**Crud Operation in Entity Framework Core In MVC**

**Step #01:**

Open Visual Studio >> Create New Project >> ASP.NET Core Web App (Model-View-Controller)>> Project Name >> choose Framework version >> OK.

After opening the project, first, we have to create a domain class. Right-click on Models folder and add a class like below. Adding Country, State, City & Employee class.

**Country Class:**

public class Country

{

public Country()

{

States = new Collection<State>();

Employees= new Collection<Employee>();

}

public int Id { get; set; }

[Required]

[Display(Name = "Country Name")]

public string CountryName { get; set; }

public ICollection<State> States { get; set; }

public ICollection<Employee> Employees { get; set; }

}

**State Class:**

public class State

{

public State()

{

Cities = new Collection<City>();

Employees= new Collection<Employee>();

}

public int Id { get; set; }

[Required]

[Display(Name = "State Name")]

public string StateName { get; set; }

[ForeignKey("Country")]

[Display(Name = "Country Name")]

public int CountryId { get; set; }

public Country Country { get; set; }

public ICollection<City> Cities { get; set; }

public ICollection<Employee> Employees { get; set; }

}

**City Class:**

public class City

{

public City()

{

Employees= new Collection<Employee>();

}

public int Id { get; set; }

[Required]

[Display(Name = "City Name")]

public string CityName { get; set; }

[ForeignKey("State")]

[Display(Name = "State Name")]

public int StateId { get; set; }

public State State { get; set; }

public ICollection<Employee> Employees { get; set; }

}

**Employee Class:**

public class Employee

{

public int Id { get; set; }

[Required]

public string Name { get; set; }

[Required]

public string Address { get; set; }

[Required]

public string Gender { get; set; }

public Boolean Ssc { get; set; }

public Boolean Hsc { get; set; }

public Boolean Bsc { get; set; }

public Boolean Msc { get; set; }

public string Picture { get; set; }

[Required]

[ForeignKey("Country")]

[Display(Name = "Country Name")]

public int CountryId { get; set; }

public Country Country { get; set; }

[Required]

[ForeignKey("State")]

[Display(Name = "State Name")]

public int StateId { get; set; }

public State State { get; set; }

[Required]

[ForeignKey("City")]

[Display(Name = "City Name")]

public int CityId { get; set; }

public City City { get; set; }

}

**Step #02**

Now, I am going to add a DbContext class. For that, right-click the Models folder and add a class and give the name AppDbContext:DbContext.

**AppDbContext Class:**

public class AppDbContext:DbContext

{

public AppDbContext(DbContextOptions<AppDbContext> options): base(options)

{

}

public DbSet<Country> Countries { get; set; }

public DbSet<State> States { get; set; }

public DbSet<City> Cities { get; set; }

public DbSet<Employee> Employees { get; set; }

}

Set our connection string in **appsettings.json:**

"ConnectionStrings": {

"DBConnection": "Server =KHPOLASH; user id=sa; password=khpolash; Database= CrudOperationDb; Integrated Security=True; "

}

Set connection in **Program.cs** file**:**

builder.Services.AddDbContext<AppDbContext>(options=> options.UseSqlServer(builder.Configuration.GetConnectionString("DBConnection")));

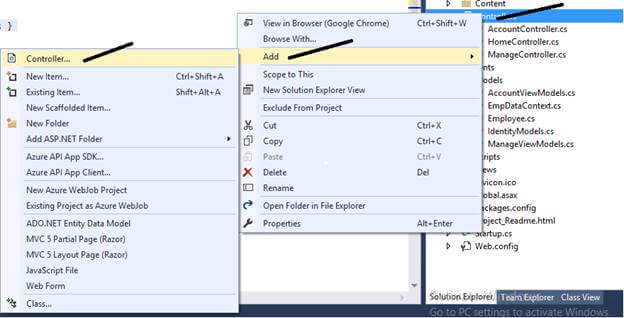
Now open Package Manager Console and run following two commands:

* add-migration “TableName”
* update-database

**Step #03:**

Now, we have to add a Controller. Go to Controllers folder and add a controller.

When we add a controller, in this controller an Index Action method is automatically created. We can change it to a user-friendly method name. Now, create the object of your Dbcontext class and write the logic for retrieval of the data.



public class EmployeeController : Controller

{

private readonly IWebHostEnvironment \_webHostEnvironment;

private readonly AppDbContext \_context;

public EmployeeController(AppDbContext context, IWebHostEnvironment webHostEnvironment)

{

\_context = context;

\_webHostEnvironment = webHostEnvironment;

}

// GET: Employee

public async Task<IActionResult> Index()

