Dependency Injection using Ninject in ASP.NET MVC

By Md. Mahedee Hasan, Software Architect, Leads
Web: **http://mahedee.net** ,
Source: http://mahedee.net/dependency-injection-using-ninject-in-asp-net-mvc/

What is Dependency Injection?
In software engineering, dependency injection is a software design pattern that implements inversion of control for resolving dependencies. – Wikipedia. It makes software components are loosely coupled.

Advantages of Dependency Injection

- Flexibility to use alternative implementation of service.
- Configurable & easy to use
- Make system loosely couple
- Code becomes more reusable, testable, readable and manageable.
- Reduction of boilerplate code in the application objects

What is Ninject?

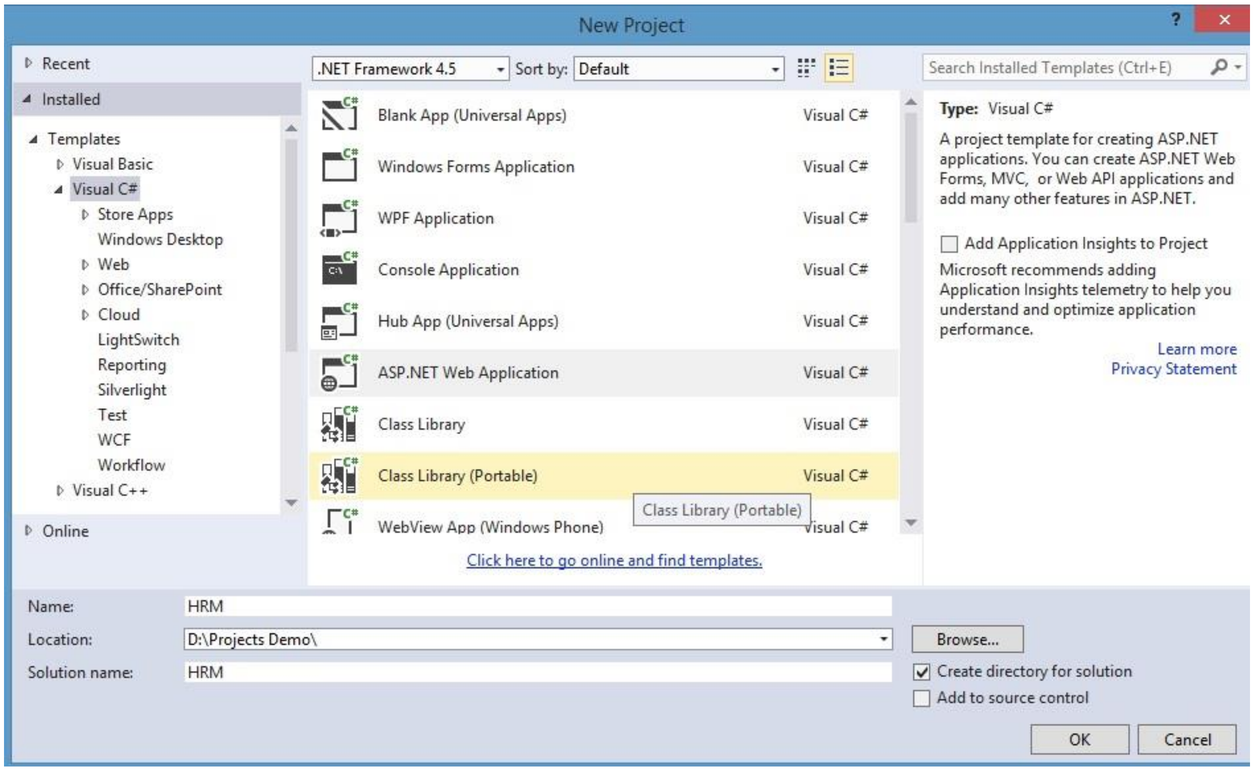
- Open Source Inversion of Control (IOC)
- It is a Dependency Injector for .NET created by Nate Kohari
- It's very easy to use.
- Easily add from nuget.
- For more information visit Ninject.org

Let's come to the implementation of DI using Ninject in ASP.NET MVC

Tools and Technology used
I used following tools and technology to develop the project –

1. Visual Studio 2013
2. Visual C#
3. ASP.NET MVC 5
4. Entity Framework 6
5. Razor view engine
6. Ninject for MVC5

Step 1: Create a ASP.net MVC Project
From Visual studio 2013, choose File->Project as below


