Professional Software Engineering

Andrea Carrara and Patrick Berggold

Hritik Singh and Mohab Hassaan – Tutors

Chair of Computational Modeling and Simulation

Lecture schedule

- LINQ
- Object Relational Mapping
- Entity Framework

andrea.carrara@tum.de; patrick.berggold@tum.de

LINQ

andrea.carrara@tum.de; patrick.berggold@tum.de

Purpose of LINQ

- Integration of queries into C#
- Unification of datasources
- Similar to SQL statements
- Fluent style
- PLINQ since C# 4.0

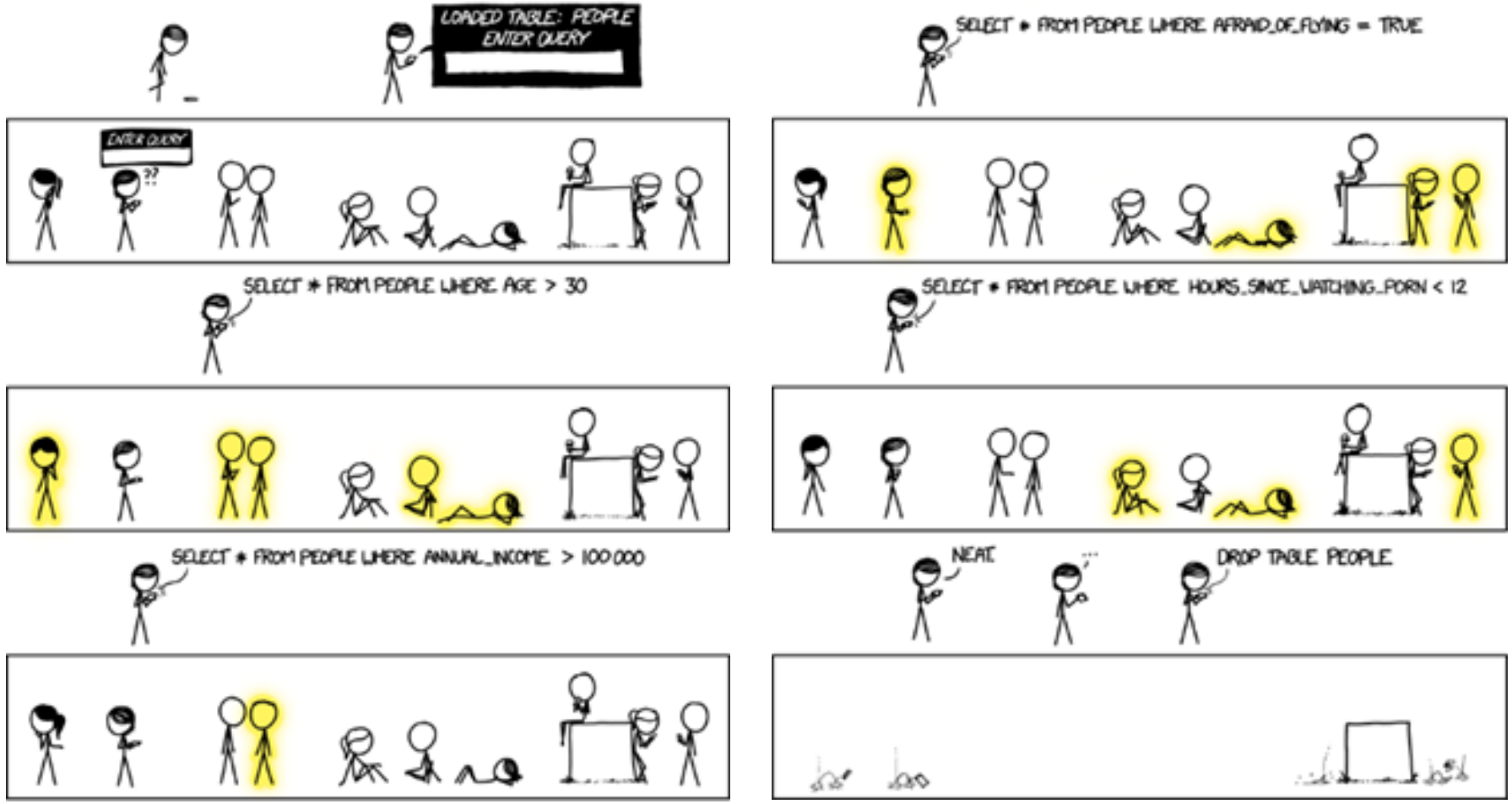
- **In a LINQ query, you are always working with objects.**

