CUSTIS® ИТ-РЕШЕНИЯ ДЛЯ РАЗВИТИЯ

Загадочный EF Core, или Как написать свое расширение

Игорь Шаталкин

Программист-эксперт

DotNext, 20 ноября 2022

Что может быть проще?

- Контекст
  - DotNetCore
  - ORM: EF Core
  - Приложение почти готово к внедрению
- Задача
  - Нужны вьюхи
  - Много вьюх

EF Core поддерживает это из коробки

```
public class Class1Configuration : IEntityTypeConfi
{
    public void Configure(EntityTypeBuilder<Class1
    {
        builder.ToTable("my_table").HasKey(e => e
        // ...
    }
}

public class View1Configuration : IEntityTypeCo
{
    public void Configure(EntityTypeBuilder<Vi
    {
        builder.ToView("my_view").HasSql("SELECT * FRO
        // ...
    }
}
```

Есть готовый пакет

```
dotnet add package EntityFrameworkCore.Views
```

💡 Просто пропишем вьюху в миграцию!

```
public partial class DotNext1 : Migration
{
    protected override void Up(MigrationBuilder m
    {
        migrationBuilder.Sql(@"create or replace
as select * from migr_ext_tests.my_table;");
    }

    protected override void Down(MigrationBuild
    {
        migrationBuilder.Sql("drop view v_view
    }
}
```

Давайте введем константу

```
public class Views
{
    public const string MyView = @"create or replace view v_view_dotnext_1
as select * from migr_ext_tests.my_table;";
}

public partial class DotNext1 : Migration
{
    protected override void Up(MigrationBuilder migrationBuilder)
    {
        migrationBuilder.Sql(Views.MyView);
    }
}
```

Почему не стоит хранить вьюхи в константах?

- [ ] 1) Будут проблемы с кодировкой
- [ ] 2) Невозможно накатить заданную версию
- [ ] 3) Упремся в ограничения на максимальную длину строки
- [ ] 4) Вьюхи будут ссылаться на несуществующие колонки

Почему не стоит хранить вьюхи в константах?

- [ ] 1) Будут проблемы с кодировкой
- [x] 2) Невозможно накатить заданную версию
- [ ] 3) Упремся в ограничения на максимальную длину строки
- [x] 4) Вьюхи будут ссылаться на несуществующие колонки

Ситуация

1. Нужны вьюхи
2. Готового решения нет
   - ❌ EF Core поддерживает это из коробки
   - ❌ Есть готовый пакет
   -  Просто пропишем вьюху в миграцию!
   - 💣 Давайте введем константу
3. Напишем расширение для EF Core

