Lab: Testing and Debugging ASP.NET MVC  
4 Web Applications  
**Exercise 1: Performing Unit Tests**~~υTask 1: Create a test project.  
1. In the result pane of the Hyper-V Manager console, in the~~ **~~Name~~** ~~list of the~~ **~~Virtual Machines~~** ~~area,  
right-click~~ **~~20486B-SEA-DEV11~~**~~, and then click~~ **~~Connect~~**~~.  
2. To log on to~~ **~~20486B-SEA-DEV11~~**~~, click the~~ **~~Ctrl+Alt+Delete~~**~~button.  
3. In the~~ **~~User name~~** ~~box, type~~ **~~Admin~~**~~, in the~~ **~~Password~~** ~~box, type~~ **~~Pa$$w0rd~~**~~, and then click the~~**~~Forward~~** ~~button.  
4. On the Windows 8 Start screen, click~~ **~~Desktop~~**~~.  
5. On the taskbar, click the~~ **~~File Explorer~~** ~~icon.  
6. In the~~ **~~Libraries~~** ~~window, navigate to~~ **~~Allfiles~~**  
**Open (D):\Labfiles\Mod06\Starter\PhotoSharingApplication**, and then double-click  
**PhotoSharingApplication.sln**.

**Abrir proyecto o solucion del lab de vistas**

7. In the Solution Explorer pane of the **PhotoSharingApplication - Microsoft Visual Studio** window,  
right-click **Solution ‘Photo Sharing Application’ (1 project)**, click **Add**, and then click **New Project**.  
8. In the navigation pane of the **Add New Project** dialog box, click **Visual C#**, click **Test**, and then, in  
the result pane, click **Unit Test Project**.  
9. In the **Name** box, type **PhotoSharingTests**, and then click **OK**.  
10. In the Solution Explorer pane of the **PhotoSharingApplication - Microsoft Visual Studio** window,  
under PhotoSharingTests, right-click **References**, and then click **Add Reference**.  
11. In the navigation pane of the **Reference Manager - PhotoSharingTests**window, click **Solution**, and  
then click **Projects**.  
12. In the result pane, select the check box corresponding to **PhotoSharingApplication**, and then click  
**OK**.  
13. In the Solution Explorer pane of the **PhotoSharingApplication - Microsoft Visual Studio** window,  
under PhotoSharingTests, right-click **References**, and then click **Add Reference**.  
14. In the navigation pane of the **Reference Manager - PhotoSharingTests**window, click **~~Assemblies~~**~~,  
and then click~~ **~~Extensions~~**~~.~~

(**buscar la version de MVC que usa el proyecto de Photosharing desde la BIN)**15. In the result pane, select the check box corresponding to **System.Web.Mvc** and then click **OK**.  
**υTask 2: Write the tests.**1. In the Solution Explorer pane of the **PhotoSharingApplication - Microsoft Visual Studio** window,  
under PhotoSharingTests, right-click **UnitTest1.cs**, click **Rename**, type **PhotoControllerTests**, and  
then press Enter.  
2. In the **Microsoft Visual Studio** dialog box, click **Yes**.  
3. In the **PhotoControllerTests.cs**code window, locate the following code.public void TestMethod1()  
4. Replace the code with the following code.  
public void Test\_Index\_Return\_View()  
5. Place the mouse cursor at the end of the Microsoft.VisualStudio.TestTools.UnitTesting namespace,  
press Enter, and then type the following code.  
using System.Collections.Generic;  
using System.Web.Mvc;  
using PhotoSharingApplication.Models;  
using PhotoSharingApplication.Controllers;  
6. Place the mouse cursor in the **Test\_Index\_Return\_View**test method code block, and then type the  
following code.  
PhotoController controller =  
new PhotoController();  
var result = controller.Index()  
as ViewResult;  
Assert.AreEqual("Index",  
result.ViewName);  
7. Place the mouse cursor at the end of the **Test\_Index\_Return\_View**test method code block, press  
Enter twice, and then type the following code.  
[TestMethod]  
public void Test\_PhotoGallery\_Model\_Type()  
{  
}  
8. Place the mouse cursor in the **Test\_PhotoGallery\_Model\_Type**test method code block, and then  
type the following code.  
var controller = new PhotoController();  
var result = controller.\_PhotoGallery() as PartialViewResult;  
Assert.AreEqual(typeof(List<Photo>), result.Model.GetType());  
9. Place the mouse cursor at the end of the **Test\_PhotoGallery\_Model\_Type**test method code block,  
press Enter twice, and then type the following code.  
[TestMethod]  
public void Test\_GetImage\_Return\_Type()  
{  
}  
10. Place the mouse cursor in the **Test\_GetImage\_Return\_Type**test method code block, and then type  
the following code.  
var controller = new PhotoController();  
var result = controller.GetImage(1) as ActionResult;  
Assert.AreEqual(typeof(FileContentResult), result.GetType());  
11. On the TEST menu of the PhotoSharingApplication - Microsoft Visual Studio window, point to Run,  
and then click All Tests.  
12. In the Test Explorer pane, expand **Passed Tests (1)**, and then note that only the  
**Test\_Index\_Return\_View**test is passed.