andrea.carrara@tum.de; patrick.berggold@tum.de

What is a Query?

- A precise request for information retrieval (wiki)
- Queries can:
  - **Find specific data**
  - **Filter by a specific criteria**
  - **Summarize data**
  - **Automate data management task**
- SQL : ANSI standard since 1986 / ISO 1987

andrea.carrara@tum.de; patrick.berggold@tum.de



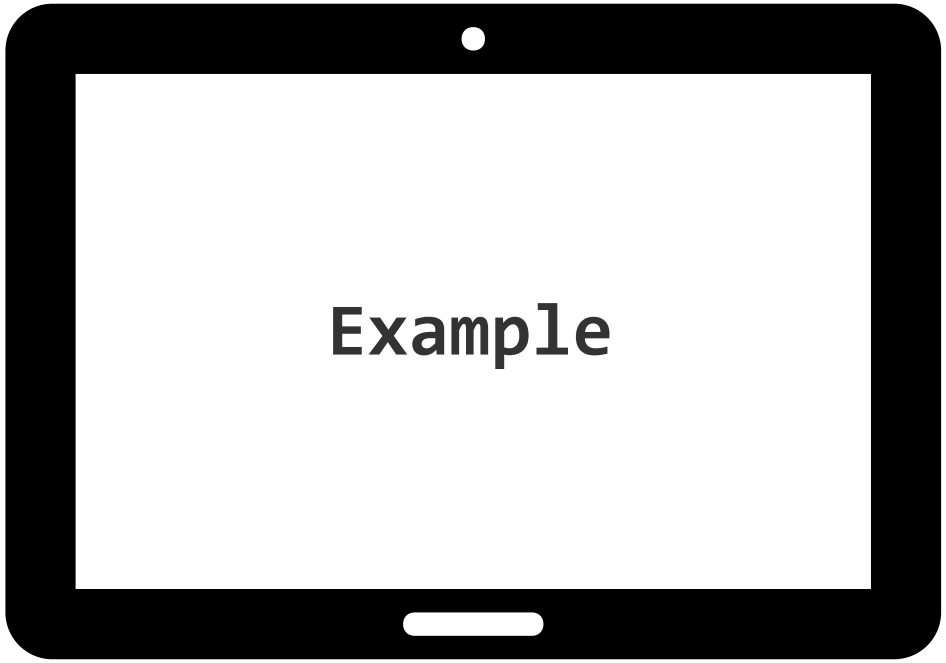
andrea.carrara@tum.de; patrick.berggold@tum.de

The „OLD“ Way

```
public IEnumerable<Student> GetAttendingStudents() {
    List<Student> attending_students;
    foreach(var student in allStudents){
        if(student.attending == true){
            attending_students.Add(student);
        }
    };
    return attending_students;
}
```


andrea.carrara@tum.de; patrick.berggold@tum.de

Linq



andrea.carrara@tum.de; patrick.berggold@tum.de

Extension Methods

- syntactic sugar
- simplified syntax
- static non-generic !

