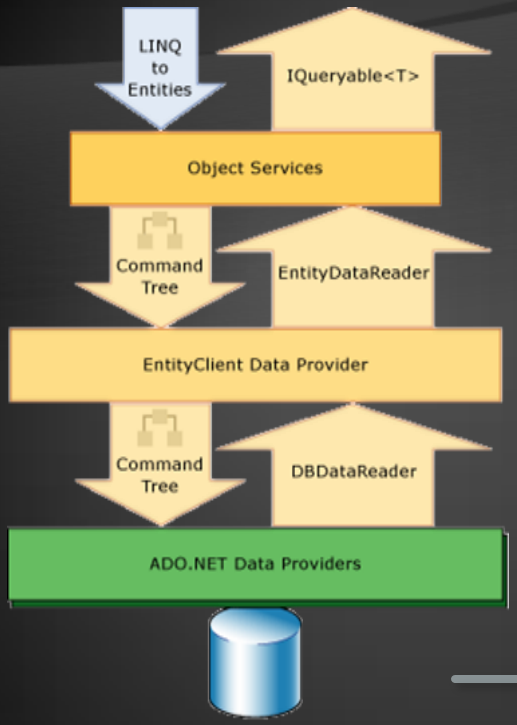
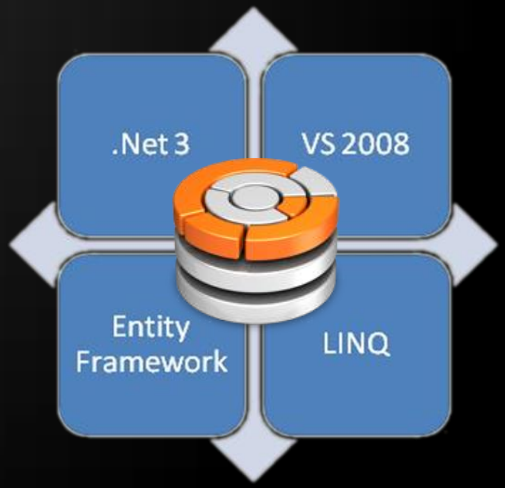
Telerik Academy



