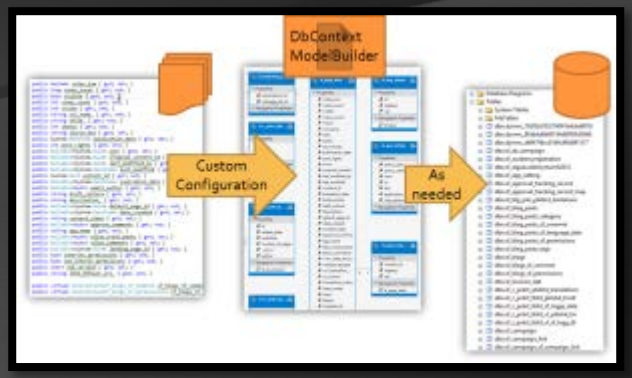
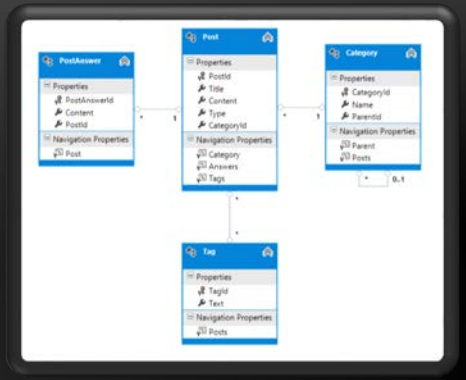
Telerik Academy





Entity Framework Code First

Databases

Telerik Software Academy

http://academy.telerik.com

Telerik Academy

Table of Contents

- Modeling Workflow
- Code First Main Parts
  - Domain Classes (Models)
  - DbContext and DbSets
  - Database connection
- Using Code First Migrations
- Configure Mappings
- LINQPad
- Repository Pattern

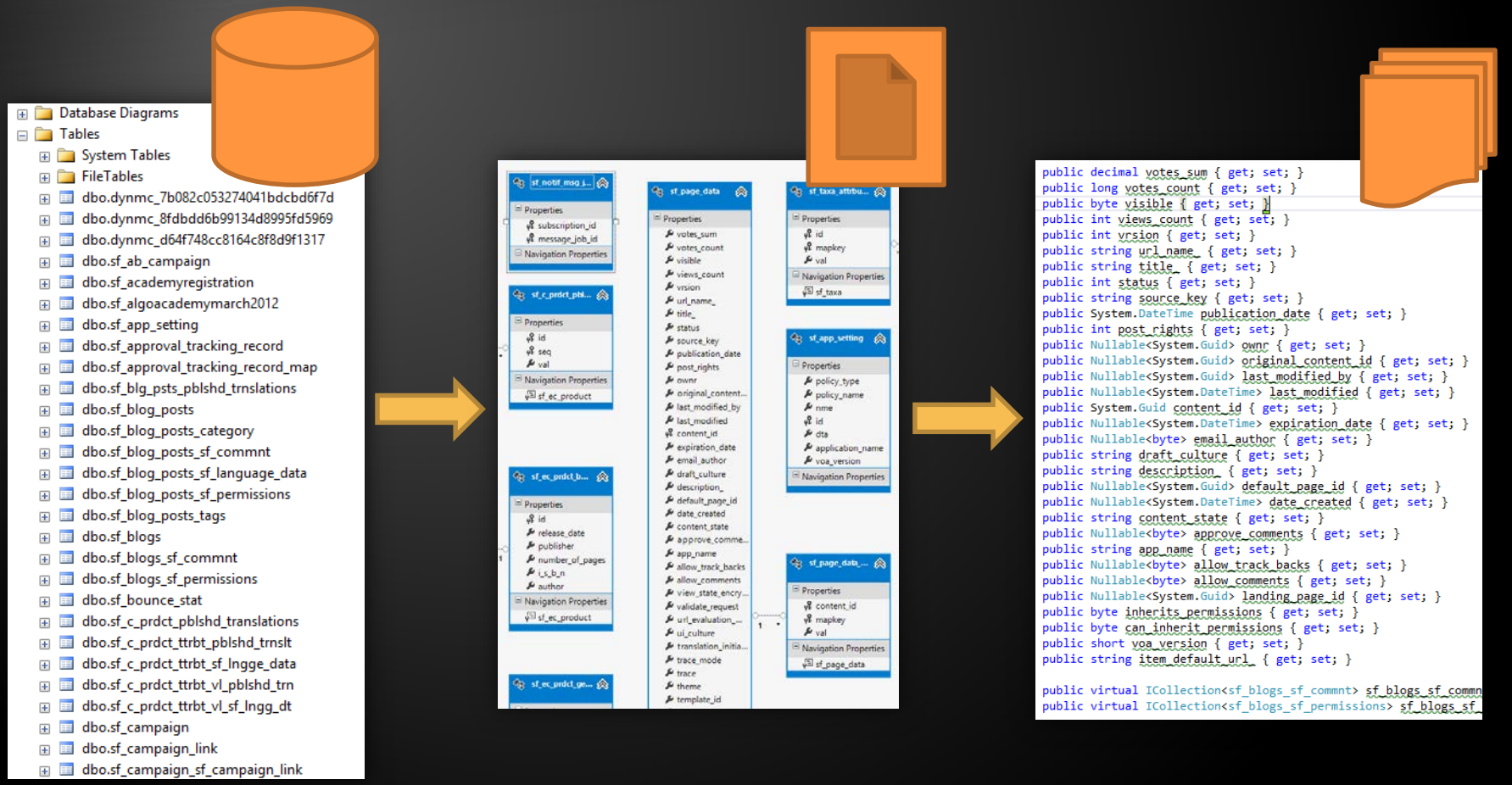
Modeling Workflow

- Entity Framework supports three types of modeling workflow:
  - Database first
    - Create models as database tables
    - Use Management Studio or native SQL queries
  - Model first
    - Create models using visual EF designer in VS
  - Code first
    - Write models and combine them in DbContext

