

Namespace Wolfgang.DbContextBuilderCore

Classes

[DbContextBuilder<T>](#)

Uses the Builder pattern to create instances of DbContext types seeded with specified data.

[SqliteForSqlServerModelCustomizer](#)

An implementation of SqliteModelCustomizer specifically configured to handle a DbContext designed for Microsoft SQL Server databases.

[SqliteModelCustomizer](#)

Overrides the default model creation process in the DbContext{T} with configurations suitable for SQLite.

Interfaces

[IGenerateRandomEntities](#)

Provides an API to generate random entities for seeding databases.

Class DbContextBuilder<T>

Namespace: [Wolfgang.DbContextBuilderCore](#)

Assembly: Wolfgang.DbContextBuilder-Core.dll

Uses the Builder pattern to create instances of DbContext types seeded with specified data.

```
public class DbContextBuilder<T> where T : DbContext
```








Type Parameters

T

Inheritance

[object](#)  ← DbContextBuilder<T>

Inherited Members

[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  , [object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  , [object.ToString\(\)](#) 

Methods

BuildAsync()

Creates a new instance of T seeded with specified data."/>.

```
public Task<T> BuildAsync()
```

Returns

[Task](#)  <T>

instance of {T}

Exceptions

[NotSupportedException](#) 

The specified database provider is not supported

SeedWithRandom<TEntity>(int)

Populates the specified DbSet with random entities of type TEntity.

```
public DbContextBuilder<T> SeedWithRandom<TEntity>(int count) where TEntity : class
```

Parameters

count [int](#)

The number of items to create

Returns

[DbContextBuilder<T>](#)

[DbContextBuilder<T>](#)

Type Parameters

TEntity

The type of entity to create

Exceptions

[ArgumentOutOfRangeException](#)

count is less than 1

SeedWithRandom<TEntity>(int, Func<TEntity, int, TEntity>)

Populates the specified DbSet with random entities of type TEntity.

```
public DbContextBuilder<T> SeedWithRandom<TEntity>(int count, Func<TEntity, int, TEntity>  
func) where TEntity : class
```

Parameters

count [int](#)

The number of items to create

func [Func](#) <TEntity, [int](#), TEntity>

A function that takes a TEntity and the index number of the entity and returns an updated TEntity

Returns

[DbContextBuilder](#) <T>

[DbContextBuilder](#) <T>

Type Parameters

TEntity

The type of entity to create

Exceptions

[ArgumentOutOfRangeException](#)

count is less than 1

SeedWithRandom<TEntity>(int, Func<TEntity, TEntity>)

Populates the specified DbSet with random entities of type TEntity.

```
public DbContextBuilder<T> SeedWithRandom<TEntity>(int count, Func<TEntity, TEntity> func)
where TEntity : class
```

Parameters

count [int](#)

The number of items to create

func [Func](#) <TEntity, TEntity>

A function that takes a TEntity and returns an updated TEntity

Returns

[DbContextBuilder<T>](#)

[DbContextBuilder<T>](#)

Type Parameters

TEntity

The type of entity to create

Exceptions

[ArgumentOutOfRangeException](#) 

count is less than 1

SeedWith<TEntity>(IEnumerable<TEntity>)

Populates the specified DbSet with the provided entities.

```
public DbContextBuilder<T> SeedWith<TEntity>(IEnumerable<TEntity> entities) where TEntity
: class
```

Parameters

entities [IEnumerable](#)  <TEntity>

The entities to populate the database with

Returns

[DbContextBuilder<T>](#)

[DbContextBuilder<T>](#)

Type Parameters

TEntity

Exceptions

[ArgumentNullException](#) 

entities is null

SeedWith<TEntity>(params TEntity[])

Populates the specified DbSet with the provided entities.

```
public DbContextBuilder<T> SeedWith<TEntity>(params TEntity[] entities) where TEntity
: class
```

Parameters

entities TEntity[]

The entities to populate the database with

Returns

[DbContextBuilder](#)<T>

[DbContextBuilder](#)<T>

Type Parameters

TEntity

Exceptions

[ArgumentNullException](#) 

entities is null

[ArgumentException](#) 

entities contains a null item

[ArgumentException](#) 

entities contains a string

UseAutoFixture()

Tell DbContextBuilder to use AutoFixture to create random entities.

```
public DbContextBuilder<T> UseAutoFixture()
```