Entity Framework

ORM Concepts, Entity Framework (EF), DbContext



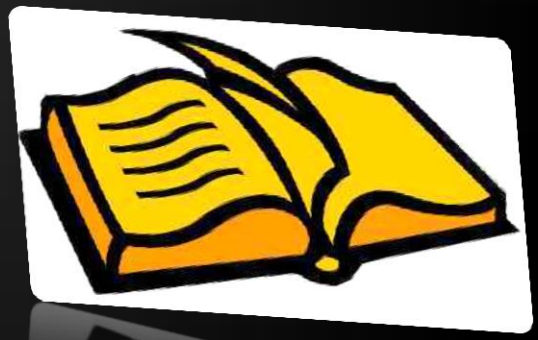
Databases

Telerik Software Academy

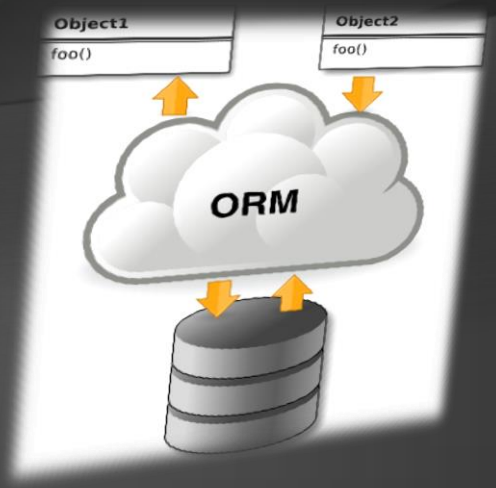
http://academy.telerik.com

Telerik Academy

Table of Contents



Telerik Academy



Introduction to ORM

Object-Relational Mapping (ORM) Technologies

ORM Technologies

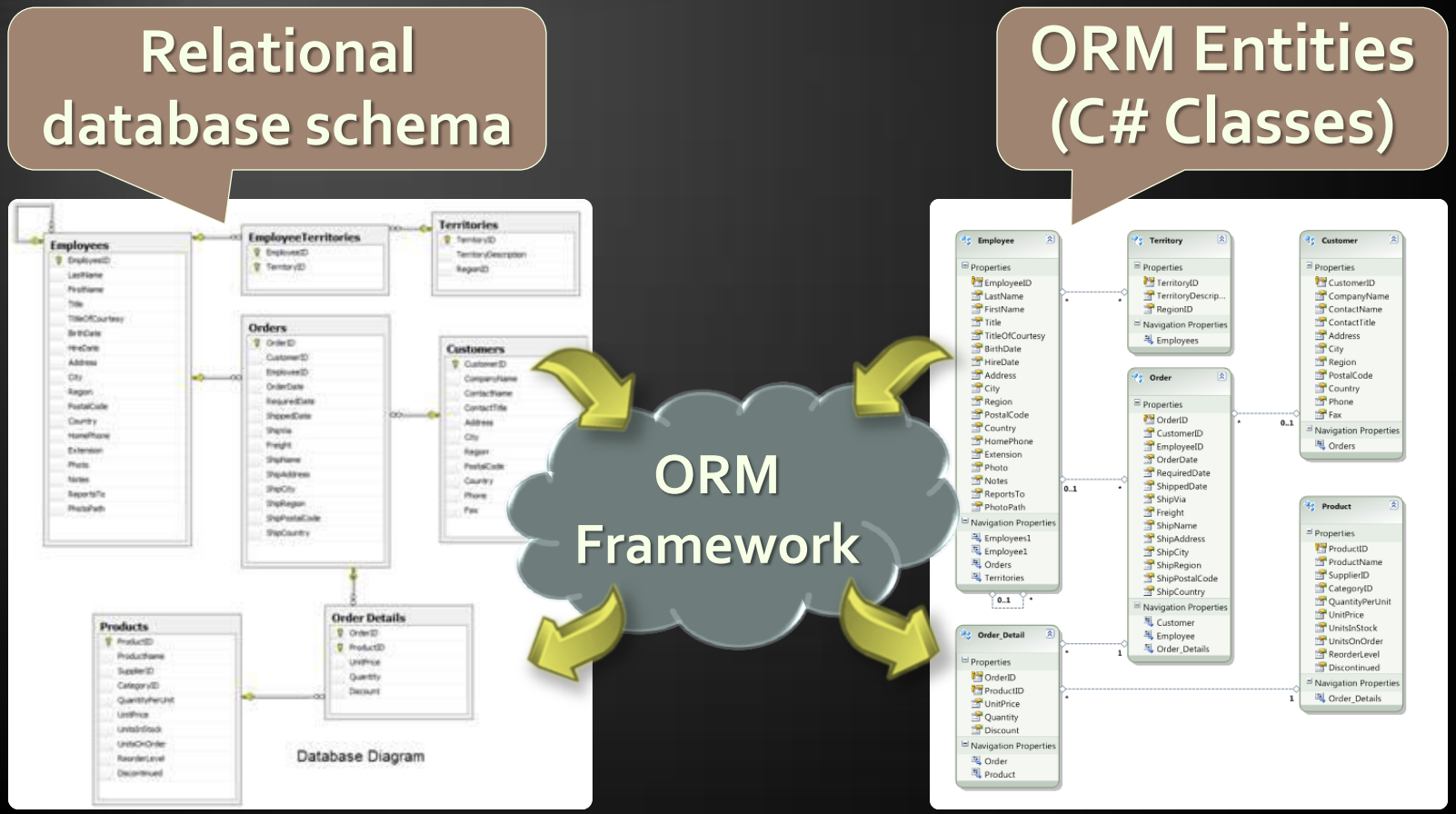
- Object-Relational Mapping (ORM) is a programming technique for automatic mapping and converting data
  - Between relational database tables and object-oriented classes and objects
- ORM creates a "virtual object database"
  - Which can be used from within the programming language, e.g. C# or Java
- ORM frameworks automate the ORM process
  - A.k.a. object-relational persistence frameworks

ORM Frameworks

- ORM frameworks typically provide the following functionality:
  - Creating object model by database schema
  - Creating database schema by object model
  - Querying data by object-oriented API
  - Data manipulation operations
    - CRUD – create, retrieve, update, delete
- ORM frameworks automatically generate SQL to perform the requested data operations

ORM Mapping – Example

- Database and Entities mapping diagrams for a subset of the `Northwind` database



ORM Advantages

- Object-relational mapping advantages
  - Developer productivity
    - Writing less code
  - Abstract from differences between object and relational world
    - Complexity hidden within ORM
  - Manageability of the CRUD operations for complex relationships
  - Easier maintainability

Telerik Academy

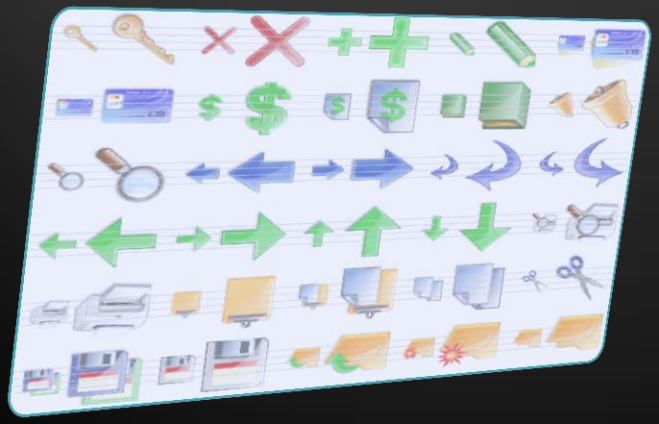
ORM Frameworks in .NET

- Built-in ORM tools in .NET Framework and VS
  - Entity Framework (LINQ-to-Entities)
  - LINQ-to-SQL
  - Both combine entity class mappings and code generation, SQL is generated at runtime
- Third party ORM tools
  - NHibernate – the old daddy of ORM
  - Telerik OpenAccess ORM

Telerik Academy

Entity Framework

Object Relation Persistence Framework



Overview of EF

- Entity Framework (EF) is a standard ORM framework, part of .NET
  - Provides a run-time infrastructure for managing SQL-based database data as .NET objects
- The relational database schema is mapped to an object model (classes and associations)
  - Visual Studio has built-in tools for generating Entity Framework SQL data mappings
    - Data mappings consist of C# classes and XML
  - A standard data manipulation API is provided

Entity Framework Features

- Maps tables, views, stored procedures and functions as .NET objects
- Provides LINQ-based data queries
  - Executed as SQL SELECTs on the database server (parameterized queries)
- Built-in CRUD operations – Create/Read/Update/Delete
- Creating or deleting the database schema
- Tracks changes to in-memory objects