- Since they are static and external they can‘t be used in Interfaces ☹
  - **A new interface can only be applied if you derive the base class!**

andrea.carrara@tum.de; patrick.berggold@tum.de

What LINQ Offers

- Language Integrated Query
- Defined in System.Linq namespace
- Can be used with IEnumerable<T> or IQueryable<T>

- A LINQ consist of:
  1. Obtain the data source.
  2. Create the query.
  3. Execute the query.

andrea.carrara@tum.de; patrick.berggold@tum.de

The Three Parts of a Query Operation

```
class IntroToLINQ {
    static void Main() {
            // The Three Parts of a LINQ Query:
            // 1. Data source.
            int[] numbers = new int[7] { 0, 1, 2, 3, 4, 5, 6 };

            // 2. Query creation.
            // numQuery is an IEnumerable<int>
            var numQuery = from num in numbers
                                   where (num % 2) == 0
                                   select num;

            // 3. Query execution.
            foreach (int num in numQuery) {
                        Console.Write("{0,1} ", num);
            }
    }
}
```

from: docs.microsoft.com

andrea.carrara@tum.de; patrick.berggold@tum.de

LINQ Operator Categories

- Filtering Operators
- Join Operators
- Projection Operations
- Quantifier Operations
- Sorting Operators
- Grouping Operators
- Conversions
- Aggregation
- Generation Operations
- Set Operations
- Equality
- Element Operators
- Partition Operations
- Concatenation

andrea.carrara@tum.de; patrick.berggold@tum.de

Similar To SQL?

```
from num in numbers where (num % 2) == 0 select num;
```

- Why is from at the beginning of the query?
  - **C# does not know where the data comes from**
  - **Static type checking !**

- Extensible operators!
- Extensible data providers e.g.:
  - **LINQ to Excel**
  - **LINQ to Amazon**

andrea.carrara@tum.de; patrick.berggold@tum.de

Example I

```
public IEnumerable<Student> GetAttendingStudents() {
     var students = from s in allStudents
                    where s.Attending (== true)
                     select s;
     return attending;
}
```

- Similar to a SQL query
- In this case you safe a simple “for-loop”.

andrea.carrara@tum.de; patrick.berggold@tum.de

Example II

```
var Streets = from s in Muenchen.Streets
                  where s.Utilization > 0.9
                  orderby s.age descending
                   select new {s.Name}.Take(10);
```

- Paging sort
- sorted column different from the retrieved column
- Only retrieves the first 10 Items

andrea.carrara@tum.de; patrick.berggold@tum.de

Deferred Execution

- A query does not execute until we capture the result
- The query is defined beforehand
- Lazy Loading is a deferred execution
  - **Lazy Loading:**

    "Don't do anything until you have to"

```
var streets = from s in Muenchen.Streets
              where s.Utilization > 0.9
              orderby s.age descending
              select new {s.Name}.Take(8);

foreach(var sn in streets) {
    // do something
}
```

Query

Deferred Execution!

andrea.carrara@tum.de; patrick.berggold@tum.de

Greedy & Lazy Operators

- Operators that execute the query immediately (Count, ToArray …)
- Most operators are lazy
- Change query after definition in a different abstraction layer!

```
var streets = from s in Muenchen.Streets
              where s.Utilization > 0.9
              orderby s.age descending
              select new {s.Name}.Take(8);

streets = s.Where(s => s.ToLower().Contains(“straße”)) // lambda function!

foreach(var sn in streets) {
         // do something with all streets selected
        }
```

only one query!

andrea.carrara@tum.de; patrick.berggold@tum.de

LINQ TO XML

andrea.carrara@tum.de; patrick.berggold@tum.de

LINQ to XML – Creations

```
XElement TumLocations = new XElement(“locations",
            new XElement("city", "Munich"),
            new XElement("city", "Garching"),
            new XElement("city", "Weihenstephan"),
            new XElement("city", "Straubing ")
          );
```

```
<locations>
        <city>Munich</city>
        <city>Garching</city>
        <city>Weihenstephan</city>
        <city>Straubing</city>
</locations>
```

