Přístup k databázím

Ing. Michal Radecký, Ph.D.

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Přístup k datům

- Využití určité míry abstrakce (dle použitého přístupu) pro přístup k externím datům
- Databázové servery, lokální databázové soubory, XML, JSON, atd.
- Různá paradigmata přístupů k datům

Relační přístup

- Databáze je založena na relačním přístupu
  - Tabulky – entitní typy
  - Řádky – záznamy
  - Sloupce – atributy
- Pracuje se přímo s dotazováním nad daty (SQL dle konkrétní DB vrstvy) a následném zpracování odpovědi
- Obecně pomocí klasických možností C# (kolekce, dynamic, object, …)

Objektově-relační přístup

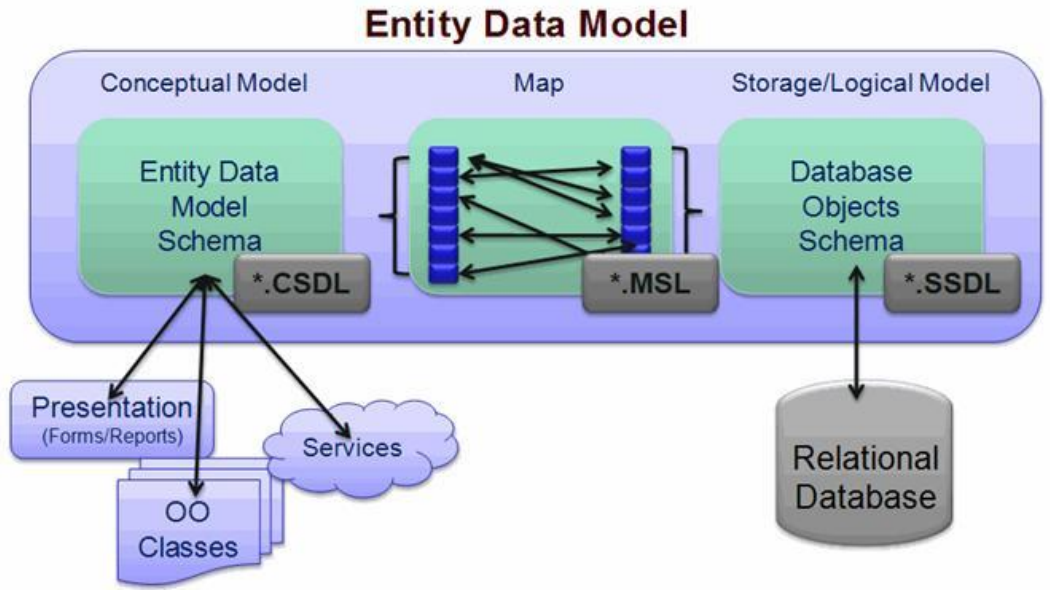
- Databáze je pořád založena na relačním přístupu
- Z pohledu implementace však pracujeme s konkrétními objekty (Properties a Metody)
- Objektově relační mapování (ORM) – vysoká míra abstrakce
- Data z datové vrstvy jsou rovnou přístupné v podobě objektů
- Nepoužívá se přímé dotazování (SQL), ale SQL se automatizovaně generuje
- Obvykle vysoká režie a možná neefektivita (efektivnější u jednoduchých DB)
- Obecně implementace přístupu CRUD (Create, Read, Update, Delete)

ORM frameworky

- Existuje celá řada implementací, obvykle jako samostatná vrstva (rozšíření) pro konkrétní platformu

- Entity Framework
  - OpenSource, součást konceptu ADO.NET, existuje již i varianta EF Core
  - Entity Data Model – mapování .NET objektů na datové objekty, vazby, apod.
  - Entity SQL, LINQ to Entities, Native SQL

- NHibernate, LINQ to SQL



Entity Framework dotazování

- LINQ to Entities

```
//Querying with LINQ to Entities
using (var context = new SchoolDBEntities())
  {
    var query = context.Students
                       .where(s => s.StudentName == "Bill")
                       .FirstOrDefault<Student>();
  }
```

```
using (var context = new SchoolDBEntities())
  {
    var query = from st in context.Students
                where st.StudentName == "Bill„
                select st; var student = query.FirstOrDefault<Student>();
  }
```

