|  |  |  |
| --- | --- | --- |
| Name: Kaustubh Wade | ER No.: 160410116050 | Class: TYIT-1 Batch-C |

# Beyond The Syllabus Practical

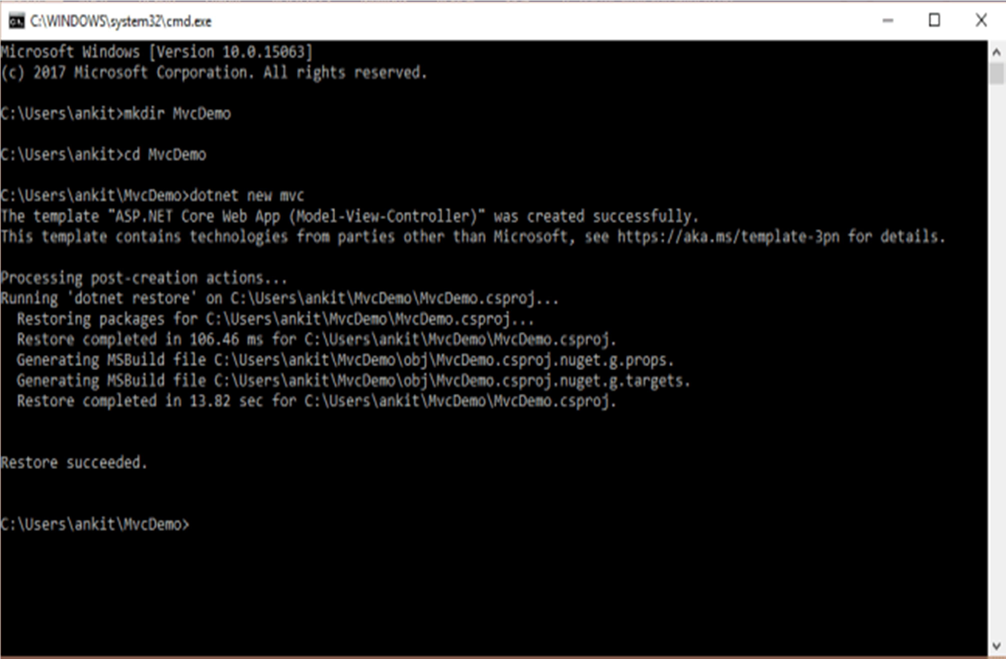
## AIM: Create an application by using MVC framework which performs CRUD operation.

Type the following Commands. It will create our MVC application, "MvcDemo":

mkdir MvcDemo

cd MvcDemo

dotnet new mvc



### Employees.cs

using System.;

using System.ComponentModel.DataAnnotations;

namespace Mvc.Models

{ public class Employees

{ public int Id

{ get; set; }

[Required]

public string Name

{ get; set; }

[Required]

public string City

{ get; set; }

[Required]

public string Department

{ get; set; }

[Required]

public int Salary

{ get; set; }

}

}

### MvcDemo.csproj

<Project Sdk="Microsoft.NET.Sdk.Web">

<PropertyGroup>

<TargetFramework>netcoreapp2.0</TargetFramework>

</PropertyGroup> <ItemGroup> <PackageReference Include="Microsoft.AspNetCore.All" Version="2.0.0" />

<PackageReference Include="Microsoft.VisualStudio.Web.CodeGeneration.Design" Version="2.0.0" /> </ItemGroup>

<ItemGroup>

<DotNetCliToolReference Include="Microsoft.VisualStudio.Web.CodeGeneration.Tools" Version="2.0.0" />

<DotNetCliToolReference Include="Microsoft.EntityFrameworkCore.Tools.DotNet" Version="2.0.0" />

</ItemGroup>

</Project>

### MVCEmployeeContext.cs

using Microsoft.EntityFrameworkCore; namespace MvcDemo.Models

{ public class MvcEmployeeContext : DbContext

{ public MvcEmployeeContext (DbContextOptions<MvcEm ployeeContext> options) : base(options)

{ }

public DbSet<MvcDemo.Models.Employees> Employee

{ get; set; }

}

}

### Startup.cs

using System;

using System.Collections.Generic; using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Builder; using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection; using Microsoft.EntityFrameworkCore;

using MvcDemo.Models; namespace MvcDemo

{ public class Startup

{ public Startup(IConfiguration configuration)

