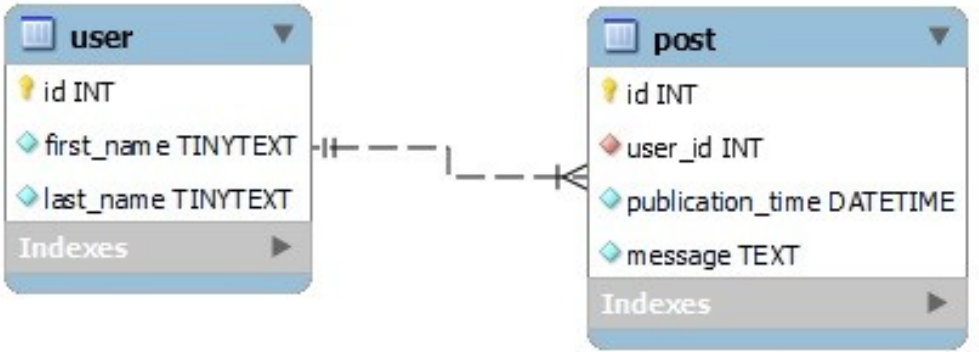
Промышленное программирование

Лекция 8

Entity Framework

Entity Framework

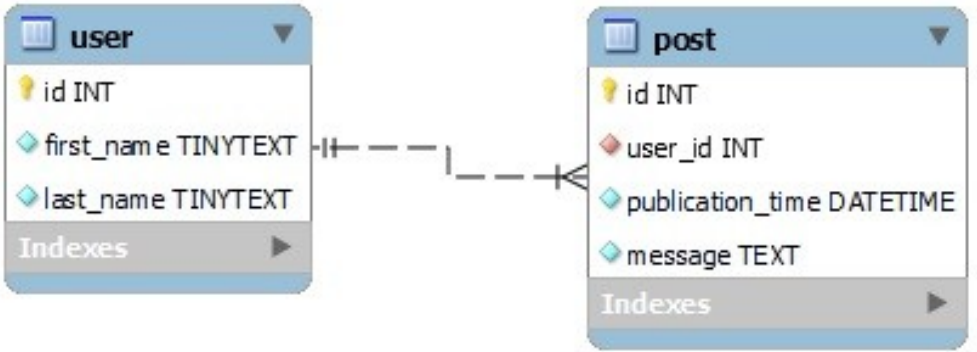


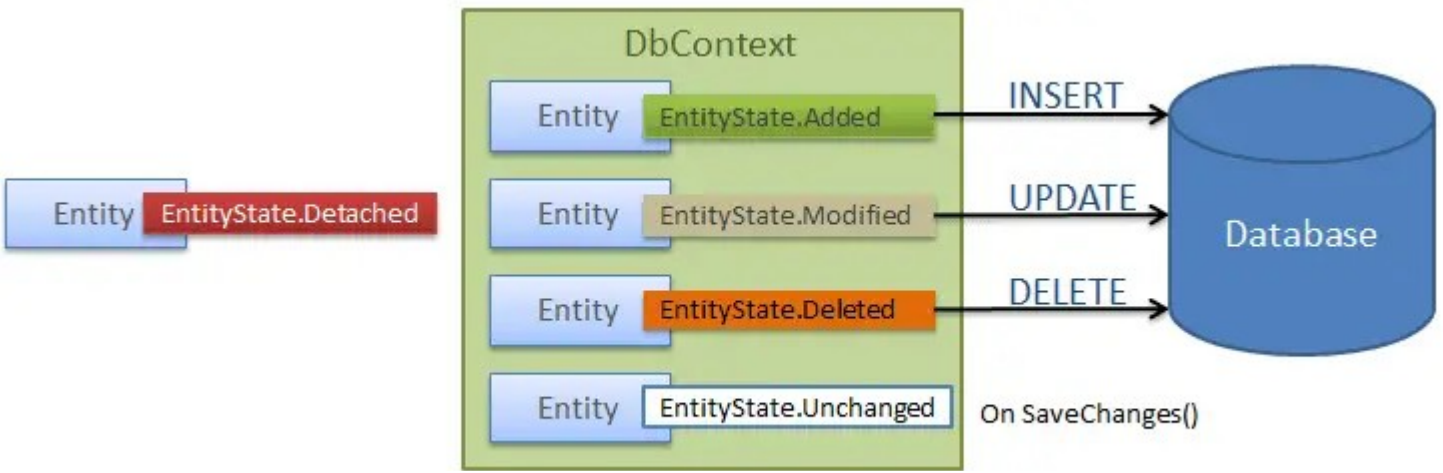

```
using Microsoft.EntityFrameworkCore;

public sealed class UserWallContext : DbContext
{
    public DbSet<Post> Posts { get; set; } = null!;
    public DbSet<User> Users { get; set; } = null!;
}
```

```
public sealed class User
{
    public int Id { get; set; }

    public string FirstName { get; set; } = string.Empty;

    public string LastName { get; set; } = string.Empty;

    public List<Post> Posts { get; set; } = null!;
}
```