13. Under Failed Tests (2), note that the **Test\_GetImage\_Return\_Type**and the  
**Test\_PhotoGallery\_Model\_Type**tests are failed.  
14. Under Failed Tests (2), click **Test\_GetImage\_Return\_Type**, and then note that the test failed while  
trying to connect to the database.  
15. Under Failed Tests (2), click **Test\_PhotoGallery\_Model\_Type**, and then note that the test failed while  
trying to connect to the database.  
16. In the Test Explorer pane, click the **Close** button.  
**υTask 3: Implement a repository.**1. In the Solution Explorer pane of the **PhotoSharingApplication - Microsoft Visual Studio** window,  
expand PhotoSharingApplication.  
2. In the Solution Explorer pane of the **PhotoSharingApplication - Microsoft Visual Studio** window,  
right-click **Models**, click **Add**, and then click **New Item**.  
3. In the navigation pane of the **Add New Item - PhotoSharingApplication**dialog box, click **Code**,  
and then, in the result pane, click **Interface**.  
4. In the **Name** box, type **IPhotoSharingContext**, and then click **Add**.  
5. In the IPhotoSharingContext.cs code window, locate the following code.  
interface IPhotoSharingContext  
6. Replace the code with the following code.  
public interface IPhotoSharingContext  
7. Place the mouse cursor in the **IPhotoSharingContext**interface code block, press Enter, and then  
type the following code.  
IQueryable<Photo> Photos { get; }  
8. Place the mouse cursor at the end of the **Photos** property code block, press Enter, and then type the  
following code.  
IQueryable<Comment> Comments { get; }  
9. Place the mouse cursor at the end of the **Comments** property code block, press Enter, and then type  
the following code.  
int SaveChanges();  
10. Place the mouse cursor at the end of the **SaveChanges**method code block, press Enter, and then  
type the following code.  
T Add<T>(T entity) where T : class;  
11. Place the mouse cursor at the end of the **Add** method code block, press Enter, and then type the  
following code:  
Photo FindPhotoById (int ID) ;  
12. Place the mouse cursor at the end of the **FindPhotoById**method code block, press Enter, and then  
type the following code.

