| **Name of Student:** Ajay Karthikesan | | | |
| --- | --- | --- | --- |
| **Roll Number:** 57 | | **Assignment Number:** 10 | |
| **Aim of Assignment:**  Model View Controller  a. Create a MVC application to demonstrate the use of  Httpget and Httppost.  b. Create a MVC application to Edit,Create, Delete and to  display the data from the database using ADO.net Entity  Model. | | | |
| **DOP:** | | **DOS:** 3.6.23 | |
| **CO Mapped:**  CO4 | **PO Mapped:**  PO3, PO5, PSO1, PSO2 | **Faculty Signature:** | **Marks:** |

## 

## Practical No. 10

**Aim:** Design MVC based Web applications

1. Create a MVC application to demonstrate the use of Httpget and Httppost.
2. Create a MVC application to Edit,Create, Delete and to display the data from the database using ADO.net Entity Model.

**Theory:**

MVC - Model View Controller:

The Model-View-Controller (MVC) is an architectural pattern

that separates an application into three main logical components:

the model, the view, and the controller. Each of these

components is built to handle specific development

aspects of an application. MVC is one of the most frequently

used industry standard web development frameworks to create

scalable and extensible projects.

MVC Components

- Model

The Model component corresponds to all the data-related

logic that the user works with. This can represent either the

data that is being transferred between the View and

Controller components or any other business logic related

data. For example, a customer object will retrieve the

customer information from the database, manipulate it and

update it data back to the database or use it to render data.

- View

The View component is used for all the UI logic of the

application. For example, the Customer view will include

all the UI components such as text boxes, dropdowns, etc.

that the final user interacts with.

- Controller

Controllers act as an interface between Model and View

components to process all the business logic and incoming

requests, manipulate data using the Model component, and

interact with the Views to render the final output.

**Code:**

Model: Employee.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Practical\_10A.Models

{

public class Employee

{

public string Name { get; set; }

public string Address { get; set; }

public int Age { get; set; }

} }

Controller: HomeController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using Practical\_10A.Models;

namespace Practical\_10A.Controllers

{

public class HomeController : Controller

{

[HttpGet]

public ActionResult Index()

{

Employee emp = new Employee()

{

Address = "Nerul",

Name = "Ajay",

Age = 89

};

return View(emp);

}

[HttpPost]

public ActionResult Index(Employee emp)

{

return View("DisplayData", emp);

}

} }

Views:

Index.cshtml

@model Practical\_10A.Models.Employee

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Index</title>

</head>

<body>

<div>

@using (Html.BeginForm("Myform"))

{

@Html.TextBoxFor(x => x.Name) <br /> <br /> <br />

@Html.TextBoxFor(x => x.Address) <br /> <br /> <br />

@Html.TextBoxFor(x => x.Age) <br /> <br /> <br />

<input type="submit" value="submit" />

}

</div>

</body>

</html>

DisplayData.cshtml

@model Practical\_10A.Models.Employee

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>DisplayData</title>

</head>

<body>

<div>

<h1>Employee Details</h1>

<p>Employee Name: @Model.Name</p>

<p>Employee Age: @Model.Age</p>

<p>Employee Address: @Model.Address</p>

</div>

</body>

</html>

B.

Step 1: Create a new ASP.Net application with MVC in core references selected and add ADO.Net Entity Data Model Select an EF designer from the DB model.

Step 2: Make a connection, test connection, and select desired database.

Step 3: Choose the desired Table which shows later in a.edmx file.

Step 4: Right-click on the controller and add a new controller and select the one with MVC5 with views and click add.

Step 5: Ensure the routeConfig.cs file in AppStart folder has correct details of controller and view you wish to run.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using System.Web.Routing;

namespace Practical\_10B

{

public class RouteConfig

{

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "Students", action =

"Index", id = UrlParameter.Optional }

);

}

}

}

Step 6: Now run the generated code file.

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Entity;

using System.Linq;

using System.Net;

using System.Web;

using System.Web.Mvc;

using Practical\_10B;

namespace Practical\_10B.Controllers

{

public class StudentsController : Controller

{

private Vesit\_40Entities db = new Vesit\_40Entities();

// GET: Students

public ActionResult Index()

{

return View(db.Students.ToList());

}

// GET: Students/Details/5

public ActionResult Details(int? id)

{

if (id == null)

{

return new

HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Student student = db.Students.Find(id);

if (student == null)

{

return HttpNotFound();

}

return View(student);

}

// GET: Students/Create

public ActionResult Create()

{

return View();

}

// POST: Students/Create

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include =

"Roll\_No,Name,Programme,Course")] Student student)

{

if (ModelState.IsValid)

{

db.Students.Add(student);

db.SaveChanges();

return RedirectToAction("Index");

}

return View(student);

}

// GET: Students/Edit/5

public ActionResult Edit(int? id)

{

if (id == null)

{

return new

HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Student student = db.Students.Find(id);

if (student == null)

{

return HttpNotFound();

}

return View(student);

}

// POST: Students/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include =

"Roll\_No,Name,Programme,Course")] Student student)

{

if (ModelState.IsValid)

{

db.Entry(student).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

return View(student);

}

// GET: Students/Delete/5

public ActionResult Delete(int? id)

{

if (id == null)

{

return new

HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Student student = db.Students.Find(id);

if (student == null)

{

return HttpNotFound();

}

return View(student);

}

// POST: Students/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

Student student = db.Students.Find(id);

db.Students.Remove(student);

db.SaveChanges();

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

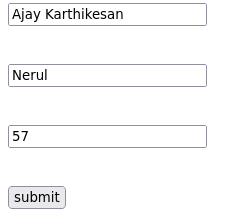
}

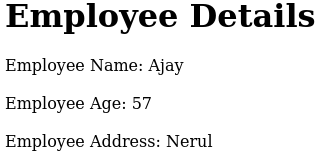
}

}

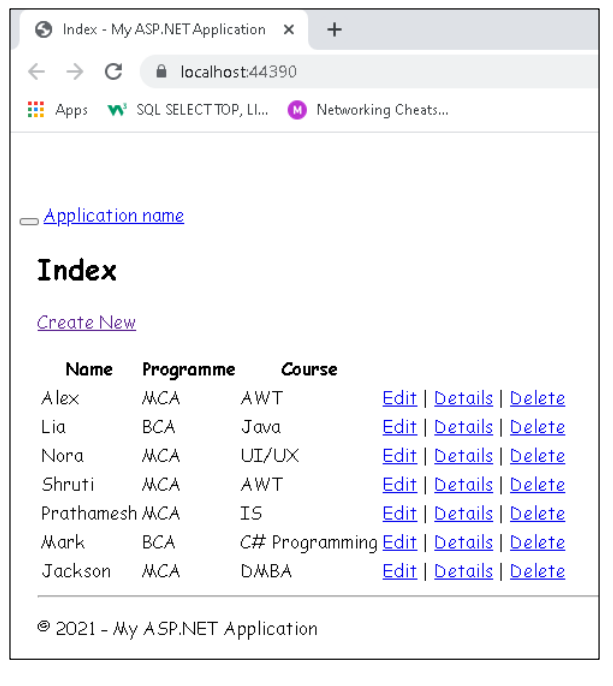
**Output:**

A.





B.



**Conclusion:**

We have successfully created MVC Application to:

A. Demonstrate the usage of HTTP Get & HTTP Post

B. Edit, Create, Delete, and display the data from the database

using ADO.net Entity Model.