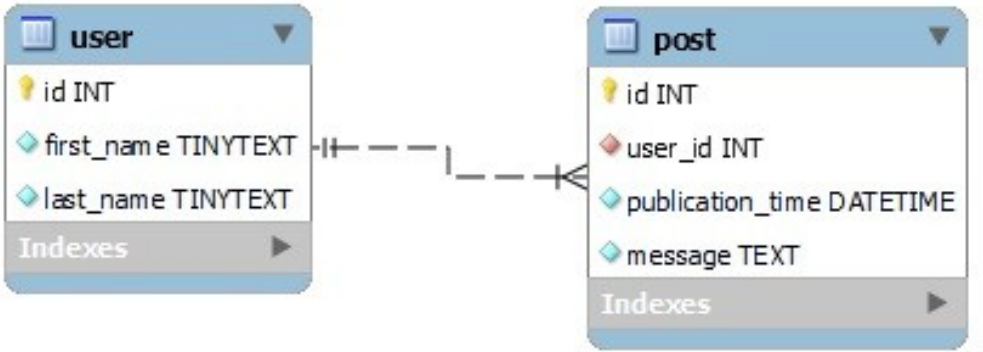
```
public sealed class Post
{
    public int Id { get; set; }

    public int UserId { get; set; }

    public DateTime PublicationTime { get; set; }

    public string Message { get; set; } = string.Empty;

    public User User { get; set; } = null!;
}
```

Entity Framework: состояния





Entity Framework: примеры запросов

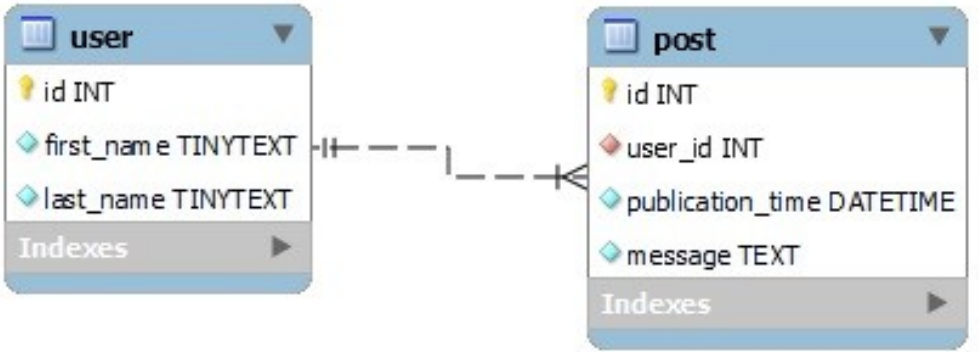



```
using var ctx = new UserWallContext();
ctx.Users.Add(new User { FirstName = "Daniil", LastName = "Ivanov" });
ctx.SaveChanges();
```

```
using var ctx = new UserWallContext();
var user = ctx.Users
    .Where(user => user.Id == 1)
    .Single();
```

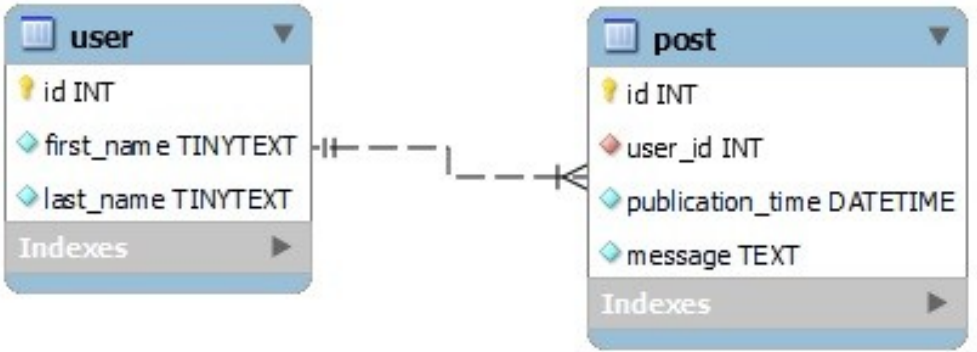
```
using var ctx = new UserWallContext();
var user = ctx.Find<User>(1);
```