Comment FindCommentById (int ID) ;  
13. Place the mouse cursor at the end of the **FindCommentById**method code block, press Enter, and  
then type the following code.  
T Delete<T>(T entity) where T : class;  
14. In the Solution Explorer pane, click **PhotoSharingContext.cs**.  
15. In the PhotoSharingContext.cs code window, locate the following code.  
public class PhotoSharingContext : DbContext  
16. Append the following code to the existing line of code.  
,IPhotoSharingContext  
17. Place the mouse cursor at the end of the **Comments DbSet**property code block, press Enter twice,  
and then type the following code.  
IQueryable<Photo>IPhotoSharingContext.Photos  
{  
get { return Photos; }  
}  
18. Place the mouse cursor at the end of the **Photos** property code block, press Enter twice, and then  
type the following code.  
IQueryable<Comment>IPhotoSharingContext.Comments  
{  
get { return Comments; }  
}  
19. Place the mouse cursor at the end of the **Comments** property code block, press Enter twice, and then  
type the following code.  
intIPhotoSharingContext.SaveChanges ()  
{  
return SaveChanges();  
}  
20. Place the mouse cursor at the end of the **SaveChanges**method code block, press Enter twice, and  
then type the following code.  
T IPhotoSharingContext.Add<T>(T entity)  
{  
return Set<T>().Add(entity);  
}  
21. Place the mouse cursor at the end of the **Add** method code block, press Enter twice, and then type  
the following code.  
Photo IPhotoSharingContext.FindPhotoById(int ID)  
{  
return Set<Photo>().Find(ID);  
}  
22. Place the mouse cursor at the end of the **FindPhotoById**code block, press Enter twice, and then type  
the following code.

Comment IPhotoSharingContext.FindCommentById(int ID)  
{  
return Set<Comment>().Find(ID);  
}  
23. Place the mouse cursor at the end of the **FindCommentById**method code block, press Enter twice,  
and then type the following code.  
T IPhotoSharingContext.Delete<T>(T entity)  
{  
return Set<T>().Remove(entity);  
}  
24. On the **FILE** menu of the **PhotoSharingApplication - Microsoft Visual Studio** window, click **Save**  
**All**.  
25. On the **BUILD** menu, click **Build Solution**.  
υTask 4: Refactor the photo controller to use the repository.  
1. In the Solution Explorer pane of the **PhotoSharingApplication - Microsoft Visual Studio** window,  
expand **Controllers**, and then click **PhotoController.cs**.  
2. In the PhotoController.cs code window, locate the following code.  
privatePhotoSharingContext context = new PhotoSharingContext();  
3. Replace the code with the following code.  
privateIPhotoSharingContext context;  
4. Place the mouse cursor at the end of the **context** object code block, press Enter twice, and then type  
the following code.  
publicPhotoController ()  
{  
context = new PhotoSharingContext();  
}  
5. Place the mouse cursor at the end of the **PhotoController**constructor code block, press Enter twice,  
and then type the following code.  
publicPhotoController (IPhotoSharingContext Context)  
{  
context = Context;  
}  
6. In the PhotoController.cs code window, locate the **Display** action method, and then locate the  
following code.  
Photo photo = context.Photos.Find(id);  
7. Replace the code with the following code.  
Photo photo = context.FindPhotoById(id);  
8. Locate the **Create** action method for the POST verb, and then locate the following code.  
context.Photos.Add(photo);  
9. Replacethecodewiththefollowingcode.

context.Add<Photo>(photo);  
10. Locate the **Delete** action method, and then locate the following code.  
Photo photo = context.Photos.Find(id);  
11. Replace the code with the following code.  
Photo photo = context.FindPhotoById(id);  
12. Locate the **DeleteConfirmed**action method, and then locate the following code.  
Photo photo = context.Photos.Find(id);  
13. Replace the code with the following code.  
Photo photo = context.FindPhotoById (id);  
14. In the **DeleteConfirmed**action method, locate the following code.  
context.Photos.Remove(photo);  
15. Replace the code with the following code.  
context.Delete<Photo>(photo);  
16. Locate the **GetImage**action method, and then locate the following code.  
Photo photo = context.Photos.Find(id);  
17. Replace the code with the following code.  
Photo photo = context.FindPhotoById(id);  
18. On the **DEBUG** menu of the **PhotoSharingApplication - Microsoft Visual Studio** window, click  
**Start Debugging**.  
19. On the Welcome to Adventure Works Photo Sharing page, click the **Display** link corresponding to  
any photo of your choice.  
20. On the Sample Photo 5 page, note that the changes are displayed, and then, in the Windows Internet  
Explorer window, click the **Close** button.  
υTask 5: Refactor the tests to use a mock repository.  
1. In the Solution Explorer pane of the **PhotoSharingApplication - Microsoft Visual Studio** window,  
right-click **PhotoSharingTests**, click **Add**, and then click **New Folder**.  
2. In the **NewFolder1** box, type **Doubles**, and then press Enter.  
3. In the Solution Explorer pane, right-click **Doubles**, click **Add**, and then click **Existing Item**.  
4. In the **Add Existing Item - PhotoSharingTests**dialog box, navigate to Allfiles  
(**D):\Labfiles\Mod06\Fake Repository\ FakePhotoSharingContext.cs**, and then click **Add**.  
5. In the Solution Explorer pane, click **PhotoControllerTests.cs**.  
6. In the PhotoControllerTests.cs code window, place the mouse cursor at the end of the  
PhotoSharingApplication.Contollers namespace, press Enter, and then type the following code.  
using System.Linq;

