Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Лабораторна робота №2

за курсом: «Робота з даними на платформи .Nеt»

за темою: «ADO.NET»

|  |  |
| --- | --- |
|  | Виконав:  студент гр. ПЗПІ-16-3  Сачек О.А. |
|  | Перевірила:  ст.викл. каф. ПІ  Олiйник О.В. |

Харків 2019 р.

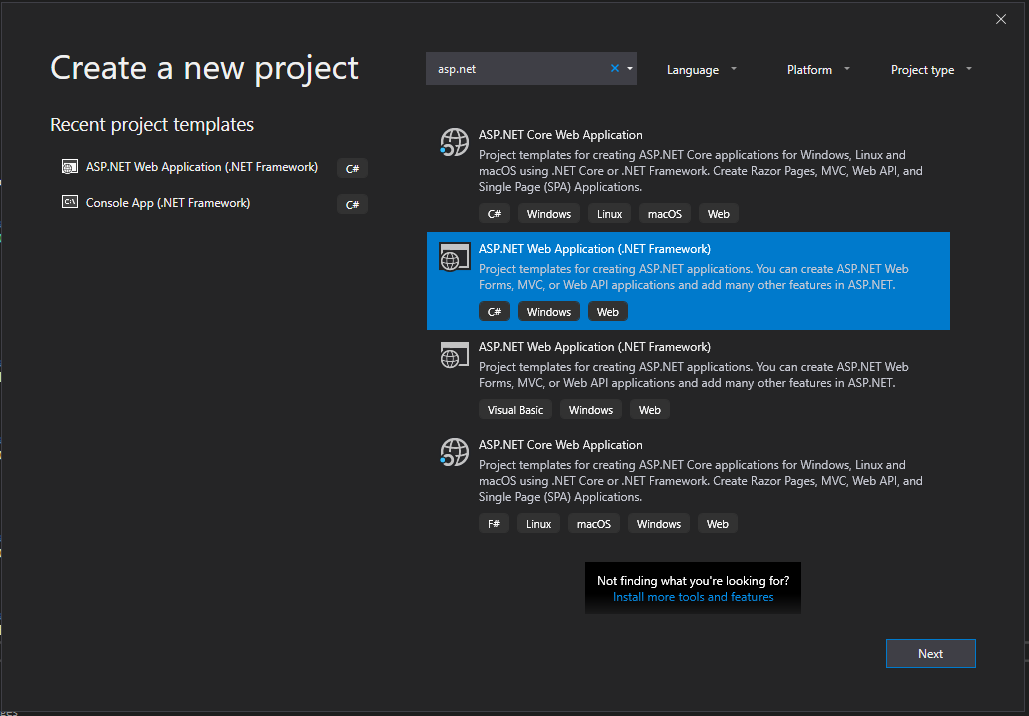
Лабораторна робота №2. ADO.NET

Створіть веб-застосування на підставі технології ADO.Net, яке використовує базу даних, створену в SQL Server.

Обмеження: використання лише компонентів ADO.Net для організації роботи з БД.

Хід роботи.

Створимо новий проект ASP.NET MVC 5.



Додамо функціонал роботи з базою даних.

public class DbContext : IDisposable

{

private SqlConnection connection;

public DbContext(string connectionString)

{

connection = new SqlConnection(connectionString);

Animals = new DbSet<Animal>(connection, nameof(Animals));

}

public DbSet<Animal> Animals { get; }

public void Dispose()

{

connection?.Dispose();

}

}

public class DbSet<T> where T : new()

{

private readonly SqlConnection connection;

private readonly string name;

public DbSet(SqlConnection connection, string name)

{

this.connection = connection;

this.name = name;

}

#region Public methods

public IList<T> Get()

{

var entities = new List<T>();

string sqlCommand = $"SELECT \* FROM {name}";

try

{

connection.Open();

using (var command = new SqlCommand(sqlCommand, connection))

using (var reader = command.ExecuteReader())

{

while (reader.Read())

{

var entity = GetEntity(reader);

entities.Add(entity);

}

}

}

finally

{

connection.Close();

}

return entities;

}

public int Add(T entity)

{

string fields = string.Join(",", GetPropertiesForEntity(entity));

string values = string.Join(",", GetValuesForEntity(entity));

string sqlCommand = $"INSERT INTO {name} ({fields}) VALUES ({values})";

try

{

connection.Open();

using (var command = new SqlCommand(sqlCommand, connection))

{

int result = command.ExecuteNonQuery();

return result;

}

}

finally

{

connection.Close();

}

}

public int Update(T entity)

{

var fields = GetPropertiesForEntity(entity).ToList();

var values = GetValuesForEntity(entity).ToList();

string setFilelds = string.Join(",", Merge(fields, values).Select(keyValue => $"{keyValue.Key}={keyValue.Value}"));

string idField = GetIdPropertyName(entity);

Guid idValue = GetIdPropertyValue(entity);

string sqlCommand = $"UPDATE {name} SET {setFilelds} WHERE {idField}='{idValue.ToString()}'";

try

{

connection.Open();

using (var command = new SqlCommand(sqlCommand, connection))

{

int result = command.ExecuteNonQuery();

return result;

}

}

finally

{

connection.Close();

}

}

public int Delete(T entity)

{

string idField = GetIdPropertyName(entity);

Guid idValue = GetIdPropertyValue(entity);

string sqlCommand = $"DELETE FROM {name} WHERE {idField}='{idValue.ToString()}'";

try

{

connection.Open();

using (var command = new SqlCommand(sqlCommand, connection))