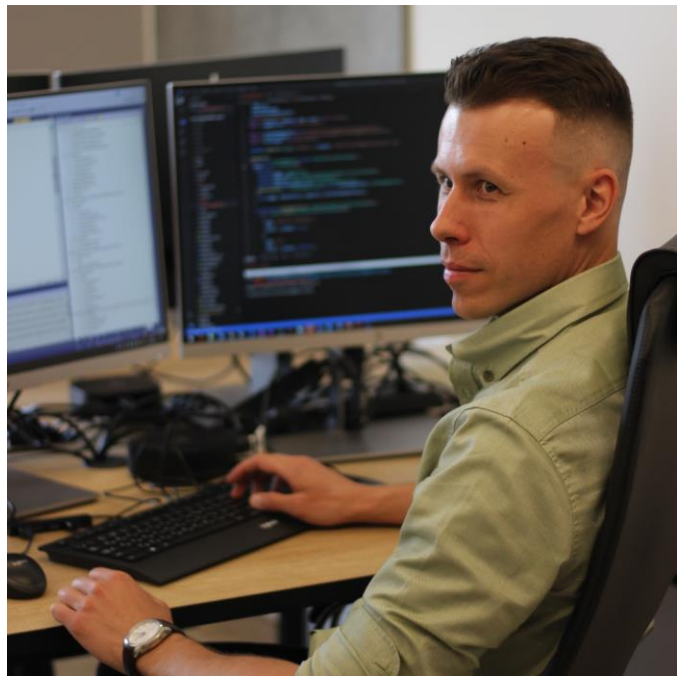
CUSTIS® ИТ-РЕШЕНИЯ ДЛЯ РАЗВИТИЯ

Загадочный EF Core, или Как написать свое расширение

Игорь Шаталкин

Программист-эксперт

DotNext, 20 ноября 2022



Игорь Шаталкин

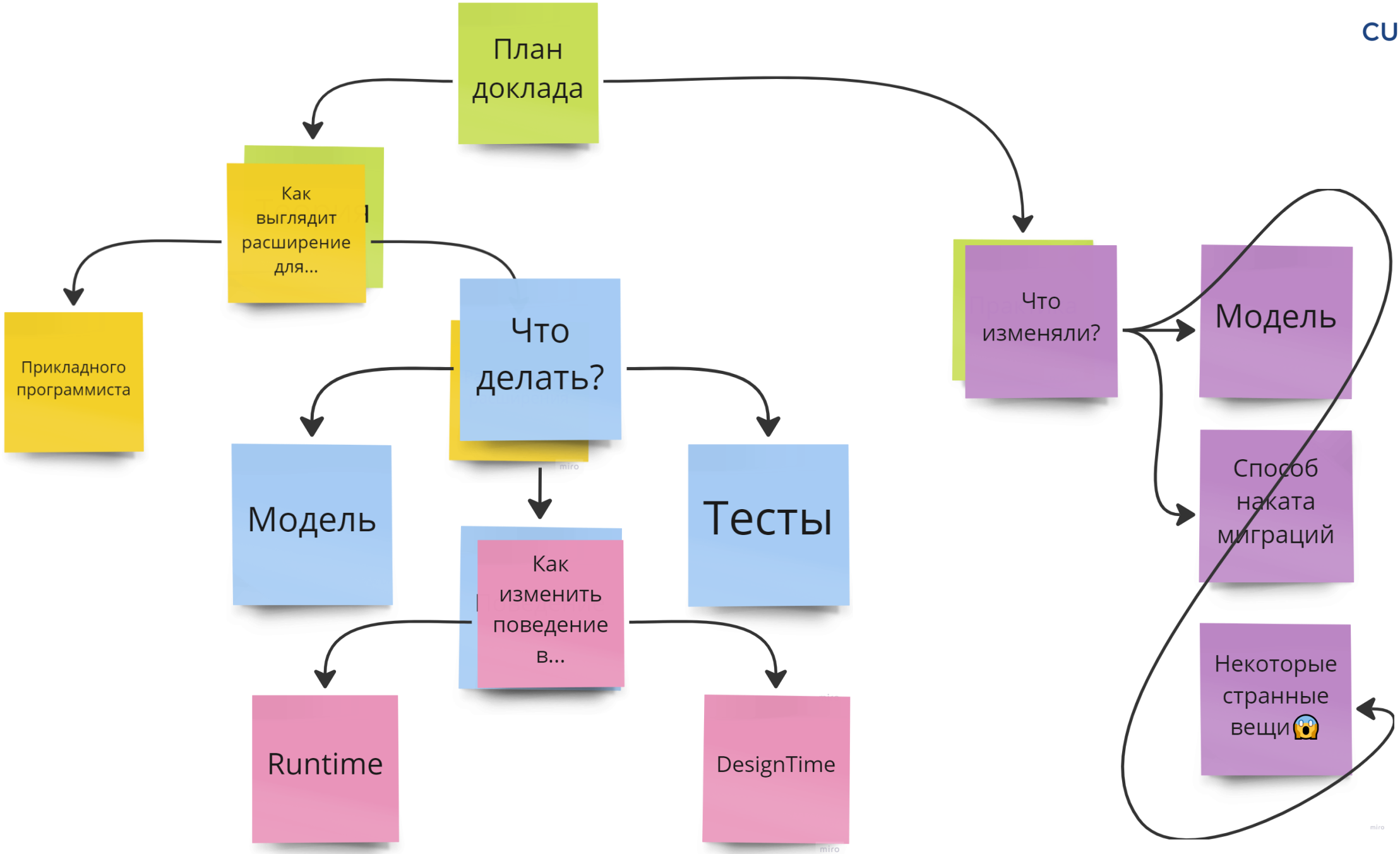
- 12 лет в ИТ
- Не люблю костыли и грабли
- Не люблю изобретать велосипеды
- Люблю делать переиспользуемые решения
- Люблю простые для прикладного программиста библиотеки

CUSTIS

- Разработка и консалтинг
- Команда: более 300 человек
- Среди клиентов: Банк России, «Спортмастер», «Газпром», крупные вузы

CUSTIS

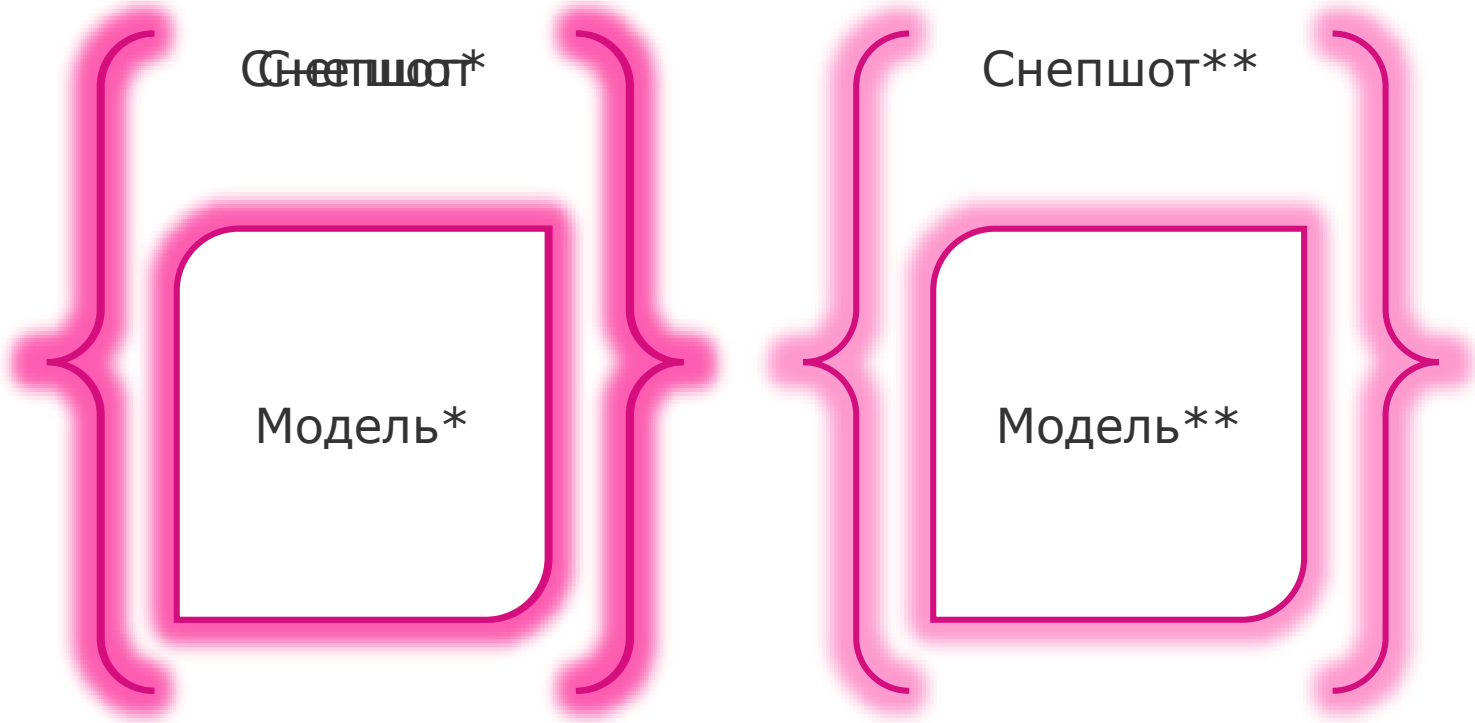
Как расширить EF Core


План доклада
Как выглядит расширение для...
Прикладного программиста
Что делать?
Модель
Тесты
Как изменить поведение в...
Runtime
DesignTime
Что изменяли?
Модель
Способ ната миграций
Некоторые странные вещи😱