using PhotoSharingTests.Doubles;  
7. Locate the **Test\_Index\_Return\_View**test method, and then locate the following code.  
PhotoController = new PhotoController();  
8. Replace the code with the following code.  
var context = new FakePhotoSharingContext();  
var controller = new PhotoController(context);  
9. Locate the **Test\_PhotoGallery\_Model\_Type**test method, place the mouse cursor at the beginning of  
the **Test\_PhotoGallery\_Model\_Type**test method code block, press Enter, and then type the  
following code.  
var context = new FakePhotoSharingContext();  
context.Photos = new [] {  
new Photo(),  
new Photo(),  
new Photo(),  
new Photo()  
}.AsQueryable();  
var controller = new PhotoController(context);  
10. In the **Test\_PhotoGallery\_Model\_Type**test method, locate the following code, select the located  
code, and then press Delete.  
var controller = new PhotoController();  
11. Locate the **Test\_GetImage\_Return\_Type**test method, place the mouse cursor at the beginning of  
the **Test\_GetImage\_Return\_Type**method, press Enter, and then type the following code.  
var context = new FakePhotoSharingContext();  
12. Place the mouse cursor at the end of the code you just typed, press Enter twice, and then type the  
following code.  
context.Photos = new [] {  
new Photo{ PhotoID = 1, PhotoFile = new byte[1], ImageMimeType = "image/jpeg" },  
new Photo{ PhotoID = 2, PhotoFile = new byte[1], ImageMimeType = "image/jpeg" },  
new Photo{ PhotoID = 3, PhotoFile = new byte[1], ImageMimeType = "image/jpeg" },  
new Photo{ PhotoID = 4, PhotoFile = new byte[1], ImageMimeType = "image/jpeg" }  
}.AsQueryable();  
13. In the **Test\_GetImage\_Return\_Type**method code block, locate the following code.  
var controller = new PhotoController();  
14. Replace the code with the following code.  
var controller = new PhotoController(context);  
15. On the **TEST** menu of the **PhotoSharingApplication - Microsoft Visual Studio** window, click **Run**,  
and then click **All Tests**.  
16. In the Test Explorer pane, note that all tests have passed.