Returns

[DbContextBuilder<T>](#)

[DbContextBuilder<T>](#)

UseCustomRandomEntityGenerator(IGenerateRandomEntities)

Allows the user to specify their own implementation of IGenerateRandomEntities for generating random entities.

```
public DbContextBuilder<T> UseCustomRandomEntityGenerator(IGenerateRandomEntities generator)
```

Parameters

generator [IGenerateRandomEntities](#)

The generator to use

Returns

[DbContextBuilder<T>](#)

[DbContextBuilder<T>](#)

UseDbContextOptionsBuilder(DbContextOptionsBuilder<T>)

Specifies a specific instance of UseDbContextOptionsBuilder to use when creating the DbContext.

```
public DbContextBuilder<T> UseDbContextOptionsBuilder(DbContextOptionsBuilder<T>  
dbContextOptionsBuilder)
```

Parameters

`dbContextOptionsBuilder` [DbContextOptionsBuilder](#) <T>

Returns

[DbContextBuilder](#) <T>

Exceptions

[ArgumentNullException](#)

callback is null

UseInMemory()

Instructs the builder to use InMemory as the database provider.

```
public DbContextBuilder<T> UseInMemory()
```

Returns

[DbContextBuilder](#) <T>

[DbContextBuilder](#) <T>

UseSqlite()

Instructs the builder to use SQLite as the database provider.

```
public DbContextBuilder<T> UseSqlite()
```

Returns

[DbContextBuilder](#) <T>

[DbContextBuilder](#) <T>

UseSqliteForSqlServer()

Configures the builder to use SQLite as the database provider with SQL Server-specific adjustments, such as default value mappings, to better mimic SQL Server behavior for testing or compatibility.

```
public DbContextBuilder<T> UseSqliteForSqlServer()
```

Returns

[DbContextBuilder<T>](#)

[DbContextBuilder<T>](#)

Interface IGenerateRandomEntities

Namespace: [Wolfgang.DbContextBuilderCore](#)

Assembly: Wolfgang.DbContextBuilder-Core.dll

Provides an API to generate random entities for seeding databases.

```
public interface IGenerateRandomEntities
```

Methods

GenerateRandomEntities<TEntity>(int)

Creates the specified number of entities of type TEntity with random data.

```
IEnumerable<TEntity> GenerateRandomEntities<TEntity>(int count) where TEntity : class
```

Parameters

count [int](#)[↗]

The number of entities to create

Returns

[IEnumerable](#)[↗]<TEntity>

An IEnumerable{TEntity}

Type Parameters

TEntity

The type of entity to create

Class SQLiteForMsSqlServerModelCustomizer

Namespace: [Wolfgang.DbContextBuilderCore](#)

Assembly: Wolfgang.DbContextBuilder-Core.dll

An implementation of [SQLiteModelCustomizer](#) specifically configured to handle a [DbContext](#) designed for Microsoft SQL Server databases.

```
public class SQLiteForMsSqlServerModelCustomizer : SQLiteModelCustomizer, IModelCustomizer
```









Inheritance

[object](#)  ← [ModelCustomizer](#)  ← [SQLiteModelCustomizer](#)  ← SQLiteForMsSqlServerModelCustomizer

Implements

[IModelCustomizer](#) 

Inherited Members

[SQLiteModelCustomizer.OverrideTableRenameRenaming](#) , [SQLiteModelCustomizer.DefaultValueMap](#) , [SQLiteModelCustomizer.OverrideDefaultValueHandling](#) , [SQLiteModelCustomizer.Customize\(ModelBuilder, DbContext\)](#) , [ModelCustomizer.Dependencies](#)  , [object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  , [object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  , [object.ToString\(\)](#) 

Remarks

This class includes overrides to map common default value SQL functions from SQL Server to their SQLite equivalents.