Telerik Academy

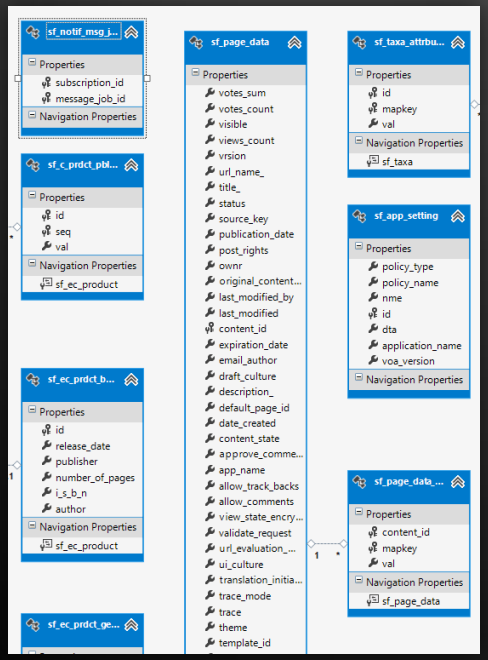
Entity Framework Features (2)

- Works with any relational database with valid Entity Framework provider
- Work with a visual model, database or with your own classes
- Has very good default behavior
  - Very flexible for more granular control
- Open source – independent release cycle
  - entityframework.codeplex.com
  - github.com/aspnet/EntityFramework

Telerik Academy

Basic Workflow

1. Define model
   - Database
   - Visual designer
   - Code
2. Express & execute query over IQueryable
3. EF determines & executes SQL query




```
var toolName = "";

var snippetOptions = DefaultToolGroup
    .Tools
    .OfType<EditorListTool>()
    .Where(t =>
        t.Name == toolName &&
        t.Items != null &&
        t.Items.Any())
    .SelectMany(
        (t, index) =>
            t.Items
            .Select(item =>
                new {
                    text = item.Text,
                    value = item.Value
                }));

if (snippetOptions.Any())
{
    options[toolName] = snippetOptions;
}
```

```
exec sp_executesql N'SELECT
[Filter2].[UserInCourseId] AS [UserInCourse
[Filter2].[UserId] AS [UserId],
[Filter2].[CourseInstanceId1] AS [CourseIns
[Filter2].[FirstCourseGroupId] AS [FirstCou
[Filter2].[SecondCourseGroupId] AS [SecondC
[Filter2].[ThirdCourseGroupId] AS [ThirdCou
[Filter2].[FourthCourseGroupId] AS [FourthC
[Filter2].[FifthCourseGroupId] AS [FifthCou
[Filter2].[IsLiveParticipant] AS [IsLivePar
[Filter2].[Accommodation] AS [Accommodatior
[Filter2].[ExcellentResults] AS [ExcellentR
[Filter2].[Result] AS [Result],
[Filter2].[CanDoTestExam] AS [CanDoTestExam
[Filter2].[CourseTestExamId] AS [CourseTest
[Filter2].[TestExamPoints] AS [TestExamPoir
[Filter2].[CanDoPracticalExam] AS [CanDoPra
[Filter2].[CoursePracticalExamId1] AS [Cour
[Filter2].[PracticalExamPoints] AS [Practic
[Filter2].[AttendancesCount] AS [Attendance
[Filter2].[HomeworkEvaluationPoints] AS [HC
FROM   (SELECT [Extent1].[UserInCourseId] A
AS [SecondCourseGroupId], [Extent1].[ThirdC
[IsLiveParticipant], [Extent1].[Accommodati
[CourseTestExamId], [Extent1].[TestExamPoir
[PracticalExamPoints], [Extent1].[Attendanc
        FROM  [courses].[UsersInCourses] AS
        INNER JOIN [courses].[CoursePractic
        WHERE ( EXISTS (SELECT
                1 AS [C1]
                FROM [courses].[CoursePract
                WHERE [Extent1].[UserInCour
        )) AND ([Extent2].[AllowExamFilesEv
INNER JOIN [courses].[CoursePracticalExams]
WHERE ([Filter2].[UserId] = @p__linq__0) AN
```

Basic Workflow (2)

4. EF transforms selected results into .NET objects
5. Modify data and call "save changes"
6. EF determines & executes SQL query

app_name	module_name
AcademyTelerikCom/	Telerik.Sitefinity.Modules.Libraries.BlobStorage.Open...
AcademyTelerikCom/	Telerik.Sitefinity.Modules.Newsletters.Data.OpenAcc...
AcademyTelerikCom/	Telerik.Sitefinity.Publishing.Data.OpenAccessPublishi...
AcademyTelerikCom/	Telerik.Sitefinity.Modules.Ecommerce.Shipping.Data....
AcademyTelerikCom/	Telerik.Sitefinity.Modules.Events.Data.OpenAccessE...
AcademyTelerikCom/	Telerik.Sitefinity.Security.Data.OpenAccessSecurityP...
AcademyTelerikCom/	Telerik.Sitefinity.Modules.GenericContent.Data.Open...
AcademyTelerikCom/	Telerik.Sitefinity.Modules.Pages.Data.OpenAccessP...
AcademyTelerikCom/	Telerik.Sitefinity.Taxonomies.Data.OpenAccessTaxo...
AcademyTelerikCom/	Telerik.Sitefinity.Scheduling.Data.OpenAccessSched...
AcademyTelerikCom/	Telerik.Sitefinity.DynamicModules.Builder.Data.Open...
AcademyTelerikCom/	Telerik.Sitefinity.Security.Data.OpenAccessProfilePro...
AcademyTelerikCom/	Telerik.Sitefinity.Services.Notifications.Data.OpenAc...
AcademyTelerikCom/	SitefinityDynamicBase
AcademyTelerikCom/	Telerik.Sitefinity.Modules.Forms.Data.OpenAccessFo...
AcademyTelerikCom/	Telerik.Sitefinity.Modules.Lists.Data.OpenAccessList...
AcademyTelerikCom/	Telerik.Sitefinity.Modules.News.Data.OpenAccessNe...
AcademyTelerikCom/	Telerik.Sitefinity.Publishing.Data.OpenAccessPublishi...
AcademyTelerikCom/	Telerik.Sitefinity.DynamicModules.Data.OpenAccess...
AcademyTelerikCom/	Telerik.Sitefinity.Modules.Blogs.Data.OpenAccessBlo...
AcademyTelerikCom/	Sitefinity
AcademyTelerikCom/	Telerik.Sitefinity.Security.Data.OpenAccessMembers...
AcademyTelerikCom/	Telerik.Sitefinity.Modules.ResponsiveDesign.Data.O...

```
private void ChangeBlogPostName(int id,
    string newName)
{
    var db = new Context();

    var post = db.Posts
        .FirstOrDefault(x => x.Id == id);

    if (post == null)
    {
        throw new ArgumentException(
            "Item with that id was not fo
            "id");
    }

    post.Name = newName;

    db.SaveChanges();
}
```

```
exec sp_executesql
    N'update [dbo].[Posts]
    set [Name] = @0
    where ([Id] = @1)',
        N'@0 nvarchar(max),
          @1 int',
            @0=N'',
            @1=1
```

