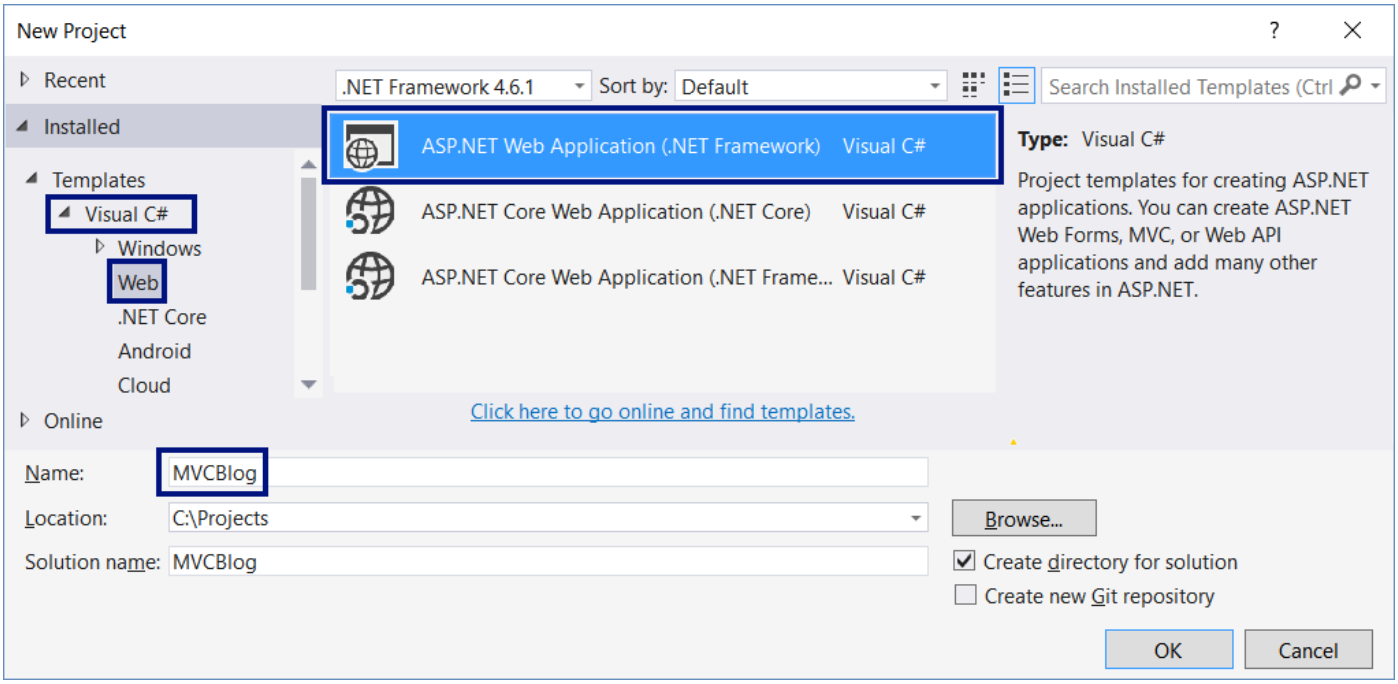
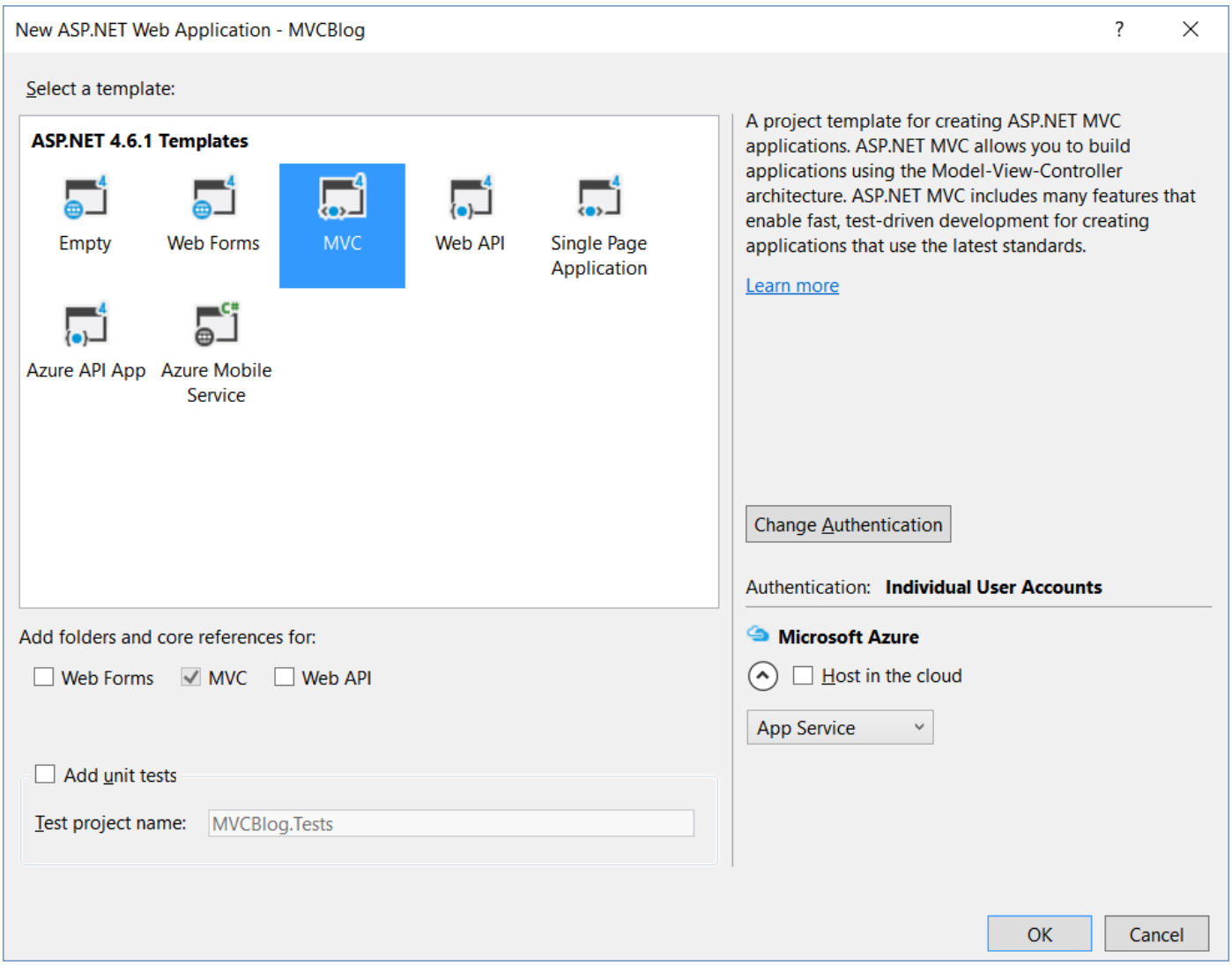
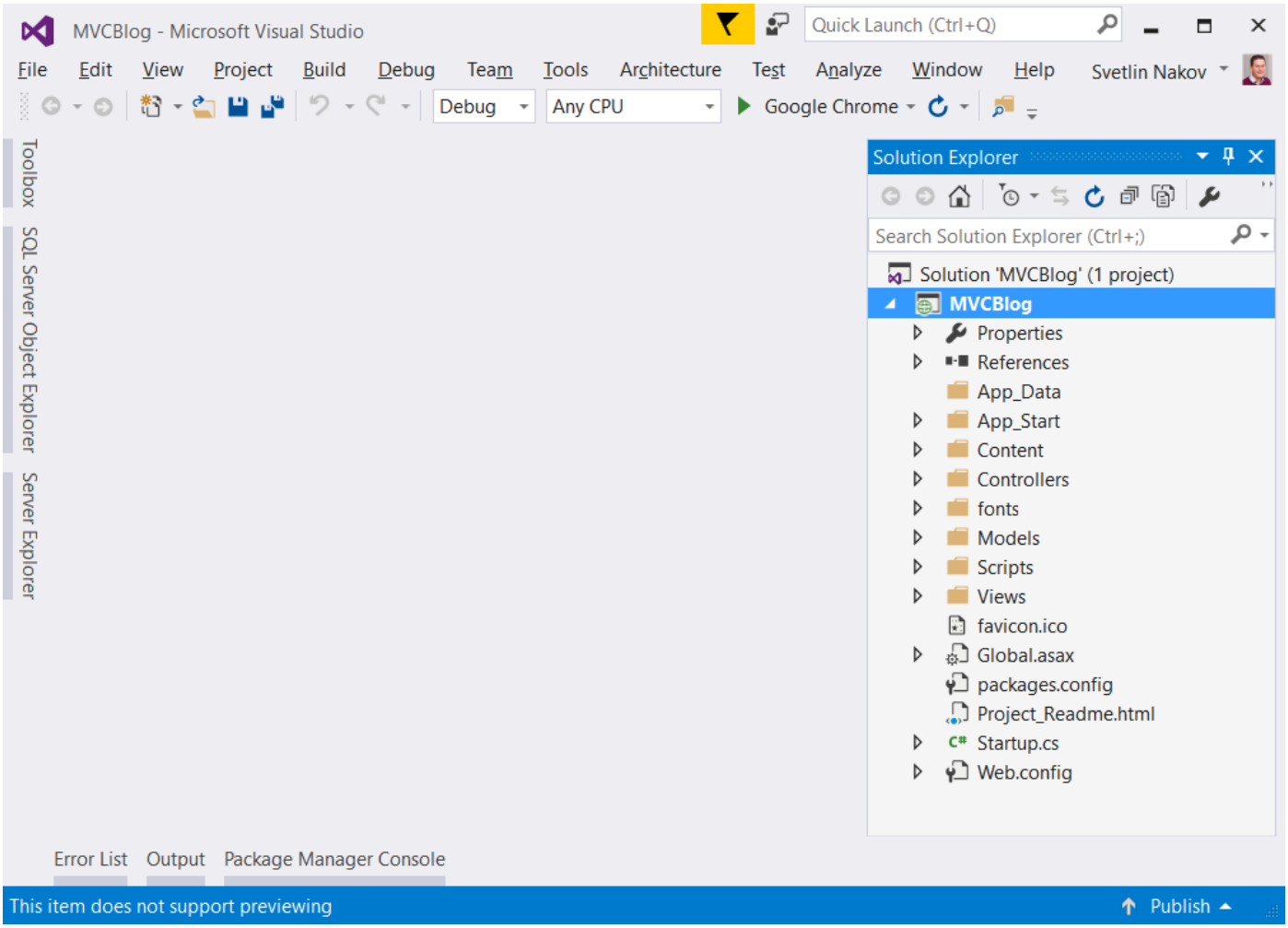
Lab: Creating a Blog with ASP.NET MVC + EF + SQL Server

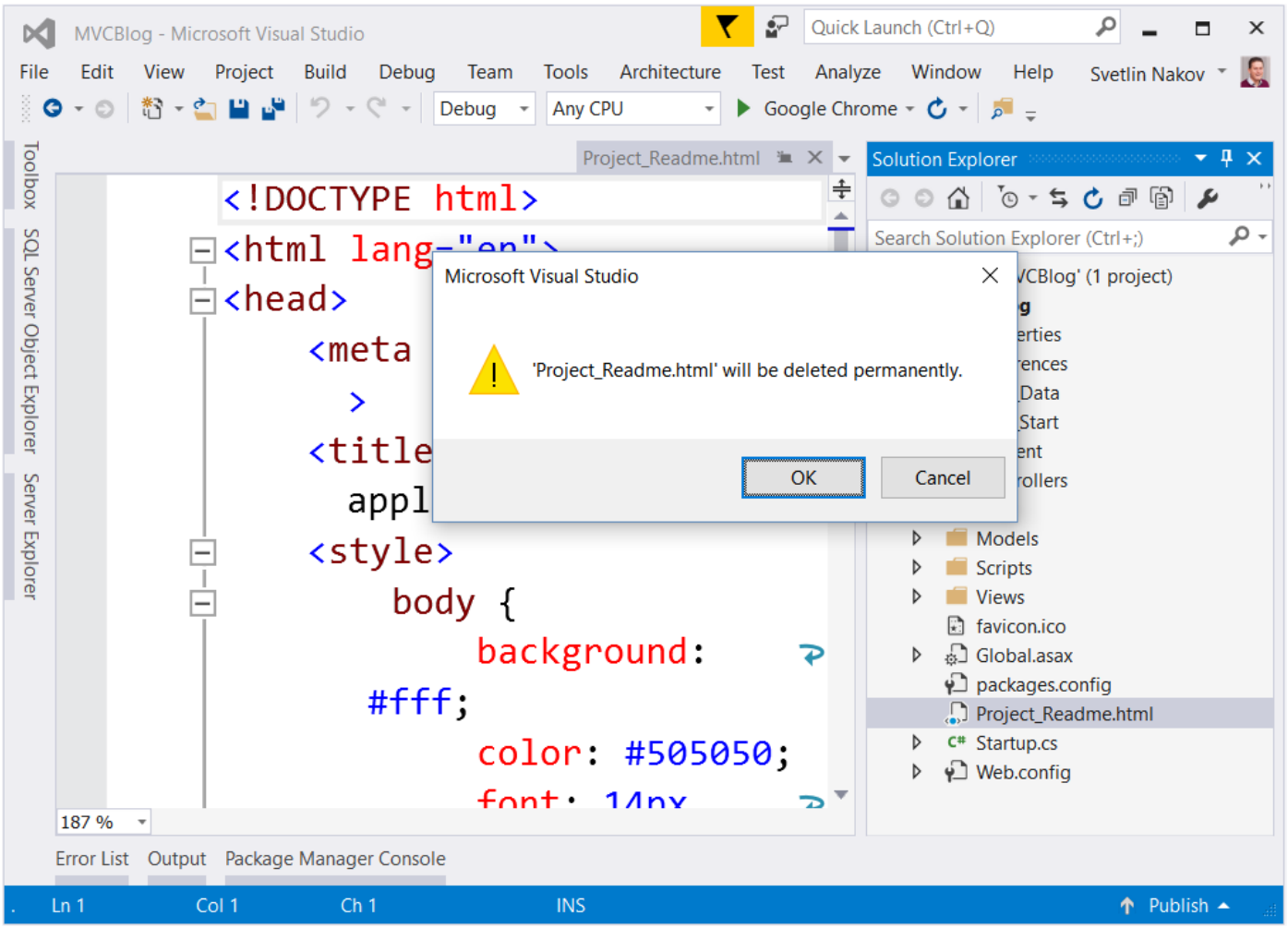
In this lab we shall create a fully-functional **Blog system** in ASP.NET MVC with SQL Server database using Entity Framework and MVC Scaffolding.

1. Create a New ASP.NET MVC Application



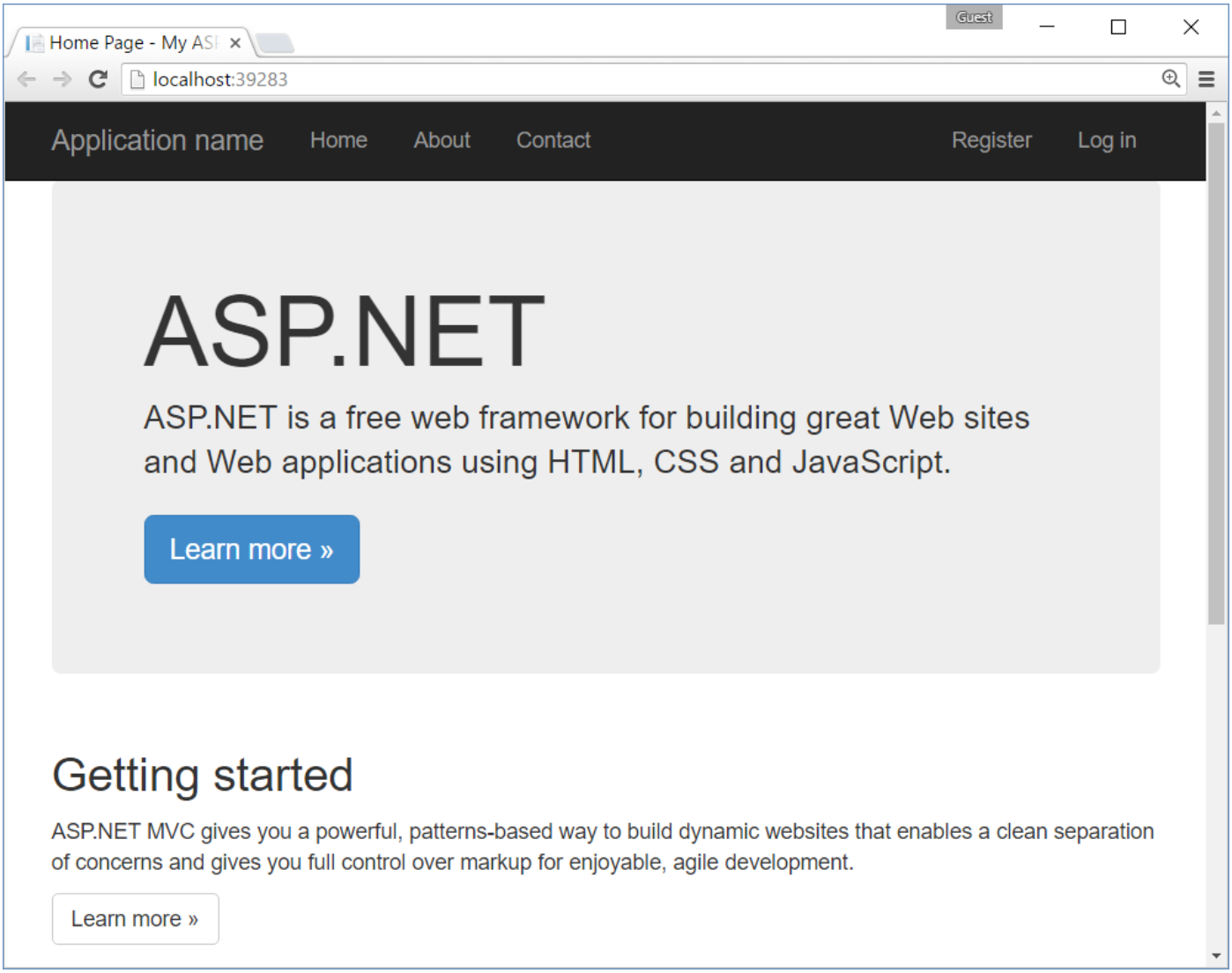

New ASP.NET Web Application - MVCBlog
Select a template:
ASP.NET 4.6.1 Templates
Empty
Web Forms
MVC
Web API
Single Page Application
Azure API App
Azure Mobile Service
A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.
Learn more
Change Authentication
Authentication: Individual User Accounts
Microsoft Azure
Host in the cloud
App Service
Add folders and core references for:
Web Forms
MVC
Web API
Add unit tests
Test project name: MVCBlog.Tests
OK
Cancel


MVCBlog - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help
Svetlin Nakov
Debug
Any CPU
Google Chrome
Toolbox
SQL Server Object Explorer
Server Explorer
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MVCBlog' (1 project)
MVCBlog
Properties
References
App_Data
App_Start
Content
Controllers
fonts
Models
Scripts
Views
favicon.ico
Global.asax
packages.config
Project_Readme.html
Startup.cs
Web.config
Error List
Output
Package Manager Console
This item does not support previewing
Publish



2. Run the Application

Run the application to see what was generated by the Visual Studio MVC application template. Press [Ctrl+F5].