- Entity SQL

```
//Querying with Object Services and Entity SQL
string sqlString = "SELECT VALUE st FROM SchoolDBEntities.Students " +
                   "AS st WHERE st.StudentName == 'Bill‘“;

var objctx = (ctx as IObjectContextAdapter).ObjectContext;
ObjectQuery<Student> student = objctx.CreateQuery<Student>(sqlString);
Student newStudent = student.First<Student>();
```

- Native SQL

```
using (var ctx = new SchoolDBEntities())
  {
    var studentName = ctx.Students.SqlQuery("Select studentid, studentname,
                          standardId from Student where studentname='Bill'").FirstOrDefault<Student>();
  }
```

Micro-ORM

- Velké ORM pokrývají komplexní problematiku datové vsrtvy (cachování, mapování, migrace, lazy loading, in memory storage, …)
- Jsou robustní, ale i náročné jak na vývoj, tak provoz.

- Micro-ORM zpravidla obsahují jen malou část zaměřenou na komunikaci s DB (načítání a přenos dat) a převod dat na objekty
  - Efektivnější vývoj a nasazení
  - Větší výkon
  - Nehodí se vždy!

  - Dapper

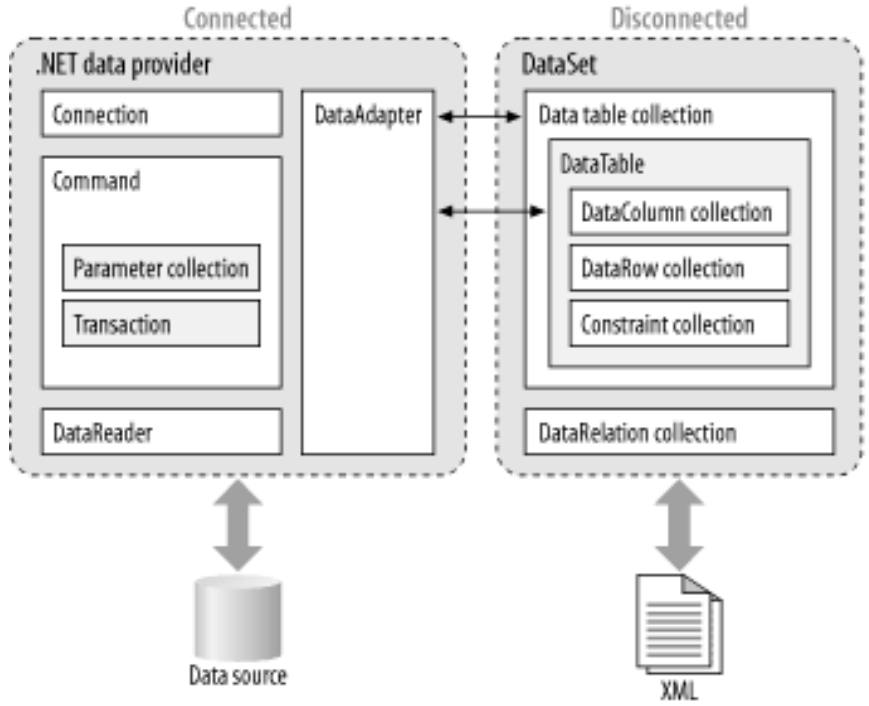
Use Case#	Query Type	ORM	Linq/Plain	.AsNoTracking() Applied	Time	Fastest
1.	Select 5k rows from 500k	Dapper	N/A	N/A	129.2 ms	Dapper
		EF Core 2	Linq	Yes	137.1 ms	
			Linq	No	148.1 ms	
			Plain SQL	Yes	132.6 ms	
			Plain SQL	No	149.2 ms	
2.	Insert new row	Dapper	N/A	N/A	5.423 ms	Dapper
		Ef Core 2	Linq	No	16.20 ms	
			Plain SQL	No	5.645 ms	
3.	Update Row at 5001 index	Dapper	N/A	N/A	4.438 ms	Dapper
		Ef Core 2	Linq	No	4.718 ms	
			Plain SQL	No	4.652 ms	

Další typy databází a přístupu k datům

- Existuje celá řada dalších přístupů
- Hraje zde roli fyzické uložení dat a způsob přístupu k nim
    - Objektové databáze (MongoDB)
    - Textová data
    - XML databáze
    - Realtime databáze
    - BigData
    - API
    - Atd.