Telerik Academy

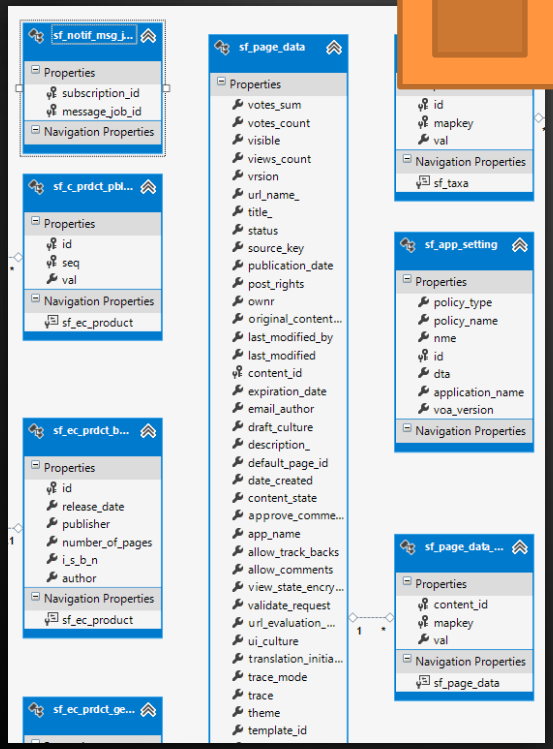
Database First Modeling Workflow

- Create models as database tables and then generate code (models) from them

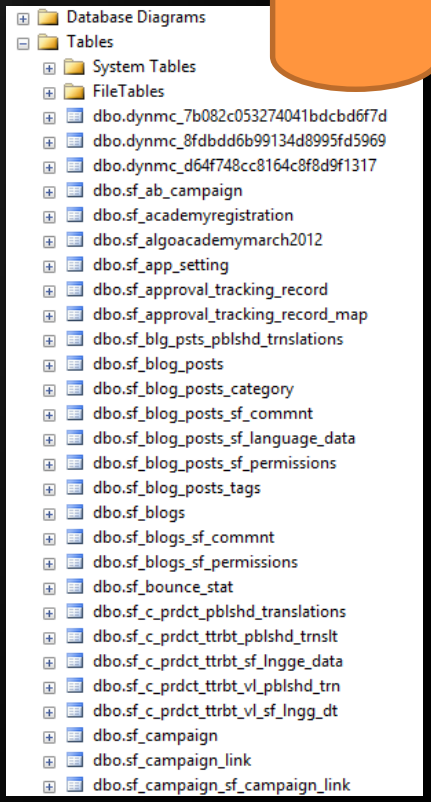


Telerik Academy

Model First Modeling Workflow



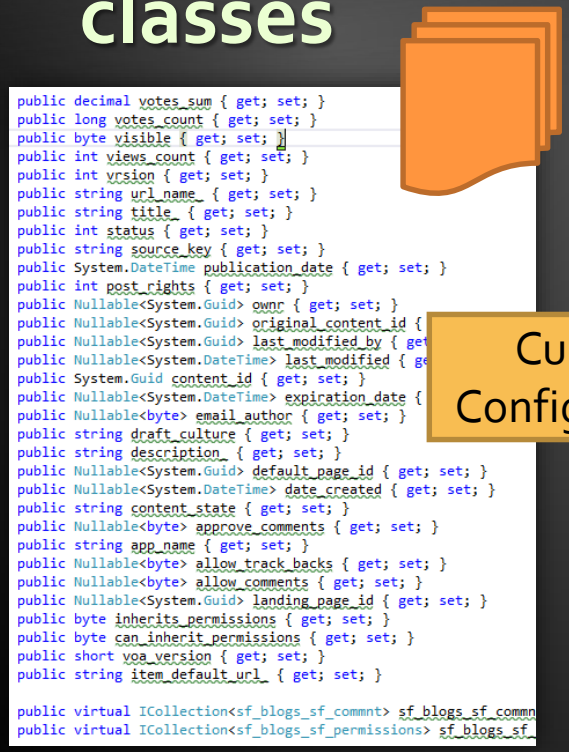

```
public decimal votes_sum { get; set; }
public long votes_count { get; set; }
public byte visible { get; set; }
public int views_count { get; set; }
public int vrsion { get; set; }
public string url_name_ { get; set; }
public string title_ { get; set; }
public int status { get; set; }
public string source_key { get; set; }
public System.DateTime publication_date { get; set; }
public int post_rights { get; set; }
public Nullable<System.Guid> ownr { get; set; }
public Nullable<System.Guid> original_content_id { get; set; }
public Nullable<System.Guid> last_modified_by { get; set; }
public Nullable<System.DateTime> last_modified { get; set; }
public System.Guid content_id { get; set; }
public Nullable<System.DateTime> expiration_date { get; set; }
public Nullable<byte> email_author { get; set; }
public string draft_culture { get; set; }
public string description_ { get; set; }
public Nullable<System.Guid> default_page_id { get; set; }
public Nullable<System.DateTime> date_created { get; set; }
public string content_state { get; set; }
public Nullable<byte> approve_comments { get; set; }
public string app_name { get; set; }
public Nullable<byte> allow_track_backs { get; set; }
public Nullable<byte> allow_comments { get; set; }
public Nullable<System.Guid> landing_page_id { get; set; }
public byte inherits_permissions { get; set; }
public byte can_inherit_permissions { get; set; }
public short voa_version { get; set; }
public string item_default_url_ { get; set; }

public virtual ICollection<sf_blogs_sf_commnt> sf_blogs_sf_commn
public virtual ICollection<sf_blogs_sf_permissions> sf_blogs_sf_
```