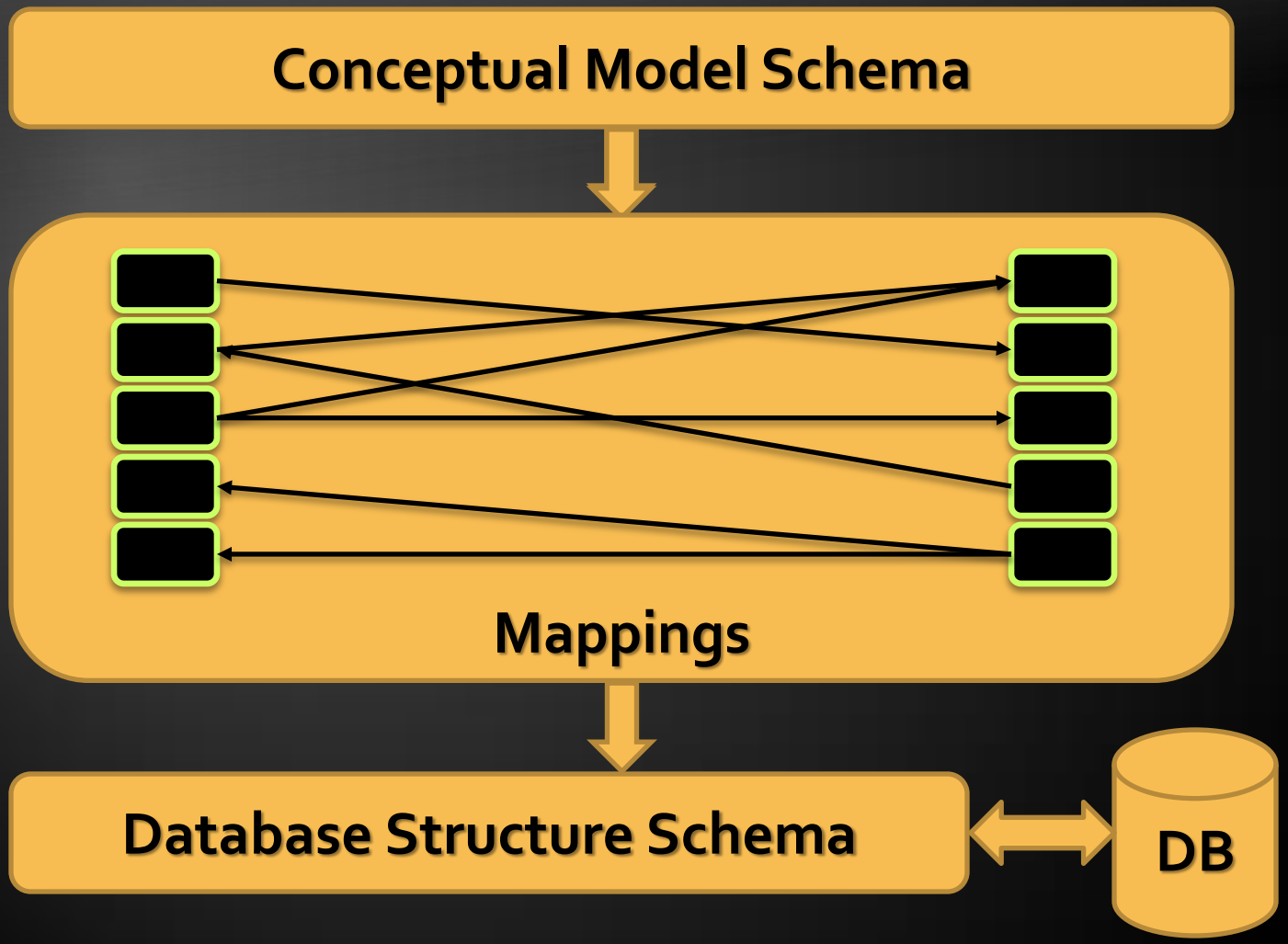
EF Components

- The `DbContext` class
  - `DbContext` holds the database connection and the entity classes
  - Provides LINQ-based data access
  - Implements identity tracking, change tracking, and API for CRUD operations
- Entity classes
  - Each database table is typically mapped to a single entity class (C# class)

EF Components (2)

- Associations (Relationship Management)
  - An association is a primary key / foreign key based relationship between two entity classes
  - Allows navigation from one entity to another, e.g. `Student.Courses`
- Concurrency control
  - Entity Framework uses optimistic concurrency control (no locking by default)
  - Provides automatic concurrency conflict detection and means for conflicts resolution

