**Task No. 1:** Create Complete calculator program using MVC.

**Solution:**

**Model**

namespace Calculator\_MVC.Models

{

public class calc

{

public int no1 { get; set; }

public int no2 { get; set; }

public int tot { get; set; }

}

}

**Controller**

namespace Calculator\_MVC.Controllers

{

public class CalcController : Controller

{

// GET: Calc

public ActionResult Index()

{

return View(new calc());

}

[HttpPost]

public ActionResult Index(calc c, string calculate)

{

if (calculate == "add")

{

c.tot = c.no1 + c.no2;

}

else if (calculate == "min")

{

c.tot = c.no1 - c.no2;

}

else if (calculate == "sub")

{

c.tot = c.no1 \* c.no2;

}

else if (calculate == "divv")

{

c.tot = c.no1 / c.no2;

}

return View(c);

}

}

}

**VIEW**

@model Calculator\_MVC.Models.calc

@{

ViewBag.Title = "Index";

}

<html>

<head></head>

<body>

<h2>Simple Calculator Using MVC</h2>

<div class="row">

@using (Html.BeginForm("/Index", "calc", FormMethod.Post))

{

<div class="form-group">

<label for="no1">Number 1 : </label>

<input type="text" class="form-control" id="no1" name="no1" value="@Model.no1" />

</div>

<div class="form-group">

<label for="no2">Number 2 : </label>

<input type="text" class="form-control" id="no2" name="no2" value="@Model.no2" />

</div>

<div class="form-group">

<label for="total">Total : </label>

<input type="text" class="form-control" id="tot" name="tot" value="@Model.tot" />

</div>

<button type="submit" class="btn btn-info" id="add" value="add" name="calculate">+</button>

<button type="submit" class="btn btn-danger" id="min" value="min" name="calculate">-</button>

<button type="submit" class="btn btn-warning" id="sub" value="sub" name="calculate">\*</button>

<button type="submit" class="btn btn-default" id="divv" value="divv" name="calculate">/</button>

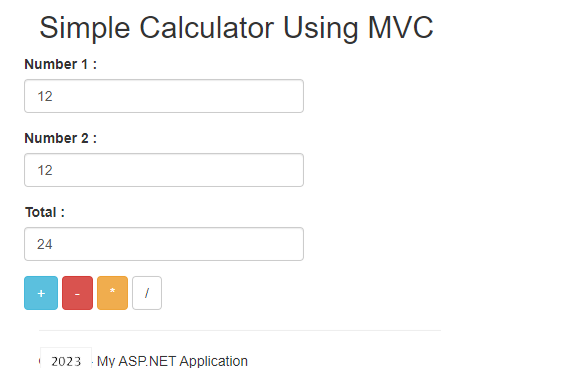
}

</div>

</body>

</html>

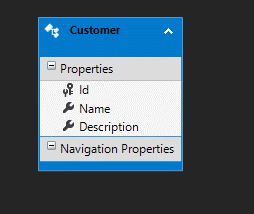
**Output:**

****

**Task No. 2:** Create Basic CRUD Application using MVC.

**Solution:**

**Model**



**Controller**

namespace CRUD.Controllers

{

public class CustomersController : Controller

{

private MvcCrudEntities db = new MvcCrudEntities();

// GET: Customers

public async Task<ActionResult> Index()

{

return View(await db.Customers.ToListAsync());

}

// GET: Customers/Details/5

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

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Customer customer = await db.Customers.FindAsync(id);

if (customer == null)

{

return HttpNotFound();

}

return View(customer);

}

// GET: Customers/Create

public ActionResult Create()

{

return View();

}

// POST: Customers/Create

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

// more details see https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Create([Bind(Include = "Id,Name,Description")] Customer customer)

{

if (ModelState.IsValid)

{

db.Customers.Add(customer);

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

return View(customer);

}

// GET: Customers/Edit/5

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

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Customer customer = await db.Customers.FindAsync(id);

if (customer == null)

{

return HttpNotFound();

}

return View(customer);

}

// POST: Customers/Edit/5

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

// more details see https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Edit([Bind(Include = "Id,Name,Description")] Customer customer)

{

if (ModelState.IsValid)

{

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

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

return View(customer);

}

// GET: Customers/Delete/5

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

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Customer customer = await db.Customers.FindAsync(id);

if (customer == null)

{

return HttpNotFound();

}

return View(customer);

}

// POST: Customers/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<ActionResult> DeleteConfirmed(int id)

{

Customer customer = await db.Customers.FindAsync(id);

db.Customers.Remove(customer);

await db.SaveChangesAsync();

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

**CREATE:**

@model CRUD.Models.Customer

@{

ViewBag.Title = "Create";

}

<h2>Create</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h4>Customer</h4>

<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.Description, htmlAttributes: new { @class = "control-label col-md-2" })

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

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

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

</div>

</div>

<div class="form-group">

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

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

</div>

</div>

</div>

}

<div>

@Html.ActionLink("Back to List", "Index")