Telerik Academy

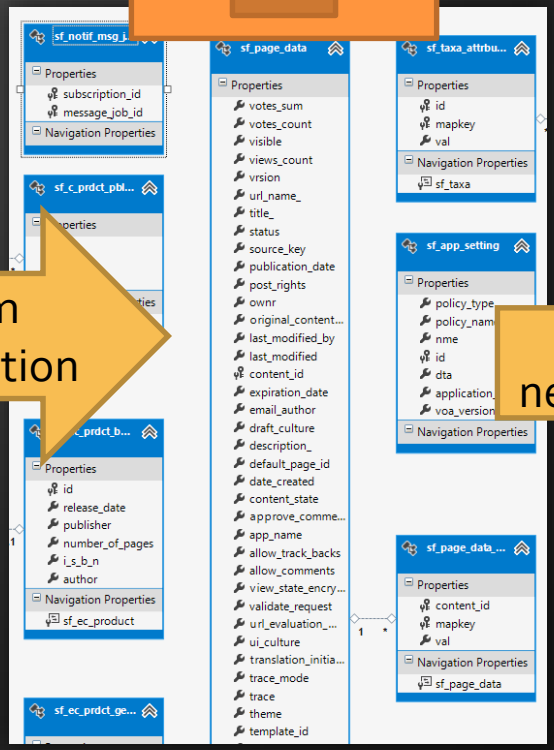
Code First Modeling Workflow

Domain classes



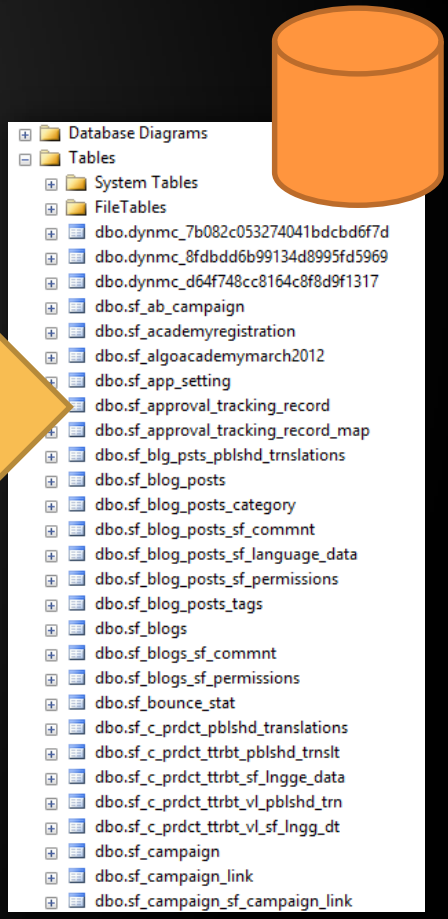
```
public decimal votes_sum { get; set; }
public long votes_count { get; set; }
public byte visible { get; set; }
public int views_count { get; set; }
public int vrsion { get; set; }
public string url_name_ { get; set; }
public string title_ { get; set; }
public int status { get; set; }
public string source_key { get; set; }
public System.DateTime publication_date { get; set; }
public int post_rights { get; set; }
public Nullable<System.Guid> ownr { get; set; }
public Nullable<System.Guid> original_content_id {
public Nullable<System.Guid> last_modified_by { ge
public Nullable<System.DateTime> last_modified { g
public System.Guid content_id { get; set; }
public Nullable<System.DateTime> expiration_date {
public Nullable<byte> email_author { get; set; }
public string draft_culture { get; set; }
public string description_ { get; set; }
public Nullable<System.Guid> default_page_id { get; set; }
public Nullable<System.DateTime> date_created { get; set; }
public string content_state { get; set; }
public Nullable<byte> approve_comments { get; set; }
public string app_name { get; set; }
public Nullable<byte> allow_track_backs { get; set; }
public Nullable<byte> allow_comments { get; set; }
public Nullable<System.Guid> landing_page_id { get; set; }
public byte inherits_permissions { get; set; }
public byte can_inherit_permissions { get; set; }
public short voa_version { get; set; }
public string item_default_url_ { get; set; }

public virtual ICollection<sf_blogs_sf_commnt> sf_blogs_sf_commn
public virtual ICollection<sf_blogs_sf_permissions> sf_blogs_sf_
```

DbContext
ModelBuilder



Custom Configuration

As needed



Why Use Code First?

- Write code without having to define mappings in XML or create database models
- Define objects in POCO
  - Reuse these models and their attributes
- No base classes required
- Enables database persistence with no configuration
  - Can use automatic migrations
- Can use Data Annotations (Key, Required, etc.)