{ Configuration = configuration; }

public IConfiguration Configuration

{ get; }

public void ConfigureServices(IServiceCollectionservices)

{ services.AddMvc(); services.AddDbContext<MvcEmployeeContext>(opt

ions =>options.UseSqlite("Data Source=MvcEmployee.db"));

}

// This method gets called by the runtime. Use th is method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IH ostingEnvironment env)

{ if(env.IsDevelopment())

{ app.UseDeveloperExceptionPage(); }

else

{ app.UseExceptionHandler("/Home/Error"); }

app.UseStaticFiles();

app.UseMvc(routes =>

{ routes.MapRoute( name: "default", template: "{controller=Home}/{action==>{ Index}/{id?}");

}

}

}

### Run the following commands:

• dotnet restore

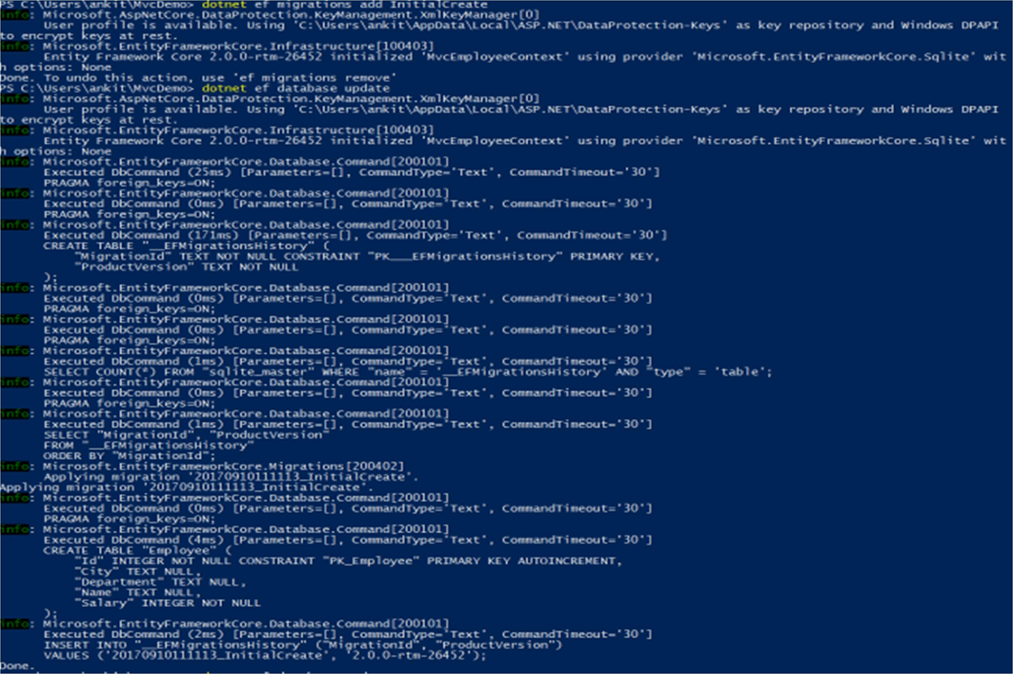
dotnet aspnet-codegenerator controller -name EmployeeCont roller -m Employees -dc MvcEmployeeContext --relativeFold erPath Controllers --useDefaultLayout –referenceScriptLibraries

### Run the following commands:

• dotnet ef migrations add InitialCreate

• dotnet ef database update

The dotnet ef migrations add InitialCreate command generates code to create the initial database schema. The schema is based on the model specified in the DbContext (in the Models/MVCEmployeeContext.cs file).



After running the second command you will get a message at end "Done." And that's it. We have created our first ASP.NET Core MVC application.

Before running the application, open launch.json and make sure that 'Program' path is set correctly:

"program": "${workspaceRoot}/bin/Debug/netcoreapp2.0/MvcD emo.dll"

Now your launch.json will look like this:

{

// Use IntelliSense to find out which attributes exist for C# debugging

// Use hover for the description of the existing attri butes

// For further information visit https://github.com/OmniSharp/omnisharp-vscode/blob/master/debugger-launchjson. md

"version": "0.2.0", "configurations": [

"name": ".NET Core Launch (web)",

"type": "coreclr",

"request": "launch", "preLaunchTask": "build",

// If you have changed target frameworks, mak e sure to update the program path.