{

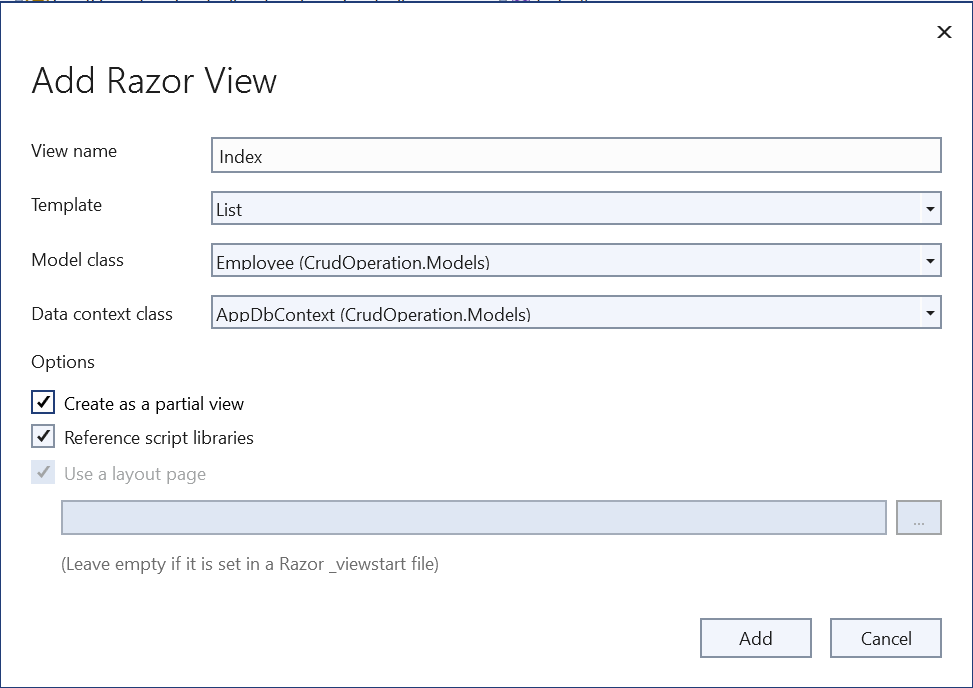
var appDbContext = \_context.Employees.Include(e => e.City).Include(e => e.Country).Include(e => e.State);

return View(await appDbContext.ToListAsync());

}

}

After that, add a View for displaying the employee records. Right-click on action method and add a View.



And, after that, we will write HTML code like below.

@model IEnumerable<CrudOperation.Models.Employee>

@{

ViewData["Title"] = "Employee List";

}

<h3>Employee List</h3>

<p>

<a class="btn btn-outline-primary btn-sm " **asp-action**="Create"><i class="bi bi-pencil-square"></i> Add New</a>

</p>

<table id="myTable" class="table table-striped-columns">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.Picture)

</th>

<th>

@Html.DisplayNameFor(model => model.Name)

</th>

<th>

@Html.DisplayNameFor(model => model.Gender)

</th>

<th colspan="4" style="text-align:center">

Education

</th>

<th>

@Html.DisplayNameFor(model => model.Address)

</th>

<th>

@Html.DisplayNameFor(model => model.Country)

</th>

<th>

@Html.DisplayNameFor(model => model.State)

</th>

<th>

@Html.DisplayNameFor(model => model.City)

</th>

<th>Action</th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>

<img class="img-thumbnail" width="50" src="~/images/@Html.DisplayFor(modelItem => item.Picture)" alt="Image" />

</td>

<td>

@Html.DisplayFor(modelItem => item.Name)

</td>

<td>

@Html.DisplayFor(modelItem => item.Gender)

</td>

<td>

@Html.DisplayNameFor(model => model.Ssc)

@Html.DisplayFor(modelItem => item.Ssc)

</td>

<td>

@Html.DisplayNameFor(model => model.Hsc)

@Html.DisplayFor(modelItem => item.Hsc)

</td>

<td>

@Html.DisplayNameFor(model => model.Bsc)

@Html.DisplayFor(modelItem => item.Bsc)

</td>

<td>

@Html.DisplayNameFor(model => model.Msc)

@Html.DisplayFor(modelItem => item.Msc)

</td>

<td>

@Html.DisplayFor(modelItem => item.Address)

</td>

<td>

@Html.DisplayFor(modelItem => item.Country.CountryName)

</td>

<td>

@Html.DisplayFor(modelItem => item.State.StateName)

</td>

<td>

@Html.DisplayFor(modelItem => item.City.CityName)

</td>

<td>

<a **asp-action**="Edit" **asp-route-id**="@item.Id"><i class="bi bi-pencil-square"></i></a> |

<a **asp-action**="Details" **asp-route-id**="@item.Id"><i class="bi bi-eye-fill"></i></a> |

<a **asp-action**="Delete" **asp-route-id**="@item.Id"><i class="bi bi-trash-fill"></i></a>

