert the result

SUPPLIER item;

Assert.IsTrue(response.TryGetContentValue<SUPPLIER>(out item));

Assert.AreEqual("Chain Merchant", item.SUPLNAME);

}

[TestMethod]

public void GetAllData()

{

// Set up Prerequisites

var controller = new SupplierController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Get();

// Assert the result

Assert.IsTrue(response.Count() > 0);

}

[TestMethod]

public void GetErrorResponseCheck()

{

// Set up Prerequisites

var controller = new SupplierController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Get("uC1");

// Assert the result

Assert.IsTrue(!response.IsSuccessStatusCode);

}

[TestMethod]

public void GetResponseCheck()

{

// Set up Prerequisites

var controller = new SupplierController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Get("C1");

// Assert the result

Assert.IsTrue(response.IsSuccessStatusCode);

}

[TestMethod]

public void GetNotFoundResponseCheck()

{

// Set up Prerequisites

var controller = new SupplierController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Get("SUP1");

// Assert the result

Assert.IsTrue(response.StatusCode == HttpStatusCode.NotFound);

}

[TestMethod]

public void PostBadRequest()

{

// Set up Prerequisites

var controller = new SupplierController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Post(new SUPPLIER { SUPLNO = "C10", SUPLNAME = "Glitter & Corporation Office", SUPLADDR = "Glitter Street, Tambaram" });

// Assert the result

Assert.IsTrue(response.StatusCode == HttpStatusCode.BadRequest);

// Assert.AreEqual(i.ITRATE, item.ITRATE);

}

[TestMethod]

public void PostRequest()

{

// Set up Prerequisites

var controller = new SupplierController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Post(new SUPPLIER { SUPLNO = "C10", SUPLNAME = "Glitter & CO", SUPLADDR = "Street,TBM" });

// Assert the result

Assert.IsTrue(response.StatusCode == HttpStatusCode.Created);

// Assert.AreEqual(i.ITRATE, item.ITRATE);

}

[TestMethod]

public void PutRequest()

{

// Set up Prerequisites

var controller = new SupplierController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Put(new SUPPLIER { SUPLNO = "C10", SUPLNAME = "Gliter & CO", SUPLADDR = "Glitter Street" });

// Assert the result

Assert.IsTrue(response.StatusCode == HttpStatusCode.OK);

}

[TestMethod]

public void PutBadRequest()

{

// Set up Prerequisites

var controller = new SupplierController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Put(new SUPPLIER { SUPLNO = "C10", SUPLNAME = "Gliter & CO Exceeds Maximum Limit", SUPLADDR = "Glitter Street" });

// Assert the result

Assert.IsTrue(response.StatusCode == HttpStatusCode.BadRequest);

}

[TestMethod]

public void DeleteNotFoundRequest()

{

// Set up Prerequisites

var controller = new SupplierController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Delete("IT09");

// Assert the result

Assert.IsTrue(response.StatusCode == HttpStatusCode.NotFound);

}

[TestMethod]

public void DeleteRequest()

{

// Set up Prerequisites

var controller = new SupplierController();

controller.Request = new HttpRequestMessage();

controller.Configuration = new HttpConfiguration();

// Act on Test

var response = controller.Delete("C10");

// Assert the result

Assert.IsTrue(response.StatusCode == HttpStatusCode.OK);

}

}

}