"program": "${workspaceRoot}/bin/Debug/netcor eapp2.0/MvcDemo.dll", "args": [],

"cwd": "${workspaceRoot}", "stopAtEntry": false, "internalConsoleOptions": "openOnSessionStart", "launchBrowser": { "enabled": true, "args": "${auto-detect-url}", "windows": { "command": "cmd.exe",

"args": "/C start ${auto-detect-url}"

},

"osx":

{ "command": "open" },

"linux":

{ "command": "xdg-open" }

},

"env":

"ASPNETCORE\_ENVIRONMENT": "Development"

},

"sourceFileMap":

{ "/Views": "${workspaceRoot}/Views" }

},

{ "name": ".NET Core Attach", "type": "coreclr",

"request": "attach",

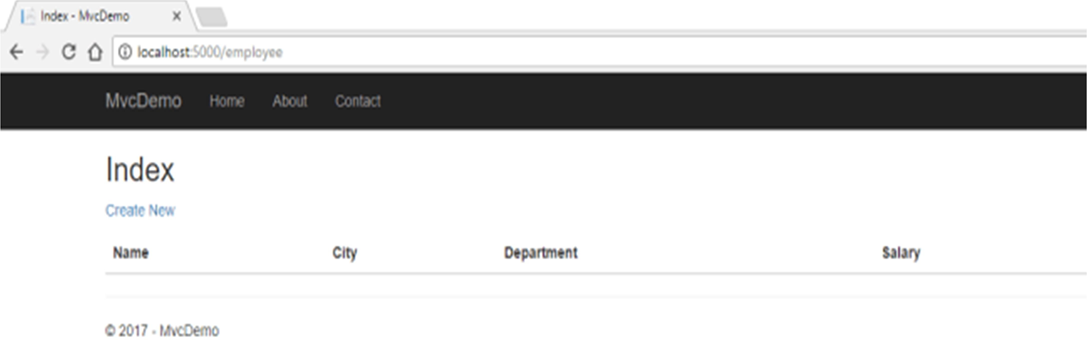
"processId": "${command:pickProcess}"