Telerik Academy

Code First Main Parts

Domain classes, DbContext and DbSets

Domain Classes (Models)

- Bunch of normal C# classes (POCO)
  - May contain navigation properties

```
public class PostAnswer
{
    public int PostAnswerId { get; set; }
    public string Content { get; set; }
    public int PostId { get; set; }
    public virtual Post Post { get; set; }
}
```

Primary key

Foreign key

Virtual for lazy loading

Navigation property

- Recommended to be in a separate class library

Telerik Academy

Domain Classes (Models) (2)

- Another example of domain class (model)

```
public class Post
{
    private ICollection<PostAnswer> ans
    public Post()
    {
        this.answers = new HashSet<PostAnswer>();
    }
    // ...
    public virtual ICollection<PostAnswer> Answers
    {
        get { return this.answers; }
        set { this.answers = value; }
    }
    public PostType Type { get; set; }
}
```

Prevents null reference exception

Navigation property

Enumeration

Telerik Academy

Demo: Creating Models

Creating domain classes (models)

DbContext Class

- A class that inherits from DbContext
  - Manages model classes using DbSet type
  - Implements identity tracking, change tracking, and API for CRUD operations
  - Provides LINQ-based data access
- Recommended to be in a separate class library
  - Don't forget to reference the Entity Framework library (using NuGet package manager)
- If you have a lot of models it is recommended to use more than one DbContext

DbSet Type

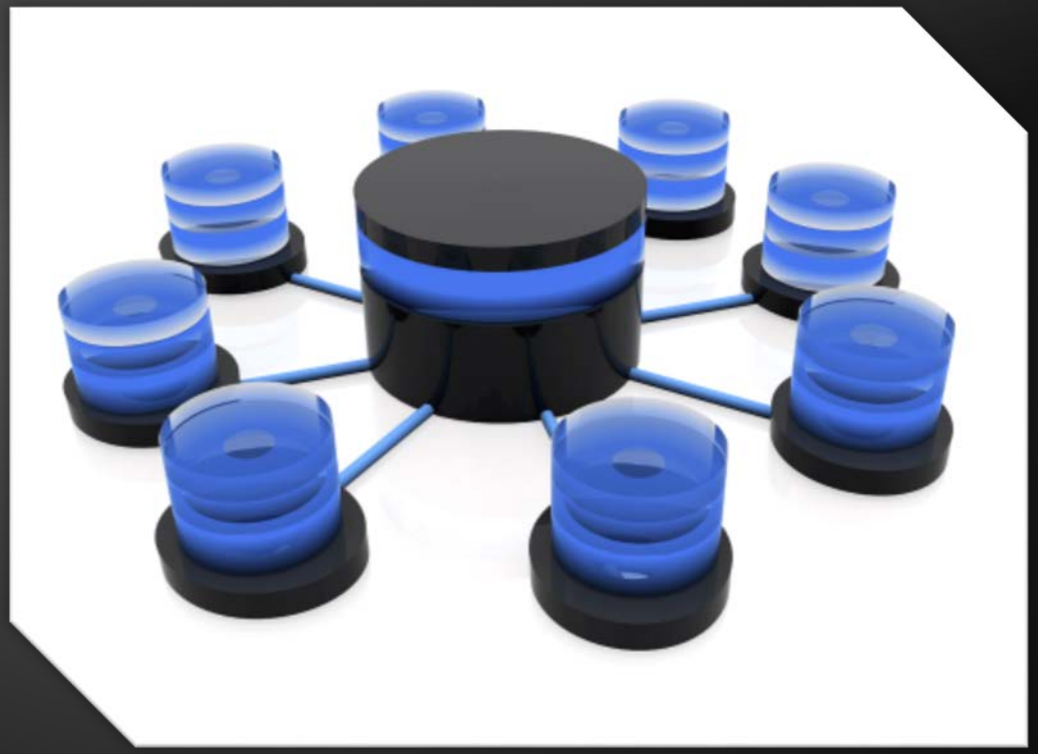
- Collection of single entity type
- Set operations: Add, Attach, Remove, Find
- Use with DbContext to query database

public class **DbSet<TEntity>** :
System.Data.Entity.Infrastructure.DbQuery<TEntity>
where **TEntity** : class
Member of System.Data.Entity

```
public DbSet<Post> Posts { get; set; }
```

DbContext Example

```
using System.Data.Entity;

using CodeFirst.Models;

public class ForumContext : DbContext
{
    public DbSet<Category> Categories { get; set; }

    public DbSet<Post> Posts { get; set; }

    public DbSet<PostAnswer> PostAnswers { get; set; }

    public DbSet<Tag> Tags { get; set; }
}
```