</div>

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

}

**EDIT:**

@model CRUD.Models.Customer

@{

ViewBag.Title = "Edit";

}

<h2>Edit</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h4>Customer</h4>

<hr />

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

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

<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.Description, htmlAttributes: new { @class = "control-label col-md-2" })

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

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

@Html.ValidationMessageFor(model => model.Description, "", 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>

}

<div>

@Html.ActionLink("Back to List", "Index")

</div>

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")}

**DELETE:**

@model CRUD.Models.Customer

@{

ViewBag.Title = "Delete";

}

<h2>Delete</h2>

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

<div>

<h4>Customer</h4>

<hr />

<dl class="dl-horizontal">

<dt>

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

</dt>

<dd>

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

</dd>

<dt>

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

</dt>

<dd>

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

</dd>

</dl>

@using (Html.BeginForm()) {

@Html.AntiForgeryToken()

<div class="form-actions no-color">

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

@Html.ActionLink("Back to List", "Index")

</div>

}

</div>

**DETAILS:**

@model CRUD.Models.Customer

@{

ViewBag.Title = "Details";

}

<h2>Details</h2>

<div>

<h4>Customer</h4>

<hr />

<dl class="dl-horizontal">

<dt>

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

</dt>

<dd>

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

</dd>

<dt>

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

</dt>

<dd>

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

</dd>

</dl>

</div>

<p>

@Html.ActionLink("Edit", "Edit", new { id = Model.Id }) |

@Html.ActionLink("Back to List", "Index")

</p>

**INDEX:**

@model IEnumerable<CRUD.Models.Customer>

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

<p>

@Html.ActionLink("Create New", "Create")

</p>

<table class="table">

<tr>

<th>

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

</th>

<th>

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

</th>

<th></th>

</tr>

@foreach (var item in Model) {

<tr>

<td>

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

</td>

<td>

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

</td>

<td>

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

@Html.ActionLink("Details", "Details", new { id=item.Id }) |

@Html.ActionLink("Delete", "Delete", new { id=item.Id })

</td>

</tr>

}

</table>

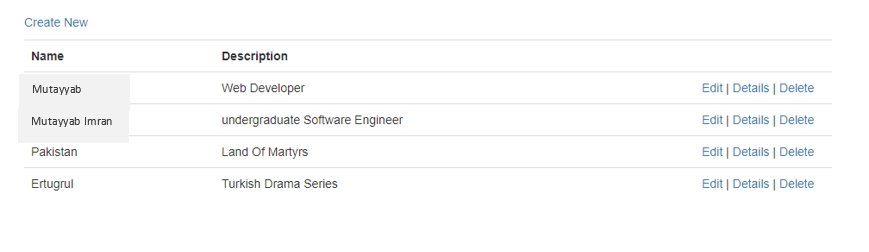
**Output:**

**Create:**

**Edit:**

* A screenshot of a computer

  Description automatically generated with medium confidence**Before:**
* **After:**

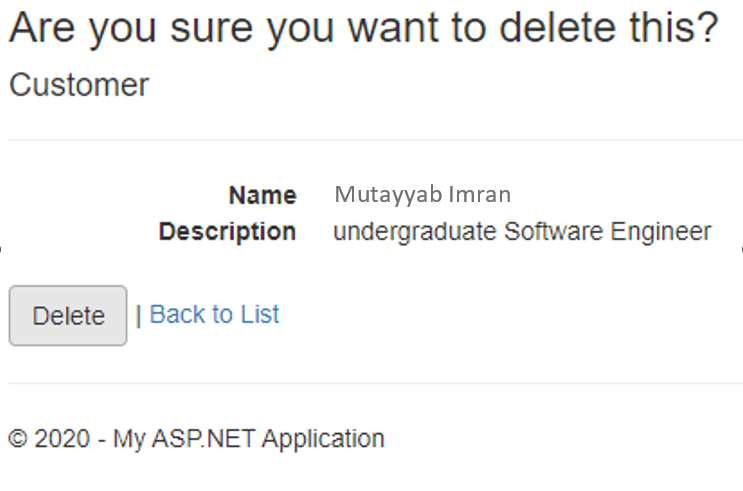
****

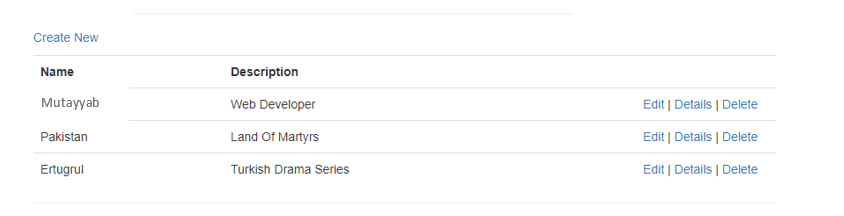
**Details:**

**A screenshot of a computer

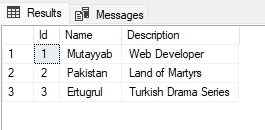
Description automatically generated with medium confidence**

**Delete:**

****

****

**Database:**

****