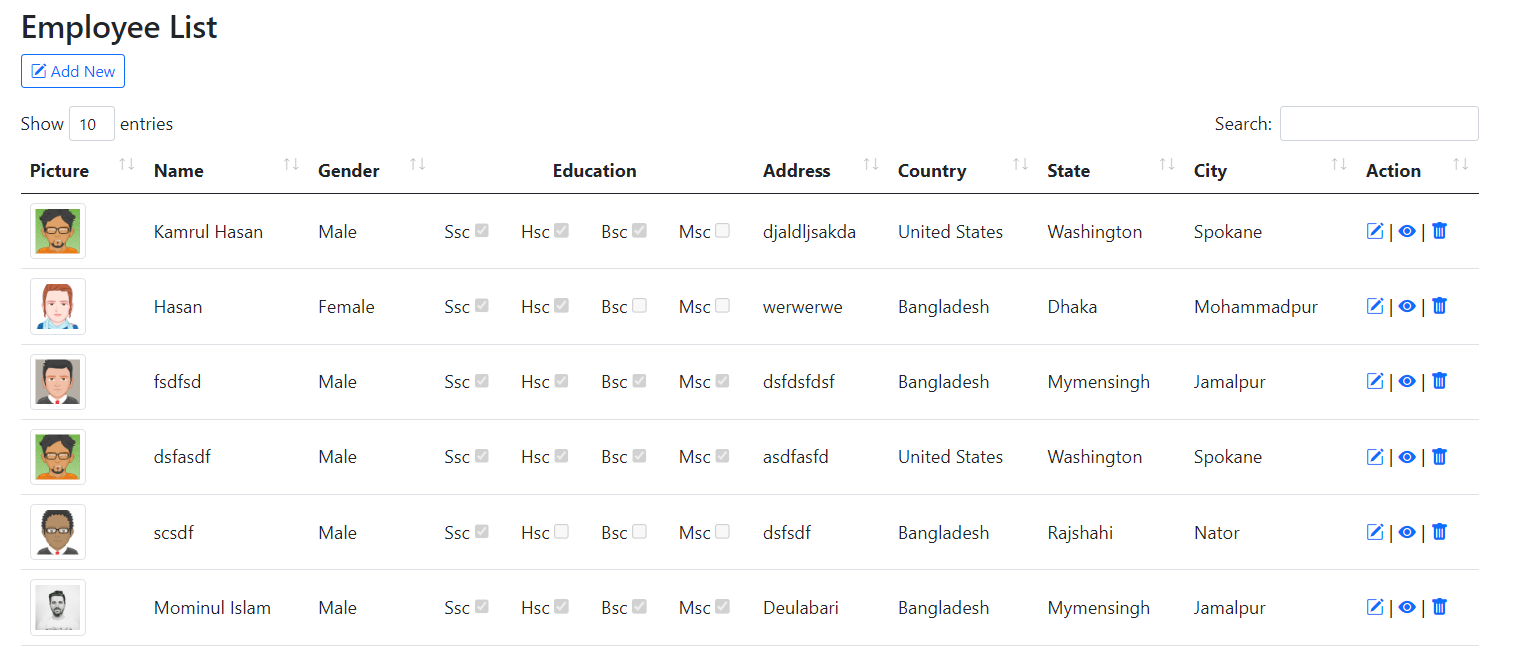
</td>

</tr>

}

</tbody>

</table>



**Create Operation:** We have to create the action method for getting the request named as Create method.

// GET: Employee/Create

public IActionResult Create()

{

ViewData["CityId"] = new SelectList(\_context.Cities, "Id", "CityName");

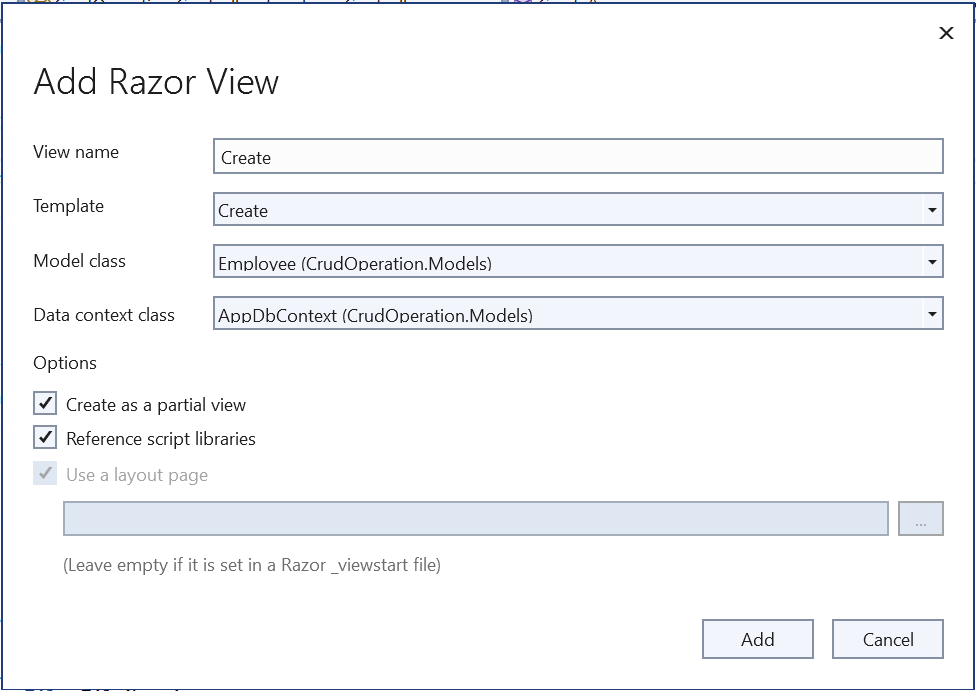
ViewData["CountryId"] = new SelectList(\_context.Countries, "Id", "CountryName");

ViewData["StateId"] = new SelectList(\_context.States, "Id", "StateName");

return View();

}

Now, right-click on Create Action Method and add View.



And, after that, we will write HTML code like below.

@model CrudOperation.Models.Employee

@{

ViewData["Title"] = "Create";

}

<h4>Employee Form</h4>

<hr />

<div class="row">

<form **asp-action**="Create" enctype="multipart/form-data">

<div **asp-validation-summary**="ModelOnly" class="text-danger"></div>

<div class="row">

<div class="col-md-6">

<div class="row mb-3">

<label **asp-for**="Name" class="col-md-3 col-form-label"></label>

<div class="col-md-9">

<input **asp-for**="Name" class="form-control" />

<span **asp-validation-for**="Name" class="text-danger"></span>

</div>

</div>

<div class="row mb-3">

<label class="col-md-3 col-form-label">Gender</label>

<div class="col-md-9 pt-2">

<label class="radio-inline mx-2">

@Html.RadioButtonFor(model => model.Gender, "Male", htmlAttributes: new { @class = "form-check-input" }) Male