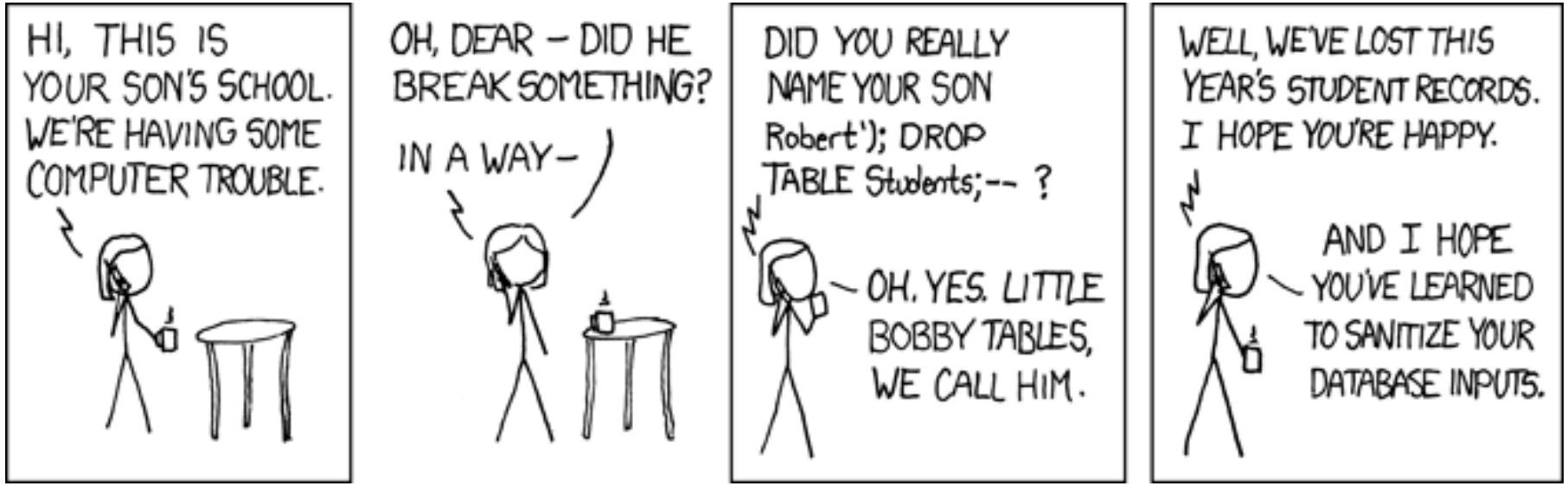

andrea.carrara@tum.de; patrick.berggold@tum.de

The Entity Framework

OBJECT RELATIONAL MAPPING

andrea.carrara@tum.de; patrick.berggold@tum.de

Remember SQL?



LINQ To SQL

- Out of the box it will only work with SQL Server

- Take a provider (NuGet):
  - **LINQ to MySql**
  - **LINQ to PostGres**
  - **LINQ to SQLite (official)**
- Converts a LINQ expression to a Database specific SQL statement!

andrea.carrara@tum.de; patrick.berggold@tum.de

Direct Data Access

- Duplicated code
- A higher potential for programming errors
- Weak typing of the business data
- Difficulty in centralizing data-related policies such as caching
- An inability to easily test the business logic in isolation from external dependencies

msdn.microsoft.com

andrea.carrara@tum.de; patrick.berggold@tum.de

What is an ORM?

- ORM – object relational mapper
- An ORM manipulates a data source using OO paradigms
- Famous librarys:
  - **Doctrine (php)**
  - **ODB (c++)**
  - **SQL Alchemy (python)**
  - **Hibernate (Java, .Net)**
- Why do I want an ORM?
  - **Abstraction layer between your code and the data source**
  - **A lot is handled automatically**
  - **Flexibility (no specific data source)**
  - **Sanitize the data**
  - **TESTED!!!**

andrea.carrara@tum.de; patrick.berggold@tum.de

Entity Framework

- Entity Framework (EF) is an ORM (Object Relational Mapper)
- Eliminates the need for data-access code
- Since version 6 it is a separate library (before it was part of .Net)
- In a way similar to LINQ to SQL