Demo: Creating DbContext

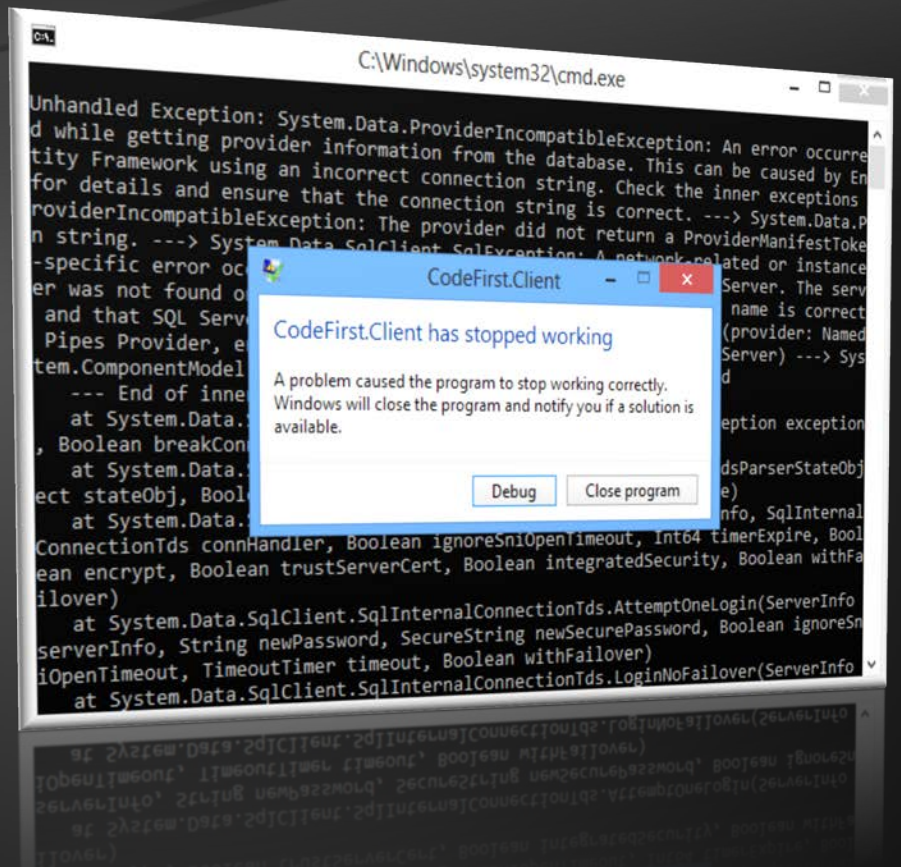
Telerik Academy

How to Interact With the Data?

- In the same way as when we use database first or model first approach

```
var db = new ForumContext();
var category = new Category { Parent = null, Name =
"Database course", };
db.Categories.Add(category);

var post = new Post();
post.Title = "Срока на домашните";
post.Content = "Моля удължете срока на домашните";
post.Type = PostType.Normal;
post.Category = category;
post.Tags.Add(new Tag { Text = "домашни" });
post.Tags.Add(new Tag { Text = "срок" });
db.Posts.Add(post);
db.SaveChanges();
```

Telerik Academy



Demo: Using The Data

Where is My Data?

- By default app.config file contains link to default connection factory that creates local db

```
<entityFramework>
  <defaultConnectionFactory
type="System.Data.Entity.Infrastructure.LocalDbConnection
Factory, EntityFramework">
    <parameters>
      <parameter value="v11.0" />
    </parameters>
  </defaultConnectionFactory>
</entityFramework>
```

- Server name by default: (localdb)\v11.0 or .\SQLEXPRESS.[full-class-name]
  - We can use VS server explorer to view database

Telerik Academy

How to Connect to SQL Server?

- First, create context constructor that calls base constructor with appropriate connection name

```
public class ForumContext : DbContext
{
    public ForumContext()
        : base("ForumDb")
    { }
    // ...
}
```

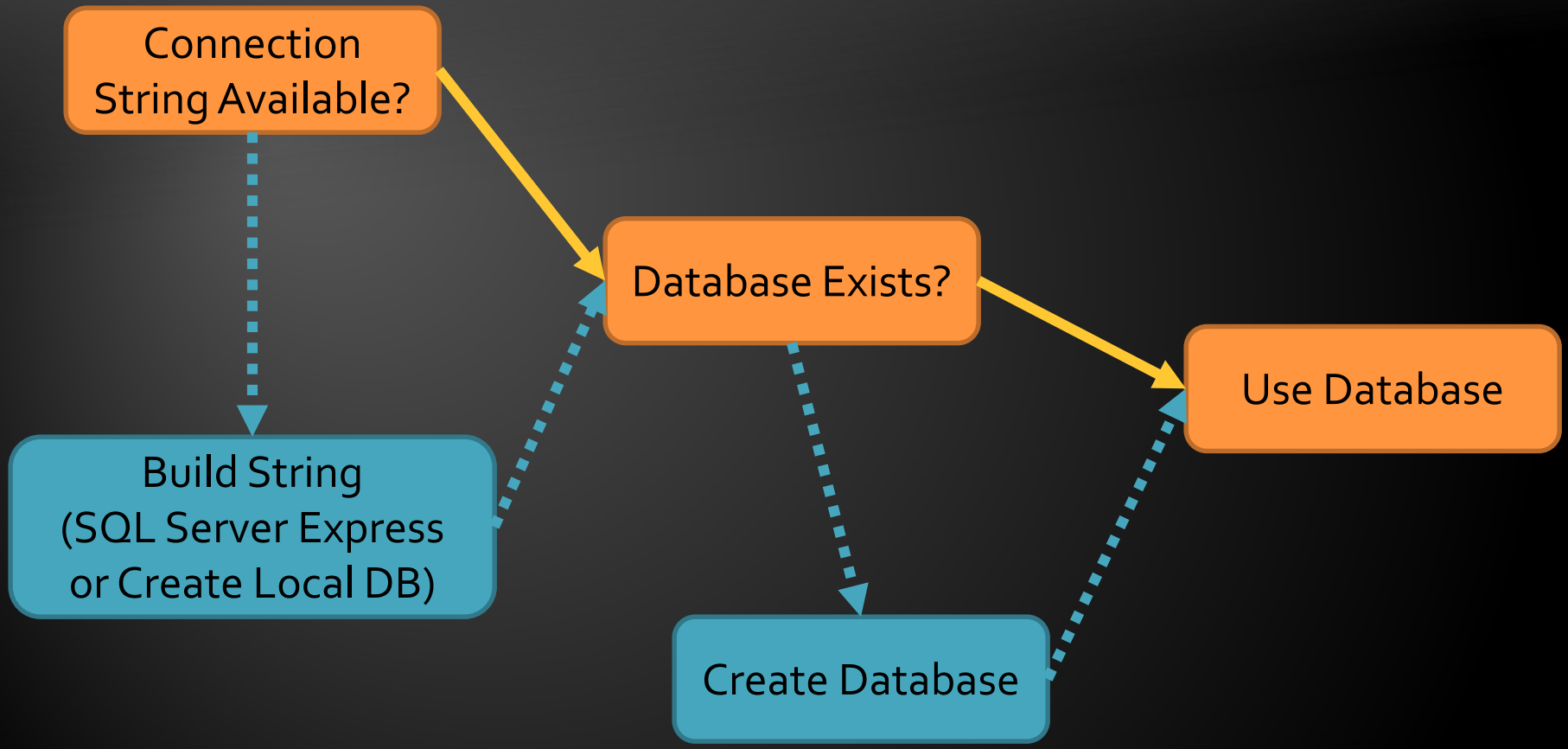
- Then a[illegible]n string in app.config