</label>

<label class="radio-inline mx-2">

@Html.RadioButtonFor(model => model.Gender, "Female", htmlAttributes: new { @class = "form-check-input" }) Female

</label>

<label class="radio-inline mx-2">

@Html.RadioButtonFor(model => model.Gender, "Other", htmlAttributes: new { @class = "form-check-input" }) Other

</label>

<br/>

<span **asp-validation-for**="Gender" class="text-danger"></span>

</div>

</div>

<div class="row mb-3">

<label class="col-md-3 col-form-label">Education</label>

<div class="col-md-9 pt-2">

<div class="form-check form-check-inline">

<input **type**="checkbox" class="form-check-input" **asp-for**="Ssc" />

<label class="form-check-label" **asp-for**="Ssc">

@Html.DisplayNameFor(model => model.Ssc)

</label>

</div>

<div class="form-check form-check-inline">

<input **type**="checkbox" class="form-check-input" **asp-for**="Hsc" />

<label class="form-check-label" **asp-for**="Hsc">

@Html.DisplayNameFor(model => model.Hsc)

</label>

</div>

<div class="form-check form-check-inline">

<input **type**="checkbox" class="form-check-input" **asp-for**="Bsc" />

<label class="form-check-label" **asp-for**="Bsc">

@Html.DisplayNameFor(model => model.Bsc)

</label>

</div>

<div class="form-check form-check-inline">

<input **type**="checkbox" class="form-check-input" **asp-for**="Msc" />

<label class="form-check-label" **asp-for**="Msc">

@Html.DisplayNameFor(model => model.Msc)

</label>

</div>

</div>

</div>

<div class="row mb-3">

<label **asp-for**="Address" class="col-md-3 col-form-label"></label>

<div class="col-md-9">

<textarea **asp-for**="Address" class="form-control"></textarea>

<span **asp-validation-for**="Address" class="text-danger"></span>

</div>

</div>

<div class="row mb-3">

<label **asp-for**="CountryId" class="col-md-3 col-form-label"></label>

<div class="col-md-9">

<select **asp-for**="CountryId" class="form-control" **asp-items**="ViewBag.CountryId">

<option **value**="">Select Country</option>

</select>

<span **asp-validation-for**="CountryId" class="text-danger"></span>

</div>

</div>

<div class="row mb-3">

<label **asp-for**="StateId" class="col-md-3 col-form-label"></label>

<div class="col-md-9">

<select **asp-for**="StateId" **asp-items**="@(ViewBag.StateId)" class="form-control">

<option **value**="">Select State</option>

</select>

<span **asp-validation-for**="StateId" class="text-danger"></span>

</div>

</div>

<div class="row mb-3">

<label **asp-for**="CityId" class="col-md-3 col-form-label"></label>

<div class="col-md-9">

<select **asp-for**="CityId" **asp-items**="@(ViewBag.CityId)" class="form-control">

<option **value**="">Select City</option>

</select>

<span **asp-validation-for**="CityId" class="text-danger"></span>

</div>

</div>

</div>

<div class="col-md-6">

<div class="row mb-3">

<label **asp-for**="Picture" class="col-md-3 col-form-label"></label>

<div class="col-md-9">

<input **type**="file" **asp-for**="Picture" class="form-control" **name**="pictureFile" accept="image/\*" id="FileUpload" onchange="PreviewImage()" />

<span **asp-validation-for**="Picture" class="text-danger"></span>

</div>

<div class="mt-3" style="text-align:end">

<img src="~/images/eset\_user.png" alt="img" id="UploadFile" class="img-thumbnail" style="width:170px;" />

</div>

</div>

<div class="mt-3" style="text-align:end">

<a class="btn btn-outline-info btn-sm" **asp-action**="Index">Back to List</a>

<input type="submit" value="Create" class="btn btn-outline-primary btn-sm" />

</div>

</div>

</div>

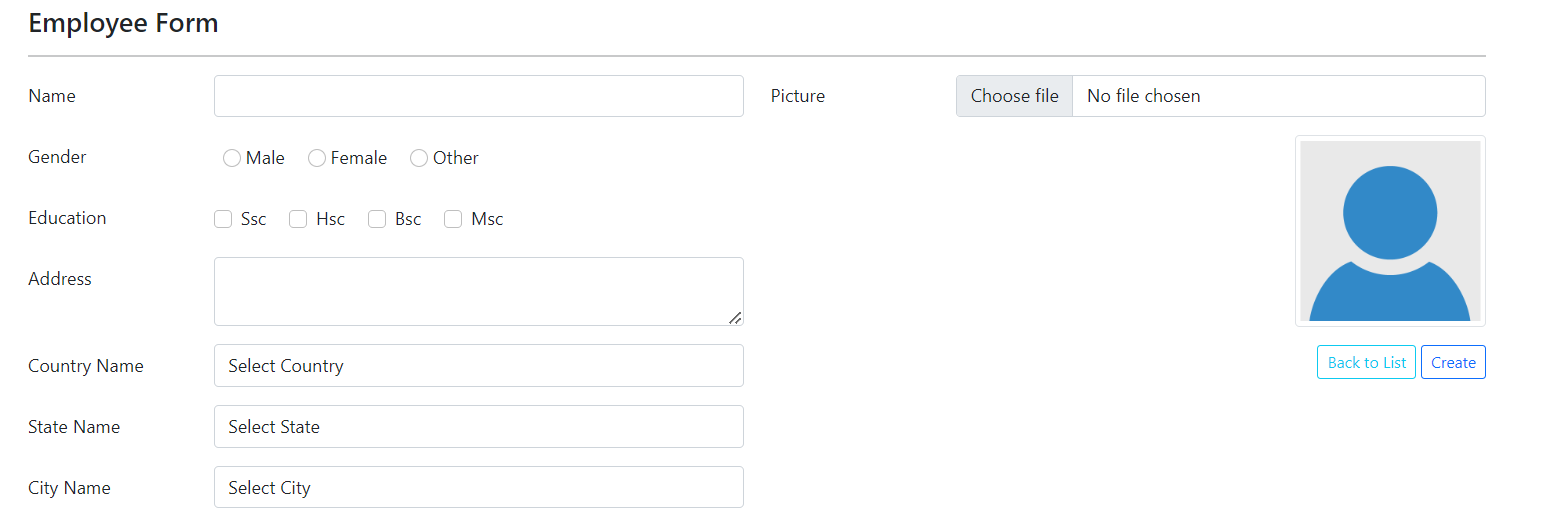
</form>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}