3. Customize the App Layout

_Layout.cshtml*

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - MVC Blog</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
```

_Layout.cshtml*

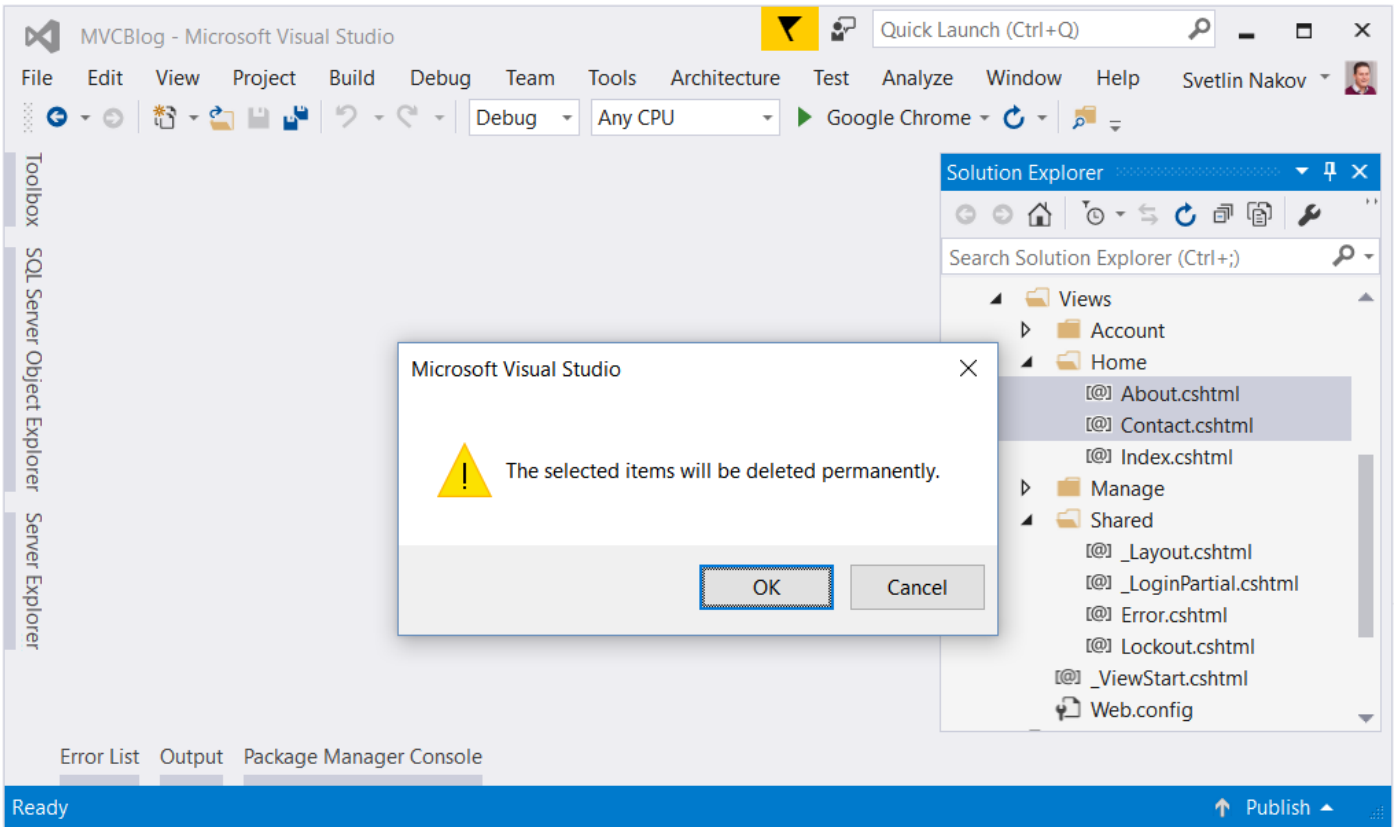
```
                </button>
                @Html.ActionLink("MVC Blog", "Index", "Home", new
                 { area = "" }, new { @class = "navbar-brand" })
            </div>
```

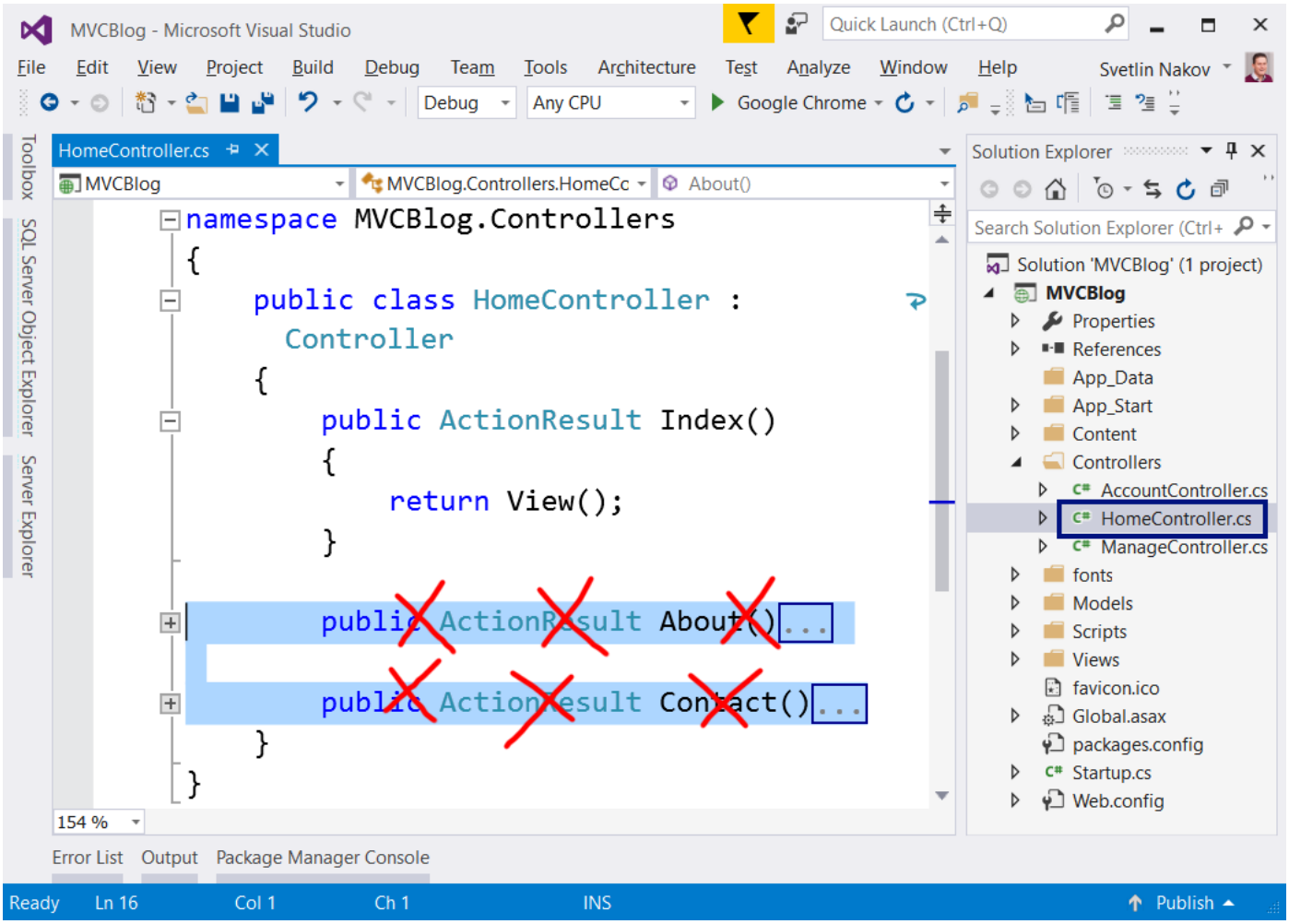
_Layout.cshtml*

