**Takeaways: WP + MVC**

* HTML/CSS/JavaScript in separate files from PHP (and each other)
* Application logic and business logic in separate functions
* Action callbacks are usually controllers
* Filter callbacks are usually models
* Output buffering to capture views into a string
* Single callback/view to markup settings, etc

Filter callbacks are usually models

public static function add\_custom\_cron\_intervals( $schedules ) {

$schedules[ 'ten\_minutes' ] = array(

'interval' => 60 \* 10,

'display' => 'Every 10 minutes'

);

return $schedules;

}

add\_filter( 'cron\_schedules', \_\_CLASS\_\_ . '::add\_custom\_cron\_intervals' );

Action callbacks are usually controllers

public static function save\_user\_fields( $userID ) {

$userFields = self::validate\_user\_fields( $userID, $\_POST );

update\_user\_meta( $userID, 'user-example-field1', $userFields[ 'user-example-field1' ] );

update\_user\_meta( $userID, 'user-example-field2', $userFields[ 'user-example-field2' ] );

}

add\_action( 'save\_post', \_\_CLASS\_\_ . '::save\_user\_fields' );

**Step 1 :**Create an MVC Application.

**Step 2:** Create Model Class

**Step 3:**  Create Controller.

**Step 4 :** Create Table and Stored procedures.

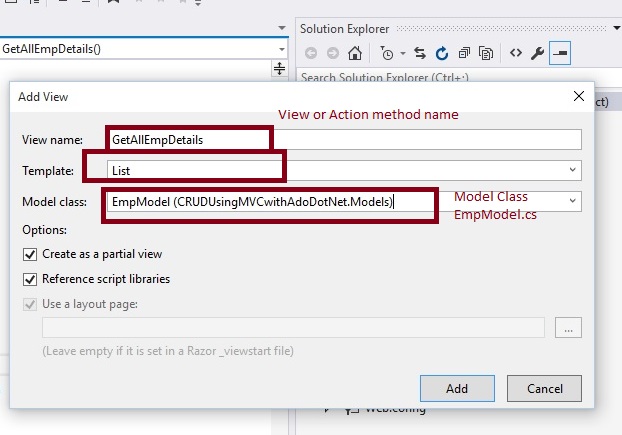
**Step 5:** Create Repository class.

**Step 6 :** Create Methods into the EmployeeController.cs file.

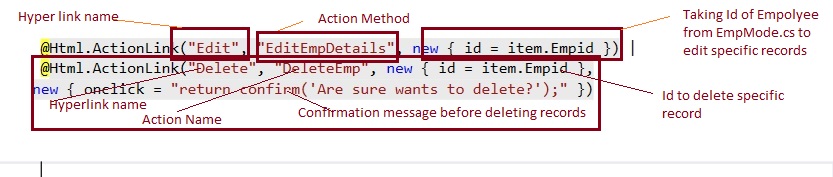
**Step 7:**Create Views.

**Create the Partial view to Add the employees**

**To View Added Employees**



**Step 8 :** Configure Action Link to Edit and delete the records as in the following figure:



**Step 9:** Configure RouteConfig.cs to set default action

**Step 10:**Run the Application

<https://www.c-sharpcorner.com/UploadFile/0c1bb2/insertupdatedelete-in-Asp-Net-mvc-5-using-ado-net/>

--To Insert Records

Create procedure [dbo].[AddNewEmpDetails]

( @Name varchar (50), @City varchar (50), @Address varchar (50) ) as

begin

Insert into Employee values(@Name,@City,@Address)

End

--To View Added Records

Create Procedure [dbo].[GetEmployees] as

begin

select \*from Employee

End

--To Update Records

Create procedure [dbo].[UpdateEmpDetails]

( @EmpId int, @Name varchar (50), @City varchar (50), @Address varchar (50)) as

begin

Update Employee set Name=@Name, City=@City, Address=@Address where Id=@EmpId

End

--To Delete Records

Create procedure [dbo].[DeleteEmpById] ( @EmpId int ) as

begin

Delete from Employee where Id=@EmpId

End

using System; using System.Data; using System.Data.SqlClient; using System.Collections.Generic;

using System.Configuration; using CRUDUsingMVC.Models;