ADO.NET

- Knihovna zastřešující přístup k datům a manipulaci s nimi
- Poskytuje abstrakci od fyzického uložení dat, využívá tzv. poskytovatele (SQL, OLEDB, ODBC)
- Součást .NET Frameworku, existuje již i pro .NET Core (.NET 5)
- Dva typy přístupu
  - Connected – přímý přístup k datovému zdroji
  - Disconnected/Connectionless – „vrstva navíc“ , která tvoří abstrakci a poskytuje služby



`System.Data`

https://docs.microsoft.com/en-us/dotnet/api/system.data?view=net-5.0

Connected přístup

- Primárně read-only přístup k datům datového zdroje a spouštění příkazů nad datovým zdrojem
- Spojení s datovým zdrojem se udržuje, data jsou vždy aktuální
- Vhodné pro „málo“ paralelní aplikace a krátké transakce

- `IDbConnection`
  - Hlavní objekt reprezentující datový zdroj
  - SqlConnection, OleDbConnection
- `IDbCommand`
  - Reprezentuje konkrétní příkaz nad zdrojem
- `IDbTransaction`
  - Reprezentuje transakci složenou z příkazů
- `IDataReader`
  - Objekt pro přístup k výsledkům dotazu
  - Umožňuje přistupovat iteračně

ConnectionString

- Řetězec specifikující parametry potřebné k připojení datového zdroje
  - Typ poskytovatele
  - Identifikace zdroje
  - Autorizace uživatele
  - Specifické parametry
- Parametry a obsah se liší dle poskytovatelů a typu zdroje (DB vs. lokální soubor)
- KeyValuePair oddělené středníky
- Umístění
  - Lokálně jako string
  - App.config, appsettings.json. Web.config
  - `ConnectionStringBuilder`

```
Data Source=190.190.200.100,1433;Network
Library=DBMSSOCN;Initial
Catalog=myDataBase;User
ID=myUsername;Password=myPassword;
```

System.Data.SqlClient

- Obecný poskytovatel pro SQL databáze
- V .Net Core nutné doinstalovat NuGet package
- Používat v `using{}` nebo metody `Close()` a `Dispose()`
- Realizace příkazu: `ExecuteNonQuery()`, `ExecuteReader()`, `ExecuteScalar()`

```
using (SqlConnection conn = new SqlConnection("connection-string"))
{
  conn.Open();
  SqlCommand command = new SqlCommand("SELECT * FROM [TableName]", conn);

  using (SqlDataReader reader = command.ExecuteReader())
  {
    while (reader.Read())
    {
      Console.WriteLine(String.Format("{0} \t | {1} \t | {2} \t | {3}",
                        reader[0], reader["state"], reader[2], reader[3]));
    }
  }
}
```

Příkaz a parametry

- Specifikace SQL dotazu jako objekt Command
- Pokud je třeba předat do SQL dotazu parametry, **nevkládát přímo do textu**, ale pomocí parametrů (s ohledem na poskytovatele)

- Typová kontrola mezi C# a DB
- Zabezpečení proti injection apod.
- Input i Output parametry

```
SqlCommand command = new SqlCommand("SELECT * FROM TableName WHERE Name LIKE @N", conn);
command.Parameters.Add(new SqlParameter("N", name));
```

Transakce

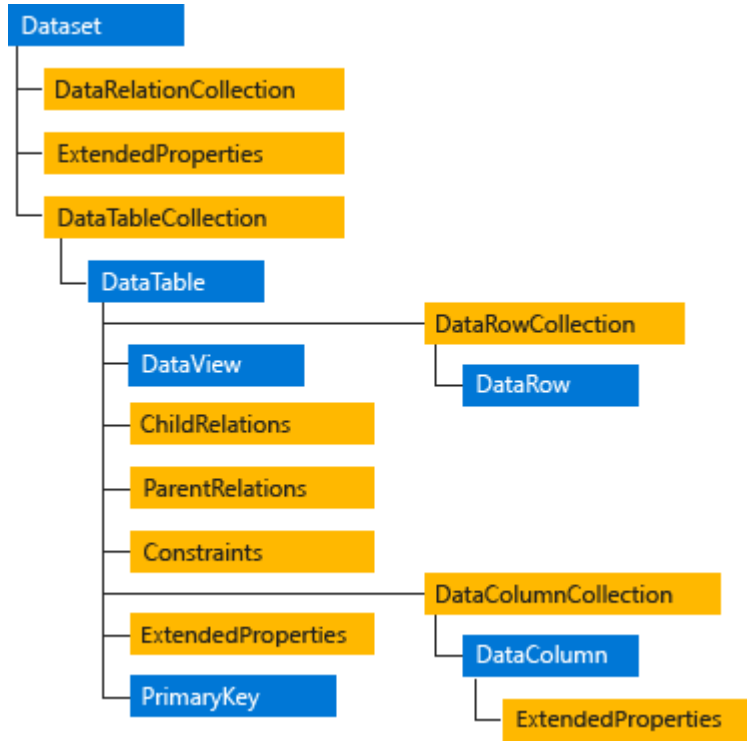
- Transakce umožňují atomizovat operace s daty
- Podpora lokálních transakcí (v rámci jednoho připojení)
- Distribuované transakce řešené pomocí `System.Transaction`
- Izolační úroveň definuje, jak se přistupuje k datům ve zdroji v průběhu transakce
  - `ReadUncommitted`
  - `ReadCommitted`
  - `RepeatableRead`
  - `Serializable`

