МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

"ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ"

Кафедра "Обчислювальна техніка та програмування"

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТП

\_\_\_\_\_\_\_\_\_\_ проф.Семенов С. Г.

"\_\_\_"\_\_\_\_\_\_\_\_\_\_2018 р.

СТВОРЕННЯ WEB API З ВИКОРИСТАННЯМ ПІДХОДУ CODE FIRST У ENTITY FRAMEWORK

Текст програми

Лист затвердження

КІТ26Б.15386-22 12 01-1-ЛЗ

Розробники:

Керівник

\_\_\_\_\_\_\_\_\_\_ Шевердін І. В.

"\_\_\_"\_\_\_\_\_\_\_\_\_\_2018 р.

Виконавець

\_\_\_\_\_\_\_\_\_\_Юрчик Д.О.

"\_\_\_"\_\_\_\_\_\_\_\_\_\_2018 р.

Харків 2018

ЗАТВЕРДЖЕНО

КІТ26Б.15386-22 12 01-1-ЛЗ

СТВОРЕННЯ WEB API З ВИКОРИСТАННЯМ ПІДХОДУ CODE FIRST У ENTITY FRAMEWORK

Текст програми

КІТ26Б.15386-22 12 01-1

Аркушів 23

Харків 2018

АНОТАЦІЯ

Даний документ входить до комплекту документів курсового проекту на тему " Створення Web API з використанням підходу Code First у Entity Framework ".

Програма розроблена як проект для середовища Visual Studio з Entity Framework..

У процесі розробки використовувалися мови програмування С#, XML, HTML.

Текст модулів та функції, що були згенеровани автоматично або незначно змінювались, цілком не приводиться.

ЗМІСТ

[1   Файл Computer.сs 4](#_Toc406601793)

[2   Файл DatabaseContext.cs 4](#_Toc406601794)

[3   Файл ComputersController.cs 4](#_Toc406601794)

[4   Файл HomeConteroller.cs 6](#_Toc406601795)

[5   Файл Index.cshtml 7](#_Toc406601796)

[6   Файл Edit.cshtml 8](#_Toc406601793)

[7   Файл Delete.cshtml 9](#_Toc406601794)

[8   Файл Add.cshtml 9](#_Toc406601794)

[9   Файл MainWindow.xaml cs 10](#_Toc406601795)

[10   Файл MainWindow.xaml 11](#_Toc406601794)

[11   Файл Program.cs 12](#_Toc406601794)

# Файл Computer.сs

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Web;

namespace ComputersCW.Models

{

public class Computer

{

[Key]

public int Index { get; set; }

public String Model { get; set; }

public double Price { get; set; }

public String CPU { get; set; }

public int RAM { get; set; }

}

}

# Файл DatabaseContext.cs

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Web;

using ComputersDBCW.Models;

namespace ComputersDBCW.Context

{

public class DatabaseContext : DbContext

{

public DatabaseContext() : base("DefaultConnection") { }

public DbSet<Computers> Computers { get; set; }

}

}

# Файл ComputersController.cs

using ComputersDBCW.Context;

using ComputersDBCW.Models;

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 System.Web.Http.Cors;

namespace ComputersDBCW.Controllers

{

[EnableCors(origins: "\*", headers: "\*", methods: "\*")]

public class ComputersController : ApiController

{

private DatabaseContext db = new DatabaseContext();

/\*[HttpGet]

public IHttpActionResult Get()

{

try

{

var result = from computer in db.Computers

select new

{

computer.Index,

computer.Model,

computer.Price,

computer.CPU,

computer.RAM

};

return Ok(result);

}

catch (Exception)

{

return InternalServerError();

}

}\*/

public IQueryable<Computers> GetComputer()

{

return db.Computers;

}

// GET: api/Computers/5

[ResponseType(typeof(Computers))]

public IHttpActionResult GetComputer(int id)

{

Computers computer = db.Computers.Find(id);

if (computer == null)

{

return NotFound();

}

return Ok(computer);

}

// PUT: api/Computers/5

[ResponseType(typeof(void))]

public IHttpActionResult PutComputer(int id, Computers computer)

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

if (id != computer.Index)

{

return BadRequest();

}

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

try

{

db.SaveChanges();

}

catch (DbUpdateConcurrencyException)

{

if (!ComputerExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return StatusCode(HttpStatusCode.NoContent);

}

// POST: api/Computers

[ResponseType(typeof(Computers))]

public IHttpActionResult PostComputer(Computers computer)

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

db.Computers.Add(computer);

db.SaveChanges();

return CreatedAtRoute("DefaultApi", new { id = computer.Index }, computer);

}

// DELETE: api/Computers/5

[ResponseType(typeof(Computers))]

public IHttpActionResult DeleteComputer(int id)

{

Computers computer = db.Computers.Find(id);

if (computer == null)

{

return NotFound();

}

db.Computers.Remove(computer);

db.SaveChanges();

return Ok(computer);

}

//метод особождения распределенных ресурсов

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

private bool ComputerExists(int id)

{

return db.Computers.Count(e => e.Index == id) > 0;

