МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГИЙ УПРАВЛІННЯ

Лабораторна робота № 6

з курсу «Стек технологій .Net»

«Робота з даними в ASP.NET 2.0»

Виконав:

студент групи КН 36-в

Макаров А.К.

Перевірила:

Літвінова Ю.С.

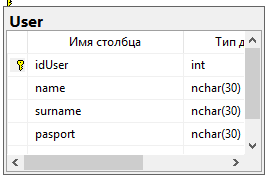
ХАРКІВ 2018

**Тема:** Робота з даними в ASP.NET.

**Ціль:** Познайомитися з рядом засобів ASP.NET для представлення та відображення даних.

**Варіант індивідуального завдання:** Телефона станція.

**Робота з базою даних**



**Опис механізму роботи з даними розробленого застосунку**

Коли клієнт заходить на сайт він може подивитися список дзвінків та додати нових. На сервері за цей процес відповідає контролер CallDescriptorModule який зберігає нових студентів, та формує їх список. Лістинг контролера показаний на лістингу 1.

Лістинг 1. Код контролера CallDescriptorModel

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.ComponentModel.DataAnnotations;

namespace Sclad.Models

{

public class UserModel

{

public int UserId { get; set; }

[Required]

[StringLength(50, MinimumLength = 3)]

[RegularExpression(@"(\S)+", ErrorMessage = "White space is not allowed")]

[Display(Name = "First Name")]

public string FirstName { get; set; }

[Required]

[StringLength(50, MinimumLength = 3)]

[RegularExpression(@"(\S)+", ErrorMessage = "White space is not allowed")]

[Display(Name = "Last Name")]

public string LastName { get; set; }

[Required]

[StringLength(50, MinimumLength = 3)]

[RegularExpression(@"(\S)+", ErrorMessage = "White space is not allowed")]

[Display(Name = "Passport data")]

public string PassportData { get; set; }

[Required]

[StringLength(50, MinimumLength = 3)]

[RegularExpression(@"(\S)+", ErrorMessage = "White space is not allowed")]

[Display(Name = "Certificate number")]

public string CertificateNumber { get; set; }

}

Далі клієнт може додати нового абітурієнта та зберегти зміни. Лістинг роботи контролеру, який відповідає за обробку даних показаний у лістингу 2.

Лістинг 2. UserDataStorage

public class HomeController : Controller

{

CallDescriptorContext db = new CallDescriptorContext();

// GET: CallDesriptor

public ActionResult Index()

{

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

}

[HttpGet]

public ActionResult Create()

{

return View();

}

[HttpPost]

public ActionResult Create(CallDescriptor cd)

{

if (ModelState.IsValid)

{

db.CallDescriptor.Add(cd);

db.SaveChanges();

return Redirect("Index");

}

return View();

}

public ActionResult Details(int? id)

{

if (id == null)

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

CallDescriptor cd = db.CallDescriptor.Find(id);

if (cd == null)

return HttpNotFound();

return View(cd);

}

[HttpGet]

public ActionResult Edit(int? id)

{

if (id == null)

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

CallDescriptor cd = db.CallDescriptor.Find(id);

if (cd == null)

return HttpNotFound();

return View(cd);

}

[HttpPost]

public ActionResult Edit(CallDescriptor cd)

{

try

{

if (ModelState.IsValid)

{

db.Entry(cd).State = System.Data.Entity.EntityState.Modified;

db.SaveChanges();

return Redirect("Index");

}

return RedirectToAction("Index");

}

catch

{

return View();

}

}

[HttpGet]

public ActionResult Delete(int? id)

{

if (id == null)

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

CallDescriptor cd = db.CallDescriptor.Find(id);

if (cd == null)

return HttpNotFound();

return View(cd);

}

[HttpPost]

public ActionResult Delete(int? id, CallDescriptor cd)

{

try

{

CallDescriptor calldescriptor = new CallDescriptor();

if (ModelState.IsValid)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

calldescriptor = db.CallDescriptor.Find(id);

if (calldescriptor == null)

return HttpNotFound();

db.CallDescriptor.Remove(calldescriptor);

db.SaveChanges();

return RedirectToAction("Index");

}

return View("Index");

}

catch

{

return View();

}

}

}

Конструктор цього класу отримує об’єкт CallDescriptor, який відповідає за роботу з дзвынками. Контролеру не важливо як конкретно реалізований цей інтерфейс і тому, логіку зберігання заявок завжди можна замінити.

Для відображення даних технологія ASP.NET MVC використовує Razor view engine, який отримує об'єкт моделі з контролеру, та дозволяє використовувати C# код всередині HTML розмітки. Файл який містить HTML та C# код називається уявленням та має розширення .cshtml. Лістинг одного з таких файлів показаний у лістингу 3.

Лістинг 3. Приклад cshtml файлу

@model IEnumerable<WebApplication3.CallDescriptor>

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

<p>

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

</p>

<table class="table">

<tr>

<th>

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

</th>

<th>

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

</th>

<th>

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

</th>

<th>

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

</th>

<th>

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

</th>

<th></th>

</tr>

@foreach (var item in Model) {

<tr>

<td>

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

</td>

<td>

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

</td>

<td>

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

</td>

<td>

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

</td>

<td>

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

</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>

Розроблений веб застосунок показаний на рисунках 1 – 4.



Рисунок 1 – Головне меню.

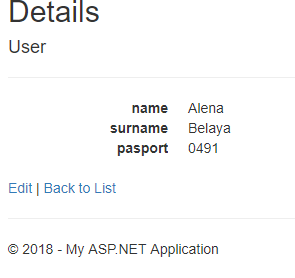


Рисунок 2 – Деталі.

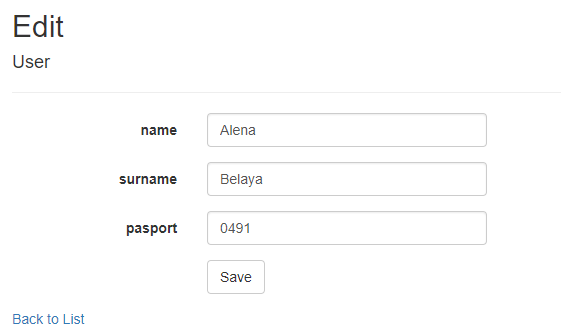


Рисунок 3 – Редагування.

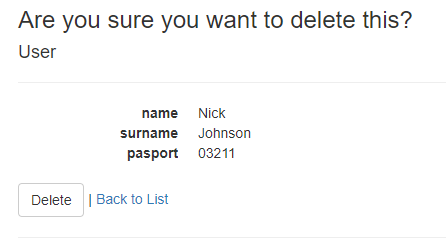


Рисунок 4 – Видалення.

**Висновок:** У даній лабораторній роботі було ознайомлено з рядом засобів ASP.NET для представлення та відображення даних.