**Note:** All the tests passed because the FakePhotoSharingContext test double can return  
test values without connecting to a database. A test double or mock object is an object used in  
test projects that behaves like the corresponding object in the web application project.  
17. In the Test Explorer pane, click the **Close** button.  
**υTask 6: Add further tests.**1. In the Solution Explorer pane of the **PhotoSharingApplication - Microsoft Visual Studio** window,  
click **PhotoControllerTests.cs**.  
2. In the PhotoControllerTests.cs code window, place the mouse cursor at the end of the  
**PhotoControllerTests**class, press Enter twice, and then type the following code.  
[TestMethod]  
public void Test\_PhotoGallery\_No\_Parameter ()  
{  
}  
3. In the **Test\_PhotoGallery\_No\_Parameter**test method code block, type the following code, and then  
press Enter twice.  
var context = new FakePhotoSharingContext();  
context.Photos = new [] {  
new Photo(),  
new Photo(),  
new Photo(),  
new Photo()  
}.AsQueryable();  
var controller = new PhotoController(context);  
4. In the **Test\_PhotoGallery\_No\_Parameter**test method code block, type the following code, and then  
press Enter.  
var result = controller.\_PhotoGallery() as PartialViewResult;  
5. In the **Test\_PhotoGallery\_No\_Parameter**test method code block, type the following code.  
varmodelPhotos = (IEnumerable<Photo>)result.Model;  
Assert.AreEqual(4, modelPhotos.Count());  
6. Select all the code in the **Test\_PhotoGallery\_No\_Parameter**method, including the **[TestMethod]**  
annotation.  
7. On the **Edit** menu of the **PhotoSharingApplication - Microsoft Visual Studio** window, click **Copy**.  
8. Place the mouse cursor at the end of the **Test\_PhotoGallery\_No\_Parameter**method.  
9. On the **Edit** menu, click **Paste**.  
10. Locate the following code in the code block you just pasted.  
public void Test\_PhotoGallery\_No\_Parameter()  
11. Replace the code with the following line of code.  
public void Test\_PhotoGallery\_Int\_Parameter()  
12. In the **Test\_PhotoGallery\_Int\_Parameter**method, locate the following code.

var result = controller.\_PhotoGallery() as PartialViewResult;  
13. In the **Test\_PhotoGallery\_Int\_Parameter**method, replace the located code with the following code.  
var result = controller.\_PhotoGallery(3) as PartialViewResult;  
14. In the **Test\_PhotoGallery\_Int\_Parameter**, method, locate the following code.  
varmodelPhotos = (IEnumerable<Photo>)result.Model;  
Assert.AreEqual(4, modelPhotos.Count());  
15. In the **Test\_PhotoGallery\_Int\_Parameter**, method, replace the located code with the following code.  
varmodelPhotos = (IEnumerable<Photo>)result.Model;  
Assert.AreEqual(3, modelPhotos.Count());  
16. On the **TEST** menu of the **PhotoSharingApplication -Microsoft Visual studio** window, click **TEST**,  
point to **Run**, and then click **All Tests**.  
17. In the Test Explorer pane, note that all tests passed.  
18. In the Test Explorer pane, click the **Close** button.  
**Results**: After completing this exercise, you will be able to add a set of PhotoController tests defined in  
the PhotoSharingTests project of the Photo Sharing application.

**Interface: IPhotoSharingContext**

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Text;

usingSystem.Threading.Tasks;

namespace PhotoSharingApplication.Models

{

public **interface** IPhotoSharingContext

{

IQueryable<Photo> Photos { get; }

IQueryable<Comment> Comments { get; }

int SaveChanges();

T Add<T>(T entity) where T : class;

Photo FindPhotoById(int ID);

Comment FindCommentById(int ID);

T Delete<T>(T entity) where T : class;

}

}

**Creación de FakePhotoSharingContext**

using System;

usingSystem.Collections.Generic;

usingSystem.Collections.ObjectModel;

usingSystem.Linq;

usingSystem.Text;

usingSystem.Threading.Tasks;

usingPhotoSharingApplication.Models;

namespace PhotoSharingTests.Doubles

{

class FakePhotoSharingContext : IPhotoSharingContext

{

//This object is a keyed collection we use to mock an

//entity framework context in memory

**SetMap \_map = new SetMap();**

publicIQueryable<Photo> Photos

{

get { return \_map.Get<Photo>().AsQueryable(); }

set { \_map.Use<Photo>(value); }

}

public IQueryable<Comment> Comments

{

get { return \_map.Get<Comment>().AsQueryable(); }

set { \_map.Use<Comment>(value); }

}

public bool ChangesSaved { get; set; }

public int SaveChanges()

{

ChangesSaved = true;

return 0;

}

public T Add<T>(T entity) where T : class

{

\_map.Get<T>().Add(entity);

return entity;

}

public Photo FindPhotoById(int ID)

{

Photo item = (from p in this.Photos

wherep.PhotoID == ID

select p).First();

return item;

}

public Comment FindCommentById(int ID)