Constructors

SQLiteForMsSqlServerModelCustomizer(ModelCustomizerDependencies)

Creates a new instance of the [SQLiteForMsSqlServerModelCustomizer](#) class.

```
public SQLiteForMsSqlServerModelCustomizer(ModelCustomizerDependencies dependencies)
```

Parameters

dependencies [ModelCustomizerDependencies](#)↗

Remarks

This class should not be created directly but rather added to the service collection by calling the `UseSqliteForSqlServerModelCustomizer` method on the `DbContextBuilder` class

Class SQLiteModelCustomizer



Namespace: [Wolfgang.DbContextBuilderCore](#)

Assembly: Wolfgang.DbContextBuilder-Core.dll

Overrides the default model creation process in the DbContext{T} with configurations suitable for SQLite.

```
public class SQLiteModelCustomizer : ModelCustomizer, IModelCustomizer
```

Inheritance

[object](#)  ← [ModelCustomizer](#)  ← SQLiteModelCustomizer









Implements

[IModelCustomizer](#) 

Derived

[SQLiteForMsSqlServerModelCustomizer](#)

Inherited Members

[ModelCustomizer.Dependencies](#)  , [object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  ,
[object.GetHashCode\(\)](#)  , [object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  ,
[object.ReferenceEquals\(object, object\)](#)  , [object.ToString\(\)](#) 

Remarks

Unless the production database you are testing is also SQLite, there will be differences between your database and the context's configuration definition and SQLite's capabilities. This class provides some basic overrides to make your DbContext work in SQLite. This class provides some basic capabilities like,

1. Renaming tables to avoid schema issues since SQLite does not support schemas.
2. Removing computed values for columns since SQLite may not support the same functions.

You can override the functionality provided in this class or if you will be frequently working in the same database engine, you can create your own ModelCustomizer, either from scratch or derived from this one, and override the desired functionality. Once you created you can reuse it over and over again. For example, you want to create a SQLiteForOracleModelCustomizer that has overrides to make an DbContext created for Oracle work in SQLite.

Constructors

SqliteModelCustomizer(ModelCustomizerDependencies)

Overrides the default model creation process in the DbContext{T} with configurations suitable for SQLite.

```
public SqliteModelCustomizer(ModelCustomizerDependencies dependencies)
```

Parameters

dependencies [ModelCustomizerDependencies](#)

Remarks

Unless the production database you are testing is also SQLite, there will be differences between your database and the context's configuration definition and SQLite's capabilities. This class provides some basic overrides to make your DbContext work in SQLite. This class provides some basic capabilities like,

1. Renaming tables to avoid schema issues since SQLite does not support schemas.
2. Removing computed values for columns since SQLite may not support the same functions.

You can override the functionality provided in this class or if you will be frequently working in the same database engine, you can create your own ModelCustomizer, either from scratch or derived from this one, and override the desired functionality. Once you created you can reuse it over and over again. For example, you want to create a SqliteForOracleModelCustomizer that has overrides to make an DbContext created for Oracle work in SQLite.

Properties

DefaultValueMap

A dictionary where the key is the default value as defined in the model and the value is the value to replace it with.

```
public IDictionary<string, string> DefaultValueMap { get; }
```

Property Value

[IDictionary](#) <[string](#), [string](#)>

Remarks

Examples of a default values that may need to be replaced it `getdate()` and `newid()` which would be replaced with `datetime('now')` and `lower(hex(randomblob(16)))` respectively.

OverrideDefaultValueHandling

This method is called for each column that has a default value defined in the model.

```
public Func<string?, string?> OverrideDefaultValueHandling { get; set; }
```

Property Value

[Func](#) <[string](#), [string](#)>

Remarks

The default implementation checks the `DefaultValueMap` dictionary to see if the existing default value is contained in the dictionary and if so, replaces it with the value in the dictionary. If the current default is not in the dictionary, it is left unchanged. You can override this behavior by assigning a custom implementation to this property

Exceptions

[ArgumentNullException](#)

OverrideTableNameRenaming

This method is called for each table in the database to rename the table since SQLite does not support schemas.

```
public Func<(string? SchemaName, string TableName), string> OverrideTableNameRenaming {  
    get; set; }
```

Property Value

[Func](#) <(string [SchemaName](#), string [TableName](#)), string>

Remarks

The default implementation renames the table by prefixing the schema name to the table name. For example, a table named "Person" in schema Personnel would be renamed to "Personnel_Person". If the table doesn't have a schema "dbo" is used as the schema name. You can override this behavior by assigning a custom implementation to this property

Exceptions

[ArgumentNullException](#) 

Methods

Customize(ModelBuilder, DbContext)

Overrides the default model creation process in the DbContext{T} with configurations suitable for SQLite.

```
public override void Customize(ModelBuilder modelBuilder, DbContext context)
```

Parameters

modelBuilder [ModelBuilder](#) 

context [DbContext](#) 