{

int result = command.ExecuteNonQuery();

return result;

}

}

finally

{

connection.Close();

}

}

#endregion

#region Private methods

private T GetEntity(SqlDataReader reader)

{

var entity = new T();

var type = entity.GetType();

var properties = type.GetProperties().Where(x => x.CanWrite);

foreach(var property in properties)

{

var value = GetValueForProperty(reader, property.Name);

property.SetValue(entity, value);

}

return entity;

}

private object GetValueForProperty(SqlDataReader reader, string propertyName)

{

var index = Enumerable.Range(0, reader.FieldCount)

.Select(i => (i, name: reader.GetName(i)))

.FirstOrDefault(x => string.Equals(x.name, propertyName));

return reader.GetValue(index.i);

}

private IEnumerable<string> GetPropertiesForEntity(T entity)

{

return entity.GetType()

.GetProperties()

.Where(x => x.CanWrite)

.Select(x => x.Name);

}

private IEnumerable<string> GetValuesForEntity(T entity)

{

return entity.GetType()

.GetProperties()

.Where(x => x.CanWrite)

.Select(x => x.GetValue(entity))

.Select(ConvertValueToSql);

}

private string ConvertValueToSql(object value)

{

switch(value)

{

case DateTime dateTimeValue:

return $"'{dateTimeValue.ToString()}'";

case string stringValue:

return $"'{stringValue}'";

case Guid guidValue:

return $"'{guidValue}'";

case null:

return "'NULL'";

default:

return value.ToString();

}

}

private string GetIdPropertyName(T entity)

{

return entity.GetType()

.GetProperties()

.FirstOrDefault(x => x.PropertyType == typeof(Guid))

.Name;

}

private Guid GetIdPropertyValue(T entity)

{

return (Guid)entity.GetType()

.GetProperties()

.FirstOrDefault(x => x.PropertyType == typeof(Guid))

.GetValue(entity);

}

private Dictionary<TKey, TValue> Merge<TKey, TValue>(IList<TKey> keys, IList<TValue> values)

{

var dic = new Dictionary<TKey, TValue>();

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

{

dic.Add(keys[i], values[i]);

}

return dic;

}

#endregion

}

Використаємо Unity для Dependency Injection.

public static class UnityBootstrapper

{

public static void Initialize()

{

var container = new UnityContainer();

container.RegisterInstance(typeof(DbContext), new DbContext(ConfigurationManager.ConnectionStrings["DefaultConnection"].ConnectionString));

DependencyResolver.SetResolver(new UnityDependencyResolver(container));

}

}

Додамо Views та Controllers

public class AnimalsController : Controller

{

private readonly DbContext context;

public AnimalsController(DbContext context)

{

this.context = context;

}

[Route("animals")]

public ActionResult AnimalsList()

{

var model = context.Animals.Get();

return View("AnimalsListView", model);

}

[Route("animals/create")]

public ActionResult AnimalsCreate()

{

return View("AnimalsCreateView");

}

[HttpPost]

[Route("animals/create")]

public ActionResult AnimalsCreate(Animal animal)

{

if (ModelState.IsValid)

{

animal.Id = Guid.NewGuid();

context.Animals.Add(animal);

return RedirectToAction("AnimalsList");

}

else

{

return View("AnimalsCreateView", animal);

}

}

[Route("animals/edit")]

public ActionResult AnimalsEdit(Guid id)

{

var model = context.Animals.Get().FirstOrDefault(x => x.Id == id);

return View("AnimalsEditView", model);

}

[HttpPost]

[Route("animals/edit")]

public ActionResult AnimalsEdit(Animal animal)

{

if (ModelState.IsValid)

{

context.Animals.Update(animal);

return RedirectToAction("AnimalsList");

}

else

{

return View("AnimalsEditView", animal.Id);

}

}

[Route("animals/delete")]

public ActionResult AnimalsDelete(Guid id)

{

var animal = context.Animals.Get().FirstOrDefault(x => x.Id == id);

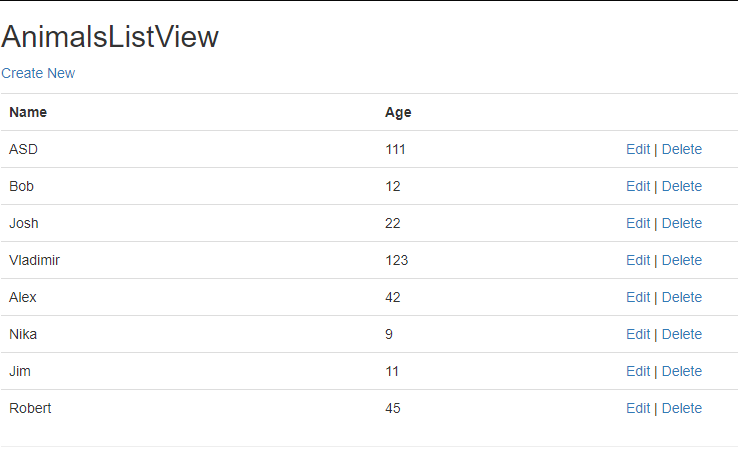
context.Animals.Delete(animal);

return RedirectToAction("AnimalsList");

}

}

Отримаємо веб-застосунок із CRUD функціоналом.



Висновки: на лабораторній роботі ми навчилися використовувати технологію ADO.NET разом із базою даних MS SQL Server, для того, щоб отримувати доступ до функціоналу роботи з даними. Також був створений веб-застосунок із використанням ASP.NET MVC 5 та ADO.NET.