Q1. 1. Index View (10 marks for showing the view correctly, including database code) Should show all the Products

ProductController :

using Exam1.Models;

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace Exam1.Controllers

{

public class ProductController : Controller

{

// GET: Product

public ActionResult Index()

{

List<Products> list = new List<Products>();

SqlConnection sc = new SqlConnection();

sc.ConnectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=Exam;Integrated Security=True;Connect ";

sc.Open();

SqlCommand icmd = new SqlCommand();

icmd.Connection = sc;

icmd.CommandType = System.Data.CommandType.Text;

//icmd.CommandText = "IndexPage";

icmd.CommandText = "select \* from Products";

try

{

SqlDataReader sdr = icmd.ExecuteReader();

while (sdr.Read())

{

list.Add(new Products { ProductId = (int)sdr["ProductId"], ProductName = (string)sdr["ProductName"] , Rate = (decimal)sdr["Rate"] , Description = (string)sdr["Description"] , CategoryName = (string)sdr["CategoryName"] });

}

sdr.Close();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

finally

{

sc.Close();

}

return View(list);

}

// GET: Product/Details/5

public ActionResult Details(int id)

{

return View();

}

// GET: Product/Create

public ActionResult Create()

{

return View();

}

// POST: Product/Create

[HttpPost]

public ActionResult Create(FormCollection collection)

{

try

{

// TODO: Add insert logic here

return RedirectToAction("Index");

}

catch

{

return View();

}

}

// GET: Product/Edit/5

public ActionResult Edit(int id = 1)

{

Products p = new Products();

SqlConnection sc = new SqlConnection();

sc.ConnectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=Exam;Integrated Security=True";

sc.Open();

SqlCommand icmd = new SqlCommand();

icmd.Connection = sc;

icmd.CommandType = System.Data.CommandType.Text;

icmd.CommandText = "select \* from Products where ProductId = @ProductId";

try

{

SqlDataReader sdr = icmd.ExecuteReader();

if(sdr.Read())

{

p = new Products { ProductId = (int)sdr["ProductId"], ProductName = (string)sdr["ProductName"], Rate = (decimal)sdr["Rate"], Description = (string)sdr["Description"], CategoryName = (string)sdr["CategoryName"] };

}

sdr.Close();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

finally

{

sc.Close();

}

return View(p);

}

// POST: Product/Edit/5

[HttpPost]

public ActionResult Edit(int id, Products p)

{

SqlConnection sc = new SqlConnection();

sc.ConnectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=Exam;Integrated Security=True;Connect Timeout=100;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";

sc.Open();

SqlCommand icmd = new SqlCommand();

icmd.Connection = sc;

icmd.CommandType = System.Data.CommandType.Text;

icmd.CommandText = "update Products set @ProductId = ProductId , @ProductName = ProductName , @Rate = Rate , @Description = Description , @CategoryName = CategoryName where @PrductId = ProductId";

icmd.Parameters.AddWithValue("@ProductId", id);

icmd.Parameters.AddWithValue("@ProductName", p.ProductName);

icmd.Parameters.AddWithValue("@Rate", p.Rate);

icmd.Parameters.AddWithValue("@Description", p.Description);

icmd.Parameters.AddWithValue("@CategoryName", p.CategoryName);

try

{

icmd.ExecuteNonQuery();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

finally

{

sc.Close();

}

return View();

}

// GET: Product/Delete/5

public ActionResult Delete(int id)

{

return View();

}

// POST: Product/Delete/5

[HttpPost]

public ActionResult Delete(int id, FormCollection collection)

{

try

{

// TODO: Add delete logic here

return RedirectToAction("Index");

}

catch

{

return View();

}

}

}

}

Q. a) Ensure that all values must be entered before submitting them. b) Give proper error messages for all validations c) The displayed value in the label should be shown from the annotation for the field. Eg "ProductName" property should be displayed as "Product Name”

Ans:

Model:

Products.cs :

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Web;

namespace Exam1.Models

{

public class Products

{

[Key]

public int ProductId { get; set; }

[DataType(DataType.Text)]

[Required(ErrorMessage = "Please Enter The Name")]

[Display(Name = "Product Name")]

public string ProductName { get; set; }

[DataType(DataType.Text)]

[Required(ErrorMessage = "Please Enter The Rate")]

public decimal Rate { get; set; }

[DataType(DataType.Text)]

[Required(ErrorMessage = "Please Enter The Description")]

public string Description { get; set; }

[DataType(DataType.Text)]

[Required(ErrorMessage = "Please Enter The Category Name")]

[Display(Name = "Category Name")]

public string CategoryName { get; set; }

}

}

Views :

Index :

@model IEnumerable<Exam1.Models.Products>

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

<p>

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

</p>

<table class="table">

<tr>

<th>

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

</th>

<th>

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

</th>

<th>

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

</th>

<th>

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

</th>

<th>

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

</th>

<th></th>

</tr>

@foreach (var item in Model) {

<tr>

<td>

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

</td>

<td>

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

</td>

<td>

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

</td>

<td>

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

</td>

<td>

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

</td>

<td>

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

@Html.ActionLink("Details", "Details", new { /\* id=item.PrimaryKey \*/ }) |

@Html.ActionLink("Delete", "Delete", new { /\* id=item.PrimaryKey \*/ })

</td>

</tr>

}

</table>

Edit :

@model Exam1.Models.Products

@{

ViewBag.Title = "Edit";

}

<h2>Edit</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h4>Products</h4>

<hr />

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

<div class="form-group">

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

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

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

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

</div>

</div>

<div class="form-group">

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

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

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

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

</div>

</div>

<div class="form-group">

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

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

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

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

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

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

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

@Html.ValidationMessageFor(model => model.CategoryName, "", 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")

}

Q. 5. (Separate Project) Create a WEB API for the Products Table. Use the Entity Framework. Test the application using POSTMAN. Send the screenshots for GET All Products, and POST method.

Ans :

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Entity;

using System.Data.Entity.Infrastructure;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

using System.Web.Http.Description;

using WebApi;

namespace WebApi.Controllers

{

public class ProductsController : ApiController

{

private ExamEntities db = new ExamEntities();

// GET: api/Products

public IQueryable<Product> GetProducts()

{

return db.Products;

}

// GET: api/Products/5

[ResponseType(typeof(Product))]

public IHttpActionResult GetProduct(int id)

{

Product product = db.Products.Find(id);

if (product == null)

{

return NotFound();

}

return Ok(product);

}

// PUT: api/Products/5

[ResponseType(typeof(void))]

public IHttpActionResult PutProduct(int id, Product product)

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

if (id != product.ProductId)

{

return BadRequest();

}

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

try

{

db.SaveChanges();

}

catch (DbUpdateConcurrencyException)

{

if (!ProductExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return StatusCode(HttpStatusCode.NoContent);

}

// POST: api/Products

[ResponseType(typeof(Product))]

public IHttpActionResult PostProduct(Product product)

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

db.Products.Add(product);

try

{

db.SaveChanges();

}

catch (DbUpdateException)

{

if (ProductExists(product.ProductId))

{

return Conflict();

}

else

{

throw;

}

}

return CreatedAtRoute("DefaultApi", new { id = product.ProductId }, product);

}

// DELETE: api/Products/5

[ResponseType(typeof(Product))]

public IHttpActionResult DeleteProduct(int id)

{

Product product = db.Products.Find(id);

if (product == null)

{

return NotFound();

}

db.Products.Remove(product);

db.SaveChanges();

return Ok(product);

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

private bool ProductExists(int id)

{

return db.Products.Count(e => e.ProductId == id) > 0;

}

}

}