//.../Repository/EmpRepository.cs // DAL //Models

namespace CRUDUsingMVC.Repository

{

public class EmpRepository

{

private SqlConnection con;

//To Handle connection related activities

private void connection()

{

string constr = ConfigurationManager.ConnectionStrings["getconn"].ToString();

con = new SqlConnection(constr);

}

//To Add Employee details

public bool AddEmployee(EmpModel obj)

{

connection();

SqlCommand com = new SqlCommand("AddNewEmpDetails", con);

com.CommandType = CommandType.StoredProcedure;

com.Parameters.AddWithValue("@Name", obj.Name);

com.Parameters.AddWithValue("@City", obj.City);

com.Parameters.AddWithValue("@Address", obj.Address);

con.Open();

int i = com.ExecuteNonQuery();

con.Close();

if (i >= 1)

{ return true; }

else

{ return false; }

}

//To view employee details with generic list

public List<EmpModel> GetAllEmployees()

{

connection();

List<EmpModel> EmpList =new List<EmpModel>();

SqlCommand com = new SqlCommand("GetEmployees", con);

com.CommandType = CommandType.StoredProcedure;

SqlDataAdapter da = new SqlDataAdapter(com);

DataTable dt = new DataTable();

con.Open();

da.Fill(dt);

con.Close();

// Bind EmpModel generic list using dataRow

foreach (DataRow dr in dt.Rows)

{

EmpList.Add(

new EmpModel

{

Empid = Convert.ToInt32(dr["Id"]),

Name = Convert.ToString(dr["Name"]),

City = Convert.ToString(dr["City"]),

Address = Convert.ToString(dr["Address"])

} );

}

return EmpList;

}

//To Update Employee details

public bool UpdateEmployee(EmpModel obj)

{

connection();

SqlCommand com = new SqlCommand("UpdateEmpDetails", con);

com.CommandType = CommandType.StoredProcedure;

com.Parameters.AddWithValue("@EmpId", obj.Empid);

com.Parameters.AddWithValue("@Name", obj.Name);

com.Parameters.AddWithValue("@City", obj.City);

com.Parameters.AddWithValue("@Address", obj.Address);

con.Open();

int i = com.ExecuteNonQuery();

con.Close();

if (i >= 1)

{ return true; }

else

{ return false; }

}

//To delete Employee details

public bool DeleteEmployee(int Id)

{

connection();

SqlCommand com = new SqlCommand("DeleteEmpById", con);

com.CommandType = CommandType.StoredProcedure;

com.Parameters.AddWithValue("@EmpId", Id);

con.Open();

int i = com.ExecuteNonQuery();

con.Close();

if (i >= 1)

{ return true; }

else

{ return false; }

}

}

}

using System.Web.Mvc;

using CRUDUsingMVC.Models;

using CRUDUsingMVC.Repository;

namespace CRUDUsingMVC.Controllers

// ../Controllers/EmployeeController.cs

{

public class EmployeeController : Controller

{

// GET: Employee/GetAllEmpDetails

public ActionResult GetAllEmpDetails()

{

EmpRepository EmpRepo = new EmpRepository();

ModelState.Clear();

return View(EmpRepo.GetAllEmployees());

}

// GET: Employee/AddEmployee

public ActionResult AddEmployee()

{

return View();

}

// POST: Employee/AddEmployee

[HttpPost]

public ActionResult AddEmployee(EmpModel Emp)

{

try

{

if (ModelState.IsValid)

{

EmpRepository EmpRepo = new EmpRepository();

if (EmpRepo.AddEmployee(Emp))

{

ViewBag.Message = "Employee details added successfully";

}

}

return View();

}

catch

{ return View(); }

}

// GET: Employee/EditEmpDetails/5

public ActionResult EditEmpDetails(int id)