Server address might be .\SQLEXPRESS

```
<connecti[illegible]
   <add name="ForumDb" connectionString="Data
Source=.;Initial Catalog=ForumDb;Integrated
Security=True" providerName="System.Data.SqlClient" />
</connectionStrings>
```

Telerik Academy

Database Connection Workflow



Telerik Academy

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration
    <section name="entityFramework" type="System.Data.Entity.In
  </configSections>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version
  </startup>
  <entityFramework>
    <defaultConnectionFactory type="System.Data.Entity.Infrastr
      <parameters>
        <parameter value="v11.0" />
      </parameters>
    </defaultConnectionFactory>
  </entityFramework>
  <connectionStrings>
    <add name="ForumDb" connectionString="Data Source=.;Initial
  </connectionStrings>
</configuration>
```

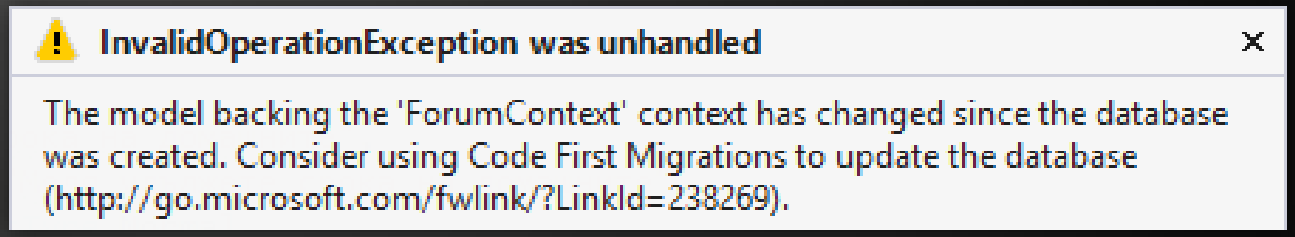
Demo: Change Database Connection

Telerik Academy

Using Code First Migrations

Changes in Domain Classes

- What happens when we change our models?
  - Entity Framework compares our model with the model in __MigrationHistory table



- By default Entity Framework only creates the database and don't do any changes after that
- Using Code First Migrations we can manage differences between models and database

Code First Migrations

- Enable Code First Migrations
  - Open Package Manager Console
  - Run Enable-Migrations command
    - This will create some initial jumpstart code
    - -EnableAutomaticMigrations for auto migrations
- Two types of migrations
  - Automatic migrations
    - Set AutomaticMigrationsEnabled = true;
  - Code-based (providing full control)
    - Separate C# code file for every migration

Telerik Academy

Database Migration Strategies

- CreateDatabaseIfNotExists (default)
- DropCreateDatabaseIfModelChanges
  - We loose all the data when change the model
- DropCreateDatabaseAlways
  - Great for automated integration testing
- MigrateDatabaseToLatestVersion
  - This option uses our migrations
- We can implement IDatabaseInitializer if we want custom migration strategy

Telerik Academy

Use Code First Migrations

- First, enable code first migrations
- Second, we need to tell to Entity Framework to use our migrations with code (or app.config)

```
Database.SetInitializer(
    new MigrateDatabaseToLatestVersion
        <ForumContext, Configuration>());
```

- We can configure automatic migration

```
public Configuration()
{
    this.AutomaticMigrationsEnabled = true;
    this.AutomaticMigrationDataLossAllowed = true;
}
```

This will allow us to delete or change properties

Seeding the Database

- During a migration we can seed the database with some data using the Seed method

```
protected override void Seed(ForumContext context)
{
    /* This method will be called after migrating to
       the latest version. You can use the
       DbSet<T>.AddOrUpdate() helper extension method
       to avoid creating duplicate seed data. E.g. */

    context.Tags.AddOrUpdate(new Tag { Text = "срок" });
    context.Tags.AddOrUpdate(new Tag { Text = "форум" });
}
```

- This method will be run every time (since EF 5)

Telerik Academy

Demo: Code First Migrations

Telerik Academy

Configure Mappings

Using Data Annotations and Fluent API

Configure Mappings

- Entity Framework respects mapping details from two sources
  - Data annotation attributes in the models
    - Can be reused for validation purposes
  - Fluent API code mapping configuration
    - By overriding OnModelCreating method
    - By using custom configuration classes
- Use one approach or the other

Data Annotations

- There is a bunch of data annotation attributes in System.ComponentModel.DataAnnotations
  - [Key] – specifies the primary key of the table
  - For validation: [StringLength], [MaxLength], [MinLength], [Required]
  - Schema: [Column], [Table], [ComplexType], [ConcurrencyCheck], [Timestamp], [ComplexType], [InverseProperty], [ForeignKey], [DatabaseGenerated], [NotMapped], [Index]
- In EF 6 we will be able to add custom attributes by using custom conventions

Fluent API for Mappings

- By overriding OnModelCreating method in DbContext class we can specify mapping configurations