Transakce

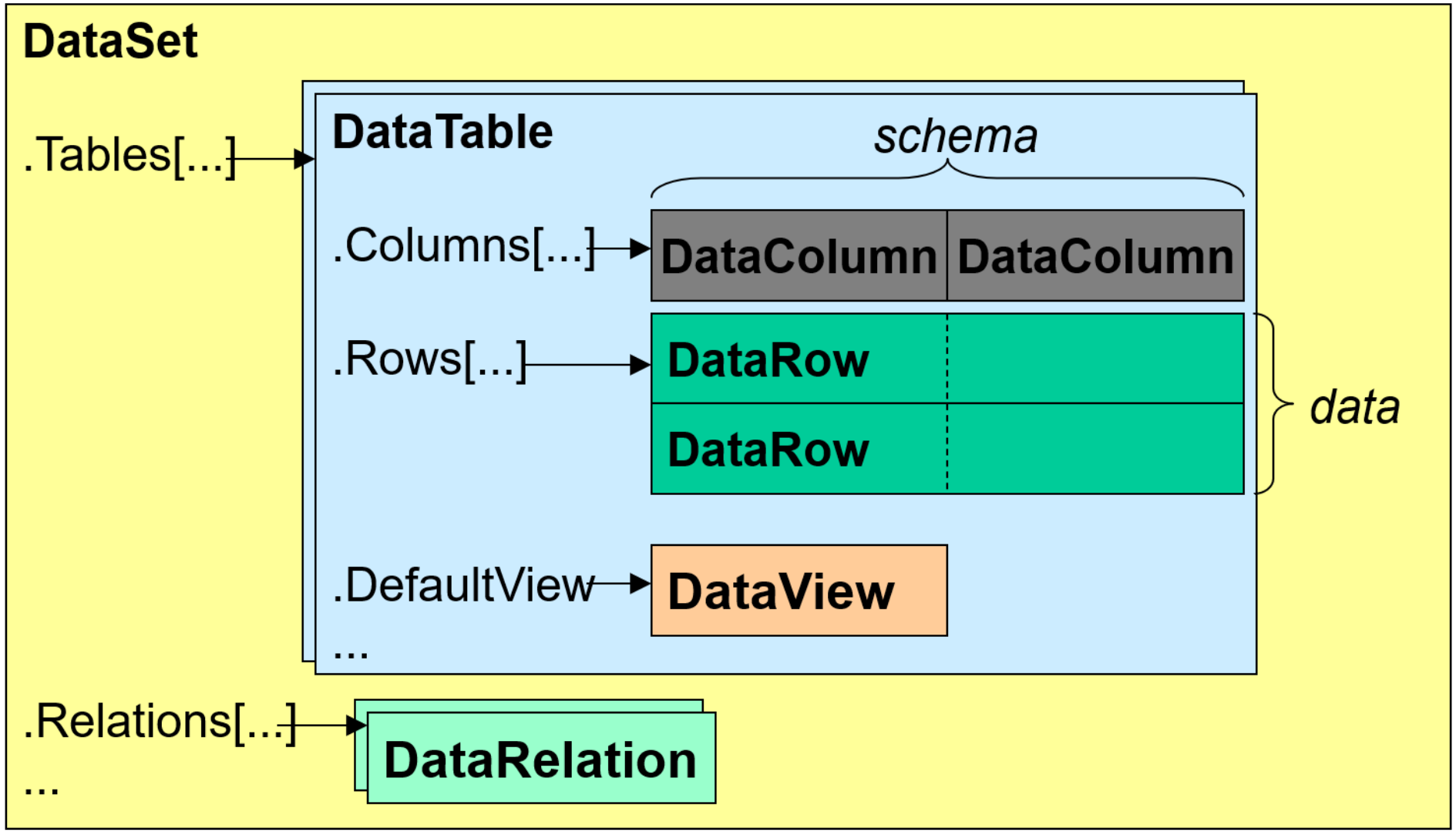
```
using (SqlConnection connection = new SqlConnection(connectionString))
{
  connection.Open(); // Start a local transaction.
  SqlTransaction sqlTran = connection.BeginTransaction();
  SqlCommand command = connection.CreateCommand();
  command.Transaction = sqlTran;
  try {
    command.CommandText = "INSERT INTO Production.ScrapReason(Name) VALUES('Wrong size')";
    command.ExecuteNonQuery();
    command.CommandText = "INSERT INTO Production.ScrapReason(Name) VALUES('Wrong color‘)“;
    command.ExecuteNonQuery();
    sqlTran.Commit();
    } catch (Exception ex) { // Handle the exception if the transaction fails to commit.
       sqlTran.Rollback();
       }
}
```