```
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Home", "Index", "Home")</li>
                    <li>@Html.ActionLink("Posts", "Index", "Posts")</li>
                    <li>@Html.ActionLink("New Post", "Create", "Posts")</li>
                </ul>
```

_Layout.cshtml*

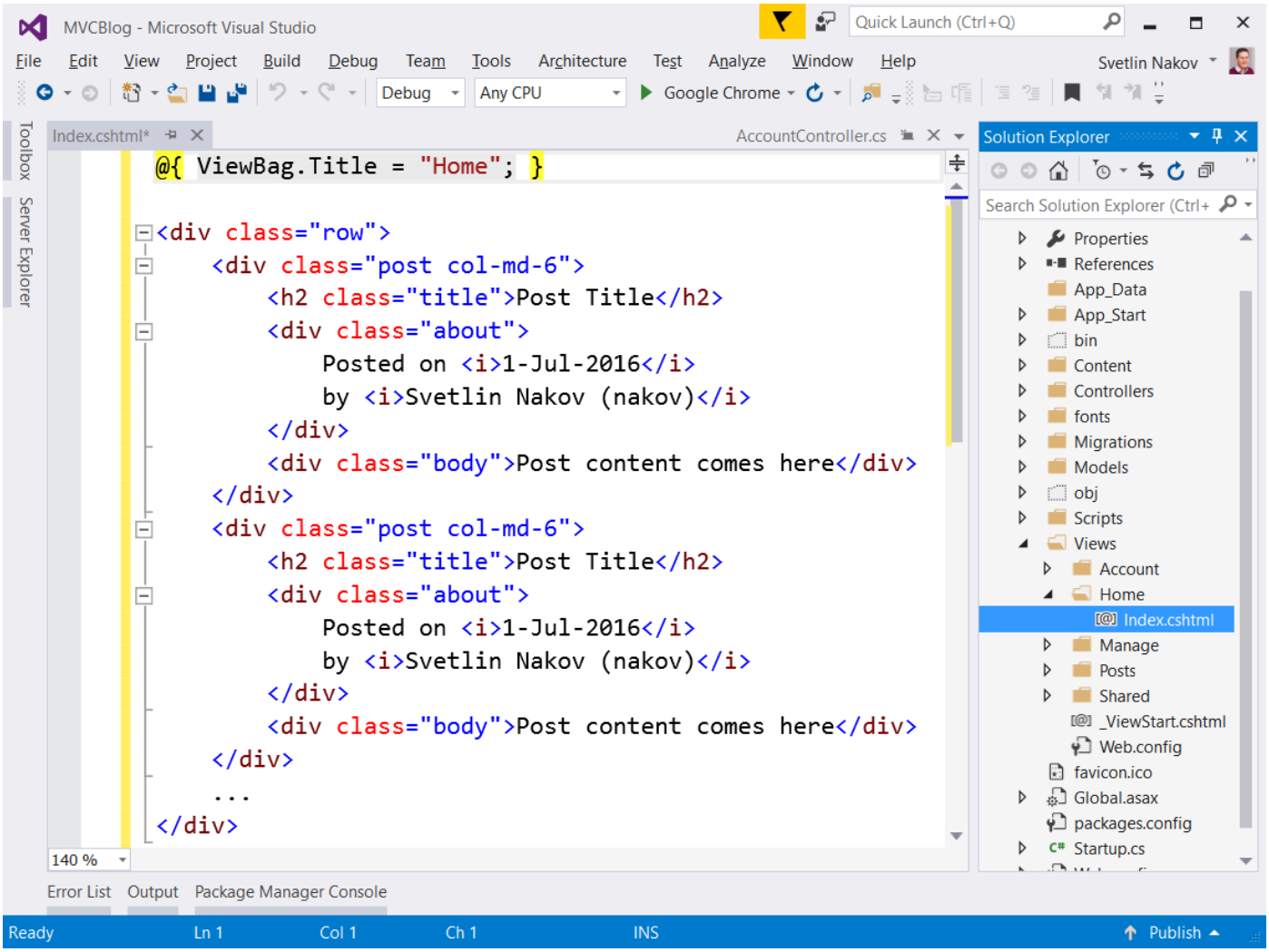
```
            <footer>
                <p>&copy; @DateTime.Now.Year - MVC Blog</p>
            </footer>
        </div>
```





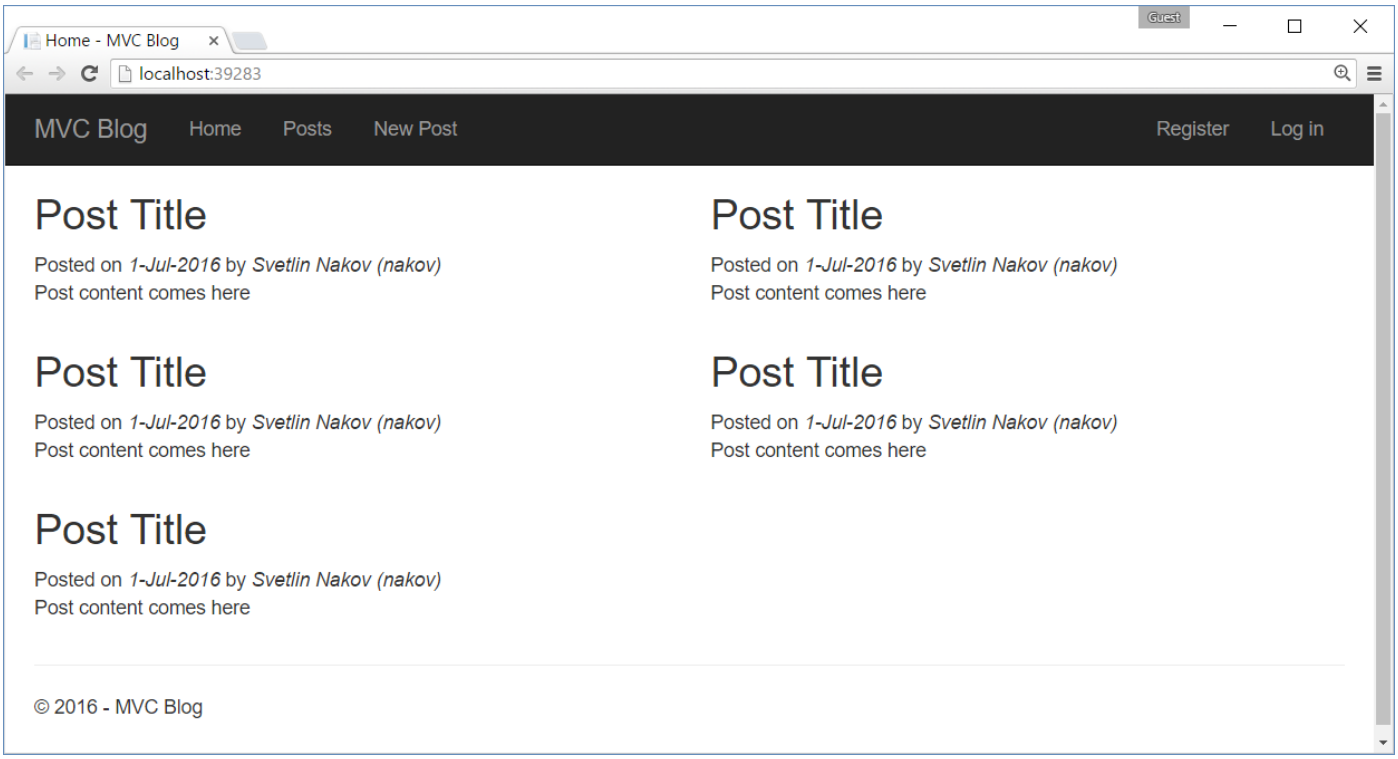
4. Create the Posts View at the Home Page

TODO: <section> / <article>

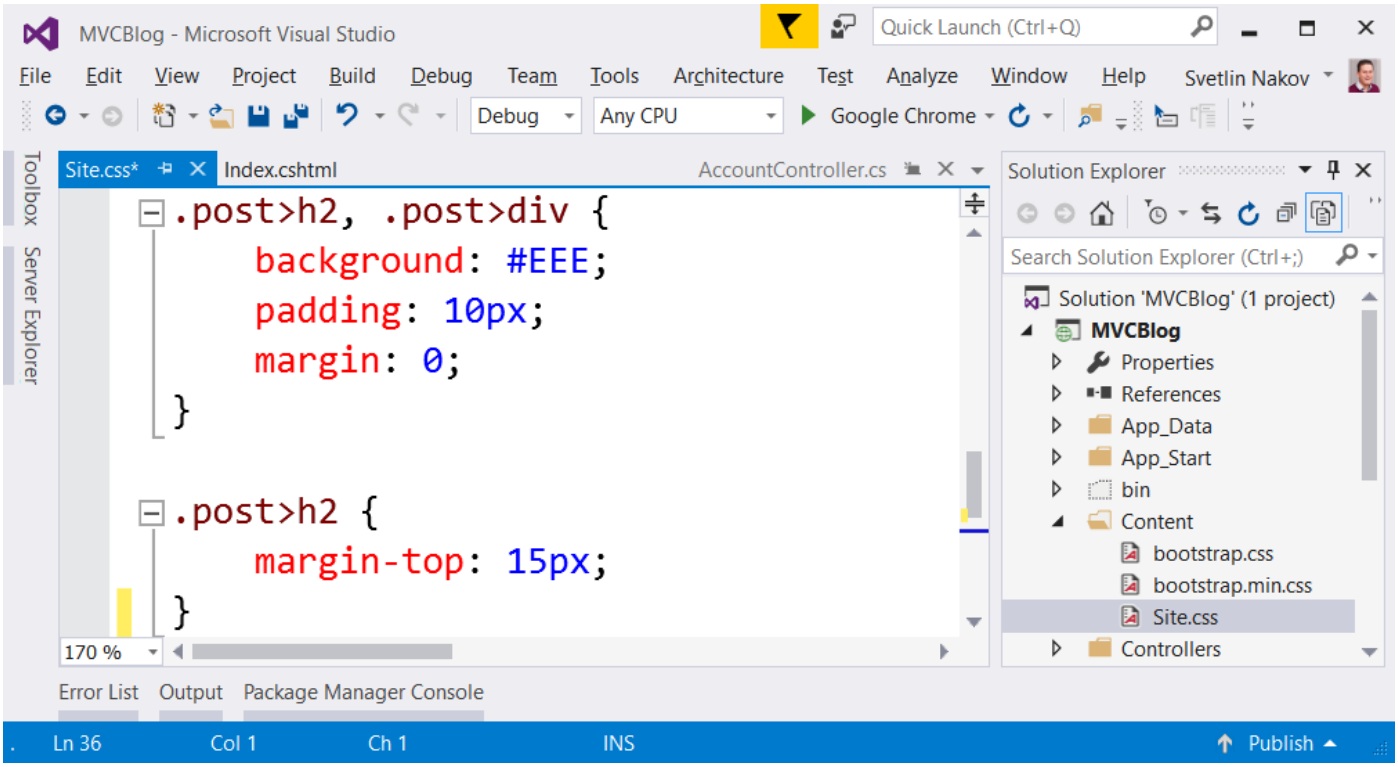


Run the App

Run the application with [Ctrl+F5] to see the results:

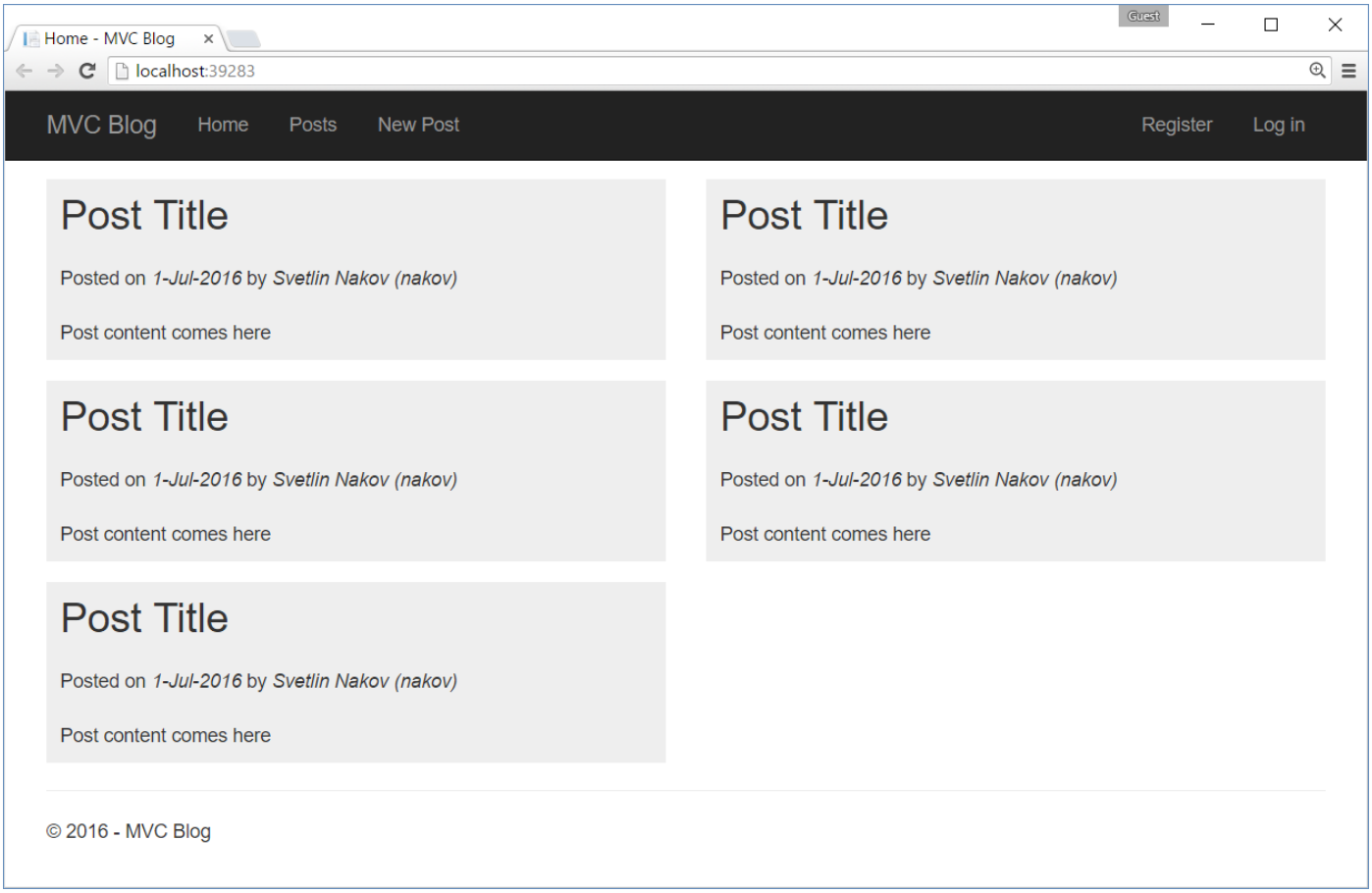


Customize the CSS



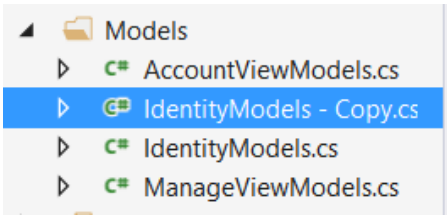
Run the App

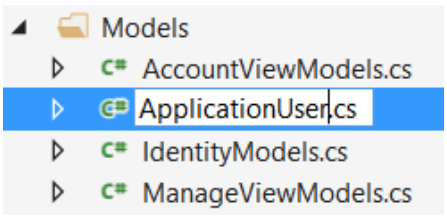
Run the app to see the CSS effects:

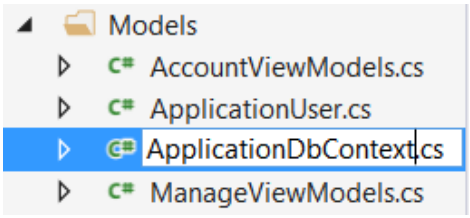


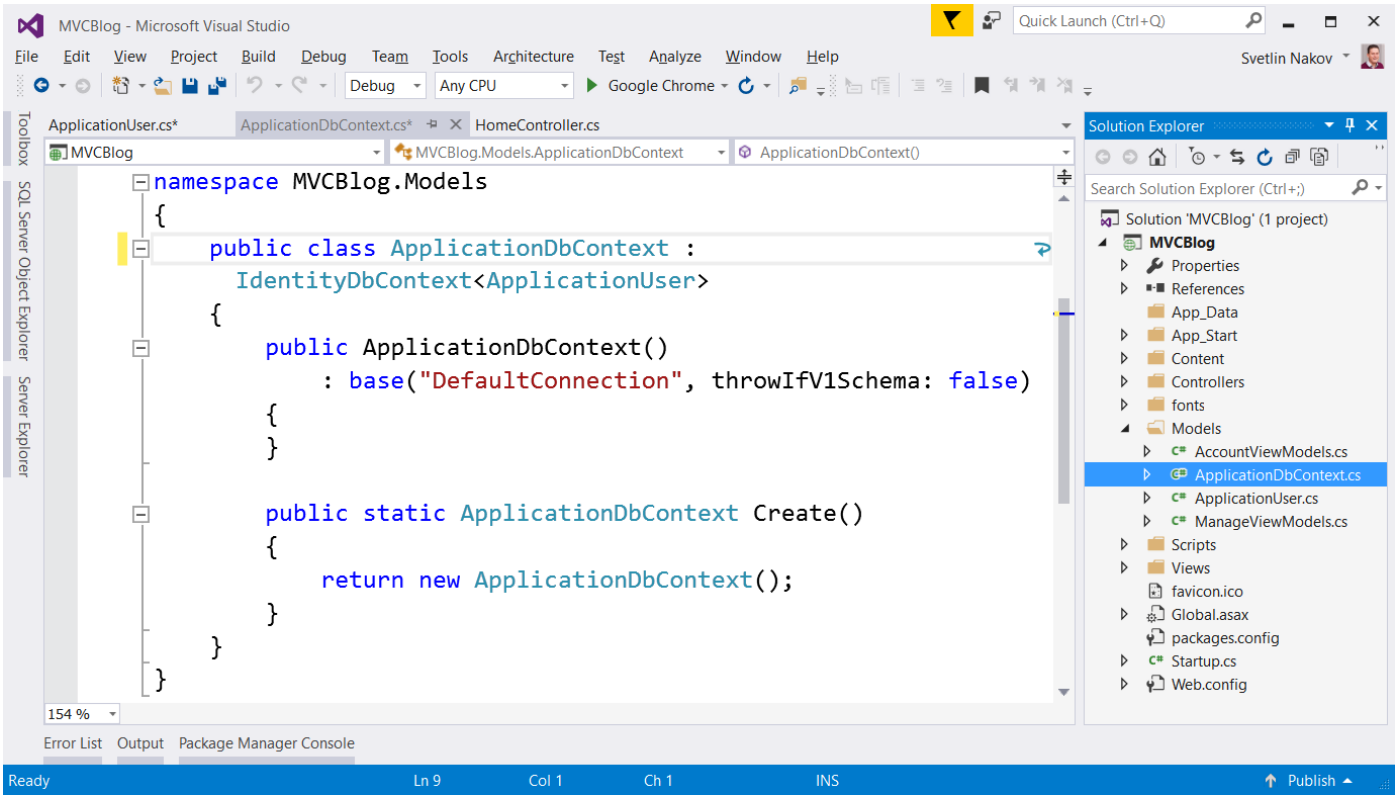
5. Refactor the Data Model Classes

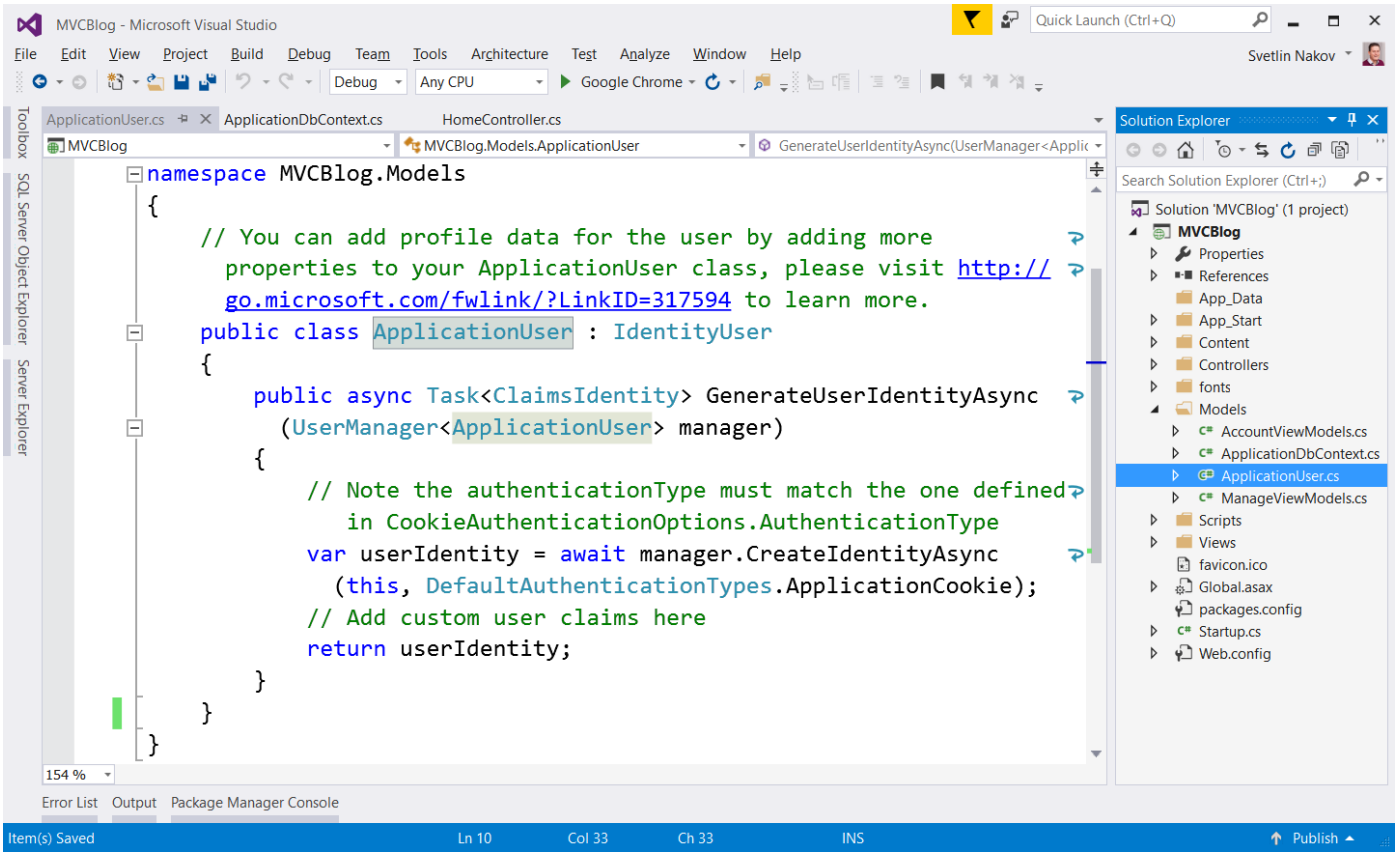
Separate the **`ApplicationUser`** and **`ApplicationDbContext`** classes in **separate files**. Initially, they both live in the **`IdentityModels.cs`** file. Extract each class in separate file:



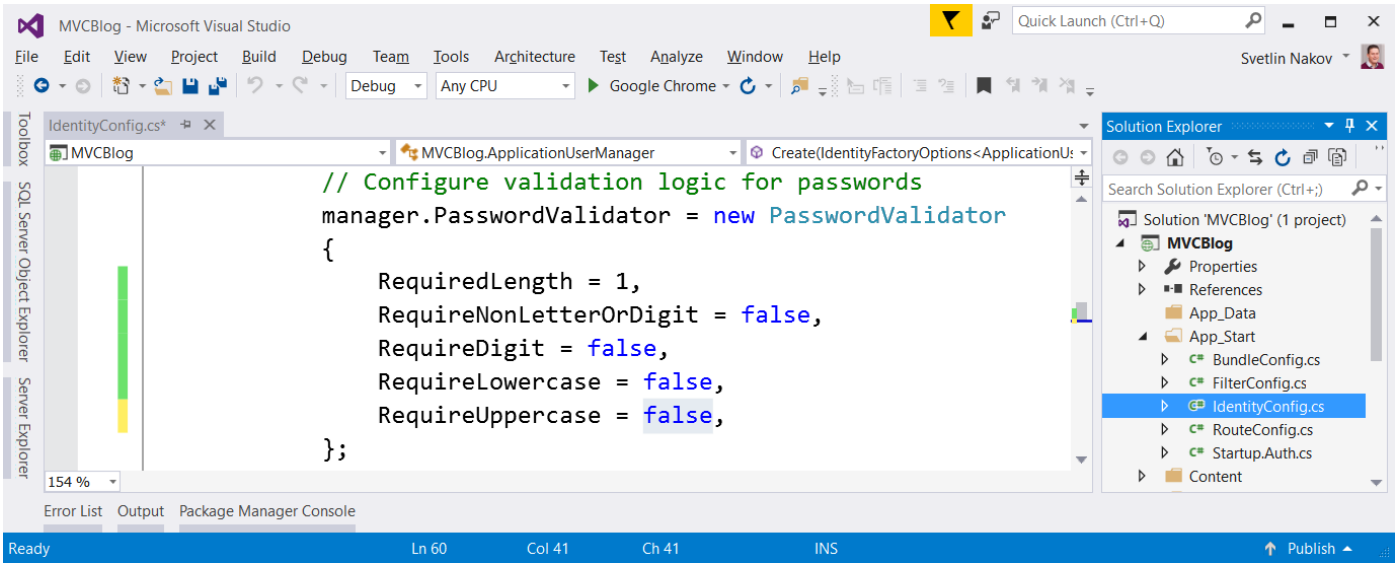


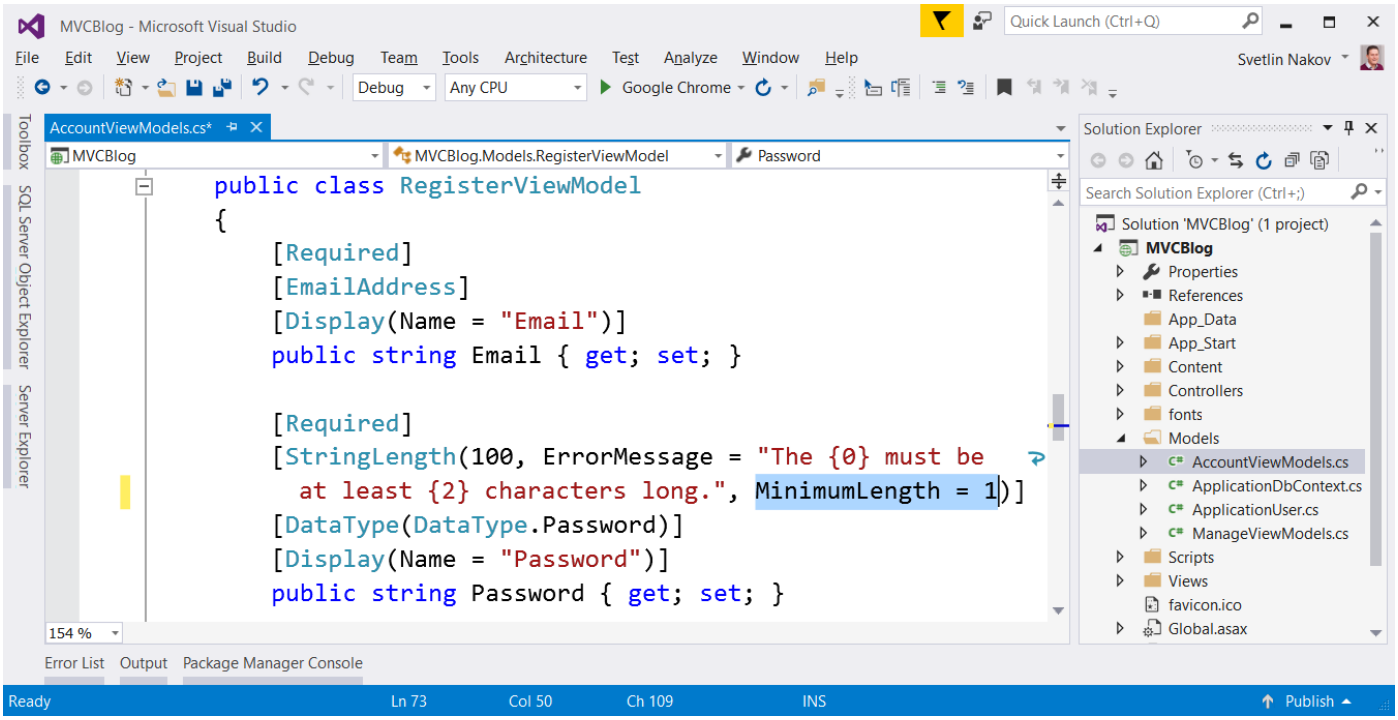


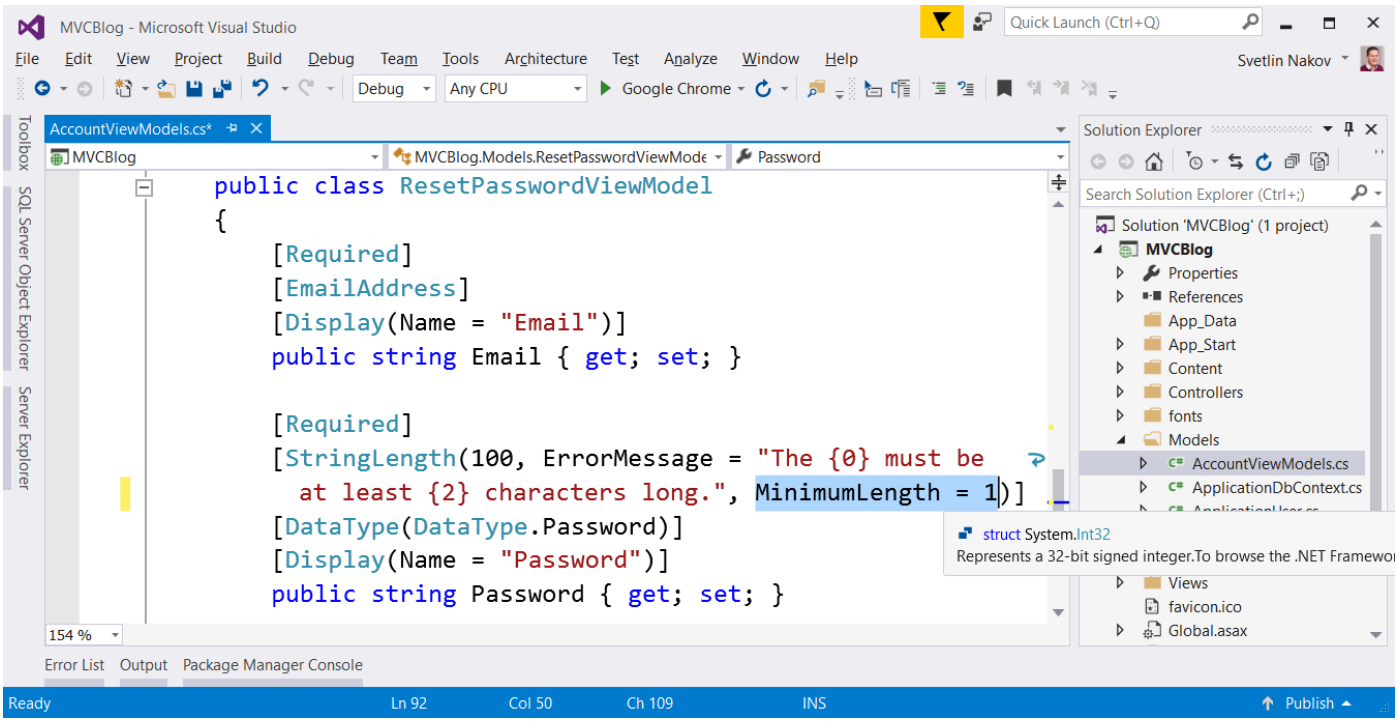




