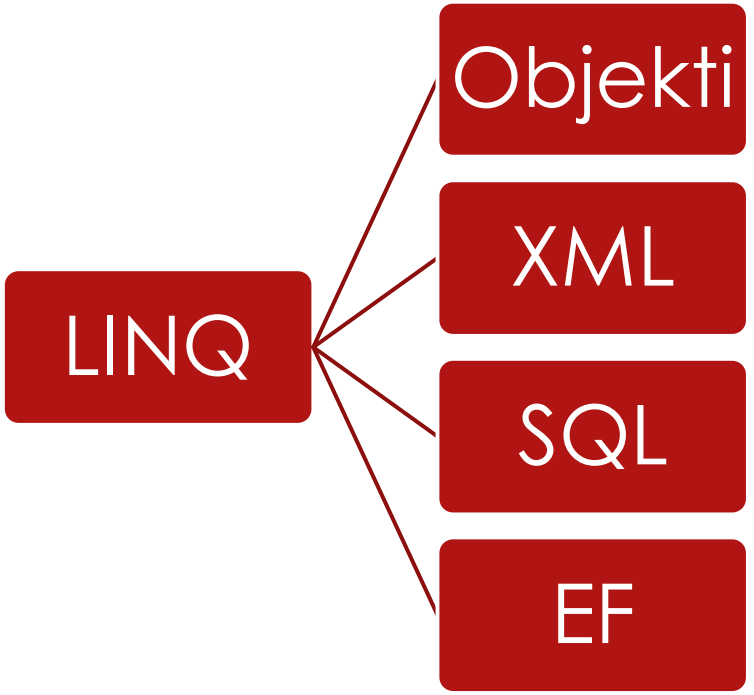
LINQ

AUTORS: ILMĀRS APEINĀNS

Kas ir LINQ?

Language-Integrated Query (LINQ) ir spēcīgs tehnoloģiju kopums, kura pamatā ir vaicājumu iespēju integrācija tieši C# valodā. LINQ Queries ir pirmās klases valodas konstrukcija C# .NET, tāpat kā klases, metodes, notikumi. LINQ nodrošina konsekventu vaicājumu pieredzi vaicājumu objektiem (LINQ uz objektiem), relāciju datu bāzēm (LINQ uz SQL) un XML (LINQ uz XML).



Pozitīvais un negatīvais

- Pozitīvi:
  - Nav jāraksta SQL vaicājumi / vaicājumu procedūras
  - Palielina produktivitāti
- Negatīvi:
  - Domāts priekš vienkāršiem vaicājumiem

LINQ darbībā

using System.Linq;

```
public static void Main(string[] args)
{
    //Linq Query Syntax
    var context = new Context();
    var query =
        from movie in context.Movies
        where movie.Title.Contains("Mat")
        select movie;
    foreach (var movie in query)
        {
        Console.WriteLine($"{movie.Title}");
    }
}
```

```
public static void Main(string[] args)
{
    //Extended query methods
    var context = new Context();
    var query = context.Movies
        .Where(m => m.Title.Contains("Mat"))
        .ToList();
    foreach (var movie in query)
        {
        Console.WriteLine($"{movie.Title}");
    }
}
```

LINQ sintakse

Darbība	LINQ sintakse	Piemērs
Kur meklējam	from c in context.<class>	from c in context.Courses
Ierobežojumi	where c.<parameter> == <variable>	where c.Level == 1
Kārtošana	orderby c.<parameter> descending	orderby c.Level descending, c.Name
Attēlošana	select c;	
Atlašīšana (grupēšana)	group c by c.<parameter> into g	group c by c.Level into g

LINQ sintakse

Darbība	LINQ sintakse	Piemērs
Apvienošana (Inner join)	from c in context.<class> join a in context.<class> on c.<parameter> equals a.<parameter> select new { <variable> = c.<parameter> , <variable> = a.<parameter> }	from c in context.Courses join a in context.Authors on c.AuthorId equals a.Id select new { CourseName = c.Name, AuthorName = a.Name}
Apvienošana (group join)	from a in context.<class> join c in context.<class> on a.<parameter> equals c. <parameter> into g select new { <variable> = a.<parameter> , <variable> = g}	from a in context.Authors join c in context.Courses on a.Id equals c.AuthorId into g select new { AuthorName = a.Name, Courses = g}

LINQ sintakse

Darbība	LINQ sintakse	Piemērs

Apvienošana
(Cross join)
	from a in context.<class>
	from c in context.<class>
	select new {
	AuthorName = a.<parameter> ,
	CourseName = c.<parameter>
	}
		from a in context.Authors
		from c in context.Courses
		select new {
		AuthorName = a.Name
		CourseName = c.Name,
		}

LINQ Extension metode

Darbība	LINQ sintakse	Piemērs
Kur meklējam	context.<class>	context.Courses
Ierobežojumi	.Where(c => c.<parameter> == <variable>)	.Where(c => c.Level == 1)
Kārtošana	.OrderBy(c => c.<parameter>) .OrderByDescending(c => c.<parameter>) .ThenBy(c => c.<parameter>) .ThenByDescending(c => c.<parameter>)	.OrderBy(c => c.Name) .OrderByDescending(c => c.Name) .ThenBy(c => c.Level) .ThenByDescending(c => c. Level)
Attēlošana	.Select c;	.Select(c => new {CourseName = c.Name, AuthorName = c.Author.Name})
Atšķiršana	.Distinct()	