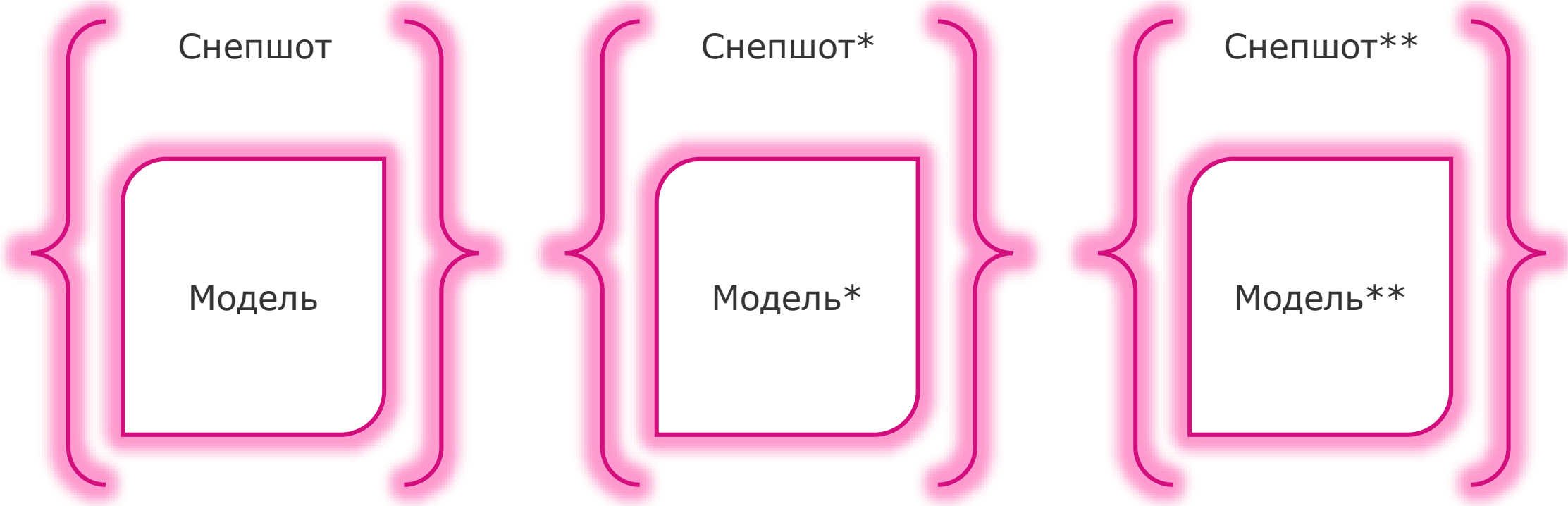
CUSTIS

Теория

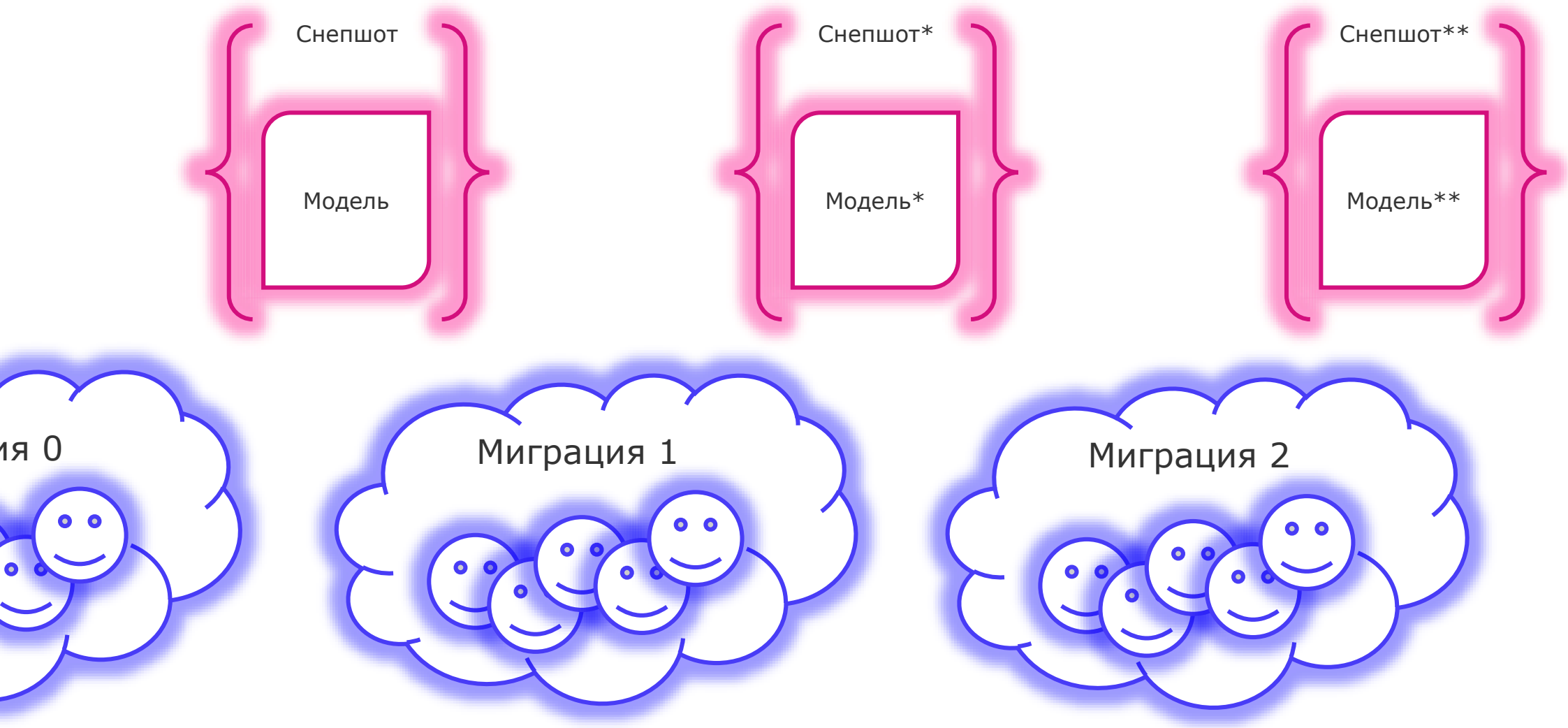
Основные понятия EF Core



Основные понятия EF Core

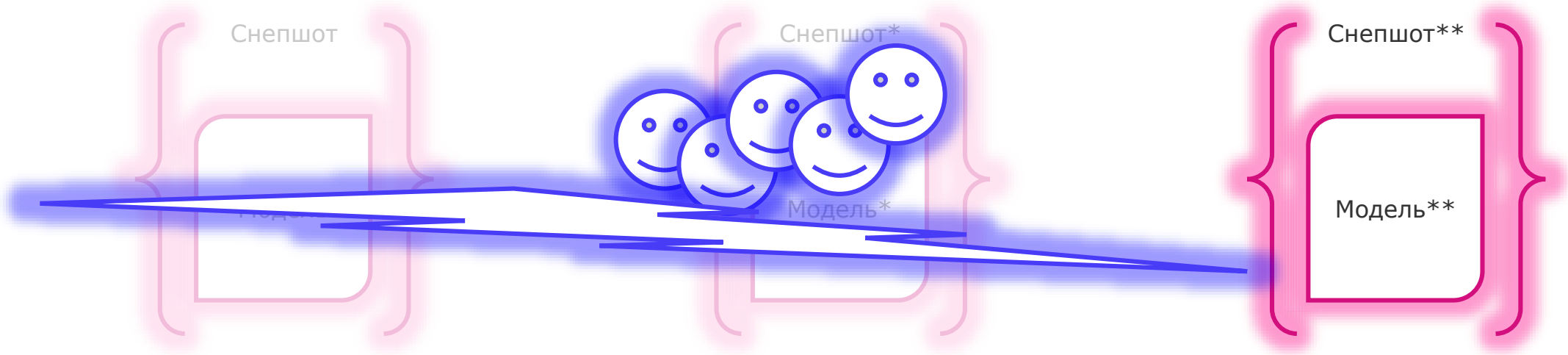


Основные понятия EF Core

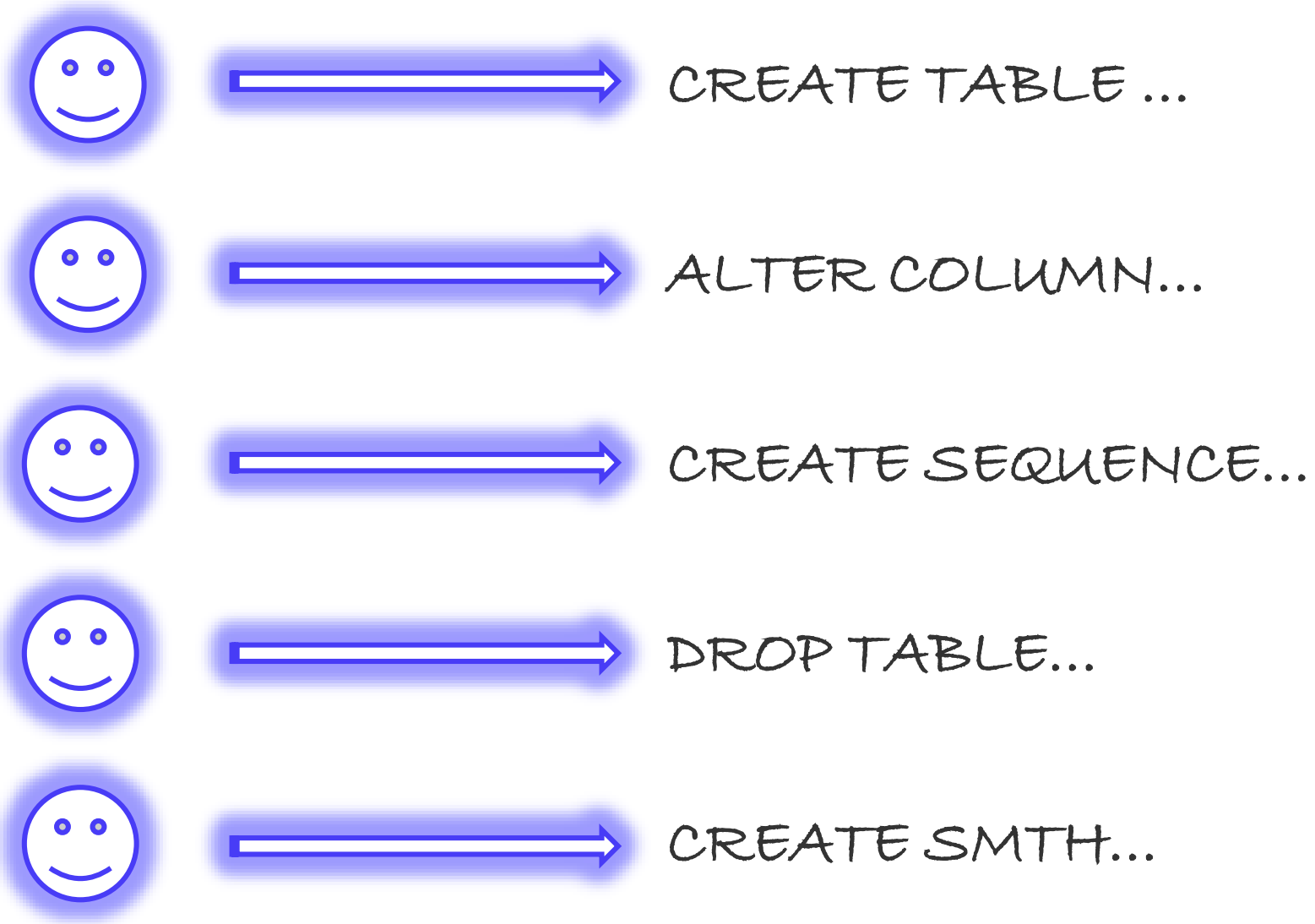


CUSTIS

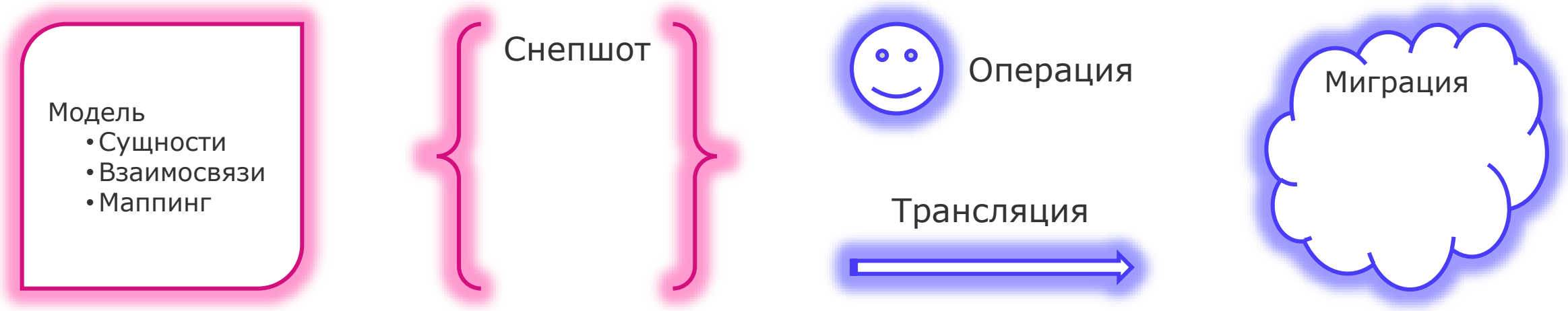
Основные понятия EF Core



Основные понятия EF Core



Основные понятия EF Core



Использование: органичность

Источник: https://t.me/loldev/2027

Простая настройка

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<TestContext>(options =>
                                            {
                                                options.UseNpgsql("my connection");
                                                options.UseViews();
                                            });
}
```

Логичность использования

```
public class TestContext : DbContext
{
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        // Add tables
        modelBuilder.ApplyConfigurationsFromAssembly(typeof(Class1Configuration).Assembly);

        // Add views
        const string Sql = @"create or replace view v_view_10
as select * from migr_ext_tests.my_table;";
