**Bazy Danych - Laboratorium**

**LINQ to Entities**

**Sprawozdanie**

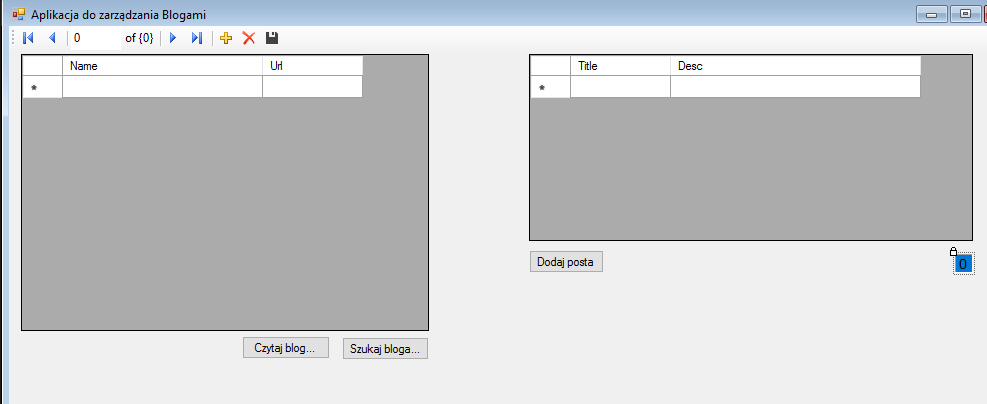
Krzysztof Suruło

Link do projektu github:

https://github.com/Archeangel/Entity-freamework-DataBasesProject

Zadanie polegało na rozwinięciu stworzonej na zajęciach aplikacji o nowe funkcjonalności wykorzystując poznane technologie.

Przede wszystkim podczas wykonywania przygotowanego skryptu napotkałem na potrzebę rozwinięcia swojej aplikacji o możliwość wygodnego dodawania postów do blogów. W tym celu zmodyfikowałem Blog form dodając kolumne wyświetlającą tytuł i opis każdego posta, przycisk dodaj nowy post, który dodaje nowego posta oraz licznik wszytkich postów. Oprócz tego w formie pojawiły się też przyciski czytaj blog i szukaj bloga.



Kod zmodyfikowanej formy:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.Entity;

using System.Diagnostics;

namespace CodeFirstLab1

{

public partial class BlogForm : Form

{

int blogId = 0;

private BlogContext bc = new BlogContext();

public BlogForm()

{

InitializeComponent();

}

private void BlogForm\_Load(object sender, EventArgs e)

{

bc.Blogs.Load();

bc.Posts.Load();

this.blogBindingSource.DataSource = bc.Blogs.Local.ToBindingList();

}

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

{

bc.SaveChanges();

}

private void blogDataGridView\_CellClick(object sender, DataGridViewCellEventArgs e)

{

bc.Database.Log = message => Trace.WriteLine(message);

foreach (DataGridViewRow row in blogDataGridView.SelectedRows)

{

blogId = Int32.Parse(row.Cells[0].Value.ToString());

}

BindingList<Post> posts = bc.Posts.Local.ToBindingList();

Console.WriteLine(blogId);

IEnumerable<Post> postsEnumerable = posts.Where(p => p.BlogId == blogId);

this.postBindingSource.DataSource = postsEnumerable;

amountOfPostsLabel.Text = postsEnumerable.Count().ToString();

}

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

{

PostForm pf = new PostForm(bc);

pf.ShowDialog();

}

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

{

FindingForm ff = new FindingForm(bc);

ff.ShowDialog();

}

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

{

ReadingBlogForm rbf = new ReadingBlogForm(blogId);

rbf.ShowDialog();

}

private void postDataGridView\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

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

{

}

}

}

Oto zakładka dodaj posta

: 

Wybieramy dane i klikamy zapisz dodając posta. Kod całej formy (w tym przycisku zapsiz wygląda tak) :

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CodeFirstLab1

{

public partial class PostForm : Form

{

private int \_blogId;

private BlogContext \_bc;

public PostForm(BlogContext bc)

{

// TODO: Complete member initialization

InitializeComponent();

this.\_bc = bc;

}

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

{

PostForm.ActiveForm.Close();

}

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

{

using (BlogContext context = new BlogContext())

{

\_blogId = (from b in context.Blogs

where b.Name == chooseBlogComboBox.SelectedItem.ToString()

select b.BlogId).FirstOrDefault();

}

Post post = new Post()

{

BlogId = \_blogId,

Desc = descriptionOfNewPostTextBox.Text,

Title = titleOfNewPostTextBox.Text

};

\_bc.Posts.Add(post);

\_bc.SaveChanges();

PostForm.ActiveForm.Close();

}

private void PostForm\_Load(object sender, EventArgs e)

{

chooseBlogComboBox.Items.Add("Choose a blog...");

chooseBlogComboBox.SelectedItem = chooseBlogComboBox.Items[0];

using (BlogContext context = new BlogContext())

{

var blogs = from b in context.Blogs

orderby b.Name

select b.Name;

foreach (var item in blogs)

{

chooseBlogComboBox.Items.Add(item);

}

}

}

private void chooseBlogComboBox\_SelectedIndexChanged(object sender, EventArgs e)

{

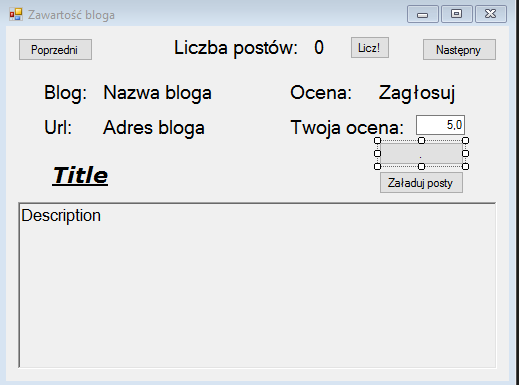
}

}

}

Wykorzystałem tutaj **Deferred Execution –** blogi zostaną załadowane do comboBoxa dopiero w czasie iterowania po pętli. W zapytaniach wykorzystałem **method Syntax**.