6. Modify the Default Password Requirements

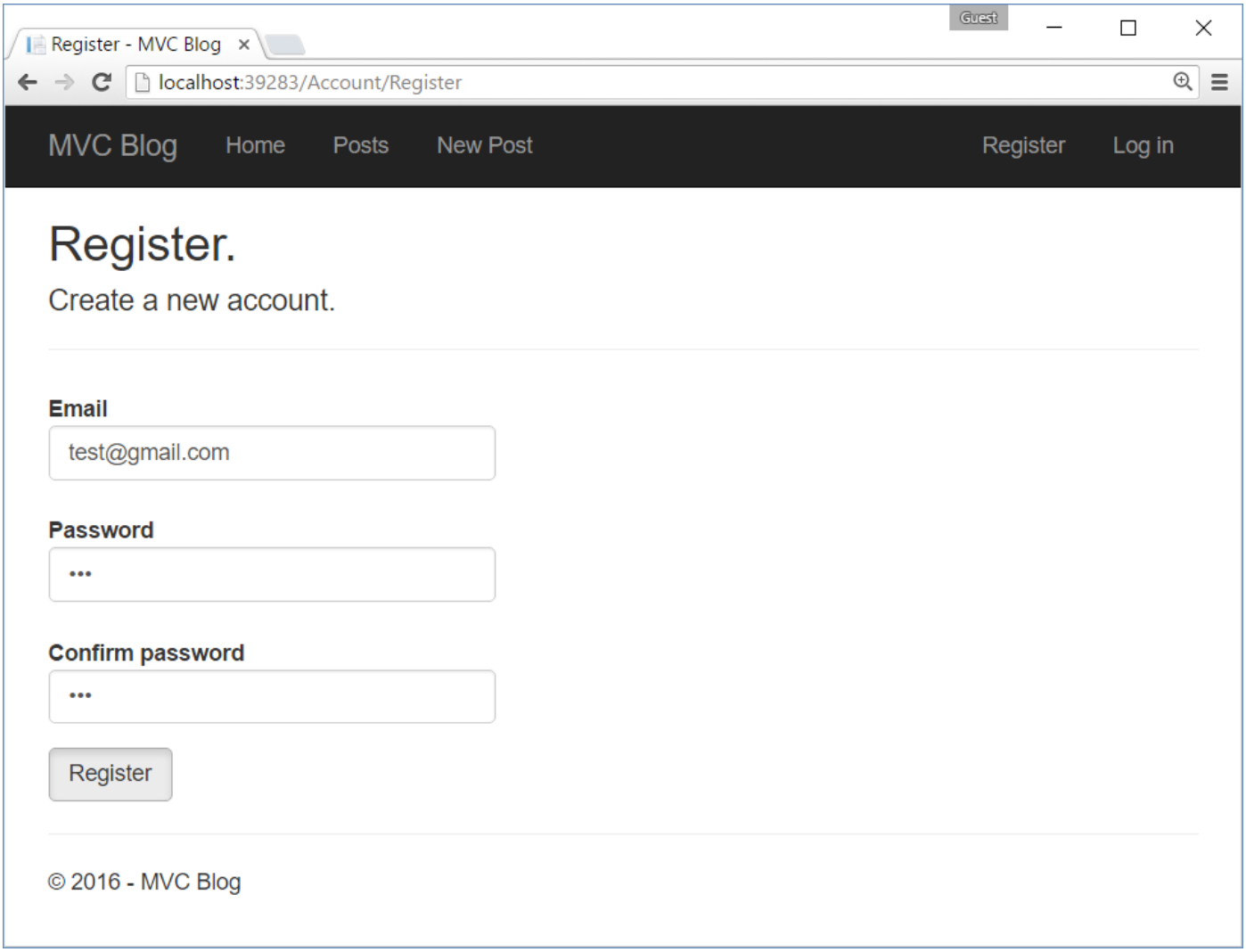


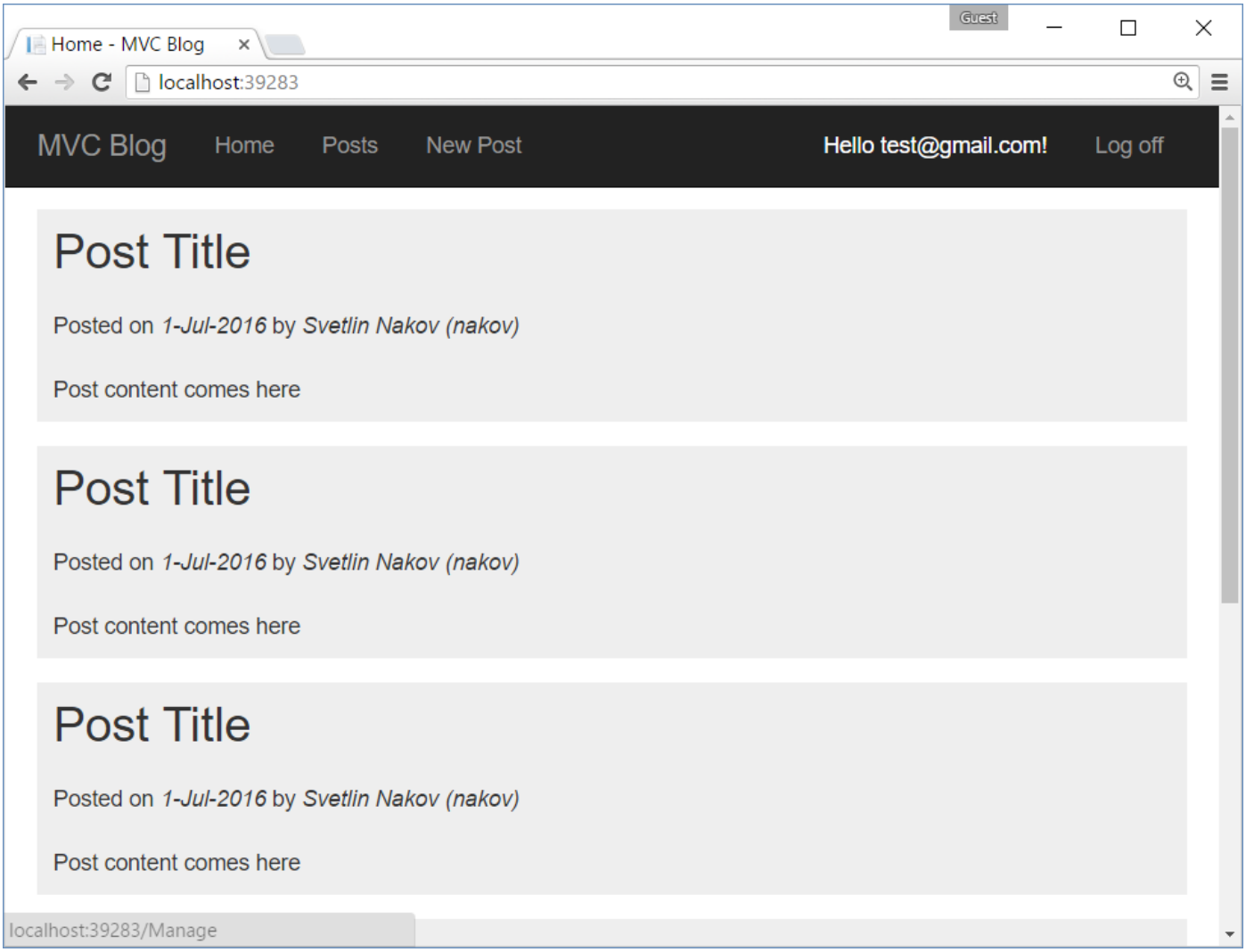




7. Register a New User

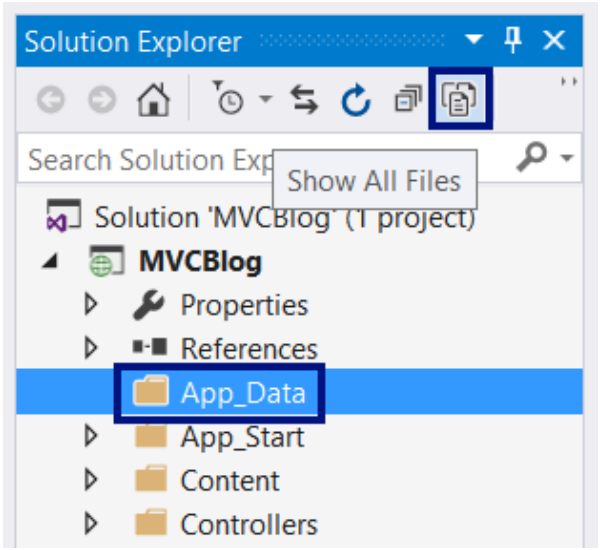
Run the application and **register a new user**:

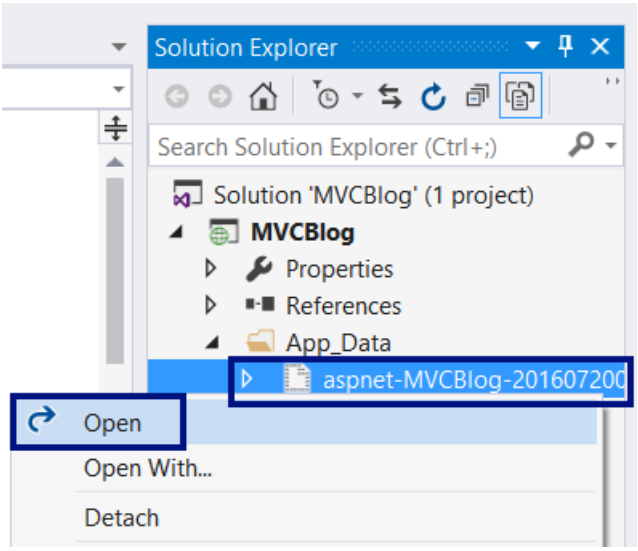

Guest
Register - MVC Blog
localhost:39283/Account/Register
MVC Blog
Home
Posts
New Post
Register
Log in
Register.
Create a new account.
Email
test@gmail.com
Password
Confirm password
Register
© 2016 - MVC Blog

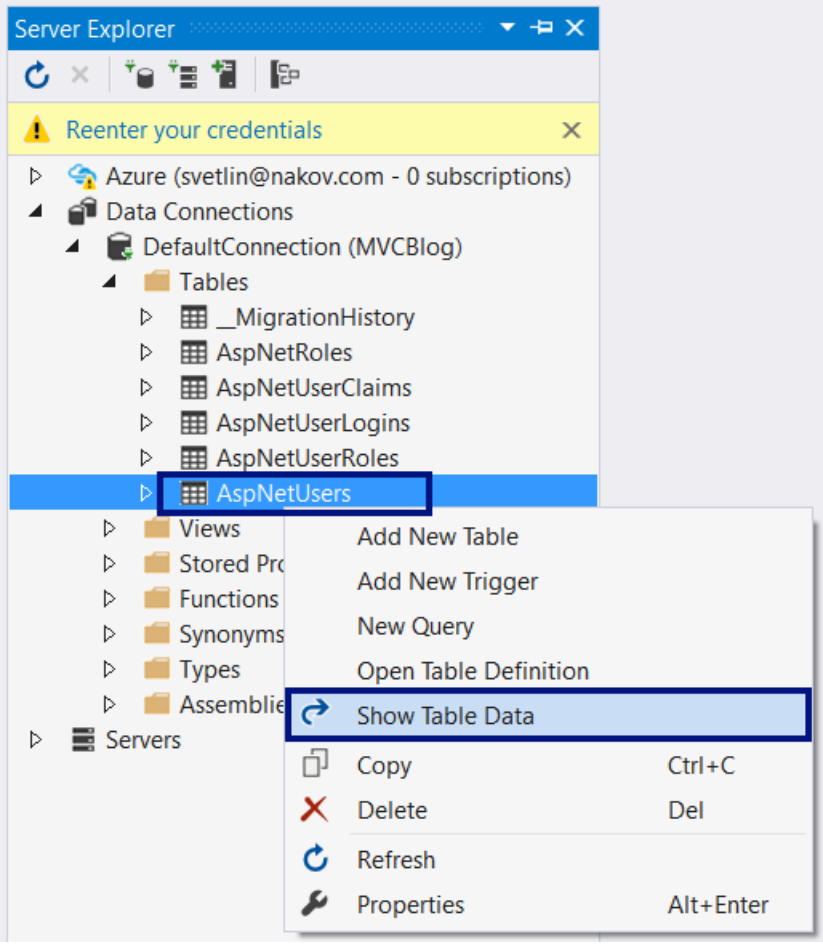


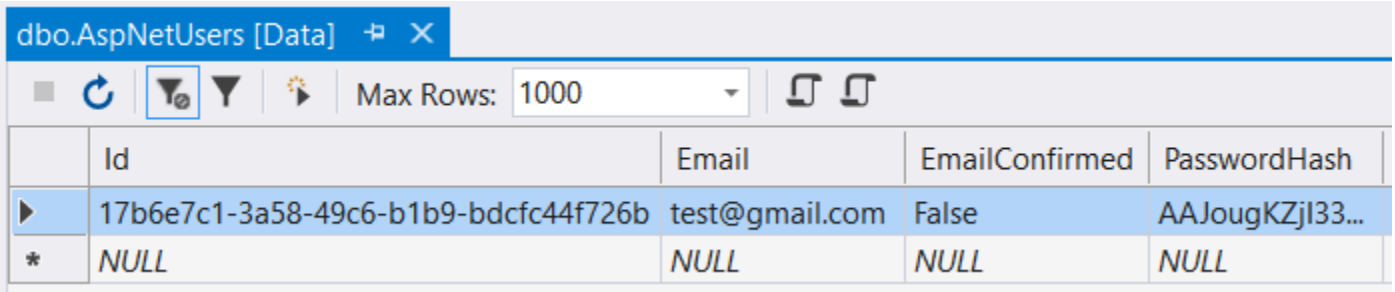
8. Check the Database

Open the **automatically-generated database** in the **`AppData`** folder and view the **`AspNetUsers`** table:

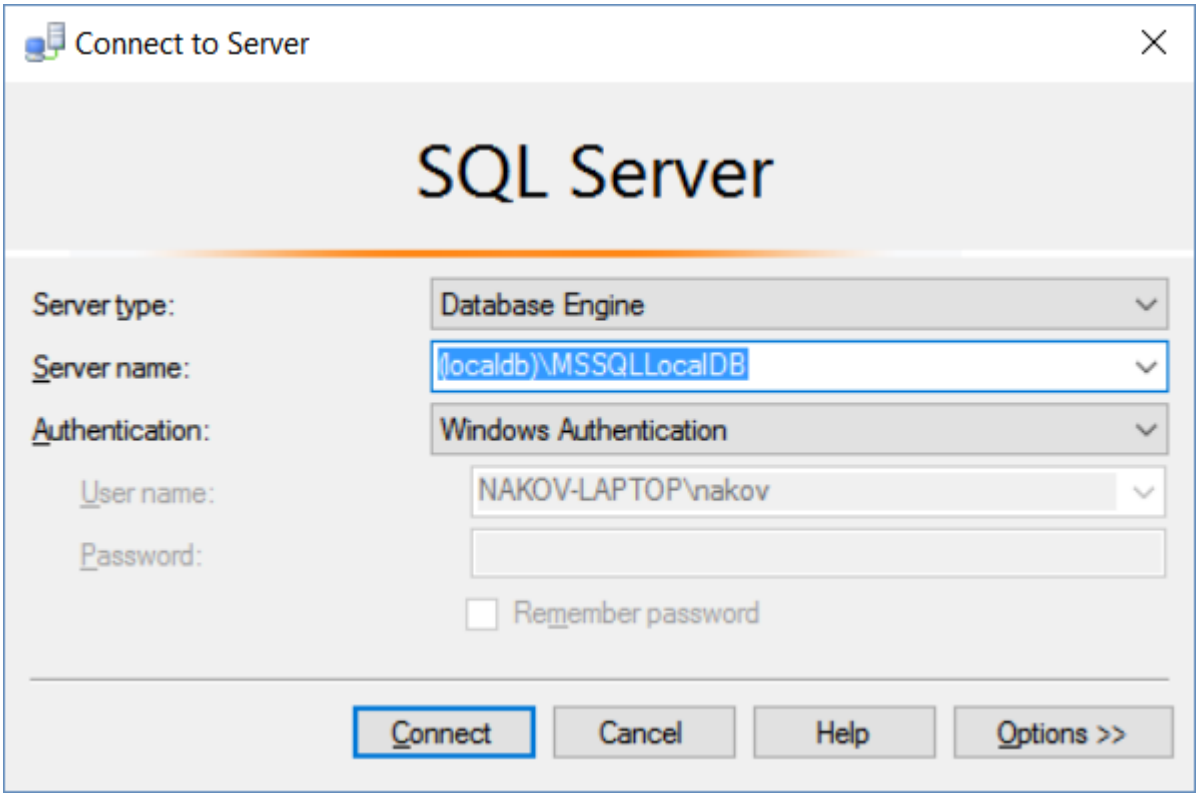


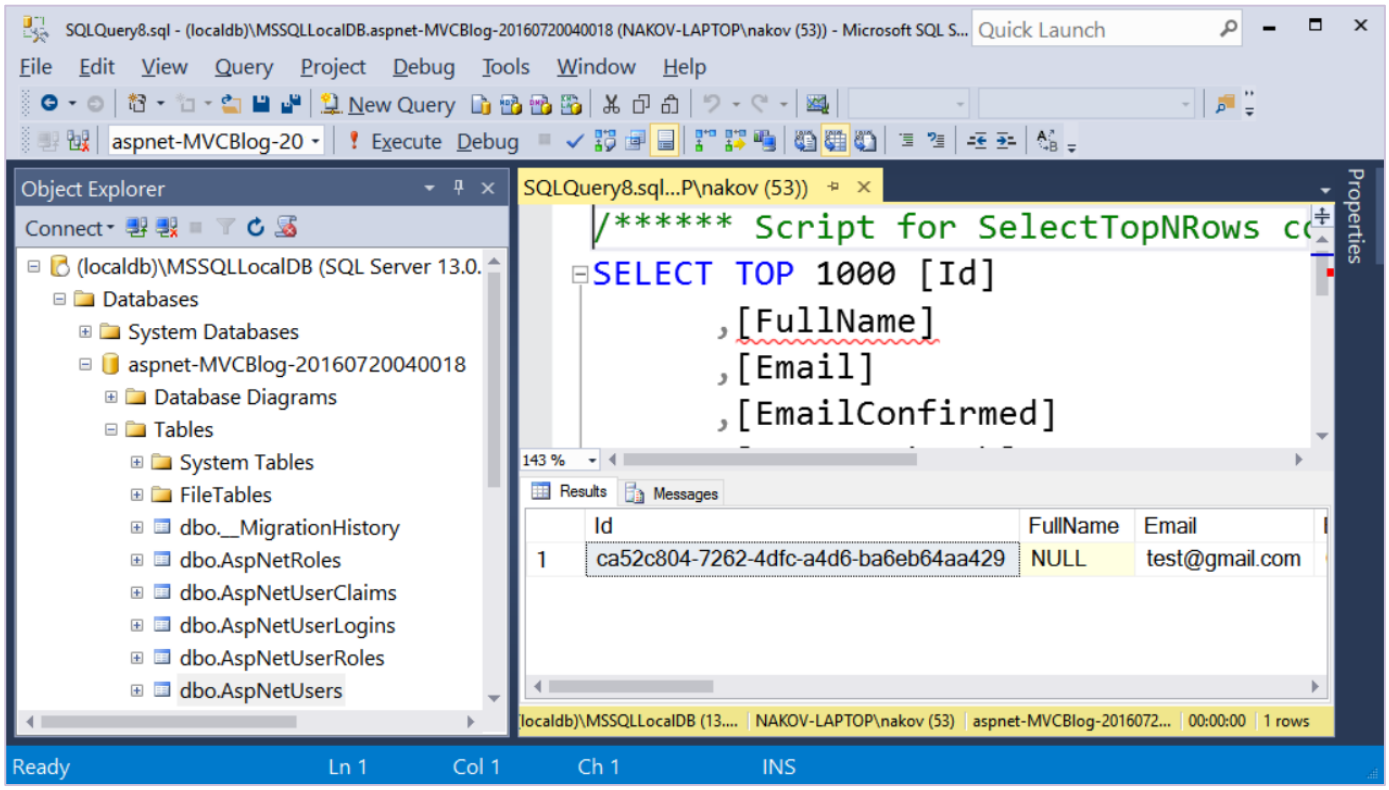






You could also open the database from **SQL Server Management Studio**:

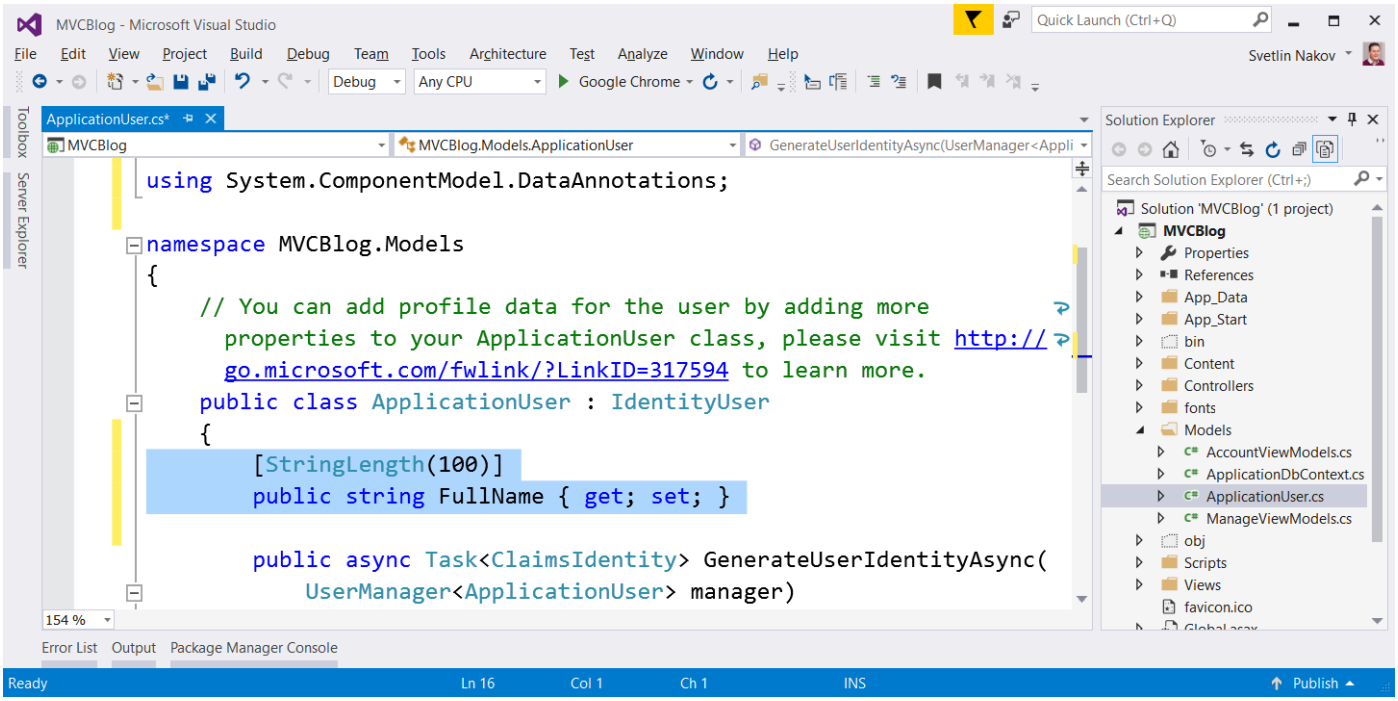




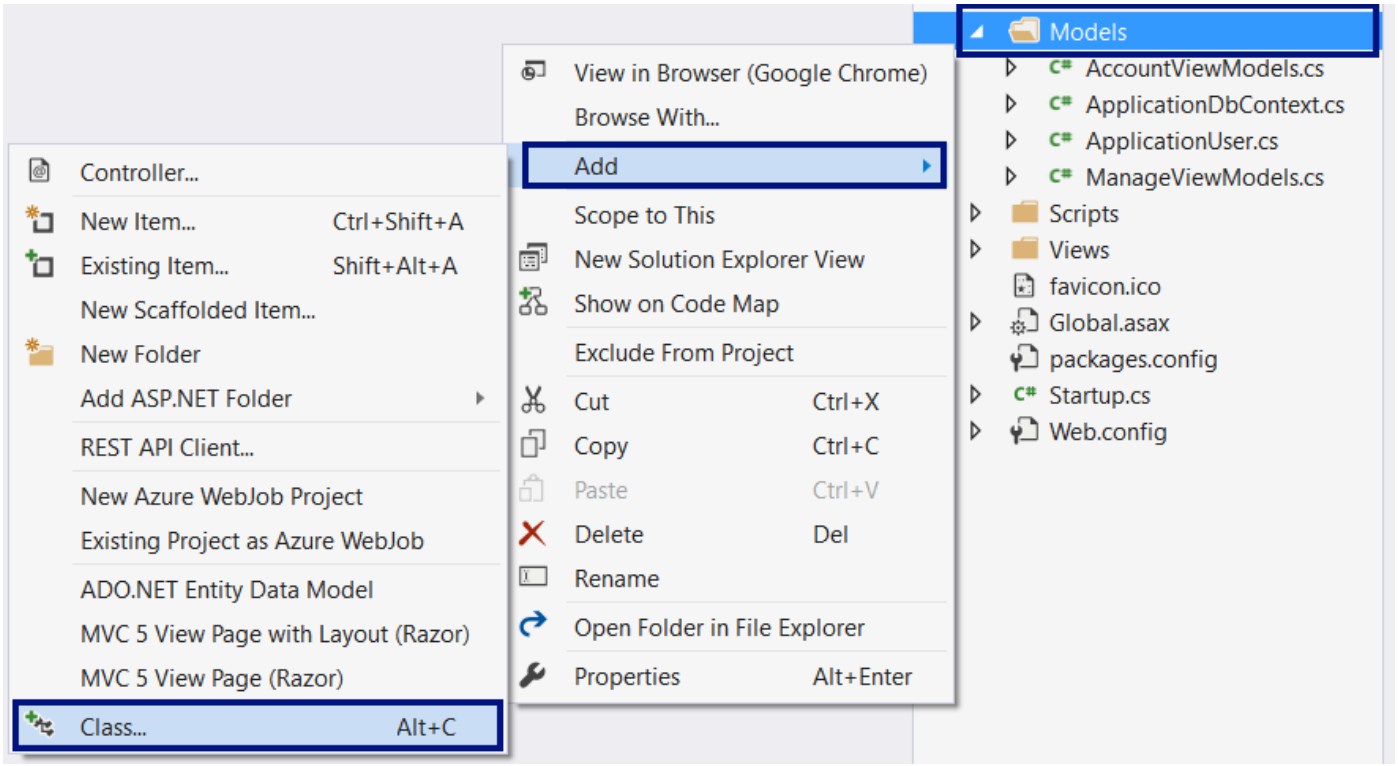
9. Modify the Data Model

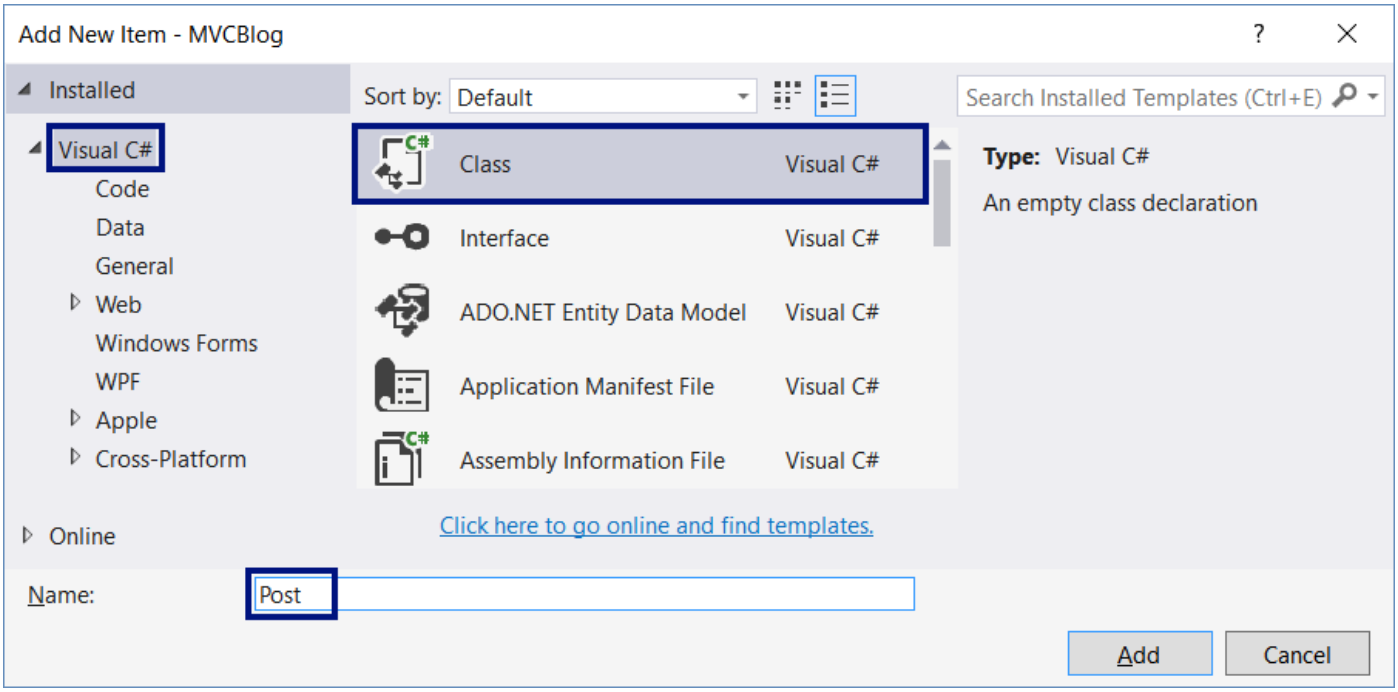
Now modify the data model to have **users** and **posts**.

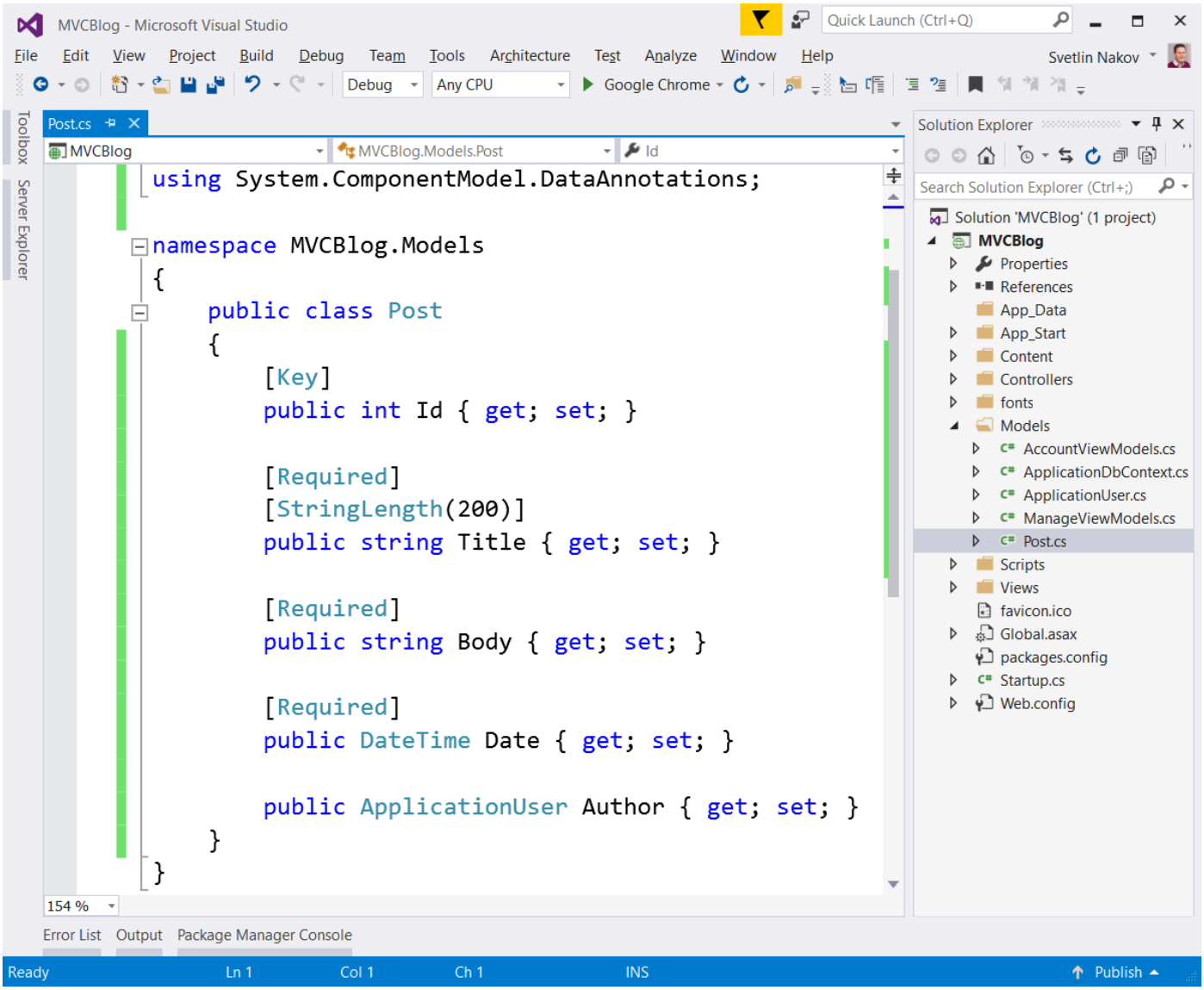
First, add **`FullName`** property to the existing **`ApplicationUser`** class:



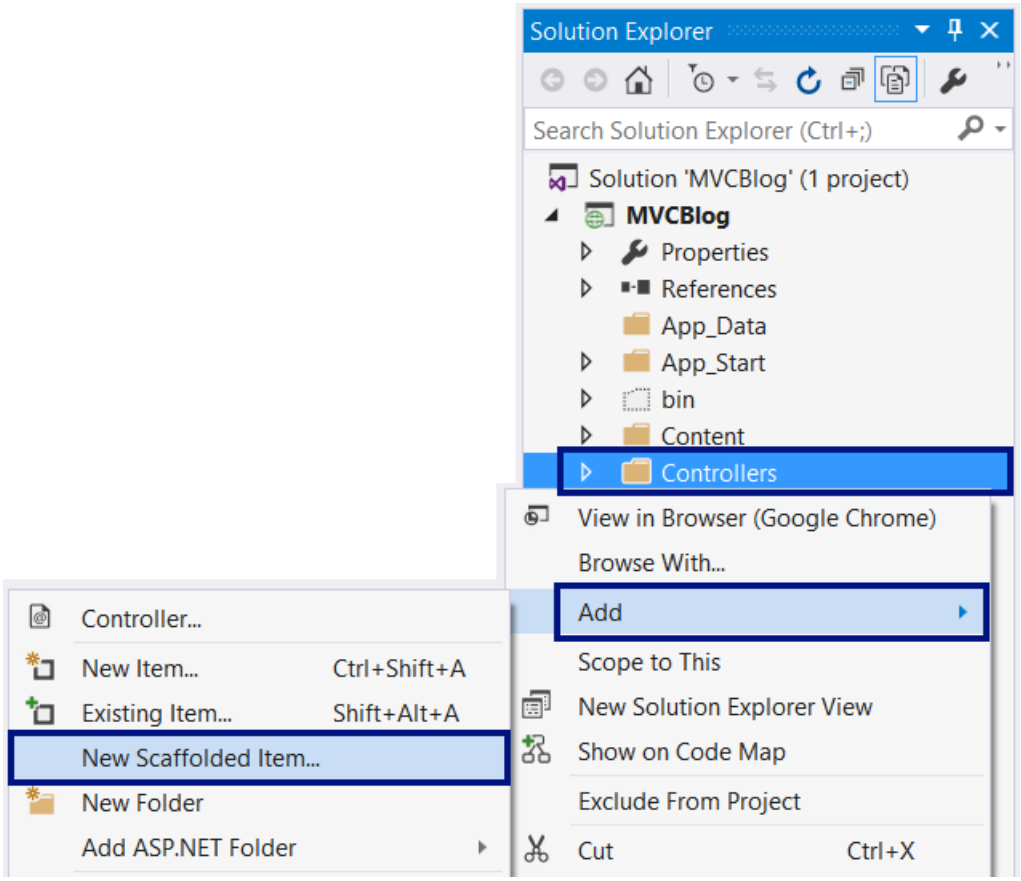
Next, create a **Post** class to hold the blog posts:

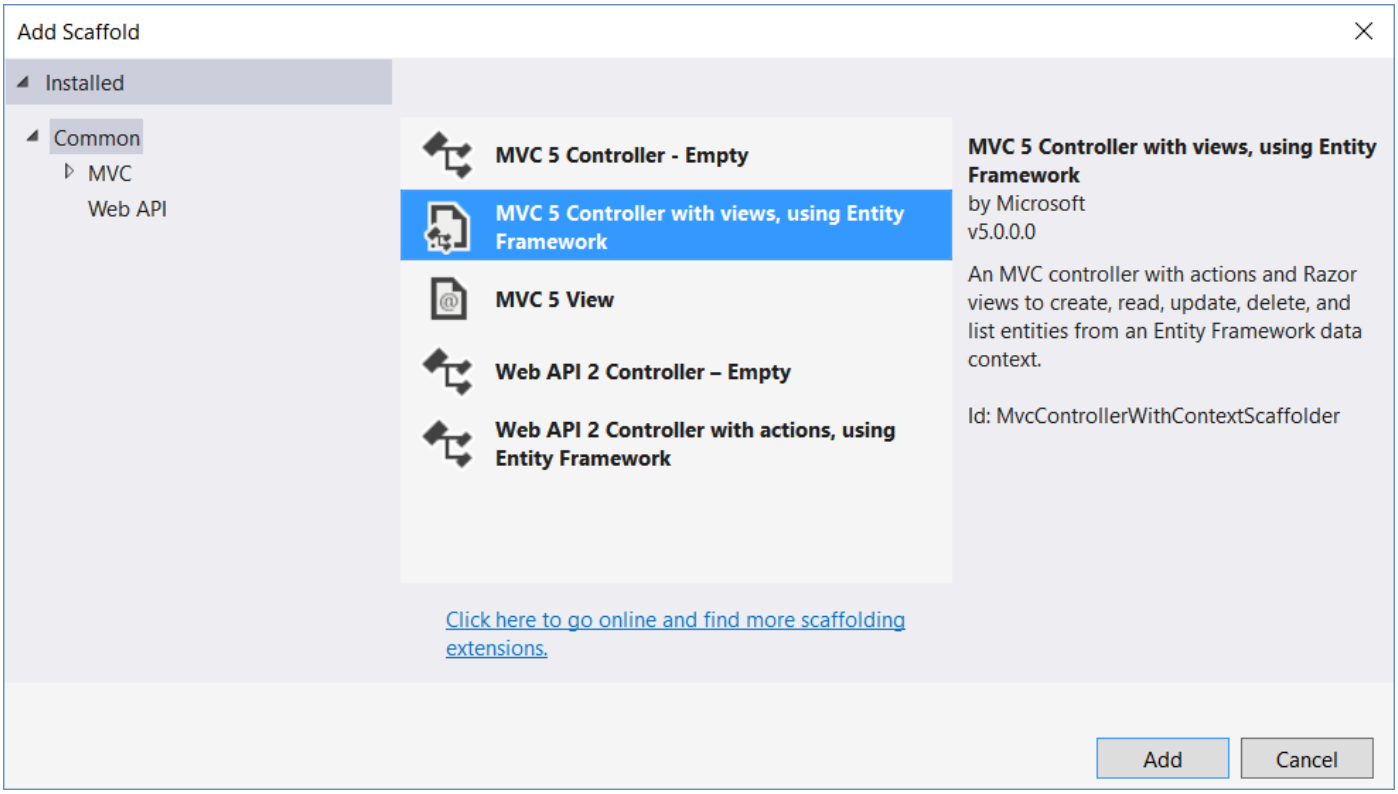