New Project
Recent
Installed
Templates
Visual Basic
Visual C#
Store Apps
Windows Desktop
Web
Office/SharePoint
Cloud
LightSwitch
Reporting
Silverlight
Test
WCF
Workflow
Visual C++
Online
.NET Framework 4.5
Sort by: Default
Search Installed Templates (Ctrl+E)
Blank App (Universal Apps) Visual C#
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
Hub App (Universal Apps) Visual C#
ASP.NET Web Application Visual C#
Class Library Visual C#
Class Library (Portable) Visual C#
Class Library (Portable)
WebView App (Windows Phone) Visual C#
Click here to go online and find templates.
Learn more
Privacy Statement
Browse...
OK
Cancel

Type: Visual C#

A project template for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

Add Application Insights to Project

Microsoft recommends adding Application Insights telemetry to help you understand and optimize application performance.

Name: HRM

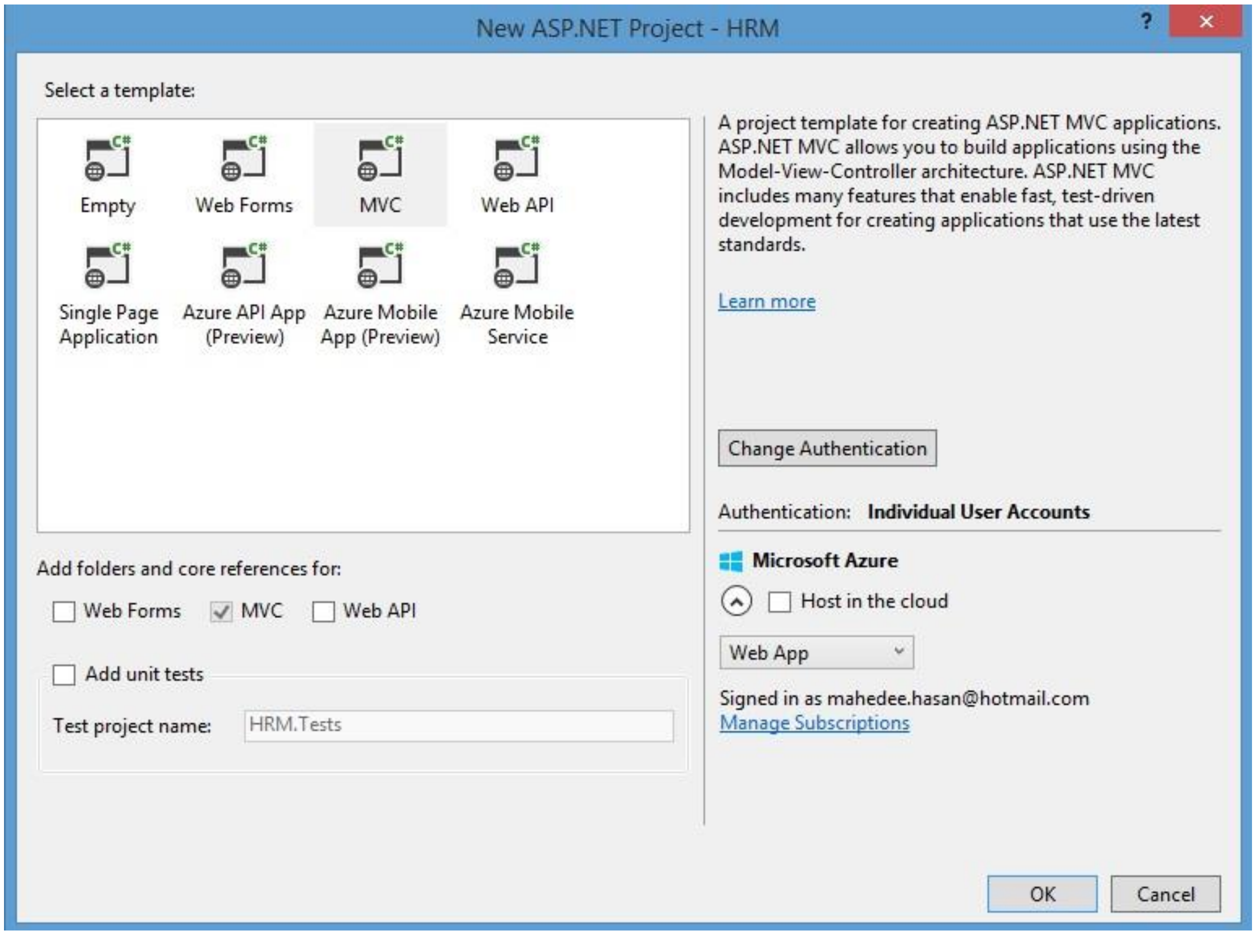
Location: D:\Projects Demo\

Solution name: HRM

[✓] Create directory for solution

[ ] Add to source control

Select MVC Template and click OK