{

EmpRepository EmpRepo = new EmpRepository();

return View(EmpRepo.GetAllEmployees().Find(Emp => Emp.Empid == id));

}

// POST: Employee/EditEmpDetails/5

[HttpPost]

public ActionResult EditEmpDetails(int id,EmpModel obj)

{

try

{

EmpRepository EmpRepo = new EmpRepository();

EmpRepo.UpdateEmployee(obj);

return RedirectToAction("GetAllEmpDetails");

}

catch

{ return View(); }

}

// GET: Employee/DeleteEmp/5

public ActionResult DeleteEmp(int id)

{

try

{

EmpRepository EmpRepo = new EmpRepository();

if (EmpRepo.DeleteEmployee(id))

{

ViewBag.AlertMsg = "Employee details deleted successfully";

}

return RedirectToAction("GetAllEmpDetails");

}

catch

{ return View(); }

}

}

}

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 = "Employee", action = "AddEmployee", id = UrlParameter.Optional }

);

}

}

using System.ComponentModel.DataAnnotations;

//.../Models/EmpModel.cs

namespace CRUDUsingMVC.Models

{

public class EmpModel

{

[Display(Name = "Id")]

public int Empid { get; set; }

[Required(ErrorMessage = "First name is required.")]

public string Name { get; set; }

[Required(ErrorMessage = "City is required.")]

public string City { get; set; }

[Required(ErrorMessage = "Address is required.")]

public string Address { get; set; }

}

}

model CRUDUsingMVC.Models.EmpModel

@using (Html.BeginForm())

{

//.../view/AddEmployee.cshtml

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h4>Add Employee</h4>

<div>

@Html.ActionLink("Back to Employee List", "GetAllEmpDetails")

</div>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.LabelFor(model => model.Name, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.City, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.City, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.City, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Address, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Address, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Address, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Save" class="btn btn-default" />

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10" style="color:green">

@ViewBag.Message

</div>

</div>

</div>

}

<script src="~/Scripts/jquery-1.10.2.min.js"></script>

<script src="~/Scripts/jquery.validate.min.js"></script>

<script src="~/Scripts/jquery.validate.unobtrusive.min.js"></script>

@model CRUDUsingMVC.Models.EmpModel

@\* // …/view/EditEmpDetails.cshtml \*@

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h4>Update Employee Details</h4>

<hr />

<div>

@Html.ActionLink("Back to Details", "GetAllEmployees")

</div>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

@Html.HiddenFor(model => model.Empid)

<div class="form-group">

@Html.LabelFor(model => model.Name, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.City, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.City, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.City, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Address, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Address, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Address, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Update" class="btn btn-default" />

</div>

</div>

</div>

}

<script src="~/Scripts/jquery-1.10.2.min.js"></script>

<script src="~/Scripts/jquery.validate.min.js"></script>

<script src="~/Scripts/jquery.validate.unobtrusive.min.js"></script>

@model IEnumerable<CRUDUsingMVC.Models.EmpModel>

@\* GetAllEmpDetails.cshtml \*@

<p>

@Html.ActionLink("Add New Employee", "AddEmployee")

</p>

<table class="table">

<tr>

<th> @Html.DisplayNameFor(model => model.Name) </th>

<th> @Html.DisplayNameFor(model => model.City) </th>

<th> @Html.DisplayNameFor(model => model.Address) </th>

<th></th>

</tr>

@foreach (var item in Model)

{

@Html.HiddenFor(model => item.Empid)

<tr>

<td> @Html.DisplayFor(modelItem => item.Name) </td>

<td> @Html.DisplayFor(modelItem => item.City) </td>

<td> @Html.DisplayFor(modelItem => item.Address) </td>

<td>

@Html.ActionLink("Edit", "EditEmpDetails", new { id = item.Empid }) |

@Html.ActionLink("Delete", "DeleteEmp", new { id = item.Empid },

new { onclick = "return confirm('Are sure wants to delete?');" })

</td>

</tr>

}

</table>