Now, we will write the code for posting the details.

// POST: Employee/Create

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see <http://go.microsoft.com/fwlink/?LinkId=317598>.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create(Employee employee, IFormFile pictureFile)

{

if (ModelState.IsValid)

{

if (pictureFile != null && pictureFile.Length > 0)

{

var path = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot/images",

pictureFile.FileName);

using (var stream = new FileStream(path, FileMode.Create))

{

pictureFile.CopyTo(stream);

}

employee.Picture = $"{pictureFile.FileName}";

}

\_context.Add(employee);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

ViewData["CityId"] = new SelectList(\_context.Cities, "Id", "CityName", employee.CityId);

ViewData["CountryId"] = new SelectList(\_context.Countries, "Id", "CountryName", employee.CountryId);

ViewData["StateId"] = new SelectList(\_context.States, "Id", "StateName", employee.StateId);

return View(employee);

}

**View Operation:**  Now we will write code for seeing the details of a particular employee,

// GET: Employee/Details/5

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

{

if (id == null || \_context.Employees == null)

{

return NotFound();

}

var employee = await \_context.Employees

.Include(e => e.City)

.Include(e => e.Country)

.Include(e => e.State)

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

if (employee == null)

{

return NotFound();

}

return View(employee);

}

Here, right click on the Action method “Details” and press Add View option. And write the below html code,

@model CrudOperation.Models.Employee

@{

ViewData["Title"] = "Details";

}

<div>

<h4>Employee Details</h4>

<hr />

<div class="row">

<div class="col-md-8">

<dl class="row">

<dt class="col-md-3">

@Html.DisplayNameFor(model => model.Name)

</dt>

<dd class="col-md-9">

@Html.DisplayFor(model => model.Name)

</dd>

<dt class="col-md-3">

@Html.DisplayNameFor(model => model.Address)

</dt>

<dd class="col-md-9">

@Html.DisplayFor(model => model.Address)

</dd>

<dt class="col-md-3">

@Html.DisplayNameFor(model => model.Gender)

</dt>

<dd class="col-md-9">

@Html.DisplayFor(model => model.Gender)

</dd>

<dt class="col-md-3">

Education

</dt>

<dd class="col-md-9">

@Html.DisplayNameFor(model => model.Ssc)

@Html.DisplayFor(model => model.Ssc)

@Html.DisplayNameFor(model => model.Hsc)

@Html.DisplayFor(model => model.Hsc)

@Html.DisplayNameFor(model => model.Bsc)

@Html.DisplayFor(model => model.Bsc)

@Html.DisplayNameFor(model => model.Msc)

@Html.DisplayFor(model => model.Msc)

</dd>

<dt class="col-md-3">

@Html.DisplayNameFor(model => model.Country)

</dt>

<dd class="col-md-9">

@Html.DisplayFor(model => model.Country.CountryName)

</dd>

<dt class="col-md-3">

@Html.DisplayNameFor(model => model.State)

</dt>

<dd class="col-md-9">

@Html.DisplayFor(model => model.State.StateName)

</dd>

<dt class="col-md-3">

@Html.DisplayNameFor(model => model.City)

</dt>

<dd class="col-md-9">

@Html.DisplayFor(model => model.City.CityName)

</dd>

</dl>

</div>

<div class="col-md-4">

<dl class="row">

<dt class="col-md-3">

@Html.DisplayNameFor(model => model.Picture)

</dt>

<dd class="col-md-9">

<img src="~/images/@Html.DisplayFor(model => model.Picture)" alt="Image" width="200" class="img-thumbnail" />