- EF provides Object Services
  - **LINQ to Entities**
  - **Change tracking**
  - **Serialization & databinding**
  - **transaction features**

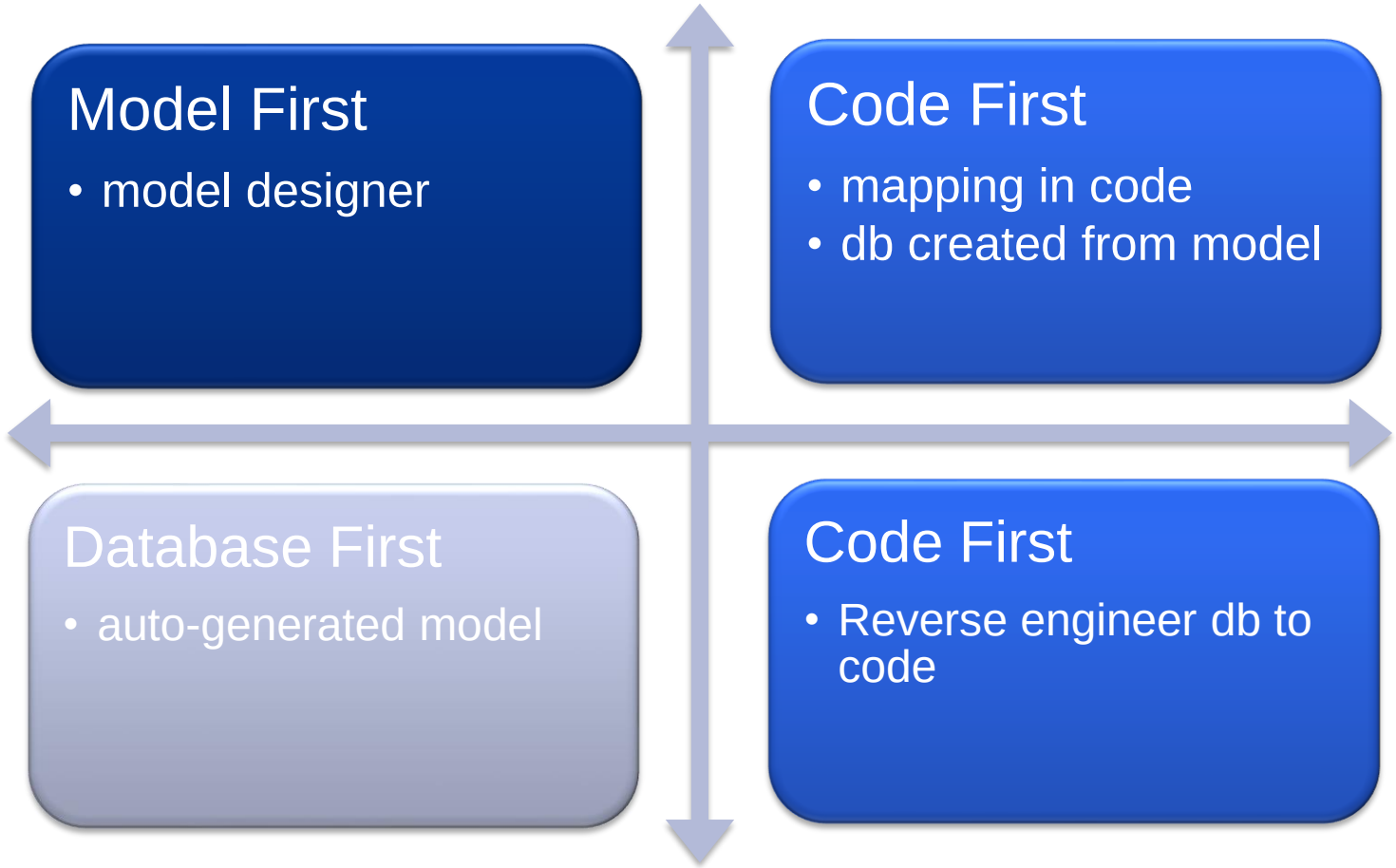
andrea.carrara@tum.de; patrick.berggold@tum.de