Disconnected přístup

- Využívá princip ORM
- Obousměrný přístup k datům v rámci speciální vrstvy (`DataSet`)
- Využívá nižší vrstvu `DataAdapter` pro connected přístup k datovému zdroji
- Není třeba neustále připojení k datovému zdroji
- Data jsou uložena v paměti (cache)
- Může dojít k nekonzistenci mezi daty

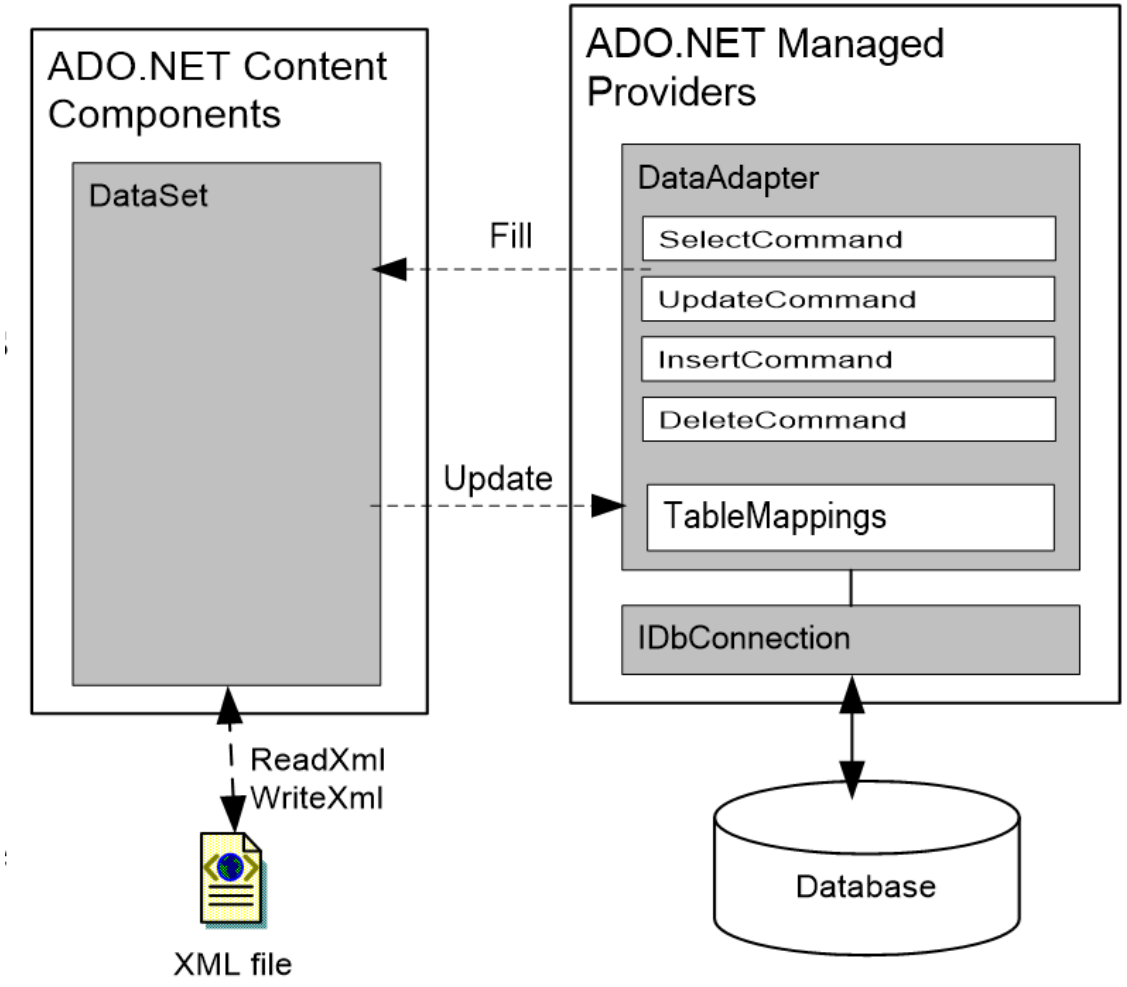


DataSet


DataSet
.Tables[...]
DataTable
schema
.Columns[...]
DataColumn
DataColumn
.Rows[...]
DataRow
DataRow
data
.DefaultView
DataView
...
.Relations[...]
DataRelation
...

DataSet

- Tvorba struktury a dat
  - Přímo v kódu
  - Podle definovaného schématu (XML)
  - Podle reálné podoby datového zdroje
- Veškeré změny se dějí na úrovni DataSetu
  - `AcceptChanges(), RejectChanges()`
  - Každý `DataRow` si udržuje svůj stav `RowState` a `RowVersion`
  - Je možné zachytávat události změny
- Synchronizace s datovým zdrojem pomocí třídy `DataAdapter`



DataSet příklad

```
//Fill Dataset
string ConString = @"Data Source=.\SQLEXPRESS;Initial
Catalog=ComputerShop;Integrated Security=True";
string Query = "SELECT * FROM Items";
SqlDataAdapter adapter = new SqlDataAdapter(Query, ConString);
DataSet set = new DataSet();
adapter.Fill(set, "Items");