EF Runtime Metadata



Telerik Academy

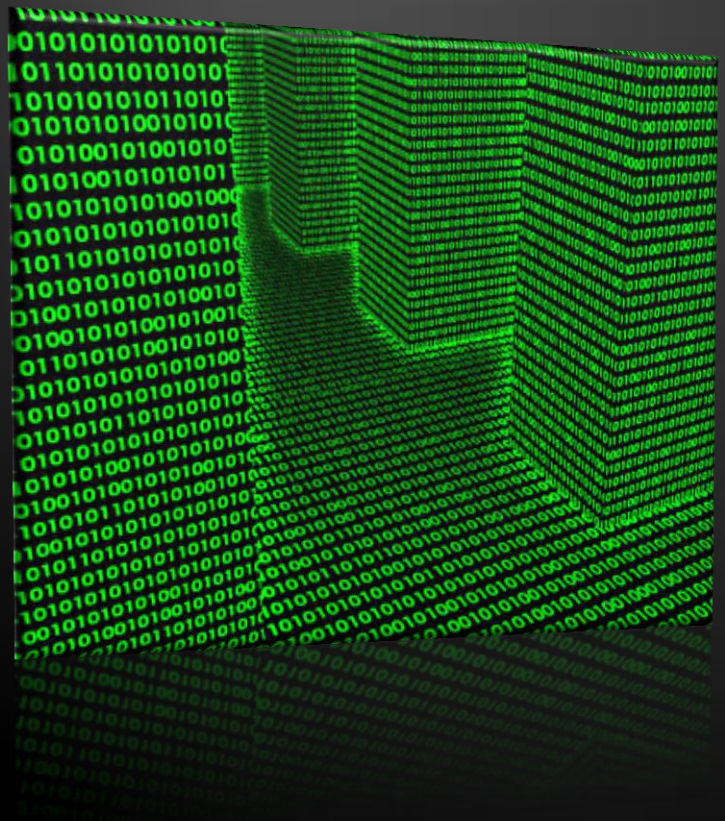
Microsoft® Visual Studio®

The Entity Framework Designer in Visual Studio

Live Demo

Telerik Academy

Reading Data with Entity Framework



The DbContext Class

- The `DbContext` class is generated by the Visual Studio designer
- `DbContext` provides:
  - Methods for accessing entities (object sets) and creating new entities (`Add()` methods)
  - Ability to manipulate database data though entity classes (read, modify, delete, insert)
  - Easily navigate through the table relationships
  - Executing LINQ queries as native SQL queries
  - Create the DB schema in the database server

Using DbContext Class

- First create instance of the DbContext:

```
NorthwindEntities northwind = new NorthwindEntities();
```

- In the constructor you can pass a database connection string and mapping source
- DbContext properties
  - Connection – the SqlConnection to be used
  - CommandTimeout – timeout for database SQL commands execution
  - All entity classes (tables) are listed as properties
    - e.g. IDbSet<Order> Orders { get; }

Telerik Academy

Reading Data with LINQ Query

- Executing LINQ-to-Entities query over EF entity:

```
using (var context = new NorthwindEntities())
{
  var customers =
    from c in context.Customers
    where c.City == "London"
    select c;
}
```

The query will be executes as SQL command in the database

- Customers property in the DbContext:

```
public partial class NorthwindEntities : DbContext
{
  public ObjectSet<Customer> Customers
  {
    get { … }
  }
}
```

Reading Data with LINQ Query

- We can also use extension methods (fluent API) for constructing the query

```
using (var context = new NorthwindEntities())
{
  var customerPhoness = context.Customers
                              .Select(c => c.Phone)
                              .Where(c => c.City == "London")
                              .ToList();
}
```

This is called projection

ToList() method executes the query

- Find element by id

```
using (var context = new NorthwindEntities())
{
  var customer = context.Customers.Find(2);
  Console.WriteLine(customer.ContactTitle);
}
```