Kolejną funkcjonalnością jest dodanie reading blog form gdzie można iterować po blogach, policzyć liczbę postów dla każdego, załadować posty i ocenić bloga. Forma wygląda tak:



Kod formy jest następujący

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CodeFirstLab1

{

public partial class ReadingBlogForm : Form

{

private BlogContext \_bc;

private int \_blogId;

private Blog blog;

private IQueryable<Post> posts;

private int currentPost = 0;

public ReadingBlogForm(int blogId)

{

InitializeComponent();

this.\_blogId = blogId;

this.\_bc = new BlogContext();

\_bc.Database.Log = message => Trace.WriteLine(message);

blog = \_bc.Blogs.Where(b => b.BlogId == \_blogId).FirstOrDefault();

if (blog == null)

{

MessageBox.Show("Brak takiego bloga.");

return;

}

else

{

urlBlogReadingBlogLabel.Text = blog.Url;

blogNameReadingBlogLabel.Text = blog.Name;

}

}

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

{

currentPost += 1;

LoadPost();

}

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

{

currentPost -= 1;

LoadPost();

}

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

{

readButton.Visible = false;

postTitleLabel.Visible = true;

titleReadingPostLabel.Visible = true;

descriptionReadingBlogRichTextBox.Visible = true;

nextPostFromBlogButton.Visible = true;

prevPostFromBlogButtom.Visible = true;

countButton.Visible = true;

// Lazy loading:

// IQueryable wspiera LazyLoading

\_bc.Database.Log = message => Trace.WriteLine(message);

posts = \_bc.Posts.Where(p => p.BlogId == \_blogId).OrderBy(p => p.PostId).AsQueryable();

currentPost = 0;

descriptionReadingBlogRichTextBox.Text = posts.FirstOrDefault().Desc;

titleReadingPostLabel.Text = posts.FirstOrDefault().Title;

}

private void LoadPost()

{

Post post;

\_bc.Database.Log = message => Trace.WriteLine(message);

if (currentPost < 0 || currentPost >= posts.Count())

{

post = null;

currentPost = 0;

ReadingBlogForm.ActiveForm.Update();

}

else

{

post = posts.Skip(currentPost).Take(1).FirstOrDefault();

}

if (post == null)

{

MessageBox.Show("Brak kolejnych postów");

return;

}

descriptionReadingBlogRichTextBox.Text = post.Desc;

titleReadingPostLabel.Text = post.Title;

}

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

{

blog.Marks += Double.Parse(yourScoreOfTheBlogTextBox.Text);

blog.AmountOfMarks += 1;

scoreOfTheBlogContainLabel.Text = (blog.Marks / blog.AmountOfMarks).ToString();

this.\_bc.SaveChanges();

}

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

{

using (BlogContext context = new BlogContext())

{

context.Database.Log = message => Trace.WriteLine(message);

var listOfPosts = (from b in context.Blogs

from p in context.Posts

where b.BlogId == \_blogId

&& b.BlogId == p.BlogId

select b.Posts)

.ToList()

.Count();

postTitleContainLabel.Text = listOfPosts.ToString();

postTitleContainLabel.Visible = true;

}

}

}

}

Widzimy tutaj zastosowanie lazy loadingu w linijce:

posts = \_bc.Posts.Where(p => p.BlogId == \_blogId).OrderBy(p => p.PostId).AsQueryable();

Kiedy rzutujemy:

var listOfPosts = (from b in context.Blogs

from p in context.Posts

where b.BlogId == \_blogId

&& b.BlogId == p.BlogId

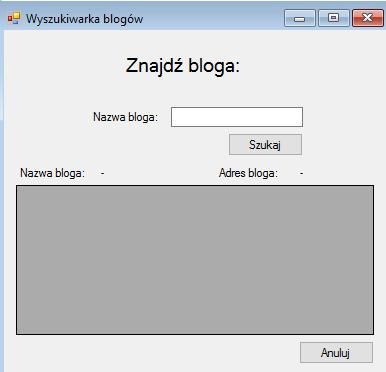
select b.Posts)

.ToList()

To przykład immidiate execution.

Użyte zostały też liczne funkcje agregujące typy count(),Take(1).FirstOrDefault();

Ostatnią przezemnie funkcjonalnością jest zakładka znajdż bloga, służąca do znajdowania bloga po nazwie:



Funkcjolaność jest oczywista. Kod formy wygląda tak:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CodeFirstLab1

{

public partial class FindingForm : Form

{

private BlogContext \_bc;

public FindingForm(BlogContext bc)

{

InitializeComponent();

this.\_bc = bc;

}

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

{

\_bc.Database.Log = message => Trace.WriteLine(message); // obserwowanie generowanego kodu SQL i momentu jego wykonania

Blog blog = \_bc.Blogs

.Include("Posts")

.Where(b => b.Name == nameOfSearchedBlogTextBox.Text.ToString())

.FirstOrDefault<Blog>(); // Include() - Eager Loading

if (blog == null)

{

MessageBox.Show("404 - Blog not found");

return;

}

foundedPostDataGridView.DataSource = blog.Posts;

foundBlogNameContainLabel.Text = nameOfSearchedBlogTextBox.Text;

foundBlogUrlContainLabel.Text = blog.Url.ToString();

}

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

{

FindingForm.ActiveForm.Close();

}

}

}

Widzimy tutaj użycie metody include aby wymusić eager loading.