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

Web App

Signed in as mahedee.hasan@hotmail.com

Manage Subscriptions

Add folders and core references for:

☐ Web Forms ✓ MVC ☐ Web API

☐ Add unit tests

Test project name: HRM.Tests

Step 2: Create a model name Employee

Create a Model name Employee in model folder

```
public class Employee
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Designation { get; set; }
    public string Dept { get; set; }
    public string BloodGroup { get; set; }
}
```

Step 3: Change or Add Connection String

Change or Add connection string in Web.config

```
<add name="DefaultConnection" connectionString="Data Source=localhost;Initial
Catalog=HRMDB;User ID=sa; Password=leads@123" providerName="System.Data.SqlClient" />
</connectionStrings>
```

Step 4: Create a Context class

Create HRMContext Class in Repository folder.

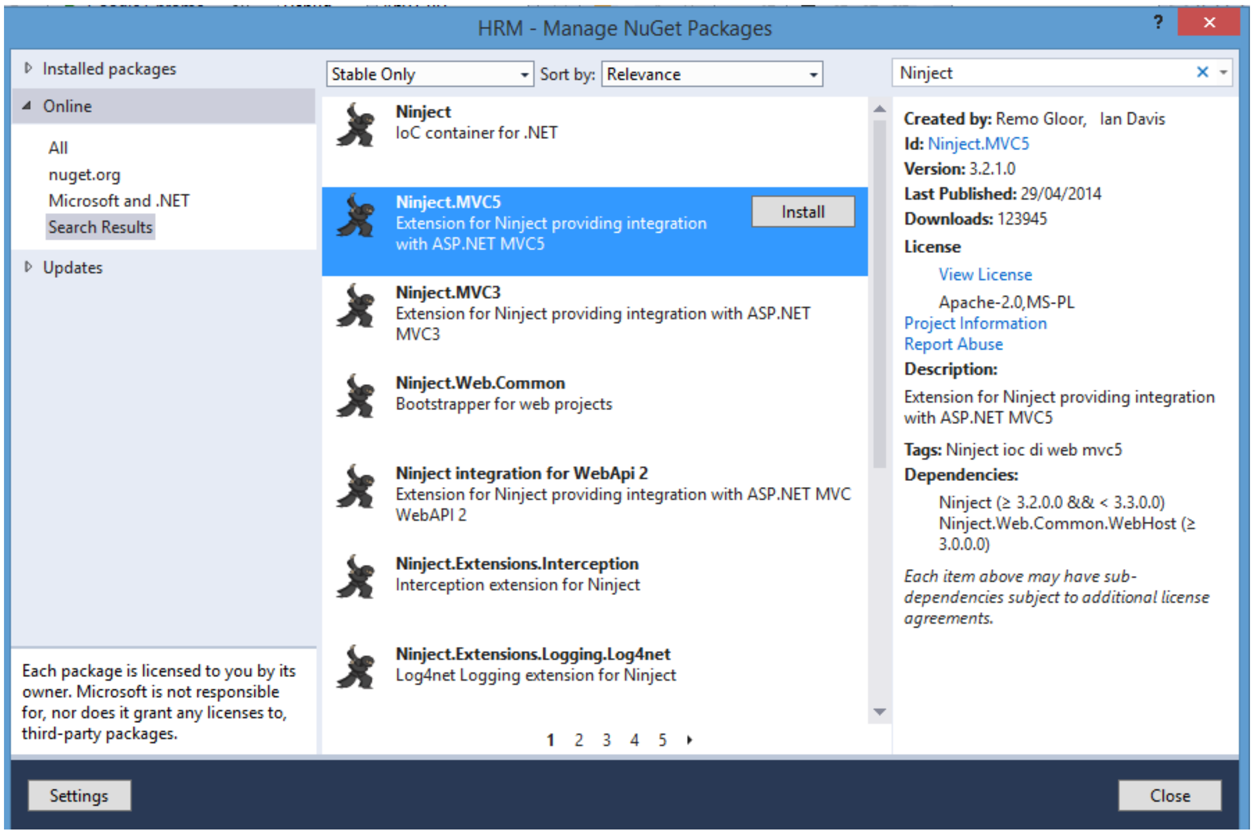
```
public class HRMContext : DbContext
{
    public HRMContext()
        : base("DefaultConnection")
    {
    }
    public DbSet<Employee> Employees { get; set; }
}
```