LINQ Extension metode

Darbība	LINQ sintakse	Piemērs
Atlašīšana (grupēšana)	.GroupBy(c => c.<parameter>)	.GroupBy(c => c.Level)
Apvienošana (Inner join)	.Join(context.<class>, c => c.<parameter>, a => a.<parameter>, (<newtempclass>,<newtempclass>) => new { <variable> = <newtempclass>.<variable> , <variable> = <newtempclass>.<variable> })	.Join(context.Authors, c => c.AuthorId, a => a.Id, (course, author) => new { CourseName = course.Name, AuthorName = author.Name})
Apvienošana (group join)	.GroupJoin(context.Courses, a => a. <parameter>, c => c. <parameter>, (<newtempclass>, <newtempclass>) => new { <variable> = <variable>, <variable> = <variable> })	.GroupJoin(context.Courses, a => a.Id, c => c.AuthorId, (author, course) => new { AuthorName = author, Course = courses })

LINQ Extension metode

Darbība	LINQ sintakse	Piemērs
Apvienošana (Cross join)	Context.<class> .SelectMany(a => context.<class>, (<newtempclass>, <newtempclass>) => new { <variable> = <newtempclass>.<parameter>, <variable> = <newtempclass>.<parameter>, })	Context.Authors .SelectMany(a => context.Courses, (author, course) => new { AuthorName = author.Name, CourseName = course.Name })

LINQ: Filtrēšana & Projekcija

- Metodes sintakse (Movies):

```
var recentSciFi = context.Movies
    .Where(m => m.Genre == "Sci-Fi" && m.ReleaseDate >= new DateTime(2010,1,1))
    .Select(m => new { m.Id, m.Title, m.ReleaseDate })
    .ToList();
```

- Vaicājumu sintakse (Movies):

```
var recentSciFiQ = (from m in context.Movies
    where m.Genre == "Sci-Fi" && m.ReleaseDate >= new DateTime(2010,1,1)
    select new { m.Id, m.Title, m.ReleaseDate }).ToList();
```

Eksistence: Any() un All()

- Metodes sintakse:

```
bool hasAnyCharacters = context.Movies.Any(m => m.Characters.Any());
bool allHaveCharacters = context.Movies.All(m => m.Characters.Any());
```

- Vaicājumu sintakse (izmanto Any() where nosacījumā):

```
var withChars = (from m in context.Movies
        where m.Characters.Any()
        select new { m.Id, m.Title }).ToList();
```

Iekļaušana: Include / ThenInclude

- Metodes sintakse (Include):

```
var moviesWithChars = context.Movies
        .Include(m => m.Characters)
        .AsNoTracking()
        .ToList();
```

- Vaicājumu sintakse (projekcija alternatīva Include vietā):

```
var moviesDto = (from m in context.Movies
        select new { m.Title, Characters = m.Characters.Select(c => c.Name) })
        .AsNoTracking().ToList();
```

Biežāk sastopamās kļūdas: Translācija

- “Client evaluation not allowed” — pārraksti vaicājumu, lai to var translēt uz SQL.
- “The LINQ expression ... could not be translated” — izmanto atbalstītas metodes/izteiksmes vai materializē agrāk.
- Izsaukumi uz .NET-only metodēm (piem., custom funkcijas) vaicājuma iekšienē netiks pārtulkotas.
- Projekcija pirms Include: “Include has been ignored because the query results in a non-entity type”.

Biežāk sastopamās kļūdas: Izsekošana

- “The instance of entity type … is already being tracked …” — tajā pašā DbContext ir divi eksemplāri ar vienādu PK.
- “The instance of entity type … cannot be tracked because another instance … is already being tracked” — sajaukta Attach/Update.
- Disasociē/Attach/AsNoTracking: izvēlies atbilstoši scenārijam (lasīšana vs modificēšana).
- 'A second operation was started on this context...' — paralēlas operācijas uz viena DbContext (asinhronā piekļuve).

Biežāk sastopamās kļūdas: FK un dzēšana

- “FOREIGN KEY constraint failed” — trūkst FK vai neatbilstošs DeleteBehavior (Cascade/Restrict/SetNull).
- “Each entity type requires a primary key” — aizmirsta PK konfigurācija.
- 'Cannot insert explicit value for identity column …' — izmanto DB ģenerētus PK vai ieslēdz IDENTITY_INSERT (piesardzīgi).

Praktiskas nianses: Include un kartēšana

- Daudzi Include var radīt 'cartesian explosion' — izmanto .AsSplitQuery() vai projekciju.
- “Cannot create a DbSet for 'X' because this type is not included in the model” — modelī nav reģistrēts tips.
- NoTracking vaicājumi + atjauninājumi — Attach pirms Update.

LINQ izņēmumi: First/Single/Nullables

- “Sequence contains no elements” — First() ir tukša; izmanto FirstOrDefault() un pārbaudi null.
- “Nullable object must have a value” — lietots .Value, kad nav vērtības; pārbaudi HasValue.
- 'InvalidOperationException: Multiple elements' — Single() atrod vairāk par vienu; izvēlies SingleOrDefault vai garantē unikālumu.

Kolācijas, datumi un .Contains

- String salīdzinājumi atkarīgi no DB kolācijas; EF.Functions.Like palīdz kontrolēt uzvedību.
- DateTime.Kind/TimeZone — izvēlies vienotu stratēģiju (parasti UTC DB).
- .Contains uz lieliem sarakstiem -> liels IN (...) — apsver Join/TVP/pagaidu tabulas.

Paldies par uzmanību

AUTORS: ILMĀRS APEINĀNS