Concept

- Program against an entity model
- Possibility to describe associations
- Data can be marked persistant or just changed locally

andrea.carrara@tum.de; patrick.berggold@tum.de

Workflow



andrea.carrara@tum.de; patrick.berggold@tum.de

Code First

- Define our entity models as classes
  - **poc plain old classes**
- Relationships are defined with C# datatypes
- Every object field will be converted to a data base field of that type

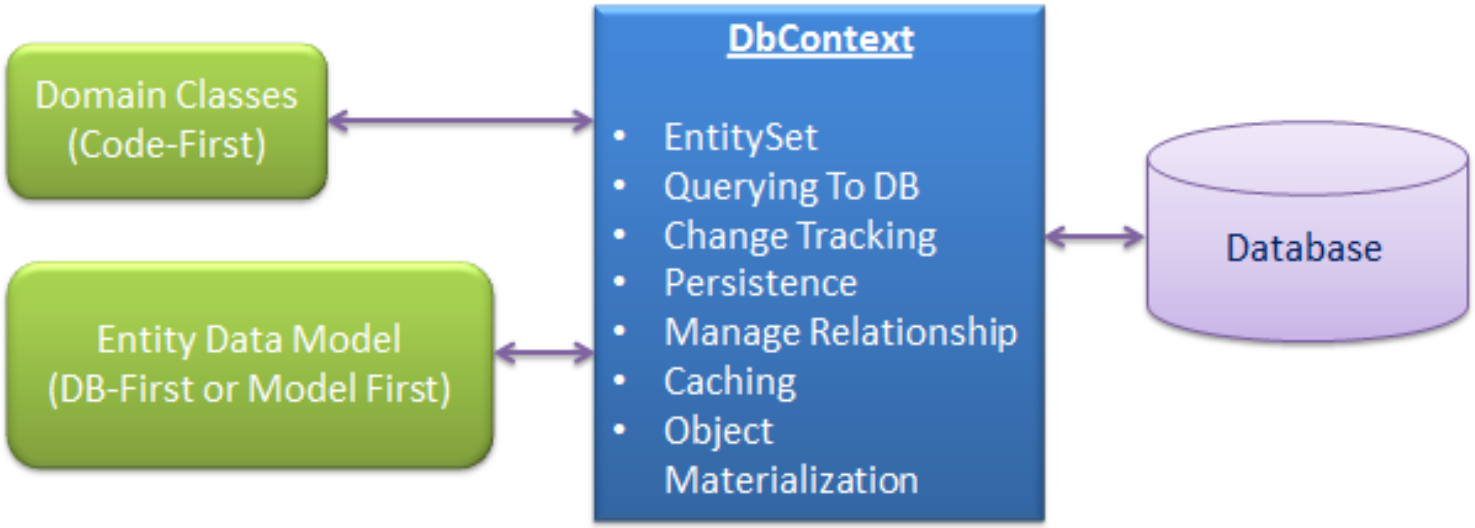
```
// Example of a plain class

public class Street{
    public int Id { get; set; }
    public string Name { get; set; }
    public float Utilization {get; set;}
    public float MaxHeight { get; set; }
    public float Condition { get; set; }
}
```


andrea.carrara@tum.de; patrick.berggold@tum.de

DbContext

- Bridge between domain and entity classes
- Responsible for the data interaction
- Holds the entity sets



From: http://www.entityframeworktutorial.net

DbSet

- If we define a context we need to tell it about the DbSets
- Entities that are not defined by DbSets can only be accessed via a relationship
- Context is interpreted at runtime

```
public class TransportContext : DbContext{
    public DbSet<Street> Streets { get; set; }
    public DbSet<District> Districts { get; set; }
    public DbSet<RepairSquad> RepairSquads { get; set; }
}
```


andrea.carrara@tum.de; patrick.berggold@tum.de