```
protected override void OnModelCreating(DbModelBuilder
modelBuilder)
{
    modelBuilder.Entity<Tag>().HasKey(x => x.TagId);
    modelBuilder.Entity<Tag>().Property(x =>
x.Text).IsUnicode(true);
    modelBuilder.Entity<Tag>().Property(x =>
x.Text).HasMaxLength(255);
    // modelBuilder.Entity<Tag>().Property(x =>
x.Text).IsFixedLength();
    base.OnModelCreating(modelBuilder);
}
```

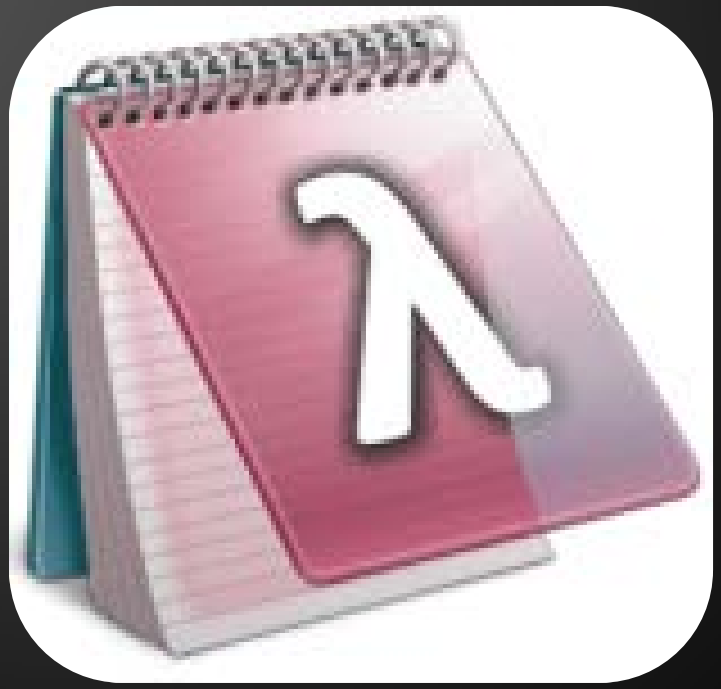
Fluent API Configurations

- .Entity()
  - Map: Table Name, Schema
  - Inheritance Hierarchies, Complex Types
  - Entity -> Multiple Tables
  - Table -> Multiple Entities
  - Specify Key (including Composite Keys)
- .Property()
  - Attributes (and Validation)
  - Map: Column Name, Type, Order
  - Relationships
  - Concurrency

Demo: Configure Mappings

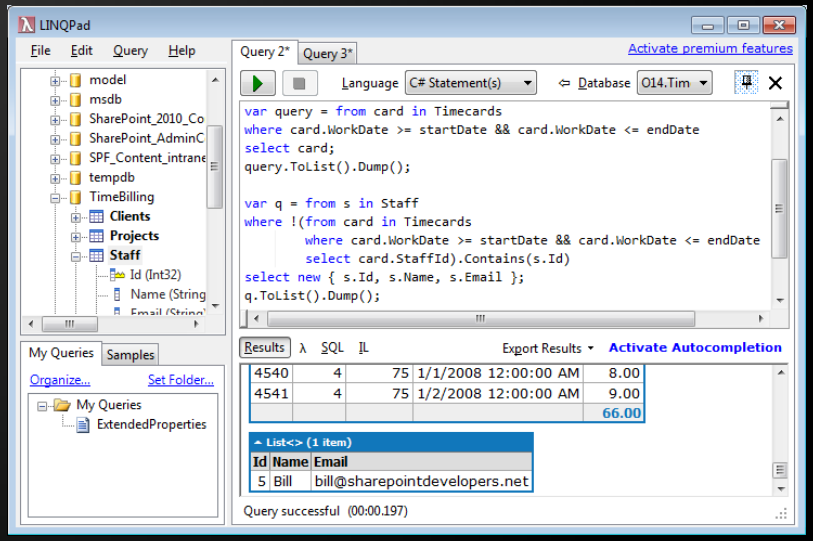
Telerik Academy

LINQPad



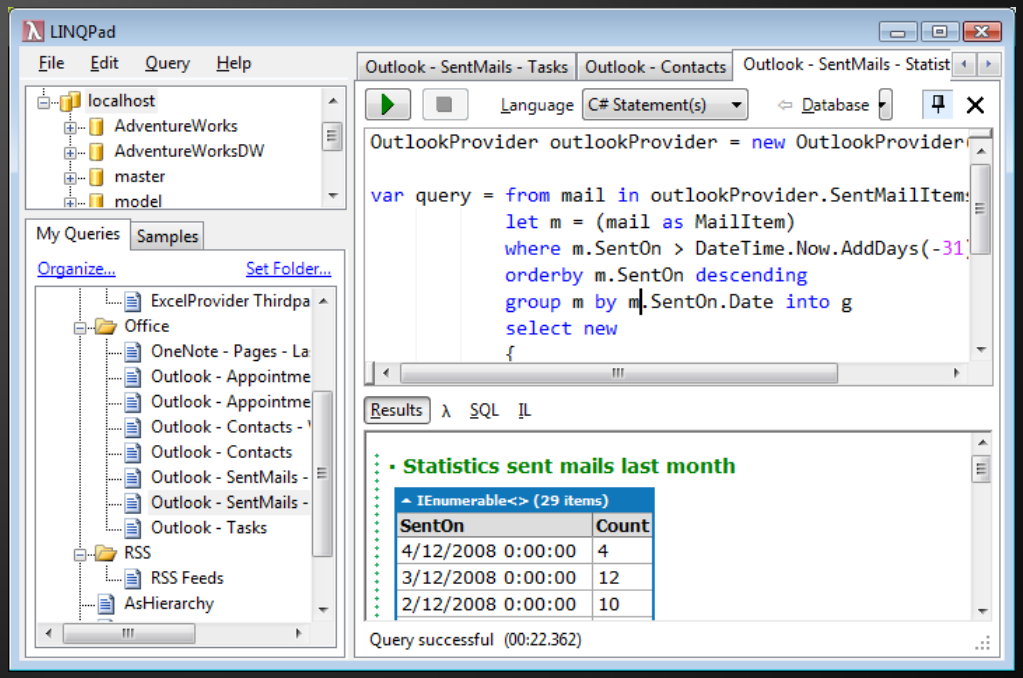
LINQPad

- Download from: http://www.linqpad.net/
- Supports native C# 5.0
- Translates it to SQL, XML, Objects
- Understands LINQ
- Free and paid version



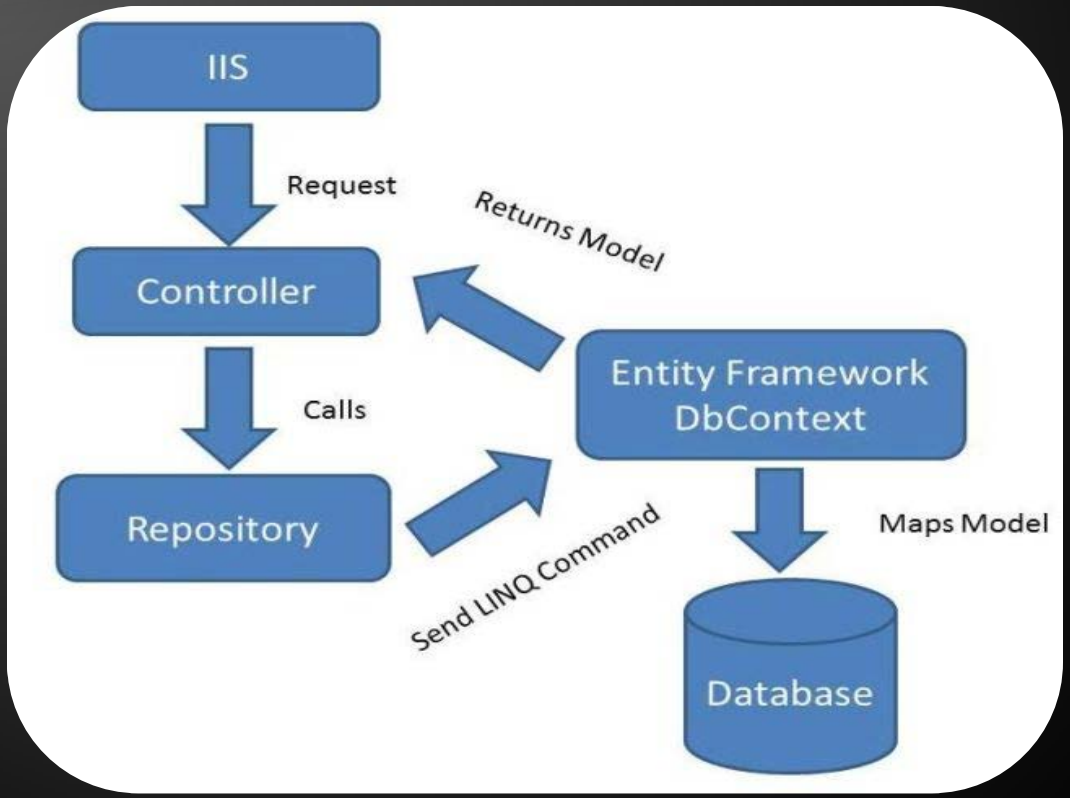