//Adding New Row to DataSet
DataRow row = set.Tables["Items"].NewRow();
row["Name"] = "4GB DDR3 RAM";
row["Price"] = "$50";
row["Date"] = "26 May 2017";
set.Tables["Items"].Rows.Add(row);

set.Tables["Items"].Rows[1]["Name"] = "Graphics Card";

dataGridView1.DataSource = set.Tables["Items"];

//Updating Database Table
SqlCommandBuilder builder = new SqlCommandBuilder(adapter);
adapter.Update(set.Tables["Items"]);
```

LINQ (Language INtegrated Query)

- Soubor nástrojů pro obecné dotazování nad daty přímo integrovaný do jazyka C#
- Nabízí vysokou míru abstrakce při určité úrovni efektivity (optimalizace, paralelismus)
- Inspirováno funkcionálním programováním
- Deklarativní jazyk vycházející z principů SQL (zajímá nás výsledek, nikoliv způsob)
- Rozšiřuje sadu metodu třídy `Enumerable` (Linq to Objects)
- Pracovní prvky
  - Sekvence – cokoliv `IEnumerable<T>`
  - Elementy – prvky sekvencí
- Syntaxe
  - Query expression – SQL syntaxe
  - Fluent syntax – metody a lambda výrazy
- Klíčové slovo `var` jako deklarace výstupu – určení datového typu na kompilátoru v době překladu, složité explicitní deklarace, dynamičnost dotazů
- Anonymní typy/třídy – pro předávání a tvorbu dočasných objektů