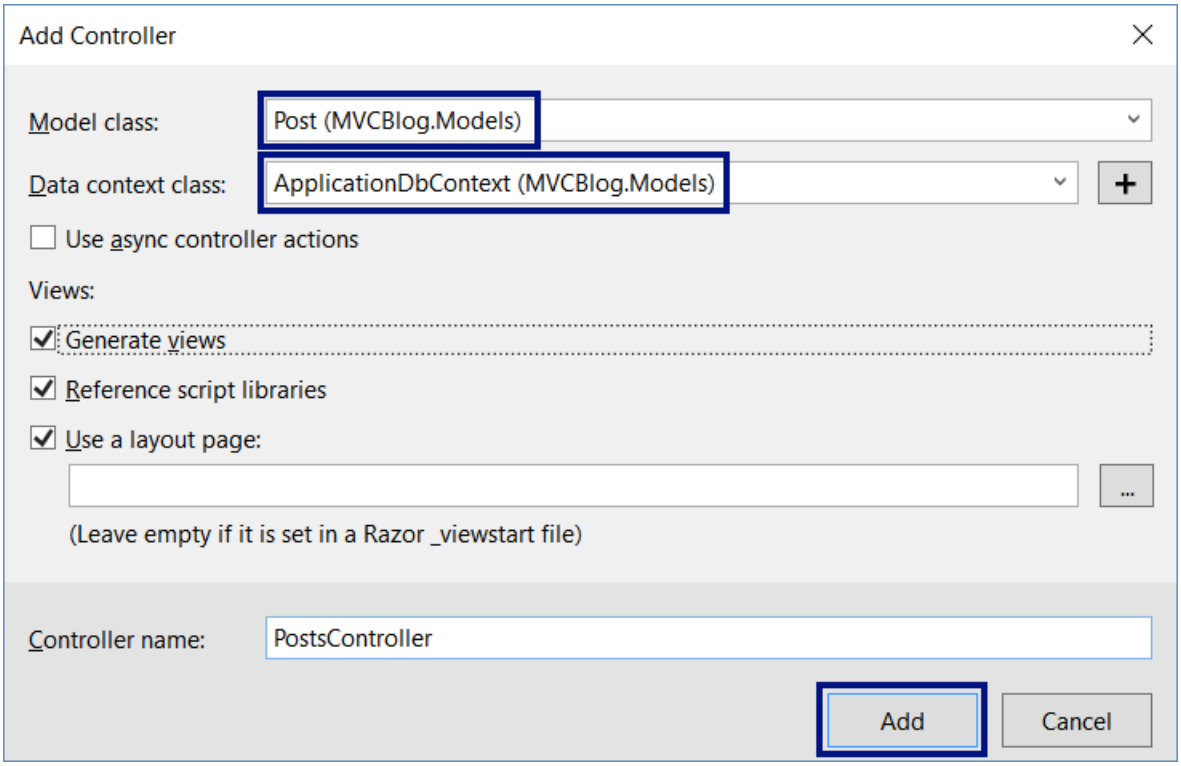

Add New Item - MVCBlog
Installed
Visual C#
Code
Data
General
Web
Windows Forms
WPF
Apple
Cross-Platform
Online
Sort by: Default
Search Installed Templates (Ctrl+E)
Class Visual C#
Interface Visual C#
ADO.NET Entity Data Model Visual C#
Application Manifest File Visual C#
Assembly Information File Visual C#
Type: Visual C#
An empty class declaration
Click here to go online and find templates.
Name: Post
Add
Cancel


MVCBlog - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help
Svetlin Nakov
Debug
Any CPU
Google Chrome
Toolbox
Server Explorer
Post.cs
MVCBlog
MVCBlog.Models.Post
Id
using System.ComponentModel.DataAnnotations;
namespace MVCBlog.Models
{
public class Post
{
[Key]
public int Id { get; set; }
[Required]
[StringLength(200)]
public string Title { get; set; }
[Required]
public string Body { get; set; }
[Required]
public DateTime Date { get; set; }
public ApplicationUser Author { get; set; }
}
}
154 %
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MVCBlog' (1 project)
MVCBlog
Properties
References
App_Data
App_Start
Content
Controllers
fonts
Models
AccountViewModels.cs
ApplicationDbContext.cs
ApplicationUser.cs
ManageViewModels.cs
Post.cs
Scripts
Views
favicon.ico
Global.asax
packages.config
Startup.cs
Web.config
Error List
Output
Package Manager Console
Ready
Ln 1