Telerik Academy

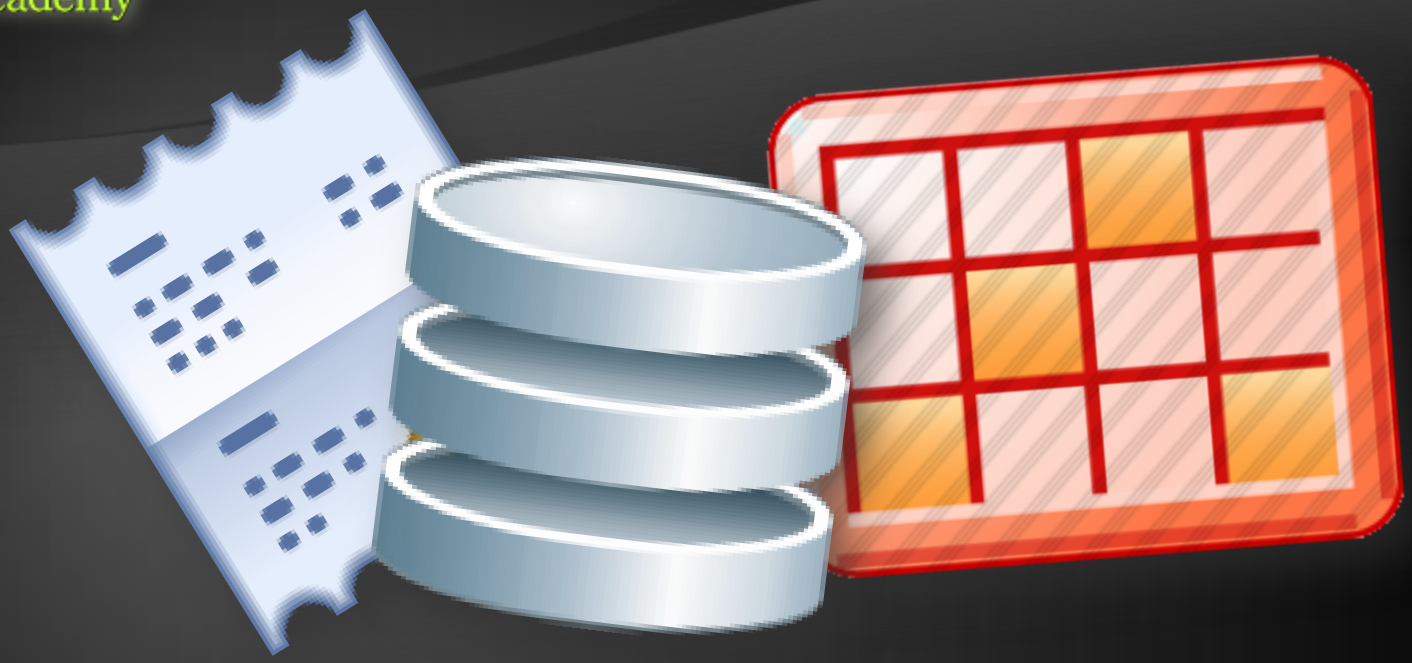
Logging the Native SQL Queries

- To print the native database SQL commands executed on the server use the following:

```
var query = context.Countries;
Console.WriteLine(query.ToString());
```

- This will print the SQL native query executed at the database server to select the Countries
  - Can be printed to file using `StreamWriter` class instead of `Console` class

Telerik Academy

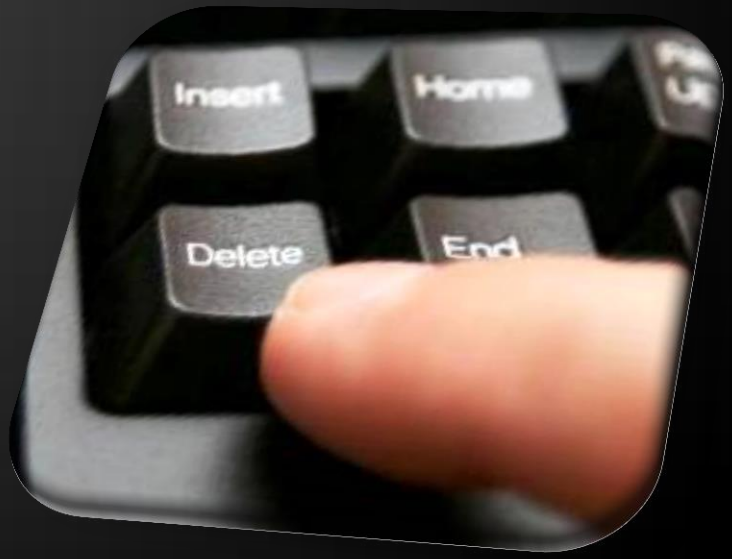


Retrieving Data with LINQ to Entities

Live Demo

Telerik Academy

Create, Update, Delete using Entity Framework



Creating New Data

- To create a new database row use the method Add(…) of the corresponding collection:

```
// Create new order object
Order order = new Order()
{
  OrderDate = DateTime.Now, ShipName = "Titanic",
  ShippedDate = new DateTime(1912, 4, 15),
  ShipCity = "Bottom Of The Ocean"
};
// Mark the object for inserting
context.Orders.Add(order);
context.SaveChanges();
```

This will execute an SQL INSERT

- SaveChanges() method call is required to post the SQL commands to the database

Cascading Inserts

- We can also add cascading entities to the database:

```
Country spain = new Country();
spain.Name = "Spain";
spain.Population = "46 030 10";
spain.Cities.Add(new City { Name = "Barcelona"} );
spain.Cities.Add(new City { Name = "Madrid"} );
countryEntities.Countries.Add(spain);
countryEntities.SaveChanges();
```

- This way we don't have to add each City individually
  - They will be added when the Country entity (Spain) is inserted to the database

Updating Existing Data

- DbContext allows modifying entity properties and persisting them in the database
  - Just load an entity, modify it and call SaveChanges()
- The DbContext automatically tracks all changes made on its entity objects

```
Order order = northwindEntities.Orders.First();
order.OrderDate = DateTime.Now;
context.SaveChanges();
```

This will execute an SQL UPDATE

This will execute an SQL SELECT to load the first order

Deleting Existing Data

- Delete is done by `Remove()` on the specified entity collection
- `SaveChanges()` method performs the delete action in the database

```
Order order = northwindEntities.Orders.First();

// Mark the entity for deleting on the next save
northwindEntities.Orders.Remove(order);
northwindEntities.SaveChanges();
```