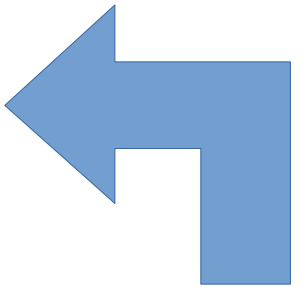
Entity Framework: навигационные свойства




```
using var ctx = new UserWallContext();

var user = ctx.Users.Where(user => user.Id == 1)
    .Include(user => user.Posts)
    .Single();
```

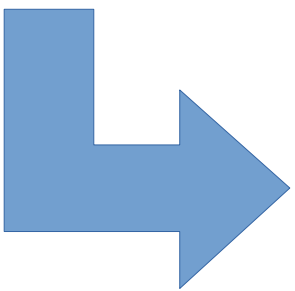
Варианты Code-First и DB-First





Code-First

DB-First



```
using Microsoft.EntityFrameworkCore;

public sealed class UserWallContext : DbContext
{
    public DbSet<Post> Posts { get; set; } = null!;
    public DbSet<User> Users { get; set; } = null!;
}

…
```

Подготовка к использованию EF

1. [Server Project] В контекстном меню проекта выбрать Manage User Secrets.

2. [Server Project] В файле secrets.json подготовить строку подключения к СУБД (пример для MySQL):

```
{
  "ConnectionStrings": {
    "UserWall": "Server=localhost;Database=user_wall;Uid=root;Pwd=1234567;"
  }
}
```

3. [Server Project] Настроить фабрику контекстов в Program.cs (пример для MySQL):

```
var connectionString = builder.Configuration.GetConnectionString(nameof(UserWall));
builder.Services.AddDbContextFactory<UserWallContext>(optionsBuilder => optionsBuilder.UseMySQL(connectionString));
```

Code-First: Пример MySQL

1. [Library Project] Установить пакет:

   MySql.EntityFrameworkCore

2. [Library Project] В класс контекста добавить конструктор с параметром:

```
public sealed class UserWallContext : DbContext
{
    …

    public UserWallContext(DbContextOptions options) : base(options)
    {
        Database.EnsureCreated();
    }
}
```

Code First Data Annotations

DB-First: Общий алгоритм

Полный гайд

1. Установить пакеты

   Microsoft.EntityFrameworkCore.Design
   Microsoft.EntityFrameworkCore.Tools
   Пакет нужного провайдера СУБД

2. В консоли PMC (Tools → NuGet Package Manager → Package Manager Console):

```
Scaffold-DbContext ConnectionString Provider
```

Документация по команде Scaffold-DbContext

DB-First: Пример MySQL

1. [Library Project] Установить пакеты

   Microsoft.EntityFrameworkCore.Design
   Microsoft.EntityFrameworkCore.Tools
   MySql.EntityFrameworkCore

2. [Library Project] Сгенерировать entity-классы в PMC:

```
Scaffold-DbContext "Server=localhost;Database=user_wall;Uid=root;Pwd=123456;"
                   MySql.EntityFrameworkCore
                   -Force
                   -NoOnConfiguring
                   -Project UserWall
                   -StartupProject UserWall
```

Заключение

[Server Project] Теперь можно внедрять фабрику и создавать контекст:

```
public class UserController : ControllerBase
{
    private readonly IDbContextFactory<UserWallContext> _contextFactory;
    private readonly IMapper _mapper;

    public UserController(IDbContextFactory<UserWallContext> contextFactory, IMapper mapper)
    {
        _contextFactory = contextFactory;
        _mapper = mapper;
    }

    [HttpGet]
    public IEnumerable<UserDto> Get()
    {
        using var ctx = _contextFactory.CreateDbContext();
        return _mapper.Map<IEnumerable<UserDto>>(ctx.Users);
    }
}
```