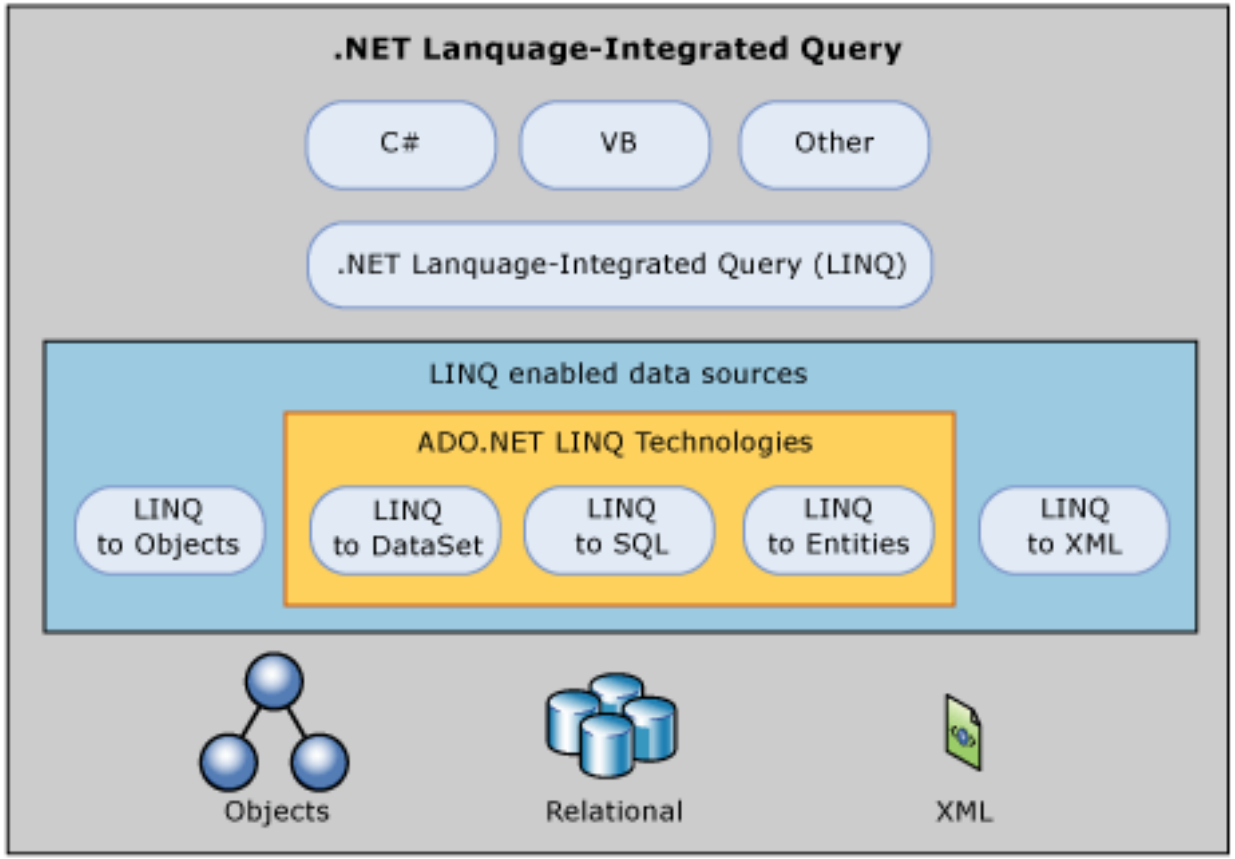
```
IEnumerable<string> filteredNames = from n in names
                                    where n.Contains("a")
                                    select n;
```

```
IEnumerable<string> filteredNames = names.Where(n => n.Contains("a"));
```

`System.Linq`
https://docs.microsoft.com/en-us/dotnet/api/system.linq?view=net-5.0

LINQ providers

- Linq to Objects
  - Nad kolekcemi, práce v paměti
- Linq to SQL/DataSet/Entities
  - Nad datovými zdroji, ORM
- Linq to XML. Linq to JSON
  - Nad strukturovanými daty



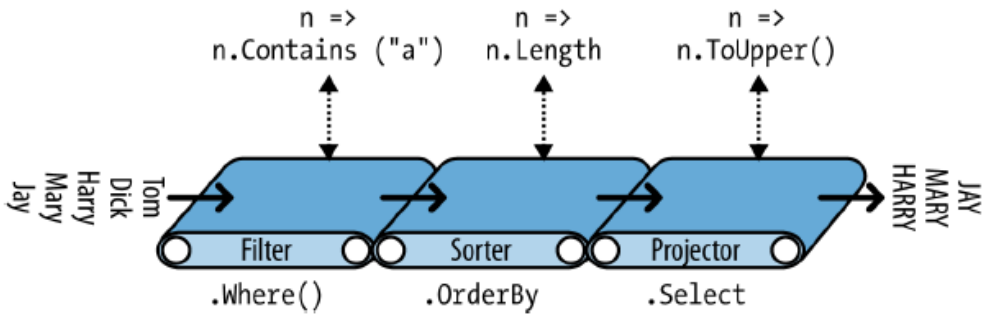
- Možnost vlastních providerů

Konstrukce dotazů

- Vždy se vytváří nová kolekce
- V případě, že chceme přistoupit k jednotlivým elementům používáme sadu metod `First<T>, Last<T>, FirstOrDefault<T>`, atd.
- Lambda výrazy – pracují s jednotlivými elementy

