

Adam Dyszy
Laboratorium 1

Bazy Danych

.Net, Entity Framework

Wykonaną na zajęciach aplikację Blog rozszerzyłem o nową funkcjonalność dodając okienka WindowsForm z wykorzystaniem mechanizmów EF i L2E.

The image shows two overlapping Windows Forms applications. The main application, titled 'BlogForm', displays two data grids. The 'Blogs' grid lists 7 blogs with columns for BlogId, Name, and Url. The 'All Posts' grid lists 9 posts with columns for PostId, Title, and Content. Below the grids are search and action buttons. The 'AddPost' dialog box is open, showing a 'Choose Blog' dropdown set to 'Blog2', an 'Enter Title' text box containing 'CoolTitle', and a 'Content' text area containing 'Cool Content' and 'Very Cool'. It has 'Add' and 'Cancel' buttons.

BlogId	Name	Url
1	Blog1	
2	Blog2	
3	Blog3	
4	Blog4	
5	Blog5	
6	Blog6	
7	b11	

PostId	Title	Content
11	S	GD
12	E2	Fee3
13	E33	e333
14	E33333	33333333
15	3g	g3g
16	f2	f22
17	3333333333	3333
18	Feeeeeer	fer2
19	b6	6666

Buttons: Search for blog by name: [text box], Add Post, Show All Posts, Get Posts for selected Blog, Refresh

AddPost Dialog:

Choose Blog: Blog2 (dropdown)
Enter Title: CoolTitle (text box)
Content: Cool Content, Very Cool (text area)
Buttons: Add, Cancel

```
public partial class AddPostForm : Form
{
    private BlogContext _context;

    public AddPostForm()
    {
        InitializeComponent();
        _context = new BlogContext();
    }
}
```

```

private void AddPostForm_Load(object sender, EventArgs e)
{
    comboBox1.Items.Add("Choose a blog...");
    comboBox1.SelectedItem = comboBox1.Items[0];
    var blogs = from b in _context.Blogs
                orderby b.Name
                select b.Name;
    foreach (var blog in blogs)
    {
        comboBox1.Items.Add(blog);
    }
}

private void add_Click(object sender, EventArgs e)
{
    var selectedBlog = (string)comboBox1.SelectedItem;
    int blogId;

    blogId = (from b in _context.Blogs where b.Name ==
selectedBlog select b.BlogId).FirstOrDefault();
    if (blogId == 0)
    {
        MessageBox.Show("No blog selected.");
        return;
    }
    if (textBox1.Text == "")
    {
        MessageBox.Show("No title!");
        return;
    }
    if (richTextBox1.Text == "")
    {
        MessageBox.Show("Post should have some content...");
        return;
    }
    var post = new Post();
    post.BlogId = blogId;
    post.Content = richTextBox1.Text;
    post.Title = textBox1.Text;
    _context.Posts.Add(post);
    _context.SaveChanges();
    Hide();
    DestroyHandle();
}

```

```

    }
    private void cancel_Click(object sender, EventArgs e)
    {
        Hide();
        DestroyHandle();
    }
}

```

- Jak widać na obrazku wyżej i w kodzie źródłowym dodana została możliwość dodawania Postów dla wybranego Bloga poprzez GUI. We wstawianiu do bazy został użyty Query Syntax, oraz standardowo Deferred Execution czyli zapytania się wykonają dopiero gdy odniesiemy się do zmiennych które je wykorzystują.

```

private void addPost_Click(object sender, EventArgs e)
{
    (new AddPostForm()).ShowDialog();
    this.refresh_Click(sender, e);
}

```

- W głównym Formie dodałem powyższy kod który pozwoli na dodanie postu i odpowiednio zaktualizowania wyświetlanych tabel dzięki Eager Loadingu w refresh_Clicku.

BlogForm

Blogs:

	BlogId	Name	Url
▶	2	Blog2	

Posts for blog Blog2:

	PostId	Title	Content
▶	5	fasdf	asdfasafshg
	13	E33	e333
	20	CoolTitle	Cool ContentVery...

- Tutaj z kolei pokazane jest wyszukanie Bloga "Blog2" oraz Posty które do niego należą.

```
private void getSelectedBlog()
{
    var selectedRows = blogDataGridView.SelectedRows;
    selectedBlogId =
int.Parse(selectedRows[0].Cells[0].FormattedValue.ToString());
    selectedBlogName =
selectedRows[0].Cells[1].FormattedValue.ToString();
}
```

- Sposób na pobieranie BlogID z grida z danymi gdzie użytkownik może wybrać rzędy w tabeli

```

private void getAllPosts_Click(object sender, EventArgs e)
{
    takeAllPosts = true;
    refresh_Click(sender, e);
    postLabel.Text = "All Posts:";
}
private void getPostsForSelectedBlog_Click(object sender,
EventArgs e)
{
    getSelectedBlog();
    takeAllPosts = false;
    refresh_Click(sender, e);
    postLabel.Text = "Posts for blog " + selectedBlogName + ":";
}
private void search_Click(object sender, EventArgs e)
{
    nameToSearch = searchBox.Text.ToString();
    refresh_Click(sender,e);
}

```

```

private void refresh_Click(object sender, EventArgs e)
{
    IQueryable<Blog> blogs = _context.Blogs;
    if (nameToSearch != "")
    {
        blogs = from b in blogs
                where b.Name.Contains(nameToSearch)
                select b;
        //blogs = blogs.Where(b =>
b.Name.Contains(nameToSearch));
    }
    blogDataGridView.DataSource = blogs.ToList();
    blogDataGridView.Update();
    if (takeAllPosts)
    {
        postsDataGridView.DataSource = _context.Posts.ToList();
    }
    else
    {
        Blog blog = _context.Blogs.Include("Posts").Where(b =>
b.BlogId == selectedBlogId).FirstOrDefault();
        postsDataGridView.DataSource = blog.Posts;
    }
    postsDataGridView.Update();
}

```

- Tutaj mamy podział na pokazywanie postów z wszystkich blogów albo z jednego bloga. Najważniejsza logika jest w funkcji refresh która wymaga ponownego wykonywania zapytań dla nowych zmian. Tym razem wykorzystałem również Method Syntax z użyciem Navigation Properties: używanie Include("Posts") pozwala na wybranie Postów od Bloga dzięki ich połączeniu w tabeli. Zastosowany jest również Eager Loading, dzięki czemu mamy od razu dane bez czekania co przyspiesza nasz update.