        modelBuilder.AddViews(new View(Name: "v_view_10", SqlCode: Sql));
    }
}
```

Поддержка фич EF Core

- Отслеживание изменений
- Генерация UP и DOWN миграций
  - `dotnet ef migrations add InitialCreate`
- Обновление БД различными способами
  - `dotnet ef database update`
  - `dotnet ef migrations script`
  - `Database.Migrate()`
  - `Database.EnsureCreated()`

CUSTIS

Требования прикладника

- Легкость подключения
- Логичность использования
- Поддержка фич EF Core

Расширение модели

```
public record View(string Name, string SqlCode);

public static class ViewsModelExtensions
{
    internal const string ViewsData = "___Views";

    public static void AddViews(this IMutableModel model, IEnumerable<View> objects)
    {
        model[ViewsData] = objects;
    }

    internal static IReadOnlyCollection<View> GetViews(this IModel model)
    {
        return model[ViewsData] as IReadOnlyCollection<View>;
    }
}

public class TestContext : DbContext
{
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.AddViews(new View(Name: "v_view_10", SqlCode: "..."));
    }
}
```

Изменение поведения

Источник: https://t.me/loldev/1510

CUSTIS

Матрица контейнеров

CUSTIS

Матрица контейнеров

	Стандартный	Внутренний
Runtime		
DesignTime		

Матрица контейнеров

	Стандартный	Внутренний
Runtime	`DbContextOptions` `DbContext`	
DesignTime		

Матрица контейнеров

	Стандартный	Внутренний
Runtime	`DbContextOptions` `DbContext`	• Выполнение команд • Получение ConnectionString
DesignTime		• Генерация миграций • Генерация снепшотов

Runtime: сервисы EF Core

```
public static class ConfigurationExtensions
{
    public static void UseViews(this DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.ReplaceService<IMigrationsModelDiffer, MyModelDiffer>();
    }
}

public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<TestContext>(options =>
                                           {
                                               options.UseNpgsql("my connection");
                                               options.UseViews();
                                           });
}
```

Runtime: наши сервисы

```
public sealed class MyModelDiffer : IMigrationsModelDiffer
{
    private readonly IMyService _myService;

    public CustomMigrationsModelDiffer(IMyService myService)
    {
        _myService = myService;
    }
}