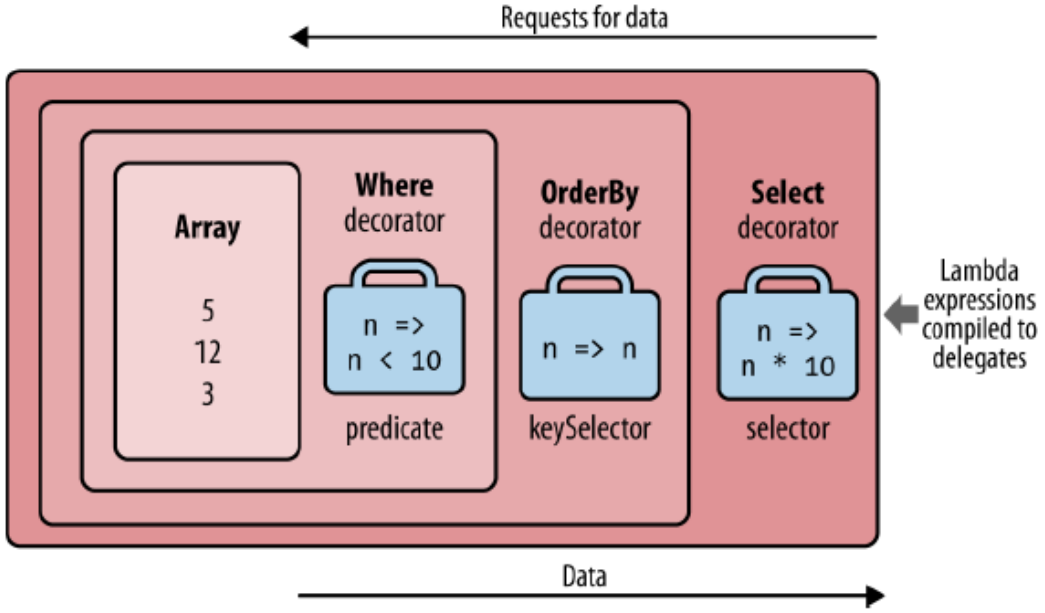
```
IEnumerable<string> filtered = names.Where(n => n.Contains("a"));
IEnumerable<string> sorted = filtered.OrderBy(n => n.Length);
IEnumerable<string> finalQuery = sorted.Select(n => n.ToUpper());

IEnumerable<string> query = names.Where(n => n.Contains("a"))
                                 .OrderBy(n => n.Length)
                                 .Select(n => n.ToUpper());
```



Konstrukce dotazů

- Při provádění se využívá tzv. dekorátorů – „obaluje“ se vstupní kolekce a výstupem je její jiná podoba v každém kroku




```
IEnumerable<int> query = new int[] { 5, 12, 3 }.Where(n => n < 10)
                                                .OrderBy(n => n)
                                                .Select(n => n * 10);
```

Další zdroje

- https://www.itnetwork.cz/csharp/kolekce-a-linq/c-sharp-tutorial-linq-dotazy

- https://www.jetbrains.com/dotnet/guide/tutorials/basics/dapper/
- http://ramirezsystems.blogspot.com/2013/04/build-your-own-orm-part-1.html
- http://www.smuda.cz/dapper-micro-orm-framework/
- https://www.infoworld.com/article/3025784/how-to-use-the-dapper-orm-in-c.html
- https://www.youtube.com/watch?v=Et2khGnrIqc, https://www.youtube.com/watch?v=uS9Sy97Su_E

Zdroje

- https://docs.microsoft.com/cs-cz/dotnet/
- https://devblogs.microsoft.com/
- https://www.codeguru.com/csharp/

- YAMIKANI FUKIZI, Kenneth, Jason DE OLIVEIRA a Michel BRUCHET. *Learn ASP.NET Core 3: Develop modern web applications*. Second edition. Packt Publishing, 2019. ISBN 978-1789610130.
- ALBAHARI, Joseph. *C# 10.0 in a Nutshell: The Definitive Reference*. O'Reilly Media; 1st edition, 2022. ISBN 978-1098121952.
- ALBAHARI, Joseph. *C# 10 and .NET 6 – Modern Cross-Platform Development: Build apps, websites, and services with ASP.NET Core 6, Blazor, and EF Core 6 using Visual Studio 2022 and Visual Studio Code*. 6th edition. Packt Publishing, 2021. ISBN 978-1801077361.