{

Comment item = (from c in this.Comments

wherec.CommentID == ID

select c).First();

return item;

}

public T Delete<T>(T entity) where T : class

{

\_map.Get<T>().Remove(entity);

return entity;

}

classSetMap : KeyedCollection<Type, object>

{

public HashSet<T> Use<T>(IEnumerable<T>sourceData)

{

var set = new HashSet<T>(sourceData);

if (Contains(typeof(T)))

{

Remove(typeof(T));

}

Add(set);

return set;

}

publicHashSet<T> Get<T>()

{

return (HashSet<T>) this[typeof(T)];

}

protected override Type GetKeyForItem(object item)

{

returnitem.GetType().GetGenericArguments().Single();

}

}

}

}

**El código debería quedar así: PhotoControllerTests**

using System;

usingMicrosoft.VisualStudio.TestTools.UnitTesting;

usingSystem.Web.Mvc;

usingPhotoSharingApplication.Controllers;

usingPhotoSharingApplication.Models;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingPhotoSharingTests.Doubles;

namespacePhotoSharingTests

{

**[TestClass]**

public class **PhotoControllerTests**

{

**[TestMethod]**

public void Test\_Index\_Return\_View()

{

//This test checks that the PhotoController Index action returns the Index View

var context = new FakePhotoSharingContext();

var controller = new PhotoController(context);

var result = controller.Index() as ViewResult;

**Assert.AreEqual("Index", result.ViewName);**

}

**[TestMethod]**

public void Test\_PhotoGallery\_Model\_Type()

{

//This test checks that the PhotoController \_PhotoGallery action passes a list of Photos to the view

var context = new FakePhotoSharingContext();

context.Photos = new[] {

new Photo(),

new Photo(),

new Photo(),

new Photo()

}.AsQueryable();

var controller = new PhotoController(context);

var result = controller.\_PhotoGallery() as PartialViewResult;

Assert.AreEqual(typeof(List<Photo>), result.Model.GetType());

}

[TestMethod]

public void Test\_GetImage\_Return\_Type()

{

//This test checks that the PhotoControllerGetImage action returns a FileResult

var context = new FakePhotoSharingContext();

context.Photos = new[] {

new Photo{ PhotoID = 1, PhotoFile = new byte[1], ImageMimeType = "image/jpeg" },

new Photo{ PhotoID = 2, PhotoFile = new byte[1], ImageMimeType = "image/jpeg" },

new Photo{ PhotoID = 3, PhotoFile = new byte[1], ImageMimeType = "image/jpeg" },

new Photo{ PhotoID = 4, PhotoFile = new byte[1], ImageMimeType = "image/jpeg" }

}.AsQueryable();

var controller = new PhotoController(context);

var result = controller.GetImage(1) as ActionResult;

Assert.AreEqual(typeof(FileContentResult), result.GetType());

}

[TestMethod]

public void Test\_PhotoGallery\_No\_Parameter()

{

//This test checks that, when you call the \_PhotoGallery action with no

//parameter, all the photos in the context are returned

var context = new FakePhotoSharingContext();

context.Photos = new[] {

new Photo(),

new Photo(),

new Photo(),

new Photo()

}.AsQueryable();

var controller = new PhotoController(context);

var result = controller.\_PhotoGallery() as PartialViewResult;

varmodelPhotos = (IEnumerable<Photo>)result.Model;

Assert.AreEqual(4, modelPhotos.Count());

}

[TestMethod]

public void Test\_PhotoGallery\_Int\_Parameter()

{

//This test checks that, when you call the \_PhotoGallery action with no

//parameter, all the photos in the context are returned

var context = new FakePhotoSharingContext();

context.Photos = new[] {

new Photo(),

new Photo(),

new Photo(),

new Photo()

}.AsQueryable();

var controller = new PhotoController(context);

var result = controller.\_PhotoGallery(3) as PartialViewResult;

varmodelPhotos = (IEnumerable<Photo>)result.Model;

Assert.AreEqual(3, modelPhotos.Count());