Step 5: Create Repository Interface and Concrete Class
Create IEmployeeRepository Interface and EmployeeRepository class in Repository folder

```
public interface IEmployeeRepository : IDisposable
{
    IQueryable<Employee> All { get; }
    Employee Find(int? id);
    void InsertOrUpdate(Employee employee);
    void Delete(int id);
    void Save();
}


public class EmployeeRepository : IEmployeeRepository
{

    HRMContext context;
    public EmployeeRepository(HRMContext context)
    {
        this.context = context;
    }

    public IQueryable<Employee> All
    {
        get { return context.Employees; }
    }

    public Employee Find(int? id)
    {
        Employee objEmployee = new Employee();
        objEmployee = context.Employees.Where(p => p.Id == id).FirstOrDefault();
        return objEmployee;
    }

    public void InsertOrUpdate(Employee employee)
    {
        if (employee.Id == default(int))
        {
            // New entity
            context.Employees.Add(employee);
        }
        else
        {
            // Existing entity
```

```
                context.Entry(employee).State = System.Data.Entity.EntityState.Modified;
            }
        }

        public void Delete(int id)
        {
            var employee = context.Employees.Find(id);
            context.Employees.Remove(employee);
        }

        public void Save()
        {
            context.SaveChanges();
        }

        public void Dispose()
        {
            context.Dispose();
        }
    }
```

Step 6: Install Ninject from nuget



Stable Only | Sort by: Relevance | Ninject

Ninject
IoC container for .NET

Ninject.MVC5
Extension for Ninject providing integration with ASP.NET MVC5

Ninject.MVC3
Extension for Ninject providing integration with ASP.NET MVC3

Ninject.Web.Common
Bootstrapper for web projects

Ninject integration for WebApi 2
Extension for Ninject providing integration with ASP.NET MVC WebAPI 2

Ninject.Extensions.Interception
Interception extension for Ninject

Ninject.Extensions.Logging.Log4net
Log4net Logging extension for Ninject

1 2 3 4 5

Created by: Remo Gloor, Ian Davis
Id: Ninject.MVC5
Version: 3.2.1.0
Last Published: 29/04/2014
Downloads: 123945
License
View License
Apache-2.0,MS-PL
Project Information
Report Abuse
Description:
Extension for Ninject providing integration with ASP.NET MVC5
Tags: Ninject ioc di web mvc5
Dependencies:
Ninject (≥ 3.2.0.0 && < 3.3.0.0)
Ninject.Web.Common.WebHost (≥ 3.0.0.0)

