Azure Class Exercise

1. Create a first MVC 4.5 application
2. Tweak it
3. Deploy to MS Azure

Tweak it Summary

1. Add a new 'time' page
    a. Display time and date
    b. Add link in the header
2. Add contact form on 'contact' page
    a. Form with:
        i. Name
        ii. Email
        iii. Message
    b. Add validation
    c. Saves to db, of course
3. Add a 'messages' page
    a. Display all the received messages
    b. Possibility for edit/ delete them
4. Cleanup
    a.

How we do it

1. Use default sample templates that MVC offers
2. Copy/ paste or write your own.
3. Modify the code to suit your needs

Exercise

First go on Windows Azure and follow the first 4 steps of the tutorial. When getting to the database section, we will make it a bit different instead of following the tutorial.

Set up the development environment

Create a web site and a SQL database in Windows Azure

Create an ASP.NET MVC 4 application

Deploy the application to Windows Azure

Add a database to the application

Add data model classes for the to-do list

Follow all the steps, but instead of the Todo example at 3 and 4, create one model class like bellow.

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Data.Entity;
```

```
namespace MvcSampleApp.Models
{
    public class MessageContext : DbContext
    {
        public DbSet<Message> Messages { get; set; }
    }

    public class Message
    {
        [Key]
        [DatabaseGeneratedAttribute(DatabaseGeneratedOption.Identity)]
        public int MessageId { get; set; }

        [Required]
        [Display(Name = "Your name")]
        [DataType(DataType.Text)]
        public string Name { get; set; }

        [Required]
        [Display(Name = "Your email")]
        [DataType(DataType.EmailAddress)]
        public string Email { get; set; }

        [Required]
        [Display(Name = "The message")]
        [DataType(DataType.MultilineText)]
        public string Message { get; set; }
    }
}
```

Enable Migrations and create the database

At number 2 of this step use the following instead what you see in the tutorial:

```
enable-migrations -ContextTypeName MvcSampleApp.Models.MessageContext

add-migration Initial

update-database
```

Create web pages that enable app users to work with to-do list items

Create a `MessageController` instead of `HomeController` controller, but we will use this only for inspiration. We are going to use the generated code to modify the existing pages.

Use code from `Views\Message\Create.cshtml` and modify `Views\Home\Contact.cshtml` so it looks like this:

```
@model MvcSampleApp.Models.Message

@{
    ViewBag.Title = "Contact";
}

<hgroup class="title">
    <h1>@ViewBag.Title.</h1>
    <h2>@ViewBag.Message</h2>
```

```
</hgroup>

<section class="contact">

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>Message</legend>

        <div class="editor-label">
            @Html.LabelFor(model => model.Name)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Name)
            @Html.ValidationMessageFor(model => model.Name)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.Email)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Email)
            @Html.ValidationMessageFor(model => model.Email)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.MessageText)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.MessageText)
            @Html.ValidationMessageFor(model => model.MessageText)
        </div>

        <p>
            <input type="submit" value="Create" />
        </p>
    </fieldset>
}
</section>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

This replaces the contact details on the contact page with a contact form. Next there is some restructuring:

1. Delete `Views\Message\Create.cshtml`, because we already moved its functionality.
2. Rename the `Views\Message\Index.cshtml` into `Views\Message\Messages.cshtml`.
3. Move the content of `Views\Message\` into `Views\Home\`.
4. Delete `Views\Message\` folder.
5. Now we have all the `Message` into `Views\Home\`.

Because we made some changes in the Views, we now need to make some changes in the Controller as well. We will modify the `HomeController`. We will use the generated code from the `MessageController`. The `HomeController` now looks like this:

```
using MvcSampleApp.Models;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MvcSampleApp.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            ViewBag.Message = "Modify this template to jump-start your ASP.NET MVC
application.";

            return View();
        }

        private MessageContext db = new MessageContext();

        //
        // GET: /Message/

        public ActionResult Messages()
        {
            return View(db.Messages.ToList());
        }

        //
        // GET: /Message/Details/5

        public ActionResult Details(int id = 0)
        {
            Message message = db.Messages.Find(id);
            if (message == null)
            {
                return HttpNotFound();
            }
            return View(message);
        }

        //
        // GET: /Message/Create

        public ActionResult Contact()
        {
            return View();
        }

        //
        // POST: /Message/Create
```

```
[HttpPost]
public ActionResult Contact(Message message)
{
    if (ModelState.IsValid)
    {
        db.Messages.Add(message);
        db.SaveChanges();
        return RedirectToAction("Index");
    }

    return View(message);
}

//
// GET: /Message/Edit/5

public ActionResult Edit(int id = 0)
{
    Message message = db.Messages.Find(id);
    if (message == null)
    {
        return HttpNotFound();
    }
    return View(message);
}

//
// POST: /Message/Edit/5

[HttpPost]
public ActionResult Edit(Message message)
{
    if (ModelState.IsValid)
    {
        db.Entry(message).State = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    return View(message);
}

//
// GET: /Message/Delete/5

public ActionResult Delete(int id = 0)
{
    Message message = db.Messages.Find(id);
    if (message == null)
    {
        return HttpNotFound();
    }
    return View(message);
}

//
// POST: /Message/Delete/5
```

```
        [HttpPost, ActionName("Delete")]
        public ActionResult DeleteConfirmed(int id)
        {
            Message message = db.Messages.Find(id);
            db.Messages.Remove(message);
            db.SaveChanges();
            return RedirectToAction("Messages");
        }

        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

Now, the left bits and cleanup

1. Remove the About page and link from the homepage
2. Add Messages link next to the remaining ones
3. Create the Time page and add link in the header
4. Edit header, featured, content and footer on the frontpage.

Deploy the application update to Windows Azure and SQL Database

Follow the steps.