}

}

}

**Modificación de PhotoSharingContext**

using System;

usingSystem.Collections.Generic;

usingSystem.Data.Entity;

usingSystem.Linq;

usingSystem.Web;

namespacePhotoSharingApplication.Models

{

public class PhotoSharingContext : DbContext, IPhotoSharingContext

{

publicDbSet<Photo> Photos { get; set; }

publicDbSet<Comment> Comments { get; set; }

IQueryable<Photo>IPhotoSharingContext.Photos

{

get { return Photos; }

}

IQueryable<Comment>IPhotoSharingContext.Comments

{

get { return Comments; }

}

intIPhotoSharingContext.SaveChanges()

{

returnSaveChanges();

}

T IPhotoSharingContext.Add<T>(T entity)

{

return Set<T>().Add(entity);

}

Photo IPhotoSharingContext.FindPhotoById(int ID)

{

return Set<Photo>().Find(ID);

}

Comment IPhotoSharingContext.FindCommentById(int ID)

{

return Set<Comment>().Find(ID);

}

T IPhotoSharingContext.Delete<T>(T entity)

{

return Set<T>().Remove(entity);

}

}

}

**Modificación de controlador: PhotoController**

usingSystem;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Web;

usingSystem.Web.Mvc;

usingSystem.Globalization;

usingPhotoSharingApplication.Models;

namespacePhotoSharingApplication.Controllers

{

[ValueReporter]

public class PhotoController : Controller

{

privateIPhotoSharingContext context;

//Constructors

publicPhotoController()

{

context = new PhotoSharingContext();

}

publicPhotoController(IPhotoSharingContext Context)

{

context = Context;

}

//

// GET: /Photo/

publicActionResult Index()

{

return View("Index");

}

[ChildActionOnly]

publicActionResult \_PhotoGallery(int number = 0)

{

List<Photo> photos;

if (number == 0)

{

photos = context.Photos.ToList();

}

else

{

photos = (from p in context.Photos

orderbyp.CreatedDate descending

select p).Take(number).ToList();

}

returnPartialView("\_PhotoGallery", photos);

}

publicActionResult Display(int id)

{

Photo photo = context.FindPhotoById(id);

//Photo photo = context.Photos.Find(id);

if (photo == null)

{

returnHttpNotFound();

}

return View("Display", photo);

}

publicActionResult Create()

{

Photo newPhoto = new Photo();

newPhoto.CreatedDate = DateTime.Today;

return View("Create", newPhoto);

}

[HttpPost]

publicActionResult Create(Photo photo, HttpPostedFileBase image)

{

photo.CreatedDate = DateTime.Today;

if (!ModelState.IsValid)

{

return View("Create", photo);

}

else

{

if (image != null)

{

photo.ImageMimeType = image.ContentType;

photo.PhotoFile = new byte[image.ContentLength];

image.InputStream.Read(photo.PhotoFile, 0, image.ContentLength);

}

context.Add<Photo>(photo);

//context.Photos.Add(photo);

context.SaveChanges();

returnRedirectToAction("Index");

}

}

publicActionResult Delete(int id)

{

Photo photo = context.FindPhotoById(id);

//Photo photo = context.Photos.Find(id);

if (photo == null)

{

returnHttpNotFound();

}

return View("Delete", photo);

}

[HttpPost]

[ActionName("Delete")]

publicActionResultDeleteConfirmed(int id)

{

Photo photo = context.FindPhotoById(id);

//Photo photo = context.Photos.Find(id);

context.Delete<Photo>(photo);

//context.Photos.Remove(photo);

context.SaveChanges();

returnRedirectToAction("Index");

}

publicFileContentResultGetImage(int id)

{

Photo photo = context.FindPhotoById(id);

//Photo photo = context.Photos.Find(id);

if (photo != null)

{

return File(photo.PhotoFile, photo.ImageMimeType);

}

else

{

return null;

}

}

publicActionResultSlideShow()

{

throw new NotImplementedException("The Slideshow action is not yet ready.");

}

}

}