This will execute an SQL DELETE command

Telerik Academy

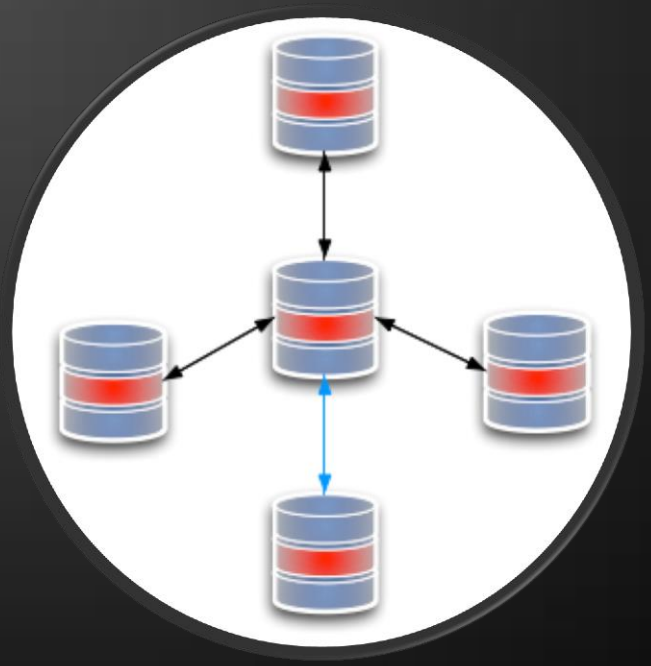
CRUD Operations with Entity Framework

Live Demo

Telerik Academy

Extending Entity Classes

Add methods like ToString(), Equals(), etc…

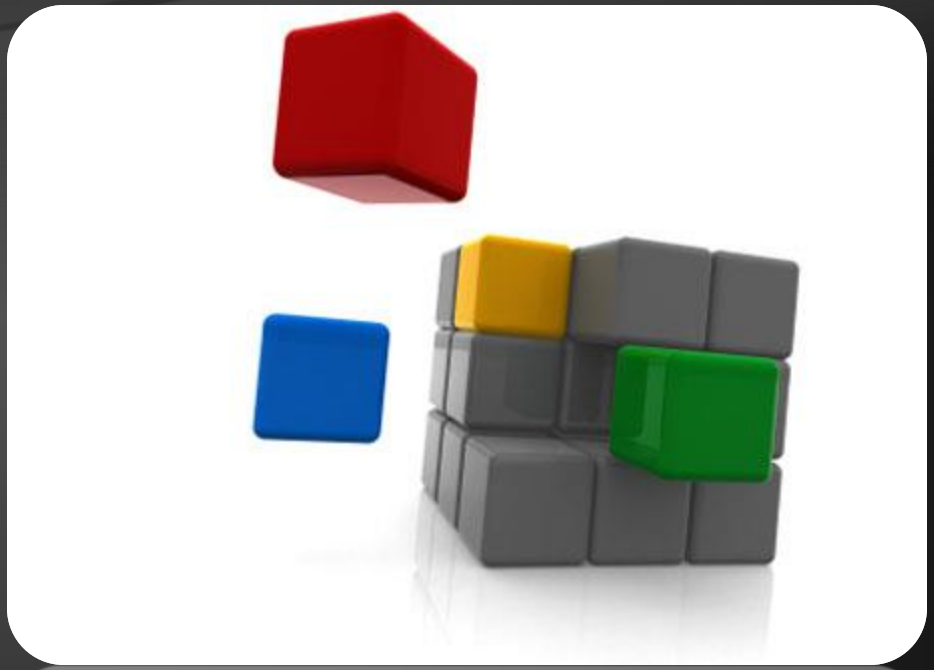


Telerik Academy

Extending Entity Classes

- When using "database first" or "model first" entity classes are separate `.cs` files that are generated by T4 tempalte `XXXModel.tt`
  - And each time we update the EntitiesModel from the database all files are generated anew
  - If we add methods like ToString(), they will be overridden and lost
  - That is why all the entity classes are "`partial`"
    - We can extend them in another file with the same partial class
- When using "code first" this is not a problem

Telerik Academy



Extending Entity Classes

Live Demo

Telerik Academy

Executing Native SQL Queries

Parameterless and Parameterized

Telerik Academy

Executing Native SQL Queries

- Executing a native SQL query in Entity Framework directly in its database store:

```
ctx.Database.SqlQuery<return-type>(native-SQL-query);
```

- Example:

```
string query = "SELECT count(*) FROM dbo.Customers";
var queryResult = ctx.Database.SqlQuery<int>(query);
int customersCount = queryResult.FirstOrDefault();
```

- Examples are shown in SQL Server but the same can be done for any other database

Telerik Academy

Executing Native SQL Queries (2)

- Native SQL queries can also be parameterized:

```
NorthwindEntities context = new NorthwindEntities();
string nativeSQLQuery =
  "SELECT FirstName + ' ' + LastName " +
  "FROM dbo.Employees " +
  "WHERE Country = {0} AND City = {1}";
object[] parameters = { country, city };
var employees = context.Database.SqlQuery<string>(
  string.Format(nativeSQLQuery, parameters));
foreach (var emp in employees)
{
  Console.WriteLine(emp);
}
```

Telerik Academy

Executing Native SQL Queries

Live Demo

Telerik Academy

Joining and Grouping Tables

Join and Group Using LINQ

Joining Tables in EF

- In EF we can join tables in LINQ or by using extension methods on `IEnumerable<T>`
  - The same way like when joining collections

```
northwindEntities.Customers.
 Join(northwindEntities.Suppliers,
  (c=>c.Country), (s=>s.Country), (c,s)=>
  new {Customer = c.CompanyName, Supplier =
  s.CompanyName, Country = c.Country });
```

```
var custSuppl =
 from customer in northwindEntities.Customers
 join supplier in northwindEntities.Suppliers
 on customer.Country equals supplier.Country
 select new {
                CustomerName = customer.CompanyName,
                Supplier = supplier.CompanyName,
                Country = customer.Country
            };
```

Grouping Tables in EF

- Grouping also can be done by LINQ
  - The same ways as with collections in LINQ
- Grouping with LINQ:

```
var groupedCustomers =
  from customer in northwindEntities.Customers
  group customer by Customer.Country;
```

- Grouping with extension methods:

```
var groupedCustomers =
  northwindEntities.Customers.GroupBy(
    customer => customer.Country);
```