}

}

}

# Файл HomeController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using ComputersDBCW.Models;

using ComputersDBCW.Context;

using System.Data.Entity;

namespace ComputerMVC.Controllers

{

public class HomeController : Controller

{

// создаем контекст данных

DatabaseContext db = new DatabaseContext();

public ActionResult Index()

{

return View(db.Computers);

}

protected override void Dispose(bool disposing)

{

db.Dispose();

base.Dispose(disposing);

}

[HttpGet]

public ActionResult EditComputer(int? id)

{

if (id == null)

{

return HttpNotFound();

}

Computers computers = db.Computers.Find(id);

if (computers != null)

{

return View(computers);

}

return HttpNotFound();

}

[HttpPost]

public ActionResult EditComputer(Computers computers)

{

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

db.SaveChanges();

return RedirectToAction("Index");

}

[HttpGet]

public ActionResult Create()

{

return View();

}

[HttpPost]

public ActionResult Create(Computers computers)

{

db.Computers.Add(computers);

db.SaveChanges();

return RedirectToAction("Index");

}

[HttpGet]

public ActionResult Delete(int id)

{

Computers b = db.Computers.Find(id);

if (b == null)

{

return HttpNotFound();

}

return View(b);

}

[HttpPost, ActionName("Delete")]

public ActionResult DeleteConfirmed(int id)

{

Computers b = db.Computers.Find(id);

if (b == null)

{

return HttpNotFound();

}

db.Computers.Remove(b);

db.SaveChanges();

return RedirectToAction("Index");

}

}

}

# Файл Index.cshtml

@model IEnumerable<ComputersDBCW.Models.Computers>

@{

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<div>

<h3>Computers</h3>

<table>

<tr class="header">

<td><p>Index</p></td>

<td><p>Model </p></td>

<td><p>Price </p></td>

<td><p>CPU </p></td>

<td><p>RAM</p></td>

<td></td>

</tr>

@foreach (ComputersDBCW.Models.Computers b in Model)

{

<tr>

<td><p>@b.Model</p></td>

<td><p>@b.Price</p></td>

<td><p>@b.CPU</p></td>

<td><p>@b.RAM</p></td>

<td><p><a href="/Home/Buy/@b.Index">Buy</a></p></td>

</tr>

}

</table>

</div>

# 6 Файл Edit.cshtml

@{

ViewBag.Title = "Edit";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

@model ComputersDBCW.Models.Computers

<h2>Computer @Model.Index</h2>

@using (Html.BeginForm("EditComputer", "Home", FormMethod.Post))

{

<fieldset>

@Html.HiddenFor(m => m.Index)

<p>

@Html.LabelFor(m => m.Model, "Model")

<br />

@Html.EditorFor(m => m.Model)

</p>

<p>

@Html.LabelFor(m => m.Price, "Price")

<br />

@Html.EditorFor(m => m.Price)

</p>

<p>

@Html.LabelFor(m => m.CPU, "CPU")

<br />

@Html.EditorFor(m => m.CPU)

</p>

<p>

@Html.LabelFor(m => m.RAM, "RAM")

<br />

@Html.EditorFor(m => m.RAM)

</p>

<p><input type="submit" value="Отправить" /></p>

</fieldset>

}

# 7 Файл delete.cshtml

@{

Layout = "~/Views/Shared/\_Layout.cshtml";

}

@model ComputersDBCW.Models.Computers

<h2>Delete</h2>

<dl>

<dt>Model</dt>

<dd>

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

</dd>

<dt>Price</dt>

<dd>

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

</dd>

<dt>CPU</dt>

<dd>

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

</dd>

<dt>RAM</dt>

<dd>

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

</dd>

</dl>

@using (Html.BeginForm())

{

<input type="submit" value="Delete" />

}

# 8 Файл Add.cshtml

@model ComputersDBCW.Models.Computers

@{

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>Add</h2>

@using (Html.BeginForm())

{

@Html.LabelFor(model => model.Model, "Model")

<br />

@Html.EditorFor(model => model.Model)

<br /><br />

@Html.LabelFor(model => model.Price, "Price")

<br />

@Html.EditorFor(model => model.Price)

<br /><br />

@Html.LabelFor(model => model.CPU, "CPU")

<br />

@Html.EditorFor(model => model.CPU)

<br /><br />

@Html.LabelFor(model => model.RAM, "RAM")

<br />

@Html.EditorFor(model => model.RAM)

<br /><br />

<input type="submit" value="Add" />

}

# 9 Файл MainWindow.xaml.cs

using System;

using System.Windows;

using System.Windows.Controls;

using System.Data.SqlClient;

using System.Data;

using System.Configuration;

using System.Data.Entity;

using ComputersDBCW.Models;

using ComputersDBCW.Context;

namespace ComputersApp

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

DatabaseContext db;

public MainWindow()

{

InitializeComponent();

db = new DatabaseContext();

db.Computers.Load(); // загружаем данные

computersGrid.ItemsSource = db.Computers.Local.ToBindingList(); // устанавливаем привязку к кэшу

this.Closing += MainWindow\_Closing;

}

private void MainWindow\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

db.Dispose();

}

private void updateButton\_Click(object sender, RoutedEventArgs e)