</dd>

</dl>

</div>

</div>

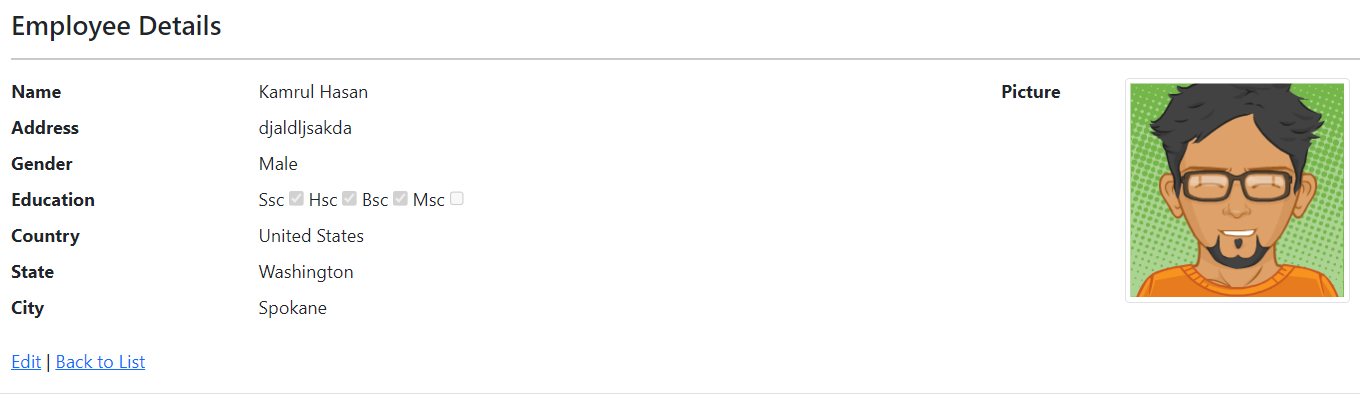
</div>

<div>

<a **asp-action**="Edit" **asp-route-id**="@Model?.Id">Edit</a> |

<a **asp-action**="Index">Back to List</a>

</div>



**Update Operation:** Now, we will perform edit functionality so for this we will create edit action method.

// GET: Employee/Edit/5

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

{

if (id == null || \_context.Employees == null)

{

return NotFound();

}

var employee = await \_context.Employees.FindAsync(id);

if (employee == null)

{

return NotFound();

}

ViewData["CityId"] = new SelectList(\_context.Cities, "Id", "CityName", employee.CityId);

ViewData["CountryId"] = new SelectList(\_context.Countries, "Id", "CountryName", employee.CountryId);

ViewData["StateId"] = new SelectList(\_context.States, "Id", "StateName", employee.StateId);

return View(employee);

}

Now, right click on the Action method “Edit” and press Add View option. And write the below html code,

@model CrudOperation.Models.Employee

@{

ViewData["Title"] = "Edit";

}

<h4>Employee</h4>

<hr />

<div class="row">

<form **asp-action**="Edit" enctype="multipart/form-data">

<div **asp-validation-summary**="ModelOnly" class="text-danger"></div>

<input **type**="hidden" **asp-for**="Id" />

<div class="row">

<div class="col-md-6">

<div class="row mb-3">

<label **asp-for**="Name" class="col-md-3 col-form-label"></label>

<div class="col-md-9">

<input **asp-for**="Name" class="form-control" />

<span **asp-validation-for**="Name" class="text-danger"></span>

</div>

</div>

<div class="row mb-3">

<label class="col-md-3 col-form-label">Gender</label>

<div class="col-md-9 pt-2">

<label class="radio-inline mx-2">

@Html.RadioButtonFor(model => model.Gender, "Male", htmlAttributes: new { @class = "form-check-input" }) Male

</label>

<label class="radio-inline mx-2">

@Html.RadioButtonFor(model => model.Gender, "Female", htmlAttributes: new { @class = "form-check-input" }) Female

</label>

<label class="radio-inline mx-2">

@Html.RadioButtonFor(model => model.Gender, "Other", htmlAttributes: new { @class = "form-check-input" }) Other

</label>

</div>

</div>

<div class="row mb-3">

<label class="col-md-3 col-form-label">Education</label>

<div class="col-md-9 pt-2">

<div class="form-check form-check-inline">

<input class="form-check-input" **asp-for**="Ssc" />

<label class="form-check-label" **asp-for**="Ssc">

@Html.DisplayNameFor(model => model.Ssc)

</label>

</div>

<div class="form-check form-check-inline">

<input class="form-check-input" **asp-for**="Hsc" />

<label class="form-check-label" **asp-for**="Hsc">

@Html.DisplayNameFor(model => model.Hsc)

</label>

</div>

<div class="form-check form-check-inline">

<input class="form-check-input" **asp-for**="Bsc" />

<label class="form-check-label" **asp-for**="Bsc">

@Html.DisplayNameFor(model => model.Bsc)

</label>

</div>

<div class="form-check form-check-inline">

<input class="form-check-input" **asp-for**="Msc" />

<label class="form-check-label" **asp-for**="Msc">

@Html.DisplayNameFor(model => model.Msc)

</label>

</div>

</div>

</div>

<div class="row mb-3">

<label **asp-for**="Address" class="col-md-3 col-form-label"></label>

<div class="col-md-9">

<textarea **asp-for**="Address" class="form-control"></textarea>

<span **asp-validation-for**="Address" class="text-danger"></span>

</div>

</div>

<div class="row mb-3">

<label **asp-for**="CountryId" class="col-md-3 col-form-label"></label>

<div class="col-md-9">

<select **asp-for**="CountryId" class="form-control" **asp-items**="ViewBag.CountryId">

<option>Select Country</option>

</select>

<span **asp-validation-for**="CountryId" class="text-danger"></span>

</div>

</div>

<div class="row mb-3">

<label **asp-for**="StateId" class="col-md-3 col-form-label"></label>

<div class="col-md-9">

<select **asp-for**="StateId" **asp-items**="@(ViewBag.StateId)" class="form-control">

<option>Select State</option>

</select>

<span **asp-validation-for**="StateId" class="text-danger"></span>

</div>

</div>

<div class="row mb-3">

<label **asp-for**="CityId" class="col-md-3 col-form-label"></label>

<div class="col-md-9">

<select **asp-for**="CityId" **asp-items**="@(ViewBag.CityId)" class="form-control">