public static class ConfigurationExtensions
{
    public static void UseViews(this DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.ReplaceService<IMigrationsModelDiffer, MyModelDiffer>(); // OK
        optionsBuilder.ReplaceService<IMyService, MyService>(); // ERROR
    }
}
```

Runtime: наши сервисы

- `DbContextOptionsBuilder.UseInternalServiceProvider(_serviceProvider)`
  - ✕ нетривиальная настройка `_serviceProvider`
- EF Core Extensions

EF Core Extensions

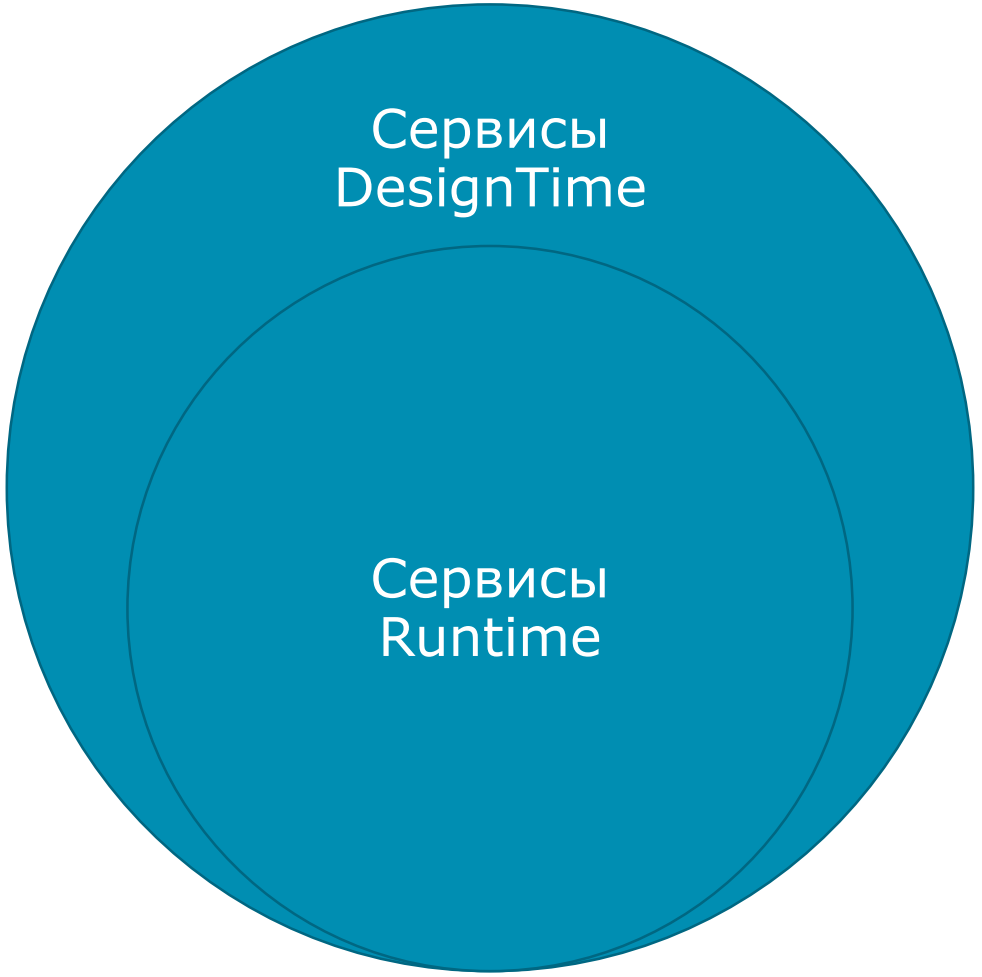
```
internal class ViewsExtension : IDbContextOptionsExtension
{
    public void ApplyServices(IServiceCollection services)
    {
        services.AddSingleton<IMyService, MyService>();
    }

    public void Validate(IDbContextOptions options) { }

    public DbContextOptionsExtensionInfo Info => new ViewsExtensionInfo(this);
}

public static class ConfigurationExtensions
{
    public static void UseViews(this DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.ReplaceService<IMigrationsModelDiffer, MyModelDiffer>();
        ((IDbContextOptionsBuilderInfrastructure)optionsBuilder).AddOrUpdateExtension(new ViewsExtension());
    }
}

public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<TestContext>(options =>
                                       {
                                           options.UseNpgsql("my connection");
                                           options.UseViews();
                                       });
}
```

CUSTIS

Изменение сервисов в Runtime

- Сервисы EF Core
  - `optionsBuilder.ReplaceService`
- Наши сервисы
  - `DbContextOptionsBuilder.UseInternalServiceProvider(_serviceProvider)`
    - ✕ нетривиальная настройка `_serviceProvider`
  - EF Core Extensions

DesignTime



DesignTime-1

```
public class CustomDesignTimeServices : IDesignTimeServices
{
    public virtual void ConfigureDesignTimeServices(IServiceCollection services)
    {
        // replace EF Core services
        services.ReplaceBySingleton<IMigrationsCodeGenerator, CustomMigrationsGenerator>();

        // add some custom services
        services.AddSingleton<ICustomSnapshotGenerator, ViewsSnapshotGenerator>();
    }

    protected static void ReplaceBySingleton<TService, TNewService>(IServiceCollection services)
        where TService : class
        where TNewService : class, TService
    {
        // ...
    }
}
```

DesignTime-2

```
[assembly: DesignTimeServicesReference
    ("TestEntryPoint.CustomDesignTimeServices, TestEntryPoint")]

public class CustomDesignTimeServices : IDesignTimeServices
{
    // ...
}
```

Изменение сервисов в DesignTime

- Реализация `IDesignTimeServices`
  - Только в startup assembly
- Использование `[DesignTimeServicesReference]`
  - Либо в startup, либо в migrations assembly

Подытог

- Расширение модели
  - Используем аннотации
- Изменение поведения
  - Контейнер из Startup не поможет
  - Runtime
  - DesignTime

Источник: https://t.me/loldev/2043

Тестирование

- Сложности
  - Много зависимостей

Примеры

```
public void OneTimeSetUp()
{
    var services = new ServiceCollection();

    services.AddEntityFrameworkDesignTimeServices();

    new CustomNpgsqlDesignTimeServices().ConfigureDesignTimeServices(services);

    _model = Mock.Of<IModel>();
    var relationalTypeMappingSource = Mock.Of<IRelationalTypeMappingSource>();
    services.AddSingleton(_model);
    services.AddSingleton(relationalTypeMappingSource);

    using var serviceScope = services.BuildServiceProvider().CreateScope();
    _generator = serviceScope.ServiceProvider.GetRequiredService<IMigrationsCodeGenerator>();
}
```

Примеры

```
public void SetUp()
{
    var relationalTypeMappingSource = Mock.Of<IRelationalTypeMappingSource>();

    var context = new DbContext(new DbContextOptionsBuilder()
                    .UseInMemoryDatabase(nameof(CustomMigrationsModelDifferTests)).Options);

    _differ = new CustomMigrationsModelDiffer(
        relationalTypeMappingSource, Mock.Of<IMigrationsAnnotationProvider>(),
        context.GetService<IChangeDetector>(),
        context.GetService<IUpdateAdapterFactory>(),
        new CommandBatchPreparerDependencies(Mock.Of<IModificationCommandBatchFactory>(),
            Mock.Of<IParameterNameGeneratorFactory>(),
            Mock.Of<IComparer<ModificationCommand>>(),
            Mock.Of<IKeyValueIndexFactorySource>(),
            context.GetService<ILoggingOptions>(),
            context.GetService<IDiagnosticsLogger<DbLoggerCategory.Update>>(),
            context.GetService<IDbContextOptions>()),
        new[] { new ViewsDiffer() });
}
```

Примеры