}

}

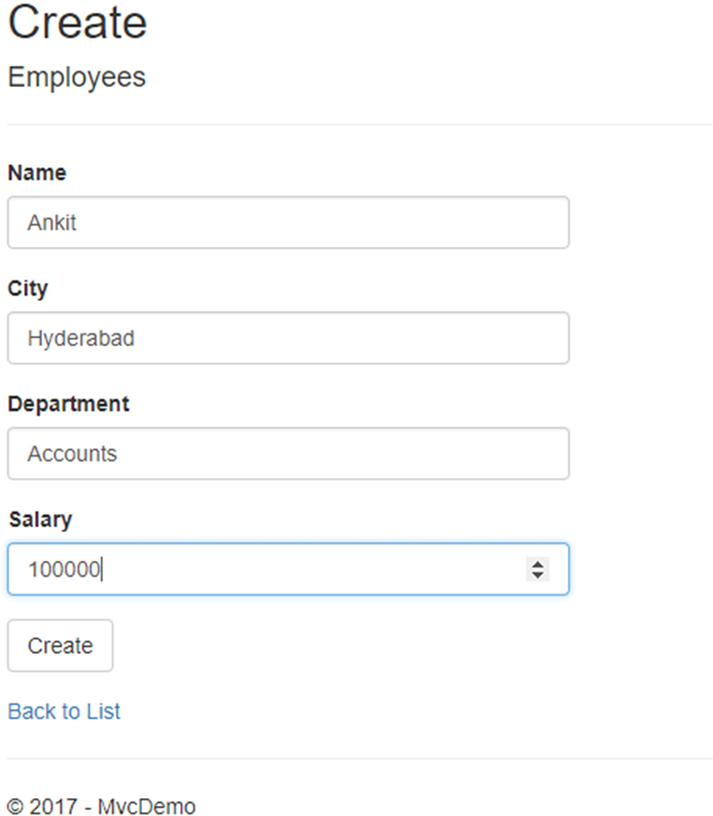
}

### Output:

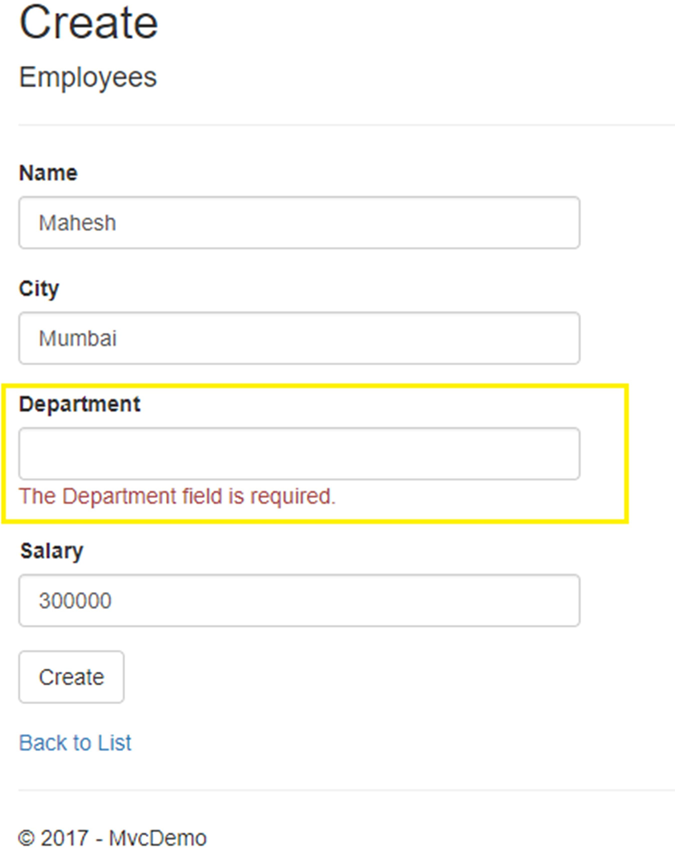


Now we will proceed with our CRUD operations.

Click on CreateNew to create a new Employee record. Add a new Employee record as shown in the image below.



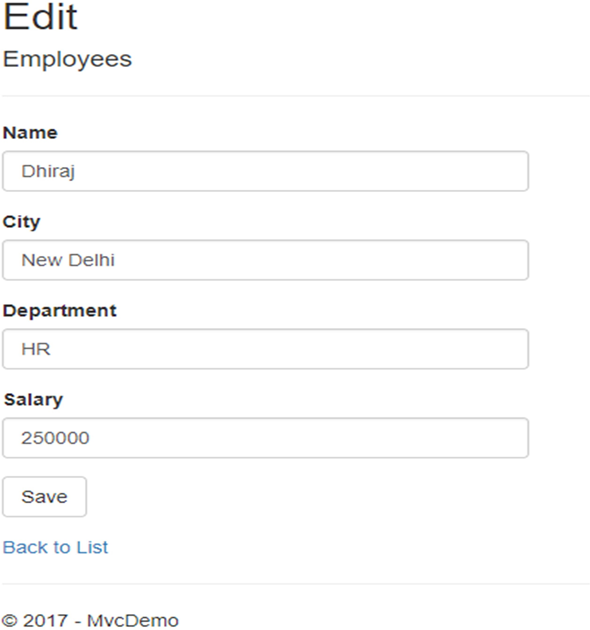
If we miss data in any fields then we will get a required field validation message.



After You click on the Create button, in Create View it will redirect us to the Index view where we can see all the employees added by us. Here, we can also see the action methods Edit, Details, and Delete.