Col 1
Ch 1
INS
Publish

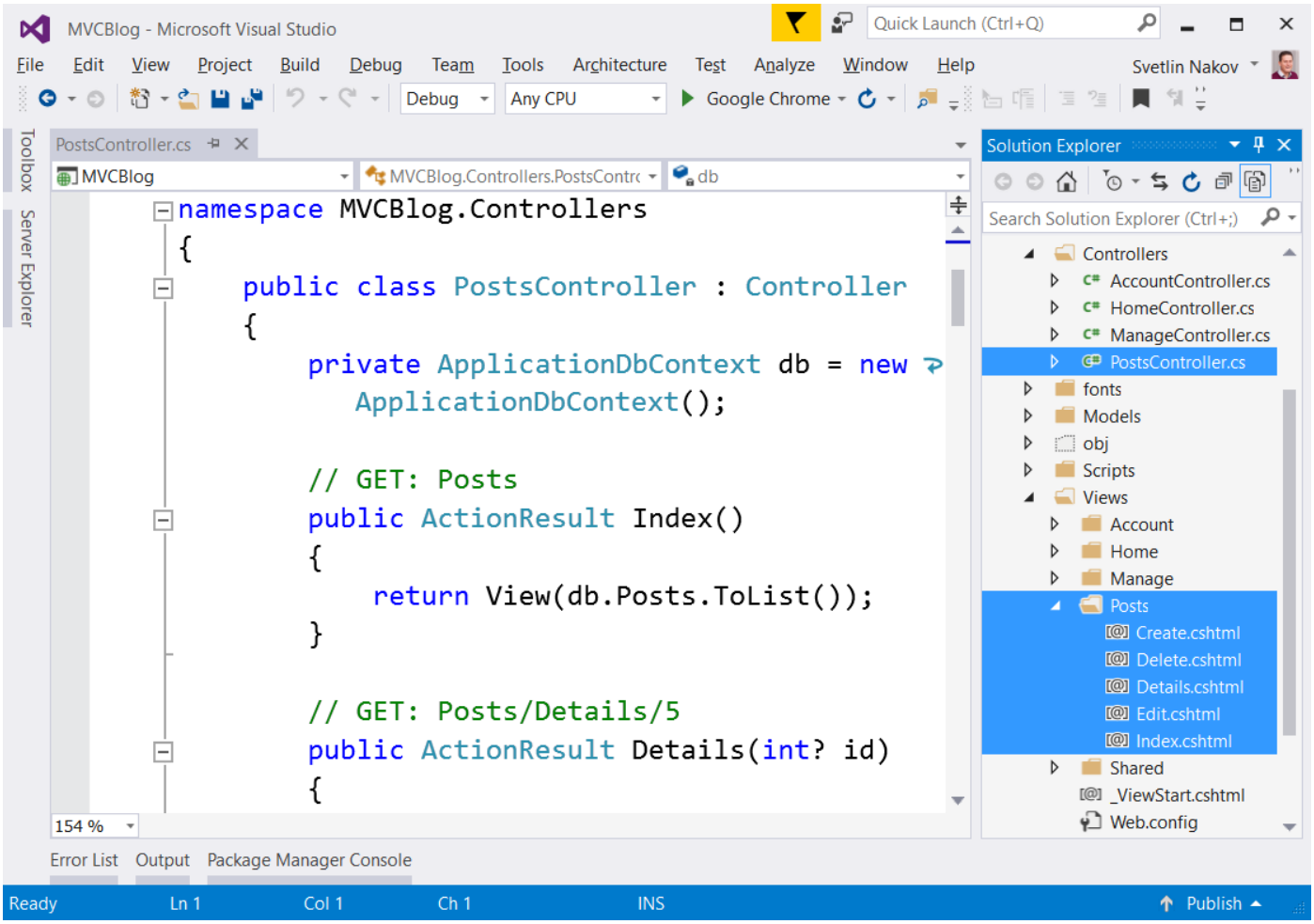
10. Scaffold the CRUD Operations for the Posts





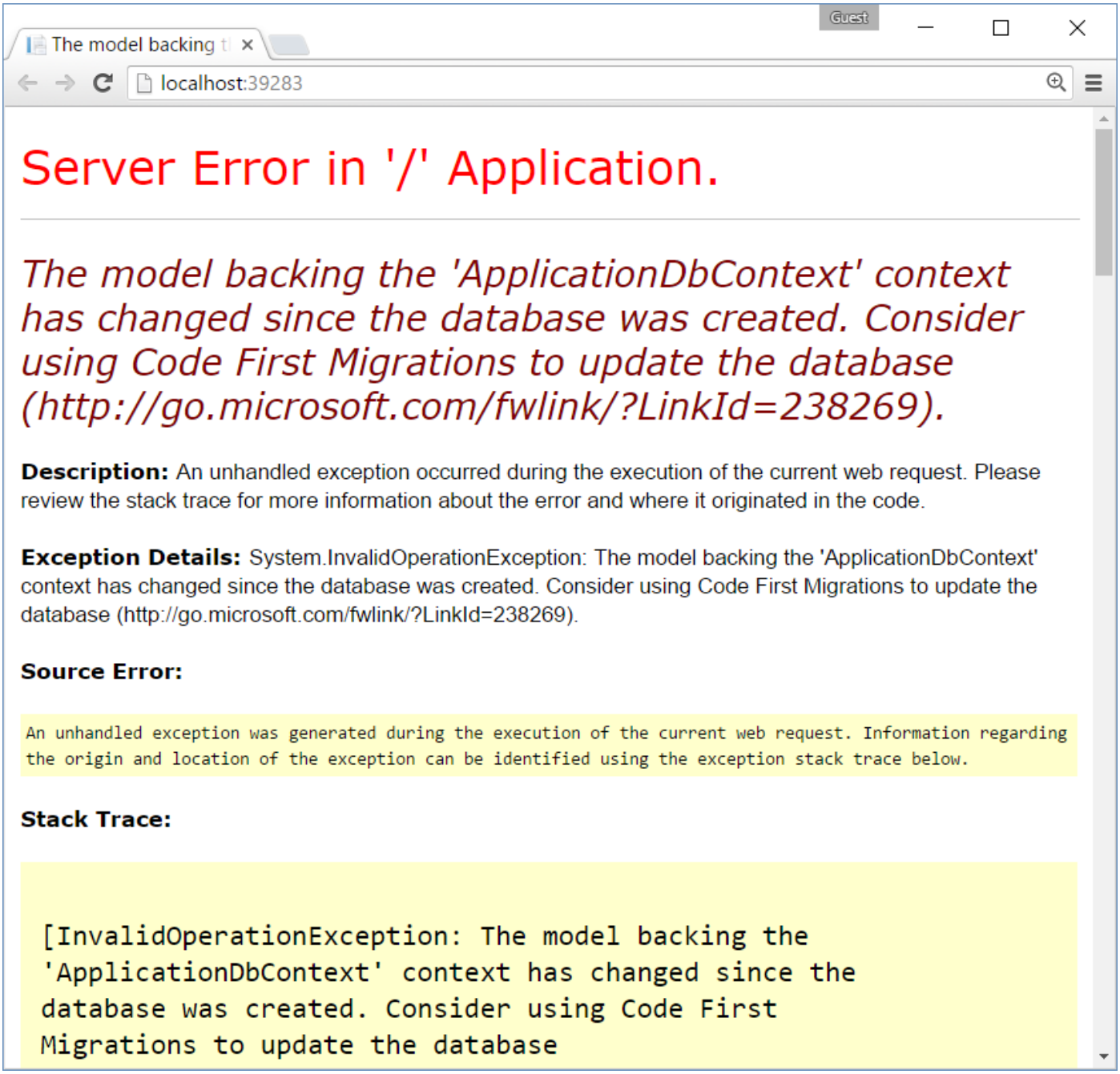


Now we have automatically-generated **`PostsController`** with **`Index`** / **`Details`** / **`Create`** / **`Edit`** / **`Delete`** **actions** + **views** for these actions:



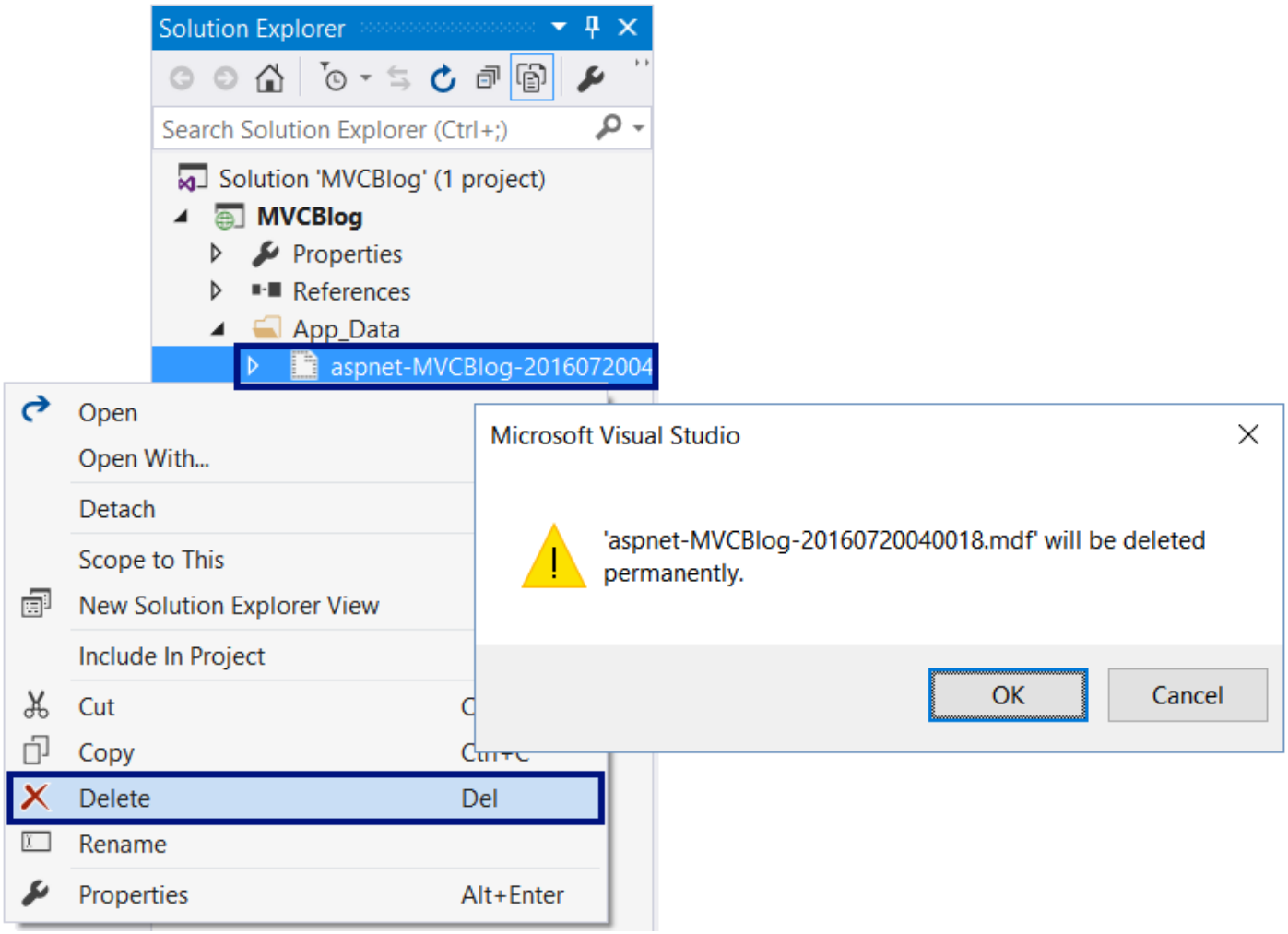
11. Test the Generated Code

If you run the app, you will see the following **error**:



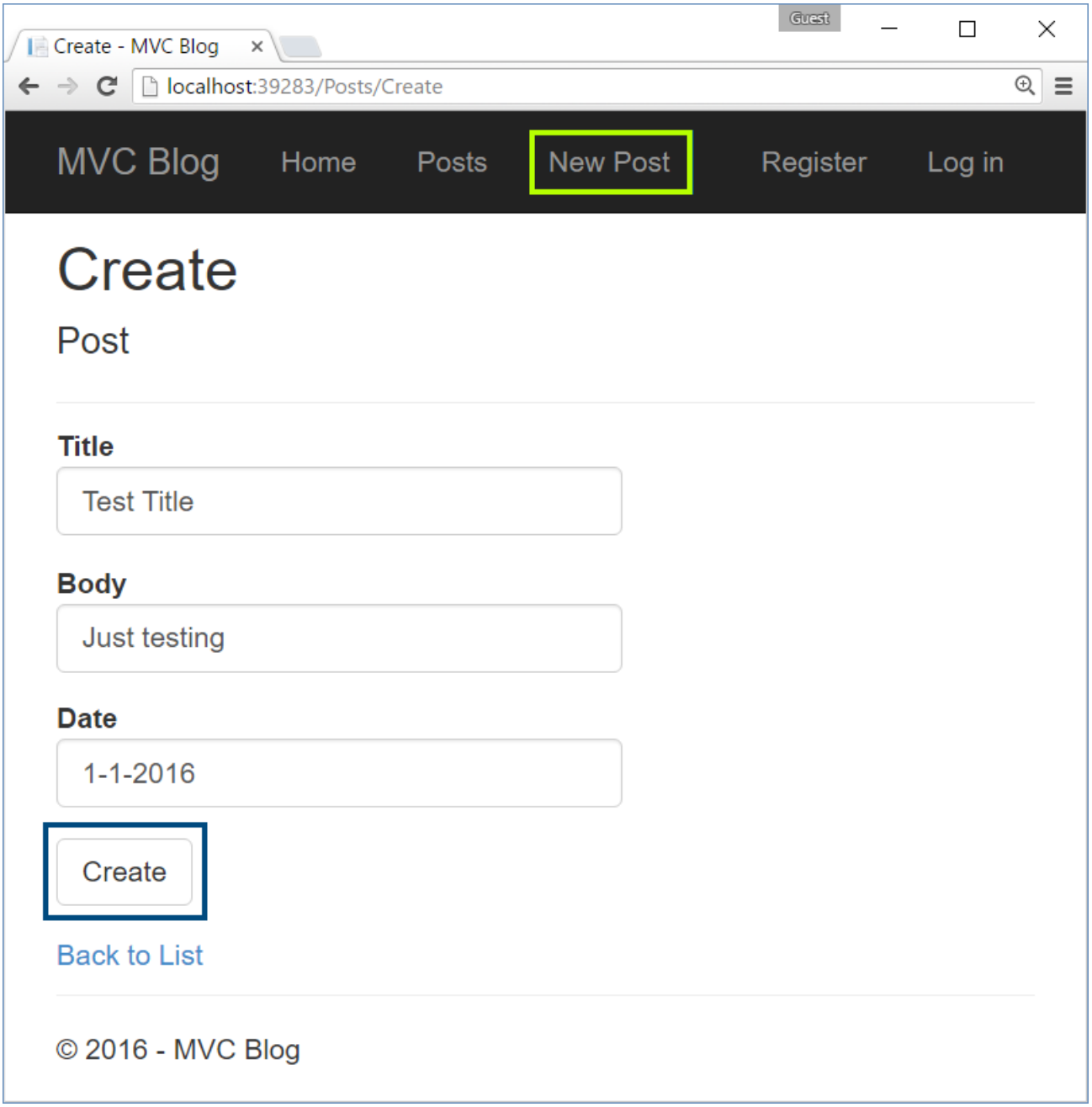
Drop the Database

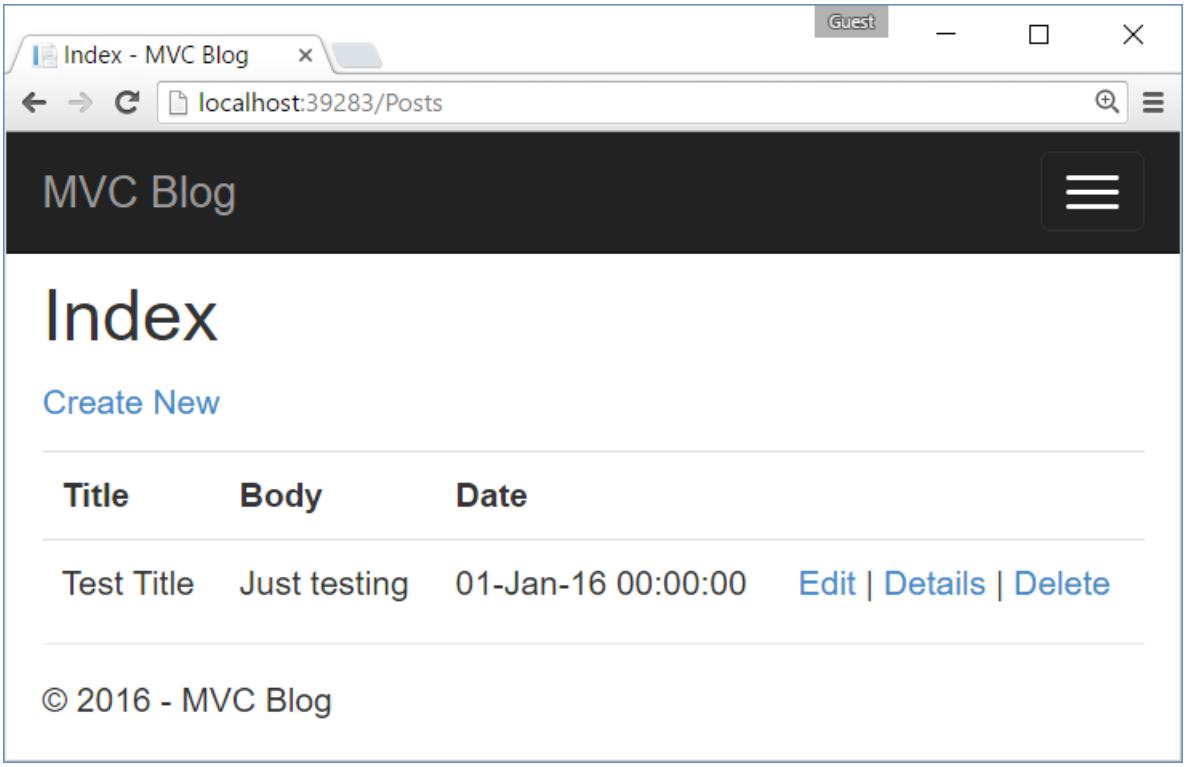
The easiest way to fix this problem is to **drop (delete) the database**:



Test the App Again

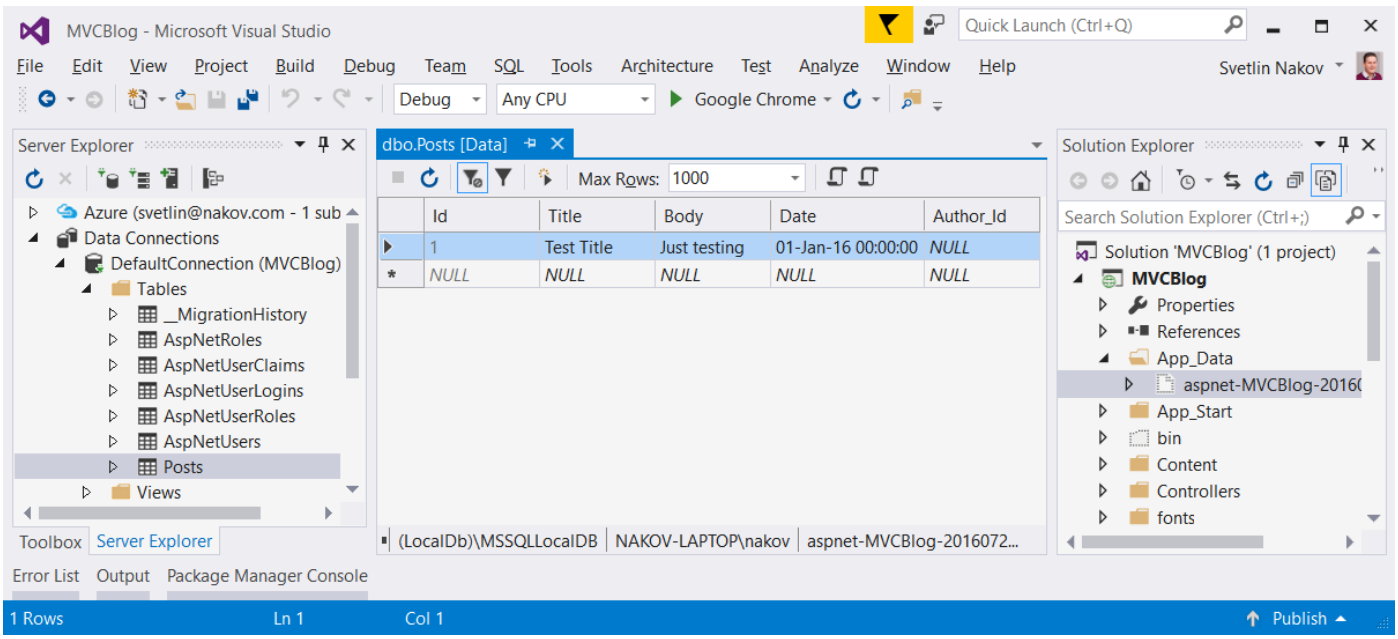
Now run the app and **create a new blog post**:


Guest
Create - MVC Blog
localhost:39283/Posts/Create
MVC Blog
Home
Posts
New Post
Register
Log in
Create
Post
Title
Test Title
Body
Just testing
Date
1-1-2016
Create
Back to List
© 2016 - MVC Blog



Check the Database

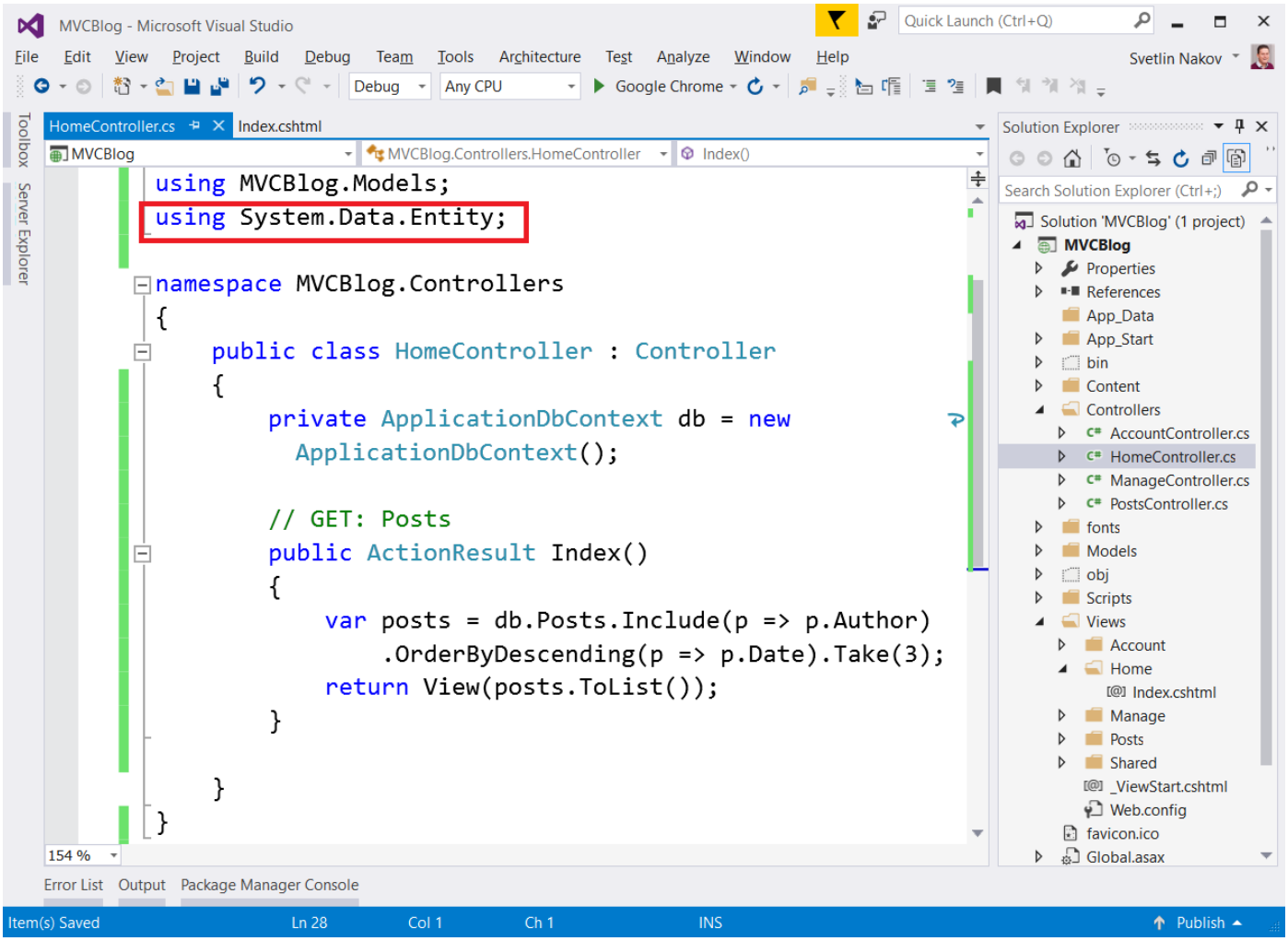
Open the database to see the table **`Posts`** and the new post inside it:



12. Implement “Last First 3 Posts” at the Home Screen

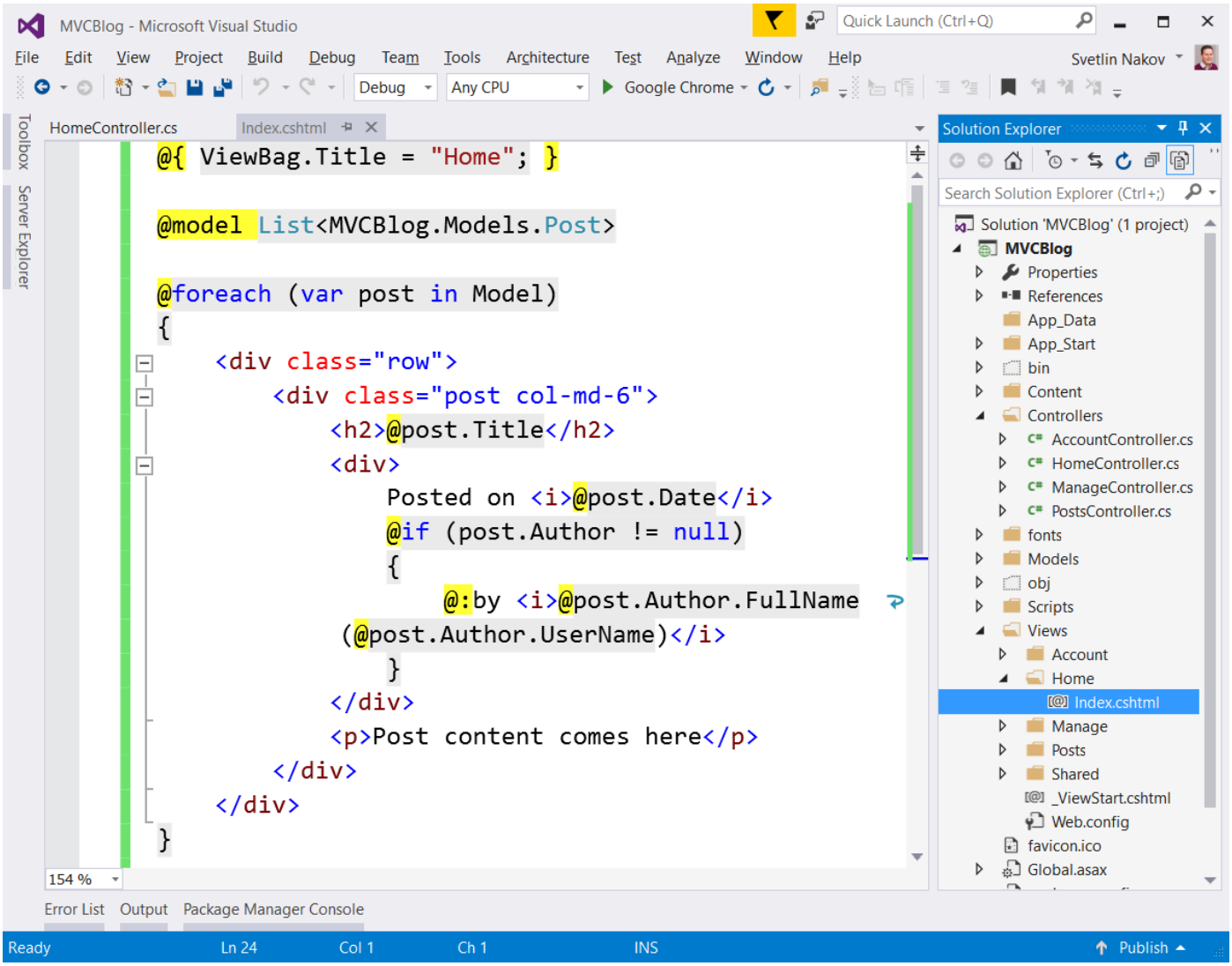
Implement the HomeController.Index()

Edit the **`Index()`** method in the **`HomeController`**. It should display the latest 3 posts along with their author:



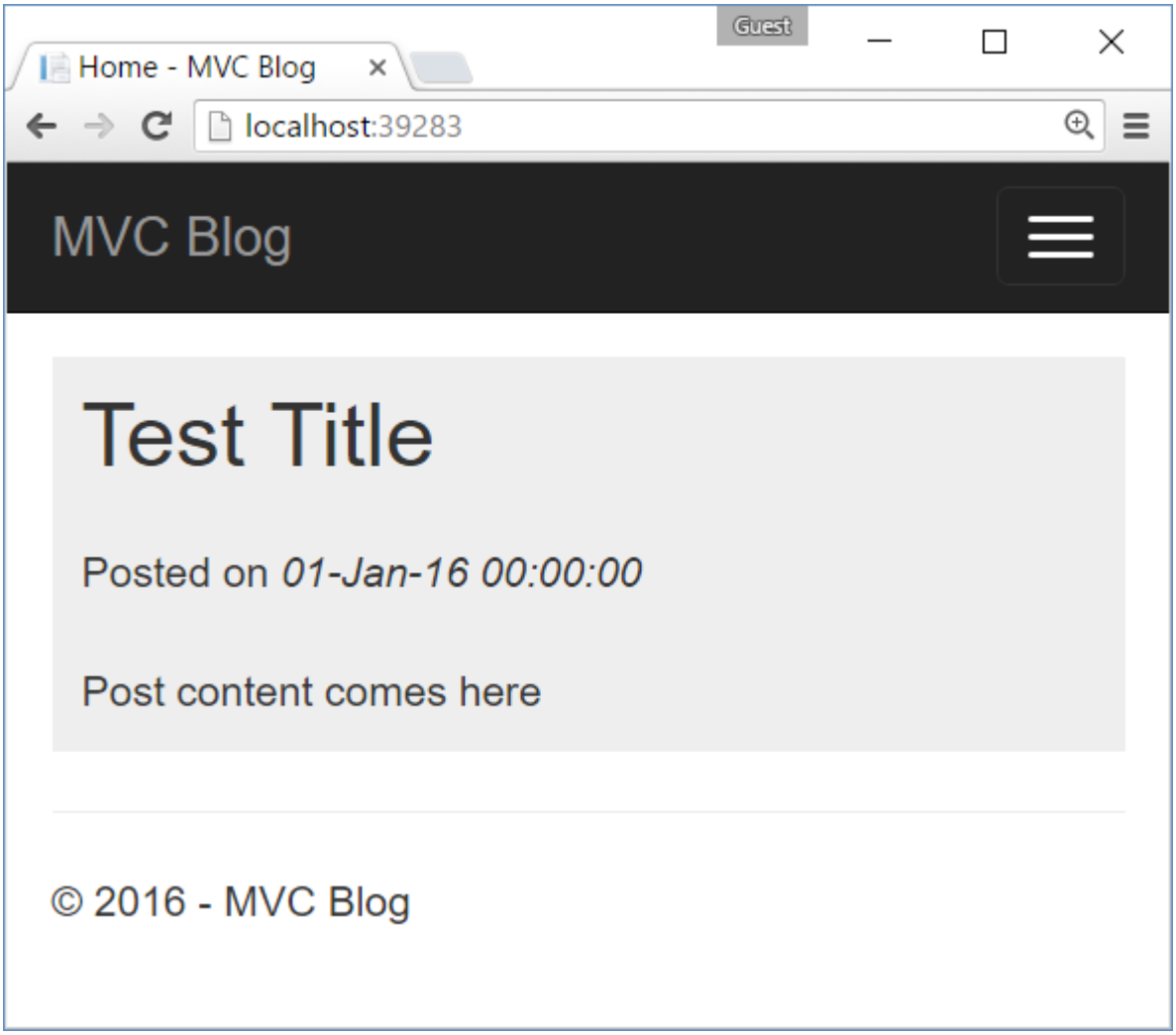
Implement the View Home\Index.cshtml

Edit the **`\Home\Index.cshtml`** view to display the posts:



Run the App

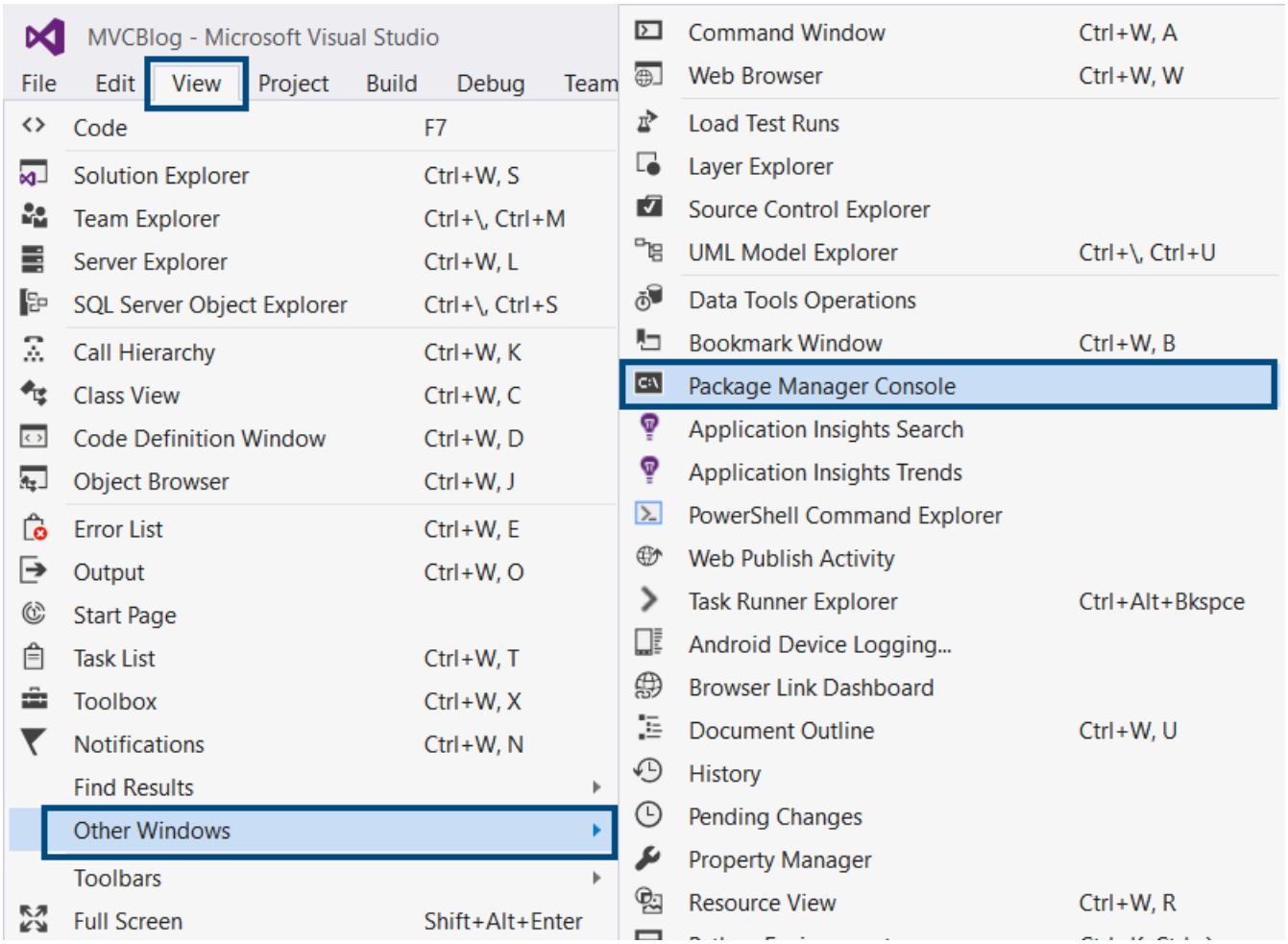
Run the application with **[Ctl+F5]** and see the posts listed by the **`HomeController`** and **`Index.cshtml`** view:

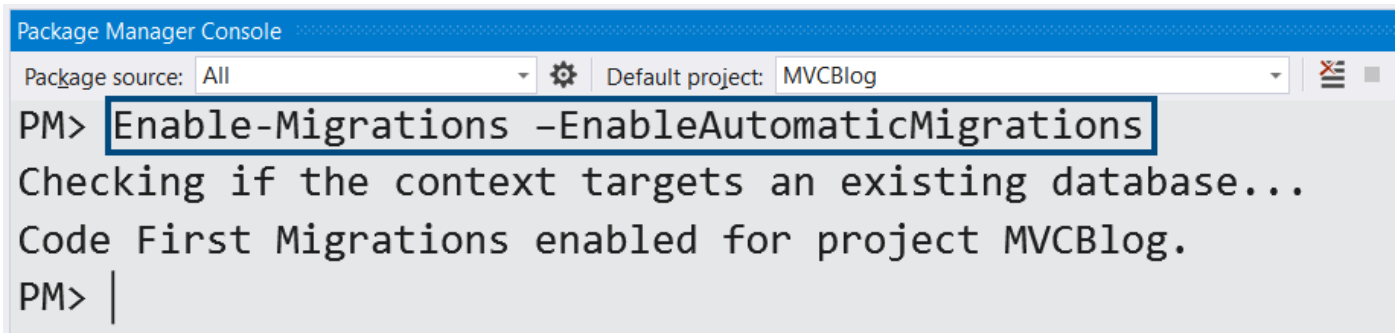


13. Fill Sample Data in the DB

Enable Automatic DB Migrations

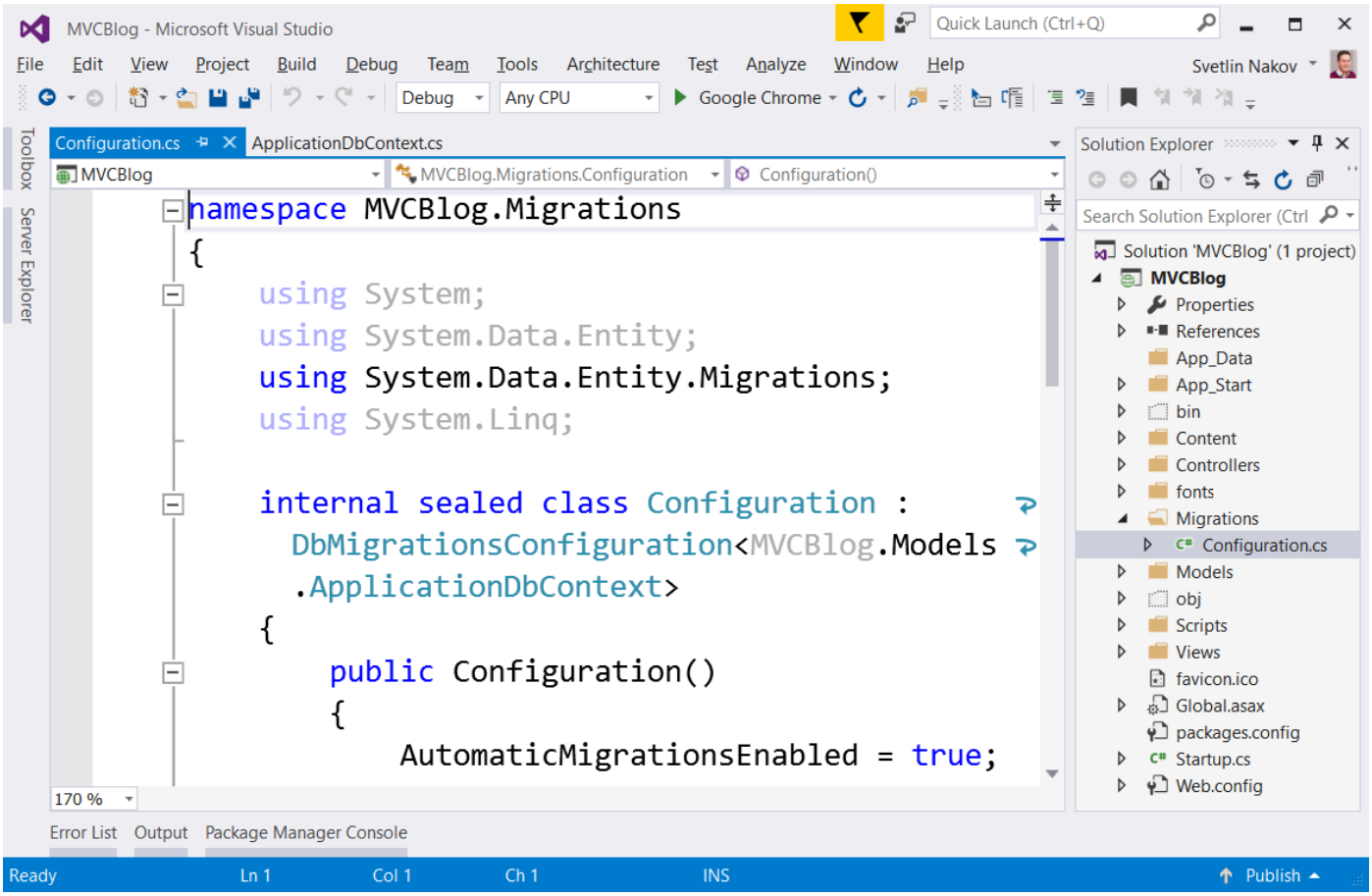
Enable the **automatic migrations** for your code-first EF data model:





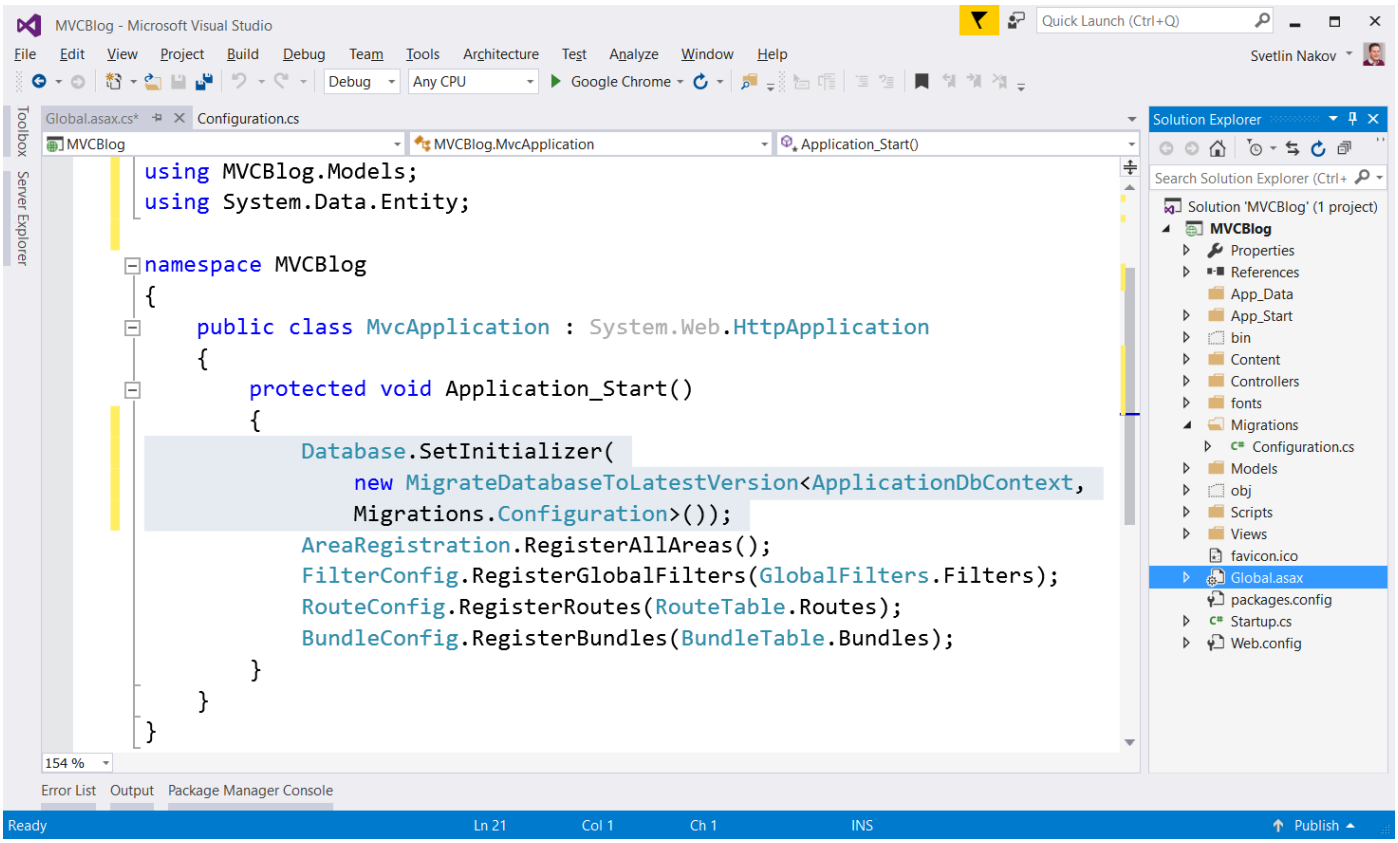
See the Auto-Generated Code

See the automatically-generated code. A new class **`Migrations\Configuration.cs`** should appear:



Configure the Database Initializer

Configure the **database initializer** to **invoke the DB migrations** at application start-up:



Populate Some in the Seed() Method

Populate some data in the database: a few **users**, **admin user** in role "**Administrators**" and a few **posts**. Just copy the following code in the file **`Configuration.cs`**:

Configuration.cs

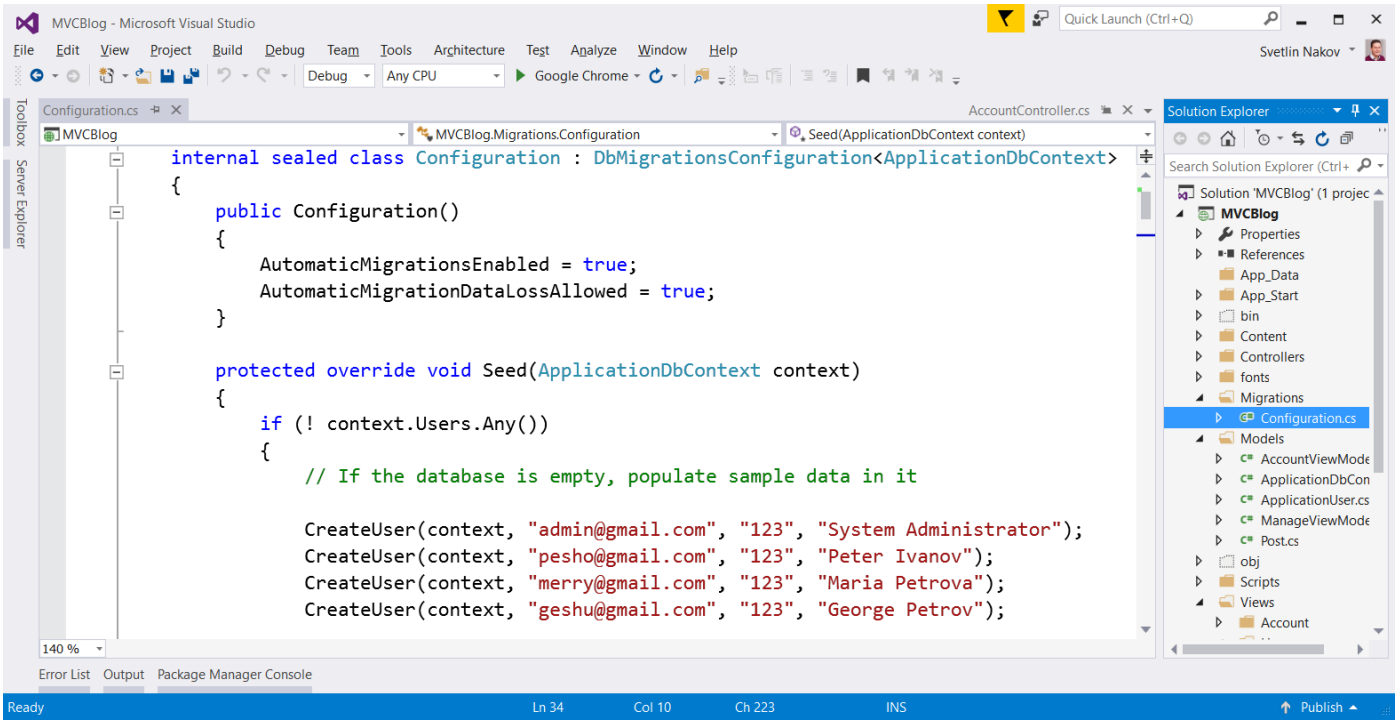
```
namespace MVCBlog.Migrations
{
    using Microsoft.AspNet.Identity;
    using Microsoft.AspNet.Identity.EntityFramework;
    using Models;
    using System;
    using System.Data.Entity.Migrations;
    using System.Linq;

