LINQ za SQL

Katedra za informatiku
Fakultet tehničkih nauka
Univerzitet u Novom Sadu

Uvod

- LINQ to SQL – .NET komponenta koja koristi mehanizme LINQ jezika za rad sa podacima smeštenim u relacionoj bazi podataka
- Dostupna od .NET verzije 3.5
- Obezbeđuje mapiranje objektnog na relacioni model (O-R mapiranje)
- Podržava samo Microsoft SQL Server
- Kao alternativa za LINQ to SQL razvijen je ADO .NET Entity Framework koji pruža širi skup funkcionalnosti i mogućnost rada sa različitim SUBP

Mapiranje objektnog modela na relacioni model

- Mapiranje se vrši anotiranjem elemenata objektnog modela (klasa i atributa)
- Mapiranje klase na tabelu u bazi podataka
  - Anotacija **`Table`**

```
[Table(Name = "STUDENT")]
public class Student
{
  ...
```

Mapiranje objektnog modela na relacioni model

- Mapiranje atributa klase na kolonu u bazi podataka
  - Anotacija **`Column`**

```
public class Student { ...
   [Column(IsPrimaryKey = true, Name = "ID",
        CanBeNull = false)]
   public int Id { get; set; }

   [Column (Name="GRAD_ID")]
   public int GradId { get; set; }
```

Mapiranje objektnog modela na relacioni model

- Mapiranje veze N:1
  - Definiše se atribut tipa **`EntityRef`** koji predstavlja referencu na drugu klasu (veza asocijacije)
  - Anotacija **`Association`** definiše atribute preko kojih se veza ostvaruje

```
public class Student { ...
  private EntityRef<Grad> grad;
   [Association(Storage = "grad",
      ThisKey ="GradId", OtherKey = "Id",
            IsForeignKey=true)]
   public Grad Grad
   {
         get { return this.grad.Entity; }
         set { this.grad.Entity = value; }
   }
```

Mapiranje objektnog modela na relacioni model

- Mapiranje veze 1:N
  - Definiše se atribut tipa **`EntitySet`** koji predstavlja listu od N objekata (veza asocijacije)
  - Anotacija **`Association`** definiše atribute preko kojih se veza ostvaruje

```
public class Grad { ...
  private EntitySet<Student> studenti =
        new EntitySet<Student>();
  [Association(Storage="studenti",
          OtherKey="GradId", ThisKey="Id")]
  public EntitySet<Student> Studenti
  {
    get { return this.studenti; }
    set { this.studenti.Assign(value); }
  }
```

Veza sa bazom podataka

- Klasa **`DataContext`**
  - Povezivanje na bazu podataka
  - Preuzimanje podataka iz baze
  - Slanje izmenjenih podataka u bazu

```
DataContext dc = new DataContext(
    @"Integrated Security=true;
        InitialCatalog=Fakultet;
        Data Source=.\SQLExpress");
```

Preuzimanje podataka

- Korišćenjem **`DataContext`** klase
  - Klasa **`Table -`** programska reprezentacija tabele iz baze podataka

```
Table<Student> studenti=dc.GetTable<Student>();
```

- LINQ upitom

```
var res = from s in dc.GetTable<Student>()
          select s;
```

- Direktnim SQL upitom

```
var res = dc.ExecuteQuery<Student>(
  "SELECT * FROM STUDENT");
foreach (Student s in res)
  Console.WriteLine(s.Ime);
```

Dodavanje sloga u tabelu u bazi podataka

- Kreira se novi objekat klase koja u objektnom modelu reprezentuje tabelu u bazi podataka
- Objekat se ubacuje u odgovarajuću tabelu DataContext objekta
- Izmena se šalje u bazu podataka

```
Student s = new Student("Petar",
  "Petrovic");
studenti.InsertOnSubmit(s);
dc.SubmitChanges();
```

Izmena sloga u bazi podataka

- Preuzima se slog koji je potrebno promeniti iz baze podataka
- Vrši se izmena nad objektom koji reprezentuje slog
- Izmena se šalje u bazu podataka

```
IEnumerable<Student> res = from s in
   studenti where s.Id == 23 select s;
Student student = res.ElementAt(0);
student.Ime = "Marko";
dc.SubmitChanges();
```

Brisanje sloga iz baze podataka

- Preuzima se slog koji je potrebno izbrisati iz baze podataka
- Objekat se izbacuje iz odgovarajuće tabele DataContext objekta
- Izmena se šalje u bazu podataka

```
IEnumerable<Student> res = from s in
    studenti where s.Id == 23 select s;
Student student = res.ElementAt(0);
studenti.DeleteOnSubmit(student);
dc.SubmitChanges();
```

Slanje izmena u bazu podataka

- Sve izmene se vrše nad objektima u memoriji, dok god se podaci eksplicitno ne pošalju u bazu podataka pozivom metode **`SubmitChanges`** klase **`DataContext`**
- U okviru metode utvrđuju se razlike između podataka u memoriji i podataka u bazi podataka
- Na osnovu utvrđenih razlika redom se formiraju SQL upiti koji izvršavaju odgovarajuće operacije nad bazom podataka

Upravljanje transakcijama

- Implicitno
  - Ako nije drugačije naznačeno, pri svakom pozivu metode **`SubmitChanges`** kreira se nova transakcija u okviru koje se obavljaju sve izmene koje metoda vrši (redom se svi SQL upiti izvršavaju u okviru ove transakcije)
- Eksplicitno
  - Moguće je eksplicitno kreirati novu transakciju (objekat klase **`System.Data.Common.DbTransaction`**)
  - Svaka **`SubmitChanges`** metoda se tada izvršava u okviru ove transakcije