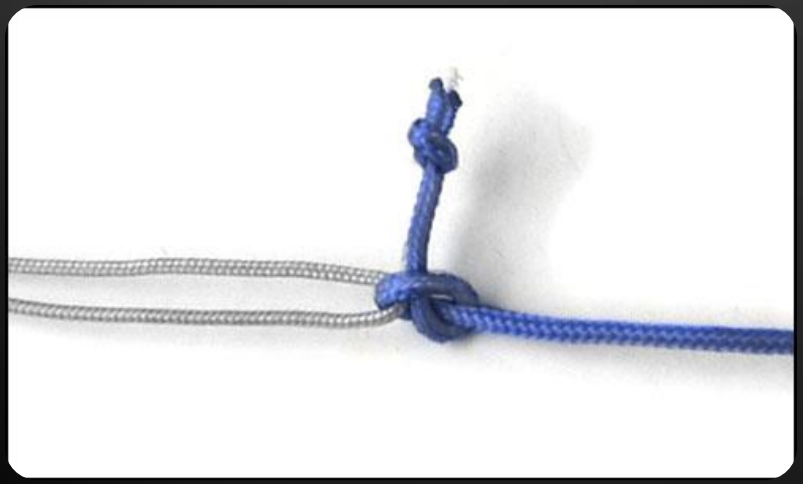
Telerik Academy

Joining and Grouping Tables

Live Demo

Telerik Academy

Attaching and Detaching Objects



Attaching and Detaching Objects

- In Entity Framework, objects can be attached to or detached from an object context
- Attached objects are tracked and managed by the DbContext
  - SaveChanges() persists all changes in DB
- Detached objects are not referenced by the DbContext
  - Behave like a normal objects, like all others, which are not related to EF

Attaching Detached Objects

- When a query is executed inside an `DbContext`, the returned objects are automatically attached to it
- When a context is destroyed, all objects in it are automatically detached
  - E.g. in Web applications between the requests
- You might later on attach to a new context objects that have been previously detached

Detaching Objects

- When an object is detached?
  - When we obtain the object from an `DbContext` and then `Dispose` it
  - Manually: by set the entry state to Detached

```
Product GetProduct(int id)
{
    using (NorthwindEntities northwindEntities =
        new NorthwindEntities())
    {
        return northwindEntities.Products.First(
            p => p.ProductID == id);
    }
}
```

Now the returned product is detached

Attaching Objects

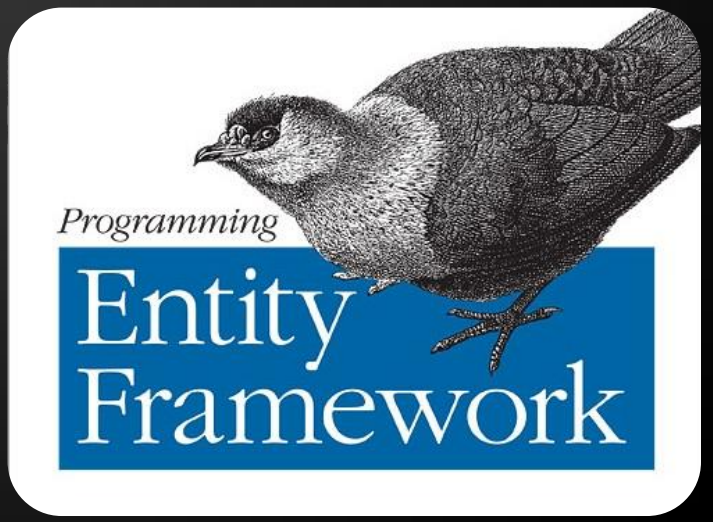
- When we want to update a detached object we need to reattach it and then update it
  - Done by the `Attached` state of the entry

```
void UpdatePrice(Product product, decimal newPrice)
{
    using (NorthwindEntities northwindEntities =
        new NorthwindEntities())
    {
        var entry = northwindEntities.Entry(product);
        entry.State = EntityState.Attached;
        product.UnitPrice = newPrice;
        northwindEntities.SaveChanges();
    }
}
```

Telerik Academy

Attaching and Detaching Objects

Live Demo



Telerik Academy

Entity Framework

http://academy.telerik.com

Homework

1. Using the Visual Studio Entity Framework designer create a `DbContext` for the `Northwind` database
2. Create a DAO class with static methods which provide functionality for inserting, modifying and deleting customers. Write a testing class.
3. Write a method that finds all customers who have orders made in 1997 and shipped to Canada.
4. Implement previous by using native SQL query and executing it through the `DbContext`.
5. Write a method that finds all the sales by specified region and period (start / end dates).

Homework (2)

6. Create a database called `NorthwindTwin` with the same structure as `Northwind` using the features from `DbContext`. Find for the API for schema generation in MSDN or in Google.
7. Try to open two different data contexts and perform concurrent changes on the same records. What will happen at `SaveChanges()`? How to deal with it?
8. By inheriting the `Employee` entity class create a class which allows employees to access their corresponding territories as property of type `EntitySet<T>`.

Homework (3)

9. Create a method that places a new order in the `Northwind` database. The order should contain several order items. Use transaction to ensure the data consistency.
10. Create a stored procedures in the `Northwind` database for finding the total incomes for given supplier name and period (start date, end date). Implement a C# method that calls the stored procedure and returns the retuned record set.

Homework (4)

11. Create a database holding users and groups. Create a transactional EF based method that creates an user and puts it in a group "Admins". In case the group "Admins" do not exist, create the group in the same transaction. If some of the operations fail (e.g. the username already exist), cancel the entire transaction.
12. * Use SQL Server Profiler to view all your queries from previous homework tasks
13. * Download and explore the full source code of Entity Framework: http://entityframework.codeplex.com/

Free Trainings @ Telerik Academy

- C# Programming @ Telerik Academy
  - csharpfundamentals.telerik.com
- Telerik Software Academy
  - academy.telerik.com
- Telerik Academy @ Facebook
  - facebook.com/TelerikAcademy
- Telerik Software Academy Forums
  - forums.academy.telerik.com