```
public class ViewsDifferTests
{
    private readonly ViewsDiffer _differ = new();
}
```

Тестирование

- «Интеграционное» тестирование
  - Изучение исходников EF Core
  - Моки
  - Создание тестового контейнера
  - Создание DbContext
- Тестирование «своих» классов
  - Все элементарно 😊

Теория: итоги

- Думаем о прикладном разработчике
- Пишем код
  - Модель
  - Поведение
  - Тесты

CUSTIS

Практика

CUSTIS

CUSTIS.NetCore.EF.MigrationGenerationExtensions

- Добавляет SQL-объекты в EF Core
  - Вьюхи
  - Синонимы
  - DBlink
  - Хранимки
- SQL-объект = «сырой» SQL

- Делает SQL объекты «родными» для EF Core
  - Отслеживает изменения
  - Генерит UP и DOWN-миграции
  - Обновляет БД различными способами

LiveCoding

- Добавление или изменение SQL-объектов
- Генерация миграций
- Как выглядит миграция
- Как выглядит снепшот
- Обновление БД
- Настройка пакета
  - Runtime
  - DesignTime

Подмена сервисов

- Runtime
  - `(1) IMigrationsModelDiffer` — находит отличия двух моделей
  - `(2) IMigrationsSqlGenerator` — транслирует операции в SQL-код
- DesignTime
  - `(3) ICSharpSnapshotGenerator` — генерирует снепшот
  - `(4) IMigrationsCodeGenerator` — генерирует код миграций
  - `(5) ICSharpMigrationOperationGenerator` — генерирует код операций внутри миграции

Расширение модели

- Неструктурированное
  - Любой объект в виде SQL
- CUSTIS.NetCore.EF.
  MigrationGenerationExtensions

- Структурированное
  - DBlinks
  - Grants
  - Synonyms
  - Views
- Не выпускали в виде NUGET

Гранты

```
/// <summary> Разрешение, выданное одной схемой другой на свой объект </summary>
99+ references | 0 changes | 0 authors, 0 changes
public record DbGrant(string Grantor, string ObjectName, string Grantee, params DbPrivilege[] Privileges);

/// <summary> Настройка грантов </summary>
1 reference | 0 changes | 0 authors, 0 changes
public IEnumerable<DbGrant> GetGrants()
{
    yield return new(Grantor: [redacted]_owner", ObjectName: "[redacted]ware", Grantee: DbSchemaNames.[redacted]Schema, params Privileges: DbPrivilege.Select, DbPrivilege.References);
    yield return new(Grantor: [redacted]_owner", ObjectName: "[redacted]ware", Grantee: DbSchemaNames.[redacted]AccSchema, params Privileges: DbPrivilege.Select, DbPrivilege.References);
    yield return new(Grantor: [redacted]_owner", ObjectName: "[redacted]re_moniker", Grantee: DbSchemaNames.[redacted]Schema, params Privileges: DbPrivilege.Select);
    yield return new(Grantor: [redacted]_owner", ObjectName: "[redacted]_view", Grantee: DbSchemaNames.[redacted]Schema, params Privileges: DbPrivilege.Select);
    yield return new(Grantor: [redacted]_owner", ObjectName: "[redacted]_attr_full", Grantee: DbSchemaNames.[redacted]Schema, params Privileges: DbPrivilege.Select);
    yield return new(Grantor: [redacted]_owner", ObjectName: "[redacted]ITRY", Grantee: DbSchemaNames.[redacted]Schema, params Privileges: DbPrivilege.Select, DbPrivilege.References);
    yield return new(Grantor: [redacted]_owner", ObjectName: "[redacted]E_COLLECTION", Grantee: DbSchemaNames.[redacted]Schema, params Privileges: DbPrivilege.Select);
```

Гранты

```
2 references | 0 changes | 0 authors, 0 changes
protected override void Generate(
    CreateGrantOperation operation,
    MigrationCommandListBuilder builder, InIdempotentWrapper inIdempotentWrapper)
{
    // GRANT SELECT, INSERT, UPDATE, DELETE ON schema.books TO books_admin
    builder
        .Append("GRANT ")
        .Append(operation.Privileges.JoinByComma())
        .Append(" ON ")
        .Append($"{operation.Grantor}.{operation.ObjectName}")
        .Append(" TO ")
        .AppendLine(operation.Grantee)
        .AppendLine(";")
        .EndCommand();
}
```

Гарантия работы одного мигратора

Наследник `Migrator`

```
public override void Migrate(string targetMigration)
{
    var connString = _connectionStrings.GetConnectionString("Admin");
    using var connection = new OracleConnection(connString);
    connection.Open();
    using var scopedTransaction = connection.BeginTransaction();

    LockDb(connection);

    try
    {
        base.Migrate(targetMigration);
        scopedTransaction.Commit();
    }
    catch
    {
        scopedTransaction.Rollback();
        throw;
    }
}
```

Перекомпиляция схем

Наследник `Migrator`

```
protected override IReadOnlyList<MigrationCommand> GenerateUpSql(Migration migration,
    MigrationsSqlGenerationOptions options = MigrationsSqlGenerationOptions.Default)
{
    var commands = base.GenerateUpSql(migration, options);
    return AddRecompile(commands);
}

2 references | 0 changes | 0 authors, 0 changes
private IReadOnlyList<MigrationCommand> AddRecompile(IReadOnlyList<MigrationCommand> commands)
{
    var recompileAcc =
        _rawSqlCommandBuilder.Build(
            sql: RecompileCommandInterceptor.GetRecompileCommand(connectionKey: DbSchemaNames.AccSchema));
    var recompile = _rawSqlCommandBuilder.Build(
        sql: RecompileCommandInterceptor.GetRecompileCommand(connectionKey: DbSchemaNames.Schema));

    return commands.Take(commands.Count - 1)
        .Concat(new[]
        {
            new MigrationCommand(recompileAcc, _currentContext.Context, _commandLogger),
            new MigrationCommand(recompile, _currentContext.Context, _commandLogger),
        })
        .Concat(commands.TakeLast(1)) // Последняя команда обновляет таблицу с историей миграций
        .ToList();
}
```

Перекомпиляция схем