    internal sealed class Configuration : DbMigrationsConfiguration<ApplicationDbContext>
    {
        public Configuration()
        {
            AutomaticMigrationsEnabled = true;
            AutomaticMigrationDataLossAllowed = true;
        }

        protected override void Seed(ApplicationDbContext context)
        {
            if (! context.Users.Any())
            {
                // If the database is empty, populate sample data in it

                CreateUser(context, "admin@gmail.com", "123", "System Administrator");
                CreateUser(context, "pesho@gmail.com", "123", "Peter Ivanov");
                CreateUser(context, "merry@gmail.com", "123", "Maria Petrova");
                CreateUser(context, "geshu@gmail.com", "123", "George Petrov");

                CreateRole(context, "Administrators");
                AddUserToRole(context, "admin@gmail.com", "Administrators");

                CreatePost(context,
                    title: "Work Begins on HTML5.1",
                    body: @"<p>The World Wide Web Consortium (W3C) has begun work on <b>HTML5.1</b>,
and this time it is handling the creation of the standard a little differently. The specification has
its <b><a href=""https://w3c.github.io/html/"">own GitHub project</a></b> where anyone can see what is
happening and propose changes.</p>
                    <p>The organization says the goal for the new specification is ""to <b>match
reality better</b>, to make the specification as clear as possible to readers, and of course to make it
possible for all stakeholders to propose improvements, and understand what makes changes to HTML
successful.""</p>
                    <p>Creating HTML5 took years, but W3C hopes using GitHub will speed up the process
this time around. It plans to release a candidate recommendation for HTML5.1 by <b>June</b> and a full
recommendation in <b>September</b>.</p>",
                    date: new DateTime(2016, 03, 27, 17, 53, 48),
                    authorUsername: "merry@gmail.com"
                );

                CreatePost(context,
                    title: "Windows 10 Preview with Bash Support Now Available",
                    body: @"<p>Microsoft has released a new <b>Windows 10 Insider Preview</b> that
includes native support for <b>Bash running on Ubuntu Linux</b>. The company first announced the new
feature at last week''s Build development conference, and it was one of the biggest stories of the
event. The current process for installing Bash is a little complication, but Microsoft has a blog
post that explains how the process works.</p>
                    <p>The preview build also includes <b>Cortana</b> upgrades, extensions support, the
new <b>Skype</b> Universal Windows Platform app and some interface improvements.</p>",
                    date: new DateTime(2016, 05, 11, 08, 22, 03),
                    authorUsername: "merry@gmail.com"
                );
```

```
                CreatePost(context,
                    title: "Atom Text Editor Gets New Windows Features",
                    body: @"<p>GitHub has released <b>Atom 1.7</b>, and the updated version of the text
editor offers improvements for Windows developers. Specifically, it is now easier to build in Visual
Studio, and it now supports the Appveyor CI continuous integration service for Windows.</p>
                    <p>Other new features include improved tab switching, tree view and crash recovery.
GitHub noted, ""Crashes are nobody''s idea of fun, but in case Atom does crash on you, it periodically
saves your editor state. After relaunching Atom after a crash, you should find all your work saved and
ready to go.""</p>
                    <p>GitHub has also released a beta preview of Atom 1.8.</p>",
                    date: new DateTime(2016, 03, 27, 17, 53, 48),
                    authorUsername: "merry@gmail.com"
                );

                CreatePost(context,
                    title: "SoftUni 3.0 Just Launched",
                    body: @"<p>The <b>Software University (SoftUni)</b> launched a new training
methodology and training program for software engineers in Sofia.</p>
                    <p>It is a big step ahead. Now SoftUni offers several professions:</p>
                    <ul>
                      <li>PHP Developer</li>
                      <li>JavaScript Developer</li>
                      <li>C# Web Developer</li>
                      <li>Java Web Developer</li>
                    </ul>",
                    date: new DateTime(2016, 02, 18, 22, 14, 38),
                    authorUsername: "pesho@gmail.com"
                );

                CreatePost(context,
                    title: "Git 2.8 Adds Security and Productivity Features",
                    body: @"<p>Version 2.8 of the open-source distributed version-control system Git
has been released. The new edition provides a variety of new features, bugfixes and other
improvements.</p>
                    <p>According to GitHub, the most notable new features include:</p>
                    <ul>
                    <li><strong>Parallel fetches of submodules:</strong> “Using ‘git submodules,’ one
Git repository can include other Git repositories as subdirectories. This can be a useful way to
include libraries or other external dependencies into your main project. The top-level repository
specifies which submodules it wants to include, and which version of each submodule,” wrote Jeff King,
a Git team member, in a <a href=""https://github.com/blog/2131-git-2-8-has-been-released"">blog
post</a>. According to him, if users have multiple submodules, fetches can be time-consuming. The
latest release allows users to fetch from multiple submodules in parallel.</li>
                    <li><strong>Don’t guess my identity: </strong>Instead of using one e-mail address
for all of a user’s open-source projects, they can now tell Git what user name and e-mail they want to
use before they commit.</li>
                    <li><strong>Convergences with Git for Windows:</strong> The Git team has been
working on making Git as easy to work with on Windows as it is on Linux and OS X. The latest release
includes Git commands rewritten in C; Windows-specific changes from the Git for Windows project; and
the ability to accept both LF and CRLF line endings. “This continuing effort will make it easier to
keep the functionality of Git in sync across platforms as new features are added,” King wrote.</li>
                    <li><strong>Security fixes: </strong>Git 2.8 addresses the vulnerability CVE-2016-
2324. There have not been any reported exploits, but the vulnerability could execute arbitrary code
when cloning a malicious repository, according to King.</li>
                    </ul>
                    <p>Other features include the ability to turn off Git’s clean and smudge filters;
the ability to see where a particular setting came from; the ability to easily diagnose end-of-line
problems; the ability to see a remote repository’s default branch; and support for cloning via the
rsync protocol has been dropped.</p>
                    <p>The full release notes are available <a
href=""https://github.com/git/git/blob/v2.8.0/Documentation/RelNotes/2.8.0.txt"">here</a>.</p>",
                    date: new DateTime(2016, 04, 11, 19, 02, 05),
                    authorUsername: "geshu@gmail.com"
                );

                CreatePost(context,
                    title: "Rogue Wave Updates Zend Framework",
```

```
                    body: @"<p>Rogue Wave is updating its open-source framework for developing Web
applications and services. According to the company, this is the first major release in four years.
Zend Framework 3 features support for PHP 7, middleware runtime and performance enhancements.</p>
                    <p>The newly released support for PHP 7 aims to simplify how developers create,
debug, monitor and deploy modern Web and mobile apps in PHP 7. “This is an exciting time to be a PHP
developer,” said Zeev Suraski, cofounder of Zend and CTO of Rogue Wave. “With Zend Framework 3, we’re
continuing our quest to make creating PHP applications simpler, more accessible and faster.”</p>
                    <p>In addition, version 3 of the framework features an architectural structure that
allows developers to use components within Zend Framework apps or any other framework in order to
reduce dependencies, and to enable reuse within the PHP ecosystem.</p>
                    <p>Another key update to the solution is a new middleware runtime. Expressive is
designed to focus on simplicity and interoperability, and it enables developers to customize their
solutions.</p>
                    <p>“I’m extremely proud of the work we’ve done with Expressive,” said Matthew Weier
O’Phinney, principal engineer and Zend Framework project lead at Rogue Wave. “Expressive signals the
future of PHP applications, composed of layered, single-purpose PSR-7 middleware.”</p>",
                    date: new DateTime(2016, 06, 30, 17, 36, 52),
                    authorUsername: "merry@gmail.com"
                );

                context.SaveChanges();
            }
        }

        private void CreateUser(ApplicationDbContext context,
            string email, string password, string fullName)
        {
            var userManager = new UserManager<ApplicationUser>(
                new UserStore<ApplicationUser>(context));
            userManager.PasswordValidator = new PasswordValidator
            {
                RequiredLength = 1,
                RequireNonLetterOrDigit = false,
                RequireDigit = false,
                RequireLowercase = false,
                RequireUppercase = false,
            };

            var user = new ApplicationUser
            {
                UserName = email,
                Email = email,
                FullName = fullName
            };

            var userCreateResult = userManager.Create(user, password);
            if (!userCreateResult.Succeeded)
            {
                throw new Exception(string.Join("; ", userCreateResult.Errors));
            }
        }

        private void CreateRole(ApplicationDbContext context, string roleName)
        {
            var roleManager = new RoleManager<IdentityRole>(
                new RoleStore<IdentityRole>(context));
            var roleCreateResult = roleManager.Create(new IdentityRole(roleName));
            if (!roleCreateResult.Succeeded)
            {
                throw new Exception(string.Join("; ", roleCreateResult.Errors));
            }
        }

        private void AddUserToRole(ApplicationDbContext context, string userName, string roleName)
        {
            var user = context.Users.First(u => u.UserName == userName);
            var userManager = new UserManager<ApplicationUser>(
                new UserStore<ApplicationUser>(context));
            var addAdminRoleResult = userManager.AddToRole(user.Id, roleName);
```

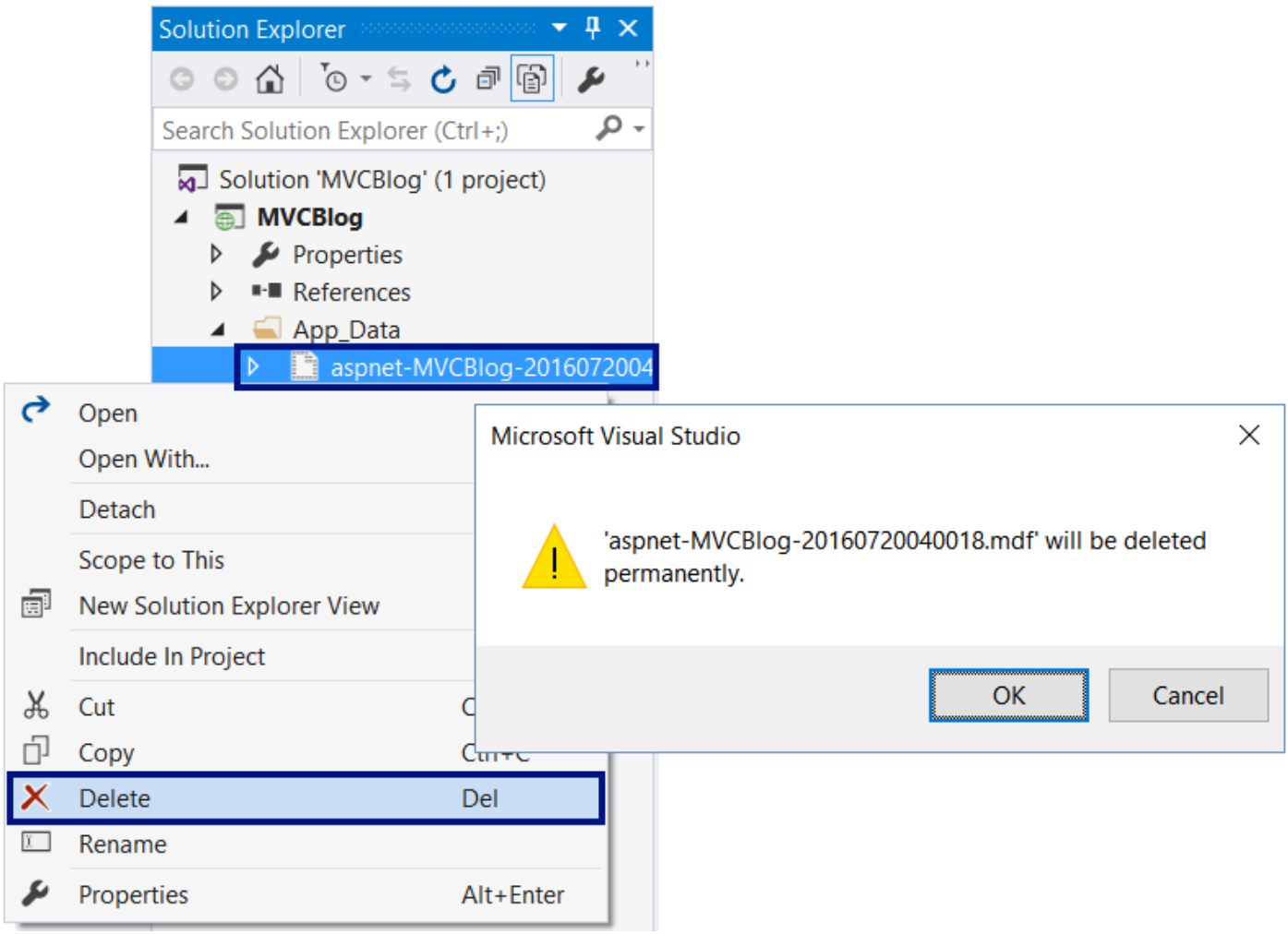
```
            if (!addAdminRoleResult.Succeeded)
            {
                throw new Exception(string.Join("; ", addAdminRoleResult.Errors));
            }
        }

        private void CreatePost(ApplicationDbContext context,
            string title, string body, DateTime date, string authorUsername)
        {
            var post = new Post();
            post.Title = title;
            post.Body = body;
            post.Date = date;
            post.Author = context.Users.Where(u => u.UserName == authorUsername).FirstOrDefault();
            context.Posts.Add(post);
        }
    }
}
```

In **Visual Studio** the code should look like this:



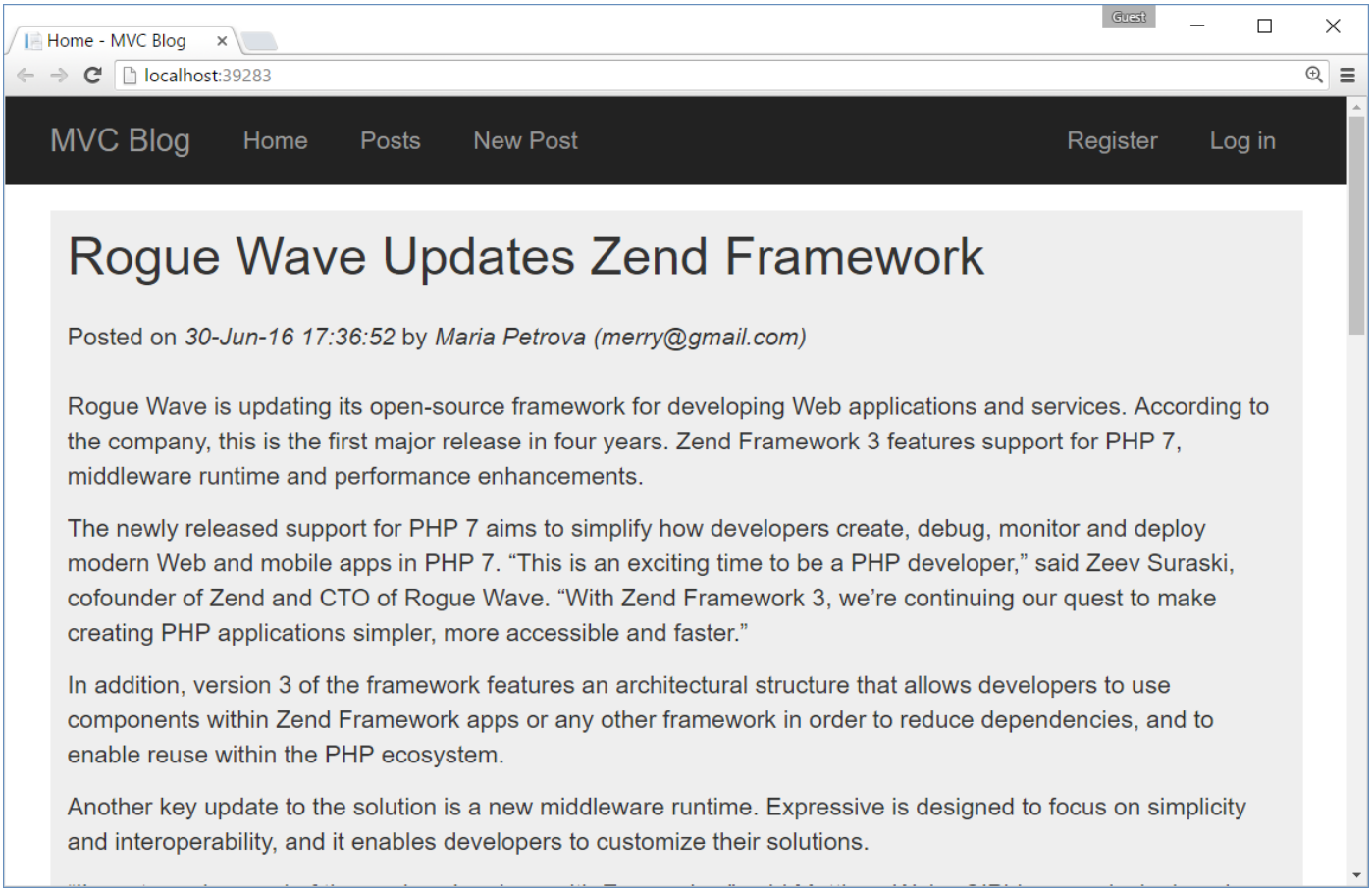
Drop the Database

If you run the application now, it will hang with a **database error**. This is due to a conflict caused by the changed database migration strategy. Just **drop the database** to ensure it is empty and there will be no conflict:



Run the App

Rebuild and **run** the application with **[Ctrl+F5]**. It should create a new database and fill some **users** and posts at **startup**. The home screen should look like this:



14. Implement "List Last 5 Posts" at the Sidebar

In the **`HomeController.Index()`** select the last 5 posts and put them in **`ViewBag.SidebarPosts`**.

In the view **`Views\Home\Index.cshtml`** render the sidebar posts (using a **`foreach`** loop) in a separate section (sidebar). Display links to **`\Posts\Details\{id}`**.

15. Add Security: Create Post for Registered Users Only

Put **`[Authorize]`** attribute before all **controller actions** allowed for authenticated users only (after login).

16. Add Security: Edit / Delete Post for Administrators Only

Put **`[Authorize(Roles="Administrators")]`** attribute before all actions aimed for administrators only.

17. Add Security: Hide Forbidden Links

At all pages where links are displayed, hide the links that are not valid for the current user:

- **Non-authenticated site visitors** should see: the home page, post details, login and register.
- **Logged-in users** should see additionally: create post, view all posts, edit own post, logout.
- **Administrators** should see additionally: edit / delete for all posts.

Just put **`@if`** in the views and some **`if`**-checks in the controllers. Checking for current user:

- **`if (User.Identity.IsAuthenticated) { … }`**
- **`if (!Roles.IsUserInRole(User.Identity.Name, "Administrators")) { … }`**

18. Add Security: Post Owners Can Edit Their Own Posts

Allow the **GET** and **POST** actions for **`PostsController.Edit(id)`** to be accessible by all authenticated users (not only administrators). Check the **ownership** of the specified post and **display error** if the current user does not own the specified post.

19. Add Authors in the Post Editor

Add **`Author_Id`** + Author fields in the **`Post`** data model class:

```
public class Post
{
    …
    public string Author_Id { get; set; }

    [ForeignKey("Author_Id")]
    public ApplicationUser Author { get; set; }
}
```

Add this code in the view **`Views\Posts\Edit.cshtml`**:

```
<select name="Author_Id">
@foreach (var author in ViewBag.Authors)
{
    <option value="@author.Id">@author.UserName</option>
}
</select>
```

Add **`Author_Id`** binding in **`PostsController.Edit(…)`**:

```
[HttpPost]
[ValidateAntiForgeryToken]
[Authorize(Roles = "Administrators")]
public ActionResult Edit([Bind(Include = "Id,Title,Body,Date,Author_Id")] Post post) { … }
```

20. Add Calendar for Date Selection

http://www.asp.net/mvc/overview/older-versions/using-the-html5-and-jquery-ui-datepicker-popup-calendar-with-aspnet-mvc/using-the-html5-and-jquery-ui-datepicker-popup-calendar-with-aspnet-mvc-part-4

…