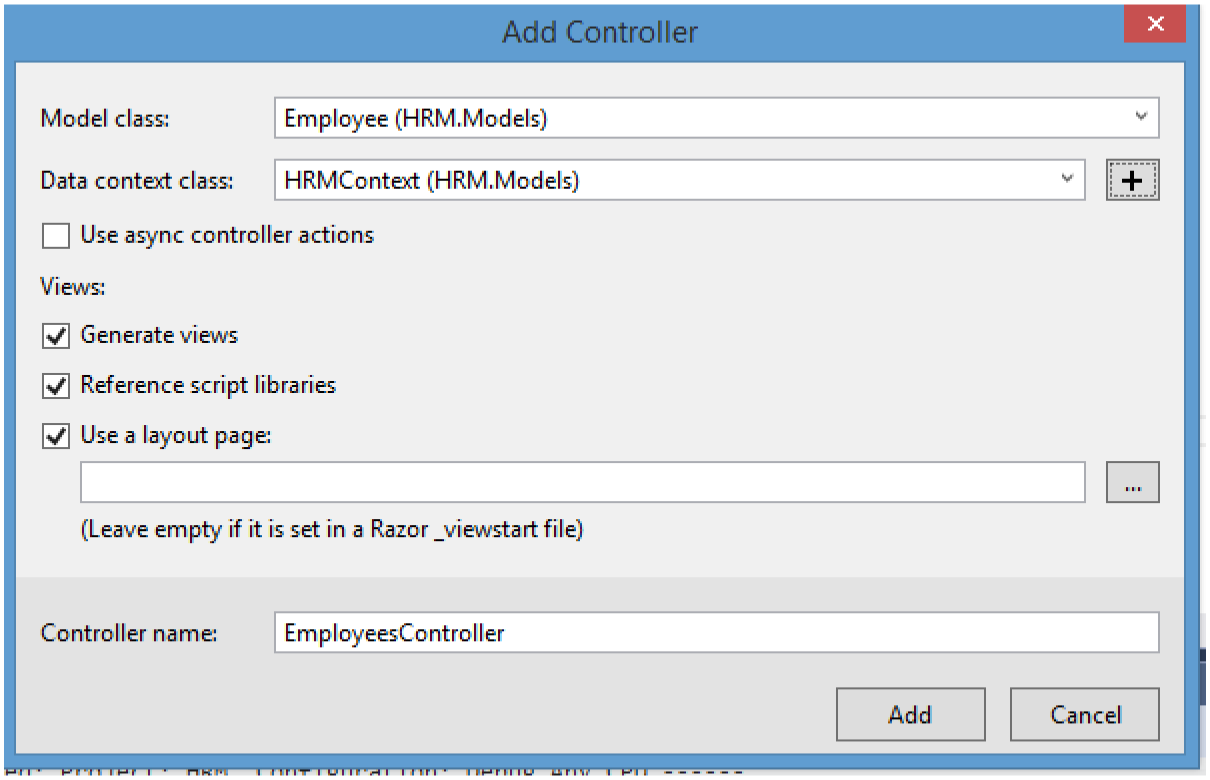
Each item above may have sub-dependencies subject to additional license agreements.

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

Step 7: Map Interface and Concrete class in Ninject
Go to NinjectWebCommon file in App_Start folder. Add the following line for IEmployee Interface and Employee concrete class.

```
public static class NinjectWebCommon
    {
        private static readonly Bootstrapper bootstrapper = new Bootstrapper();

        /// <summary>
        /// Starts the application
        /// </summary>
        public static void Start()
        {
            DynamicModuleUtility.RegisterModule(typeof(OnePerRequestHttpModule));
            DynamicModuleUtility.RegisterModule(typeof(NinjectHttpModule));
            bootstrapper.Initialize(CreateKernel);
        }

        /// <summary>
        /// Stops the application.
        /// </summary>
        public static void Stop()
        {
            bootstrapper.ShutDown();
        }

        /// <summary>
        /// Creates the kernel that will manage your application.
        /// </summary>
        /// <returns>The created kernel.</returns>
        private static IKernel CreateKernel()
        {
            var kernel = new StandardKernel();
            try
            {
                kernel.Bind<Func<IKernel>>().ToMethod(ctx => () => new
Bootstrapper().Kernel);
                kernel.Bind<IHttpModule>().To<HttpApplicationInitializationHttpModule>();

                kernel.Bind<IEmployeeRepository>().To<EmployeeRepository>();

                RegisterServices(kernel);
                return kernel;
            }
            catch
            {
                kernel.Dispose();
                throw;
            }
        }

        /// <summary>
        /// Load your modules or register your services here!
        /// </summary>
        /// <param name="kernel">The kernel.</param>
        private static void RegisterServices(IKernel kernel)
        {
        }
    }
```

Step 8: Create Controller and Views
Click Right button on Controller Folder->Add Controller. Choose its name as EmployeeController. Now choose scaffolding template for the controller as follows.