<option>Select City</option>

</select>

<span **asp-validation-for**="CityId" class="text-danger"></span>

</div>

</div>

</div>

<div class="col-md-6">

<div class="row mb-3">

<label **asp-for**="Picture" class="col-md-3 col-form-label"></label>

<div class="col-md-9">

<input **type**="file" **asp-for**="Picture" class="form-control" **name**="pictureFile" accept="image/\*" id="FileUpload" onchange="PreviewImage()" />

<span **asp-validation-for**="Picture" class="text-danger"></span>

</div>

<div class="mt-3" style="text-align:end">

<img src="~/images/@Model.Picture" alt="img" id="UploadFile" class="img-thumbnail" style="width:170px;" />

</div>

</div>

<div class="mt-3" style="text-align:end">

<a class="btn btn-outline-info btn-sm" **asp-action**="Index">Back to List</a>

<input type="submit" value="Update" class="btn btn-outline-primary btn-sm" />

</div>

</div>

</div>

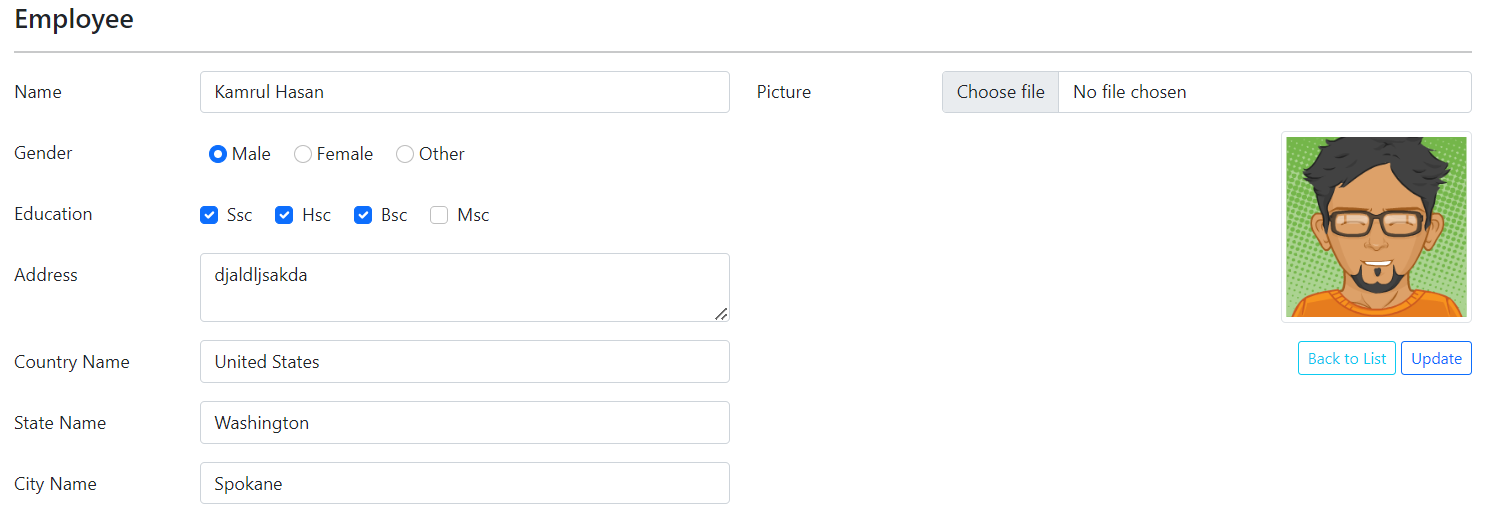
</form>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}



Now, we will write the code for updating the page.

// POST: Employee/Edit/5

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(Employee employee, IFormFile pictureFile)

{

if (ModelState.IsValid)

{

try

{

var emp = await \_context.Employees.FindAsync(employee.Id);

if (pictureFile != null && pictureFile.Length > 0)

{

var path = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot/images",

pictureFile.FileName);

using (var stream = new FileStream(path, FileMode.Create))

{

pictureFile.CopyTo(stream);

}

employee.Picture = $"{pictureFile.FileName}";

}

else

{

employee.Picture = emp.Picture;

}

emp.Picture= employee.Picture;

emp.Name = employee.Name;

emp.Address = employee.Address;

emp.Gender = employee.Gender;

emp.Ssc= employee.Ssc;

emp.Hsc= employee.Hsc;

emp.Bsc= employee.Bsc;

emp.Msc= employee.Msc;

emp.CountryId= employee.CountryId;

emp.StateId= employee.StateId;

emp.CityId= employee.CityId;

\_context.Update(emp);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!EmployeeExists(employee.Id))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

ViewData["CityId"] = new SelectList(\_context.Cities, "Id", "CityName", employee.CityId);

ViewData["CountryId"] = new SelectList(\_context.Countries, "Id", "CountryName", employee.CountryId);

ViewData["StateId"] = new SelectList(\_context.States, "Id", "StateName", employee.StateId);

return View(employee);

}

**Delete operation:** So, finally we write code for Delete operation.

// GET: Employee/Delete/5

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

{

if (id == null || \_context.Employees == null)

{

return NotFound();

}

var employee = await \_context.Employees

.Include(e => e.City)

.Include(e => e.Country)

.Include(e => e.State)

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

if (employee == null)

{

return NotFound();

}

return View(employee);

}

Right click on the Delete Action method and add the view. And we will write the HTML code.