Telerik Academy

Demo: LINQ Pad



Telerik Academy

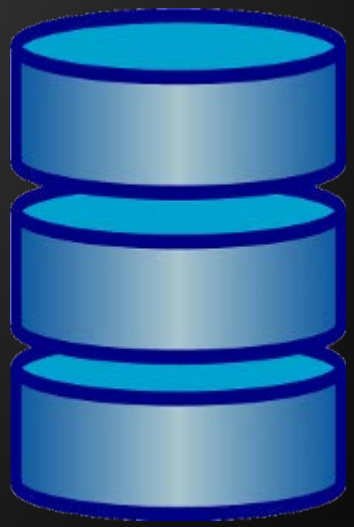
Repository Pattern



Repository Pattern

- Gives abstraction over the data layer
- Single place to make changes to your data access
- Single place responsible for a set of tables
- Easily replaceable by other implementations
- Hides the details in accessing data
- Can be implemented in various ways

Demo: Repository Pattern



Telerik Academy

Entity Framework Code First

http://academy.telerik.com

Homework

1. Using code first approach, create database for student system with the following tables:
   - Students (with Id, Name, Number, etc.)
   - Courses (Name, Description, Materials, etc.)
   - StudentsInCourses (many-to-many relationship)
   - Homework (one-to-many relationship with students and courses), fields: Content, TimeSent
   - Annotate the data models with the appropriate attributes and enable code first migrations
2. Write a console application that uses the data
3. Seed the data with random values

Telerik Academy

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