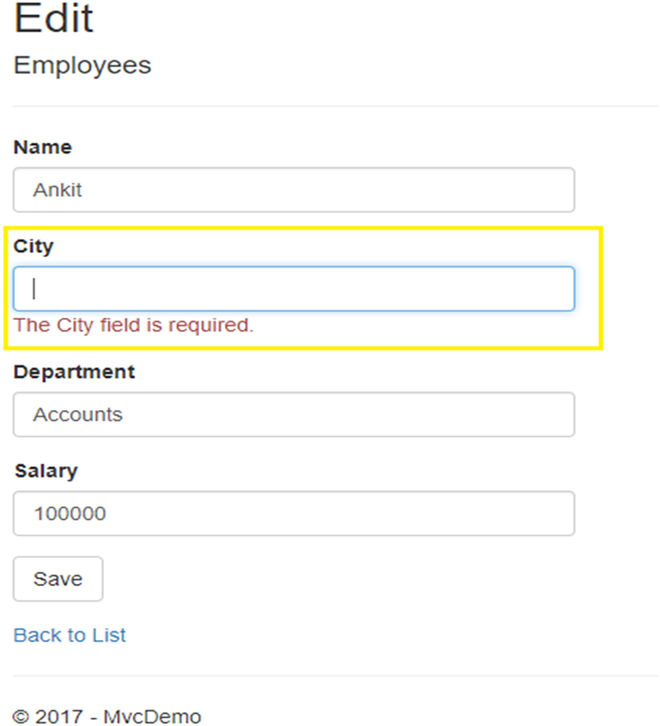
If we want to edit any existing employee records, then click the Edit action link. This will open the Edit view, as shown below, where we can change the employee data.



Here, we have changed the Salary of the employee with name Dhiraj from 200000 to 250000. Click on Save to return to the Index view and see the updated changes as highlighted in the image below.



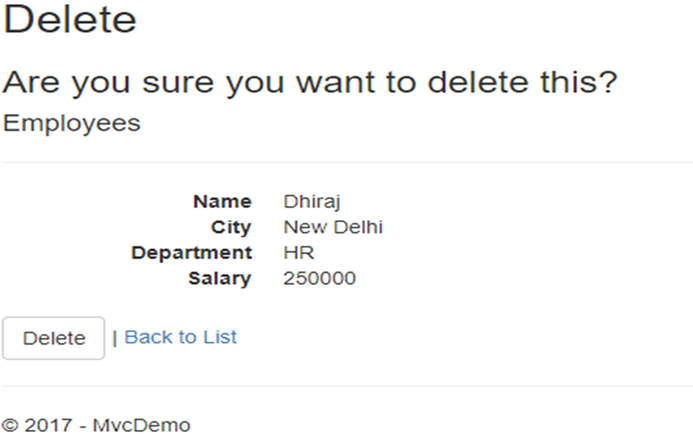
If we miss any fields while editing employee records, then the edit view will also throw the required field validation error message.



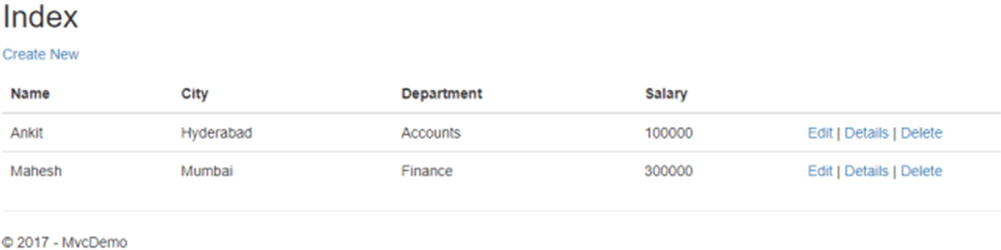
If you want to see the details of any Employee, then click on the Details action link, which will open the Details view as shown in the image below.



Click on Back to List to go back to the Index view. Now, we will perform the Delete operation on the Employee with the name Dhiraj. Click on the Delete action link which will open the Delete view and ask for a confirmation to delete.



Once we click on the Delete button, the employee record gets deleted and we will be redirected to the Index view. Here, we can see that the employee with the name Dhiraj has been removed from our record.



### EmployeeController.cs

// GET: Employee/Delete/5

public async Task<IActionResult> Delete(int? id)

{ if (id == null)

{ return NotFound(); }

var employees = await \_context.Employee

.SingleOrDefaultAsync(m => m.Id == id); if (employees == null)

{ return NotFound(); }

return View(employees);

}

// POST: Employee/Delete/5 [HttpPost, ActionName("Delete")] [ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{ var employees = await \_context.Employee.Single

OrDefaultAsync(m => m.Id == id);

\_context.Employee.Remove(employees);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}