```
internal class RecompileCommandInterceptor : DbCommandInterceptor
{
    private const string RecompilePrefix = "-- DIRTY HACK: RECOMPILE ";
    – references | 0 changes | 0 authors, 0 changes
    public static string GetRecompileCommand(string connectionKey)
    {
        return $"{RecompilePrefix}{connectionKey}";
    }
    – references | 0 changes | 0 authors, 0 changes
    public override async ValueTask<InterceptionResult<int>> NonQueryExecutingAsync(DbCommand command,
        CommandEventData eventData, InterceptionResult<int> result, CancellationToken cancellationToken = default)
    {
        if (IsRecompileCommand(command, out var connectionKey))
        {
            await Recompile(connectionKey, cancellationToken);
            return InterceptionResult<int>.SuppressWithResult(0);
        }
        return result;
    }
    – references | 0 changes | 0 authors, 0 changes
    private static bool IsRecompileCommand(DbCommand command, out string connectionKey)
    {
        var result = command.CommandText.StartsWith(RecompilePrefix);
        connectionKey = result ? command.CommandText.Replace(oldValue: RecompilePrefix, newValue: string.Empty) : string.Empty;
        return result;
    }
```

Подключение к разным схемам

```
/// <summary> Операция, которую необходимо выполнять в соединении, отличном от стандартного </summary>
6 references | 0 changes | 0 authors, 0 changes
public interface IScopedConnectionOperation
{
    /// <summary> Ключ соединения </summary>
    – references | 0 changes | 0 authors, 0 changes
    string ConnectionKey { get; }
}
```

```
protected override void Up(MigrationBuilder migrationBuilder)
{
    using (migrationBuilder.OpenConnection("  WARE"))
    {
        migrationBuilder.CreateGrant(
            grantor: "  WARE",
            objectName: "T_TM",
            grantee: "  ",
            privileges: new[] { "SELECT", "REFERENCES" });
    }

    migrationBuilder.CreateIndex(
        name: "IX_T_                    _TRADE_MARK_ID",
        table: "T_                    _PRIORITY",
        column: "TRADE_MARK_ID");
```

Подключение к разным схемам

```
internal static MigrationConnectionScope OpenConnection(this MigrationBuilder migrationBuilder, string connectionKey)
{
    return new(migrationBuilder, connectionKey);
}

internal sealed class MigrationConnectionScope : IDisposable
{
    private readonly MigrationBuilder _migrationBuilder;

    1 reference | 0 changes | 0 authors, 0 changes
    public MigrationConnectionScope(MigrationBuilder migrationBuilder, string connKey)
    {
        _migrationBuilder = migrationBuilder;
        migrationBuilder.Sql(ScopedConnectionCommandInterceptor.GetConnectSql(connKey));
    }

    0 references | 0 changes | 0 authors, 0 changes
    public void Dispose()
    {
        _migrationBuilder.Sql(ScopedConnectionCommandInterceptor.GetDisconnectSql());
    }
}
```

Подключение к разным схемам

```
private async Task<InterceptionResult<int>> ExecuteNonQuery(DbCommand command,
    InterceptionResult<int> result, CancellationToken cancellationToken)
{
    if (IsConnectToDb(command, out var connectionKey))
    {
        await OpenConnection(connectionKey, cancellationToken);
        return InterceptionResult<int>.SuppressWithResult(0);
    }

    if (IsDisconnectFromDb(command))
    {
        await CloseConnection(cancellationToken);
        return InterceptionResult<int>.SuppressWithResult(0);
    }

    if (_scopedConnection is not null)
    {
        var sql = command.CommandText;
        command.CommandText = "USING SCOPED CONNECTION";

        return await ExecuteNonQueryInScopedConnection(sql, cancellationToken);
    }

    return result;
}
```

Запуск SqlPlus

- EF Core отслеживает изменения в файле SqlPlus
- При создании миграции — создает слепок файла
- При удалении миграции — уничтожает слепок файла
- При накате миграции при необходимости выполняется скрипт SqlPlus
- Реализация — через `DbCommandInterceptor`

```
5 references | 0 changes | 0 authors, 0 changes
public override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
    base.OnConfiguring(optionsBuilder);

    var connectionStrings = new ConnectionStrings(_configuration);
    optionsBuilder.AddInterceptors(new ScopedConnectionCommandInterceptor(connectionStrings),
        new SqlPlusCommandInterceptor(_configuration),
        new RecompileCommandInterceptor(connectionStrings));
```

CUSTIS

Что можно расширять?

- Модель
  - Неструктурированно
  - Структурированно
  - Динамически
- Накат миграций
  - Гарантия работы только одного мигратора
  - Перекомпиляция схем
  - Идемпотентность миграций
- ~~Странное использование~~ Верх мастерства
  - Подключение к разным схемам
  - SqlPlus

CUSTIS

Заключение

Выводы

- Плюсы
  - EF Core имеет множество точек расширения
  - Можно реализовать любые задумки
- Минусы
  - Точки расширения плохо документированы
  - Не все просты для использования
  - Сложно писать тесты
  - Расширения следует документировать

Выводы

- Область применения
  - Частота либо критичность процесса
- Альтернативы
  - Скрипты
  - Ручные правки

Ссылки

- Загадочный EF Core, или Как написать свое расширение
- Контейнеры в EF Core
  - Dependency Injection in EF Core 1.1
- EF Core Exensions (добавление сервисов в Runtime-контейнер)
  - Расширенная настройка EF Core
- Про идемпотентность миграций
  - EF Core + Oracle: как сделать миграции идемпотентными
- Пакет CUSTIS.NetCore.EF.MigrationGenerationExtensions
  - GitHub / Nuget
- Пакет CUSTIS.OracleIdempotentSqlGenerator
  - GitHub / Nuget

CUSTIS® ИТ-РЕШЕНИЯ ДЛЯ РАЗВИТИЯ

Спасибо за внимание!

Игорь Шаталкин

ishatalkin@custis.ru

@ishatalkin