PV178 – Lab10

Projekt

- Dohodnout se na tématu
- Prezentace aktuálního stavu projektu: poslední týden na cvičení (max 5. minut)
- Více v interaktivní osnově

Agenda

- LINQ To XML
- Reflexe
- Expression Trees

LINQ to XML preview

Element vs. Atribut

```
<subjects>
    <subject AKA="C# basic" IsBestSubject="true">PV178</subject>
    <subject AKA="Inf. security and cryptography">PV080</subject>
    <subject AKA="JAVA ☹">PB162</subject>
</subjects>
```

LINQ to XML

- Rozhraní pro práci s XML
- Využívá rozšiřujících metod (jako LINQ to OBJECTS)
- Třída Xdocument (nástupce třídy XmlDocument)
- Pracujeme se stromem elementů

Příklad

```
var rootElement = XElement.Load(<path_to_file>);

// Metody
<x_element>.Element("Customer");      // Vybere první child element „Customer“
<x_element>.Elements("People");       // Vybere všechny child elementy typu „People“ (přímé)
<x_element>.Attribute(("Id").Value)   // Vybere hodnotu atributu „Id“
<x_element>.Descendants("Car")        // Všechny následovníky „Car“ (i nepřímé)



// Samozřejmě zde fungují i standardní metody LINQu
<x_elements>.Select()
<x_elements>.Where()
<x_elements>.OrderBy()
```

Samostatná práce

- LinqToXml

Reflexe

- Prozkoumávání neznámých sestav (assemblies)
- Získávání informací, které nejsou běžně přístupné v programu
- Umožňuje např.: volat private metody mimo třídy, nastavovat private proměnné apod.
- Použití:
  - Debugging (reference a metody DLL knihoven)
  - Testing?!
  - Frameworky, knihovny

Reflexe – příklad 1

```
public class Person
{
        private int age;
}
```

```
Type personType = typeof(Person);

var fields = personType.GetFields();
var constructors = personType.GetConstructors()
var methods = personType.GetMethods()
```

Reflexe – příklad 2

```
public class Person
{
        public int age { get; }; // bez setteru
}


// Vytvoří člověka
var person = new Person();

// Alfa-omega reflexe
Type personType = typeof(Person);

// Zplnoletnění osoby
var nameField = personType.GetProperty("age");
nameField.SetValue(customer, 18);
```

Samostatná práce

- Reflexe

Expression trees (strom výrazů)

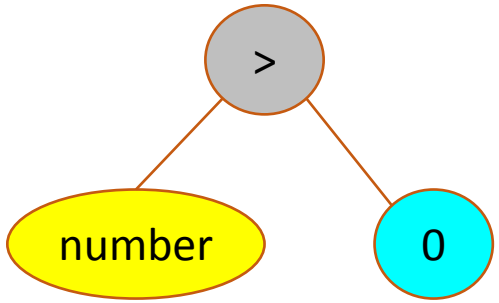
- Expression tree reprezentuje kód ve stromové datové struktuře, kde každý uzel je výraz, např.: volání metody, binární operace (např. x < y)
- Používá se k:
  - dynamická modifikace kódu (za běhu)
  - sestavování dynamický lambda výrazů (i celých LINQ dotazů)
- Sestavení stromu je poměrně náročné (hodně řádků kódu, hodně přemýšlení)
- Reálné použití – při sestavování dotazů pro SQL

Expression Trees (strom výrazu) - příklad

```
// Expression tree – můžu měnit lambda výraz dynamicky
var parExpr = Expression.Parameter(typeof(int), "number");
var consExpr = Expression.Constant(0);
var binExpr = Expression.GreaterThan(parExpr, consExpr);
var lambdaExpr = Expression.Lambda<Func<int, bool>>(binExpr, parExpr);
var func1 = lambdaExpr.Compile();

// Static lambda – nemůžů měnit dynamicky
Func<int, bool> func2 = number => number > 0;

// Použití je stejné
func1(3);
func2(3);
```



Samostatná práce

- ExpressionTrees