

第23期-配置迁移操作

2019年5月19日 15:18

多个提供程序的迁移

EF Core 工具仅为当前活动的提供程序创建迁移文件，有时，你可能需要分别为多个提供程序创建迁移，有两种方法来处理这种情况：1、维护两个不同版本的迁移集；2、合并到单个迁移集，针对不同提供程序个性化设置。

维护两个不同版本的迁移集

```
Add-Migration InitialCreate -Context MyDbContext -OutputDir Migrations
\SqlServerMigrations
Add-Migration InitialCreate -Context MySqliteDatabaseContext -OutputDir Migrations
\SqliteMigrations
```

```
dotnet ef migrations add InitialCreate --context MyDbContext --output-dir
Migrations/SqlServerMigrations
dotnet ef migrations add InitialCreate --context MySqliteDatabaseContext --output-dir
Migrations/SqliteMigrations
```

合并到单个迁移集，针对不同提供程序个性化设置

```
Id = table.Column<int>(nullable: false)
    .Annotation("SqlServer:ValueGenerationStrategy",
        SqlServerValueGenerationStrategy.IdentityColumn)
    .Annotation("Sqlite:Autoincrement", true),

if (migrationBuilder.ActiveProvider == "Microsoft.EntityFrameworkCore.SqlServer")
{
    migrationBuilder.CreateSequence( name: "EntityFrameworkHiLoSequence");
}
```

自定义迁移历史记录表

默认情况下，EF Core 将已应用到数据库的 (update-database) 迁移，记录在 __EFMigrationsHistory 迁移历史记录表中，由于各种原因，你可能想要自定义此表，以更好地满足你的需求。

更改架构和表名

```
options.UseSqlServer(connectionString, x =>
x.MigrationsHistoryTable("__MyMigrationsHistory", "mySchema"));
```

其它更详细的配置

若要配置的表的其它方面，需要重写并替换 `IHistoryRepository` 服务。

```
class MyHistoryRepository : SqlServerHistoryRepository
{
    public MyHistoryRepository(HistoryRepositoryDependencies dependencies)
        : base(dependencies)
    {
    }

    protected override void ConfigureTable(EntityTypeBuilder<HistoryRow>
history)
    {
        base.ConfigureTable(history);

        history.Property(h => h.MigrationId).HasColumnName("Id");
    }
}

options .UseSqlServer(connectionString).ReplaceService<IHistoryRepository,
MyHistoryRepository>();
```

创建和删除API

```
// Drop the database if it exists
dbContext.Database.EnsureDeleted();

// Create the database if it doesn't exist
dbContext.Database.EnsureCreated();
```

微软同时提供了这些方法的异步版本。

若要使用的 `EnsureCreated` SQL，您可以使用 `GenerateCreateScript` 方法。

```
var sql = dbContext.Database.GenerateCreateScript();
```

仅当数据库中不存在任何表时，`EnsureCreated` 才会起作用。如果有必要，也可以编写自己的检查逻辑，来查看架构是否需要初始化，可通过实现 `IRelationalDatabaseCreator` 接口来初始化架构。

```
// TODO: Check whether the schema needs to be initialized
// Initialize the schema for this DbContext
var databaseCreator = dbContext.GetService<IRelationalDatabaseCreator>();
```

```
databaseCreator.CreateTables();
```

在生产平台中迁移

```
myDbContext.Database.EnsureCreated()
```

```
myDbContext.Database.Migrate(); 推荐
```

使用 SQL 脚本