@model CrudOperation.Models.Employee

@{

ViewData["Title"] = "Delete";

}

<h1>Delete</h1>

<h3>Are you sure you want to delete this?</h3>

<div>

<h4>Employee</h4>

<hr />

<dl class="row">

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Name)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Name)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Address)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Address)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Gender)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Gender)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Ssc)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Ssc)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Hsc)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Hsc)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Bsc)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Bsc)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Msc)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Msc)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Picture)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Picture)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Country)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Country.CountryName)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.State)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.State.StateName)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.City)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.City.CityName)

</dd>

</dl>

<form **asp-action**="Delete">

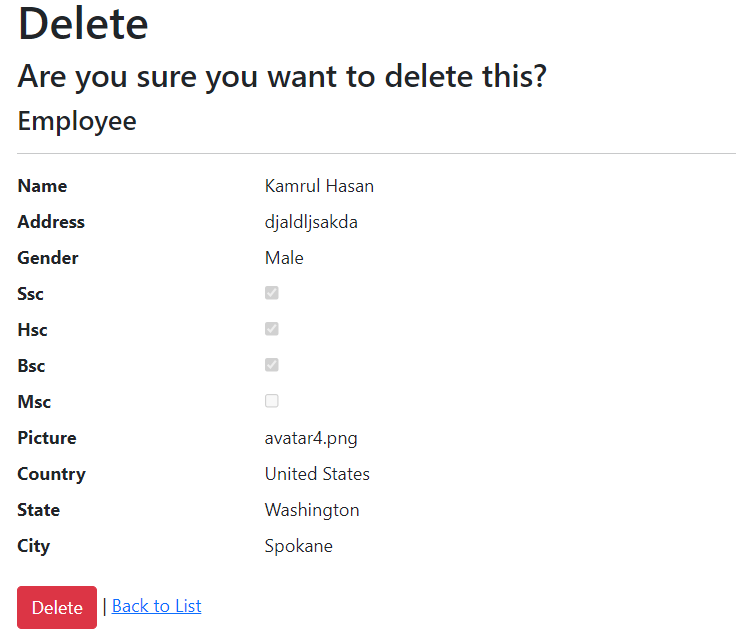
<input **type**="hidden" **asp-for**="Id" />

<input type="submit" value="Delete" class="btn btn-danger" /> |

<a **asp-action**="Index">Back to List</a>

</form>

</div>



// POST: Employee/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

if (\_context.Employees == null)

{

return Problem("Entity set 'AppDbContext.Employees' is null.");

}

var employee = await \_context.Employees.FindAsync(id);

if (employee != null)

{

\_context.Employees.Remove(employee);

}

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

Here additional Code for cascading dropdown

public JsonResult GetStatesByCountryId(int countryId)

{

List<State> statesList = new List<State>();

statesList = (from state in \_context.States

where state.CountryId == countryId

select state).ToList();

return Json(statesList);

}

public JsonResult GetCitiesByStateId(int stateId)

{

List<City> citiesList = new List<City>();

citiesList = (from city in \_context.Cities

where city.StateId == stateId

select city).ToList();

return Json(citiesList);

}

jQuery Code:

// Please see documentation at https://docs.microsoft.com/aspnet/core/client-side/bundling-and-minification

// for details on configuring this project to bundle and minify static web assets.

// Write your JavaScript code.

$(document).ready(function () {

// Get States by Country ID

$('#CountryId').change(function () {

$.ajax({

type: "get",

url: "/Employee/GetStatesByCountryId",

data: { countryId: $('#CountryId').val() },

datatype: "json",

traditional: true,

success: function (data) {

var state = "<select id='StateId'>";

state = state + '<option value="">Select State</option>';

for (var i = 0; i < data.length; i++) {

state = state + '<option value=' + data[i].id + '>' + data[i].stateName + '</option>';

}

state = state + '</select>';

$('#StateId').html(state);

}

});

});

// Get Cities by State ID

$('#StateId').change(function () {

$.ajax({

type: "get",

url: "/Employee/GetCitiesByStateId",

data: { stateId: $('#StateId').val() },

datatype: "json",

traditional: true,

success: function (data) {

var city = "<select id='CityId'>";

city = city + '<option value="">Select City</option>';

for (var i = 0; i < data.length; i++) {

city = city + '<option value=' + data[i].id + '>' + data[i].cityName + '</option>';

}

city = city + '</select>';

$('#CityId').html(city);

}

});

});

//Checkbox Checked

var $ssc = $("#Ssc");

var $hsc = $("#Hsc");

var $bsc = $("#Bsc");

var $msc = $("#Msc");

$hsc.on("click", function () {

var anyChecked = $hsc.is(":checked");

$ssc.prop("checked", anyChecked);

});

$bsc.on("click", function () {

var anyChecked = $bsc.is(":checked");

$ssc.prop("checked", anyChecked);

$hsc.prop("checked", anyChecked);

});

$msc.on("click", function () {

var anyChecked = $msc.is(":checked");

$ssc.prop("checked", anyChecked);

$hsc.prop("checked", anyChecked);

$bsc.prop("checked", anyChecked);

});

});

//Image Preview

function PreviewImage() {

var oFReader = new FileReader();

oFReader.readAsDataURL(document.getElementById("FileUpload").files[0]);

oFReader.onload = function (oFREvent) {

document.getElementById("UploadFile").src = oFREvent.target.result;

};

};

Thanks…...!

Happy Codding.