Model class: Employee (HRM.Models)

Data context class: HRMContext (HRM.Models)

☐ Use async controller actions

Views:

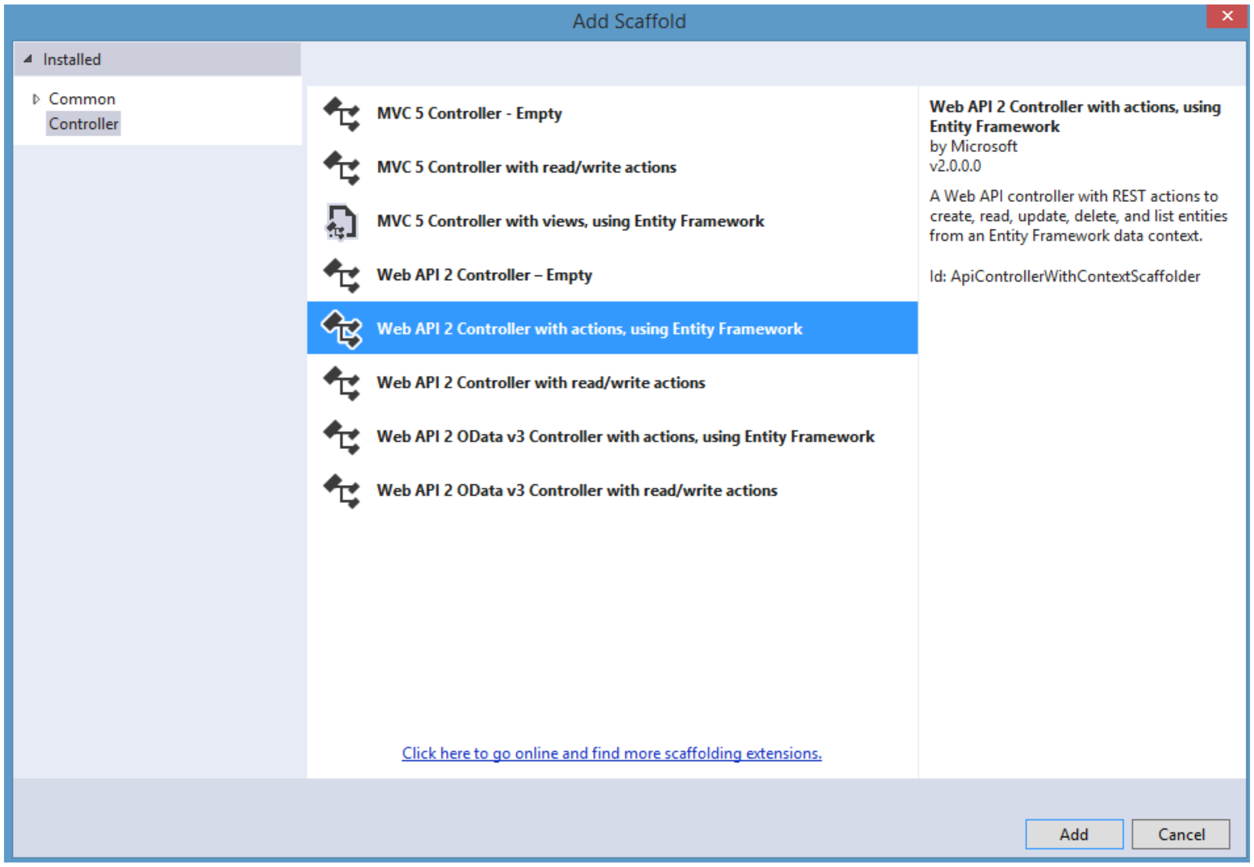
✓ Generate views

✓ Reference script libraries

✓ Use a layout page:

(Leave empty if it is set in a Razor _viewstart file)

Controller name: EmployeesController



MVC 5 Controller - Empty
MVC 5 Controller with read/write actions
MVC 5 Controller with views, using Entity Framework
Web API 2 Controller – Empty
Web API 2 Controller with actions, using Entity Framework
Web API 2 Controller with read/write actions
Web API 2 OData v3 Controller with actions, using Entity Framework
Web API 2 OData v3 Controller with read/write actions

Click here to go online and find more scaffolding extensions.

Web API 2 Controller with actions, using Entity Framework
by Microsoft
v2.0.0.0

A Web API controller with REST actions to create, read, update, delete, and list entities from an Entity Framework data context.

Id: ApiControllerWithContextScaffolder

After clicking Add button, Employee Controller and Corresponding actions and views will be created automatically.