{

db.SaveChanges();

}

private void button1\_Click(object sender, EventArgs e)

{

System.Diagnostics.Process.Start(@"C:\Users\Дарья\source\repos\ComputersDBCW\ComputersCA\bin\Debug\ComputersCA.exe");

this.Close();

}

private void deleteButton\_Click(object sender, RoutedEventArgs e)

{

if (computersGrid.SelectedItems.Count > 0)

{

for (int i = 0; i < computersGrid.SelectedItems.Count; i++)

{

Computers computer = computersGrid.SelectedItems[i] as Computers;

if (computer != null)

{

db.Computers.Remove(computer);

}

}

}

db.SaveChanges();

}

private void ComputersGrid\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

}

}

}

# 10 Файл MainWindow.xaml

<Window x:Class="ComputersApp.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:ComputersApp"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="800">

<Window.Resources>

<Style TargetType="Button">

<Setter Property="Margin" Value="20 8 20 8" />

<Setter Property="Width" Value="100" />

<Setter Property="Height" Value="30" />

</Style>

</Window.Resources>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="325\*"/>

<ColumnDefinition Width="40\*"/>

<ColumnDefinition Width="32\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="\*" />

<RowDefinition Height="Auto" />

</Grid.RowDefinitions>

<DataGrid AutoGenerateColumns="False" x:Name="computersGrid" SelectionChanged="ComputersGrid\_SelectionChanged" Margin="0,0,3.6,46" Grid.RowSpan="2" Grid.ColumnSpan="3">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Index}" Header="Index" Width="120"/>

<DataGridTextColumn Binding="{Binding Model}" Header="Model" Width="125"/>

<DataGridTextColumn Binding="{Binding Price}" Header="Price" Width="80"/>

<DataGridTextColumn Binding="{Binding CPU}" Header="CPU" Width="125"/>

<DataGridTextColumn Binding="{Binding RAM}" Header="RAM" Width="80"/>

</DataGrid.Columns>

</DataGrid>

<StackPanel HorizontalAlignment="Center" Grid.Row="1" Orientation="Horizontal" Margin="187,0.4,43,0">

<Button x:Name="updateButton" Content="Update" Click="updateButton\_Click" />

<Button x:Name="deleteButton" Content="Remove" Click="deleteButton\_Click" />

<Button x:Name="button1" Content="Console" Click="button1\_Click" />

</StackPanel>

<Grid HorizontalAlignment="Left" Height="40" Margin="580,215,0,0" VerticalAlignment="Top" Width="145" Grid.ColumnSpan="2"/>

</Grid>

</Window>

# 11 program.cs

using System;

using System.Collections.Generic;

using System.Net;

using System.Net.Http;

using System.Net.Http.Headers;

using System.Threading.Tasks;

namespace ComputersCA

{

class Program

{

static void Main(string[] args)

{

RunAsync().Wait();

}

static async Task RunAsync()

{

using (var client = new HttpClient())

{

client.BaseAddress = new Uri("http://localhost:62494/");

client.DefaultRequestHeaders.Accept.Clear();

client.DefaultRequestHeaders.Accept.Add(

new MediaTypeWithQualityHeaderValue("application/json"));

HttpResponseMessage response;

Console.WriteLine("POST");

Computers newComputers = new Computers()

{

Index = 41,

Model = "Asus",

Price = 12300.0,

CPU = "Intel",

RAM = 4

};

response = await client.PostAsJsonAsync("api/computers", newComputers);

if (response.IsSuccessStatusCode)

{

Uri computerUrl = response.Headers.Location;

Console.WriteLine(computerUrl);

Console.WriteLine("{0}\t{1}\t{2}\t{3}\t{4}", newComputers.Index, newComputers.Model,

newComputers.Price, newComputers.CPU, newComputers.RAM);

Console.WriteLine("Change Price (PUT)");

newComputers.Price = 7999.9;

response = await client.PutAsJsonAsync(computerUrl, newComputers);

Console.WriteLine("{0}\t{1}\t{2}\t{3}\t{4}", newComputers.Index, newComputers.Model,

newComputers.Price, newComputers.CPU, newComputers.RAM);

Console.WriteLine("DELETE");

response = await client.DeleteAsync(computerUrl);

}

Console.WriteLine("GET");

response = await client.GetAsync("api/computers/10");

if (response.IsSuccessStatusCode)

{

Computers computers = await response.Content.ReadAsAsync<Computers>();

Console.WriteLine("{0}\t{1}\t{2}\t{3}\t{4}", computers.Index, computers.Model,

computers.Price, computers.CPU, computers.RAM);

}

Console.WriteLine("Get all");

response = await client.GetAsync("api/computers");

if (response.IsSuccessStatusCode)

{

List<Computers> computers = await response.Content.ReadAsAsync<List<Computers>>();

for (int i = 0; i < computers.Count; i++)

{

Console.WriteLine("{0}\t{1}\t{2}\t{3}\t{4}", computers[i].Index, computers[i].Model,

computers[i].Price, computers[i].CPU, computers[i].RAM);

}

}

Console.ReadLine();

}

}

}

}