Step 9: Modify the controller

Modify the controller - use repository instead of context directly.

```
    public class EmployeesController : Controller
    {
        private readonly IEmployeeRepository repository;

        public EmployeesController(IEmployeeRepository objIrepository)
        {
            repository = objIrepository;
        }

        // GET: Employees
        public ActionResult Index()
        {
            return View(repository.All.ToList());
        }

        // GET: Employees/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Employee employee = repository.Find(id);
            if (employee == null)
            {
                return HttpNotFound();
            }
            return View(employee);
        }

        // GET: Employees/Create
        public ActionResult Create()
        {
            return View();
        }

        // POST: Employees/Create
        // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include =
"Id,Name,Designation,Dept,BloodGroup")] Employee employee)
        {
            if (ModelState.IsValid)
            {
                repository.InsertOrUpdate(employee);
                repository.Save();
                return RedirectToAction("Index");
            }

            return View(employee);
        }

        // GET: Employees/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Employee employee = repository.Find(id);
            if (employee == null)
            {
                return HttpNotFound();
            }
            return View(employee);
        }

        // POST: Employees/Edit/5
        // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include = "Id,Name,Designation,Dept,BloodGroup")]
Employee employee)
        {
            if (ModelState.IsValid)
            {
                repository.InsertOrUpdate(employee);
                repository.Save();
                return RedirectToAction("Index");
            }
            return View(employee);
        }

        // GET: Employees/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Employee employee = repository.Find(id);
            if (employee == null)
            {
                return HttpNotFound();
            }
            return View(employee);
        }

        // POST: Employees/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            repository.Delete(id);
            repository.Save();
            return RedirectToAction("Index");
        }

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

Step 10: Run command in Package Manager Console

To find Package manager console go to

Tool->NuGet Package Manager -> Package Manager Console

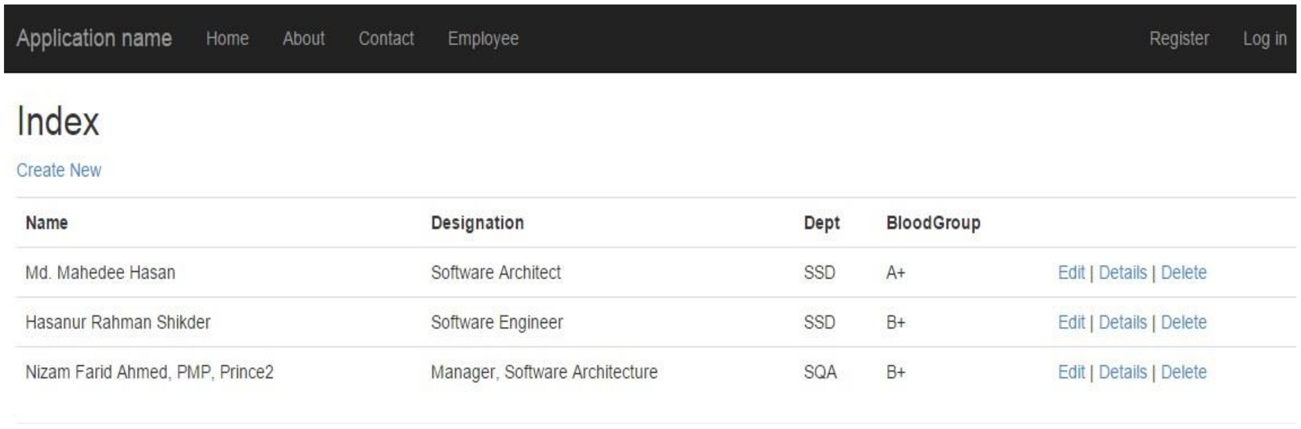
Now, build the application and run the following command one by one in Package Manager Console.

```
PM> Enable-Migrations -ContextTypeName HRMContext
PM> Add-Migration initialmigration
PM> Update-Database -Verbose
```

Step 10: Add a menu
Add a menu name employee in _Layout.cshtml page to create a menu.

```
<ul class="nav navbar-nav">
    <li>@Html.ActionLink("Home", "Index", "Home")</li>
    <li>@Html.ActionLink("About", "About", "Home")</li>
    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
    <li>@Html.ActionLink("Employee", "Index", "Employees")</li>
</ul>
```

Run the application and click “Employee” menu. Now you can create, delete, read update employee information.



Index

Create New

Name	Designation	Dept	BloodGroup	
Md. Mahedee Hasan	Software Architect	SSD	A+	Edit | Details | Delete
Hasanur Rahman Shikder	Software Engineer	SSD	B+	Edit | Details | Delete
Nizam Farid Ahmed, PMP, Prince2	Manager, Software Architecture	SQA	B+	Edit | Details | Delete