Controlling Database Creation and Schema Changes with Migrations

Julie Lerman
MOST TRUSTED AUTHORITY ON ENTITY FRAMEWORK

@julielerman thedatafarm.com

Module Overview

Overview of EF Core Migrations API

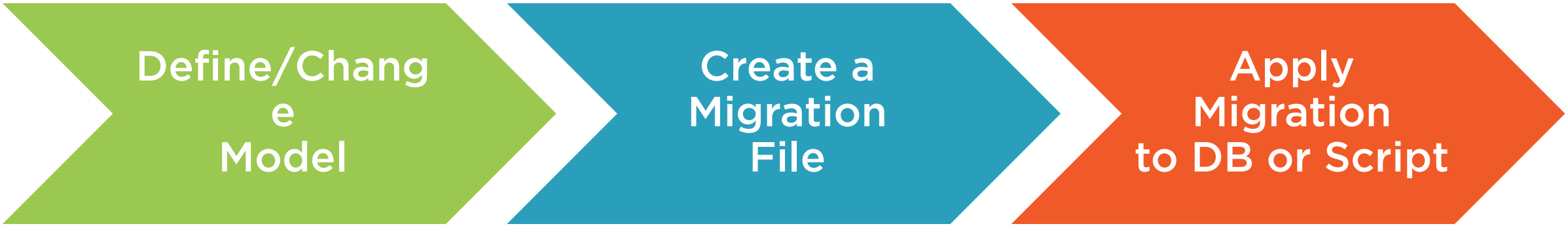
Create and inspect a migration file

Using EF Core Migrations to create a database or database scripts

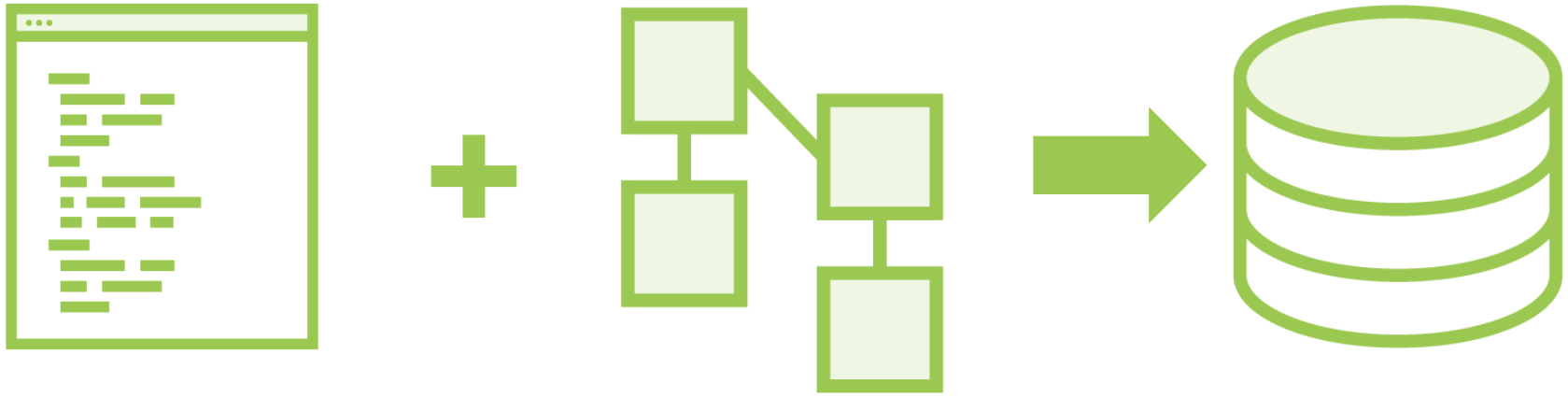
Reverse engineer and existing database into classes and DbContext

Understanding EF Core Migrations

EF Core Basic Migrations Workflow



Mapping Your Data Model to the Database



EF Core Migrations

are

source-control friendly

Adding Your First Migration

NuGet Packages for Migrations

Migrations Commands

PowerShell in VS: Microsoft.EntityFrameworkCore.Tools

dotnet EF Core CLI: A separate installation

```
dotnet tool install --global dotnet-ef
```

Migrations APIs (Microsoft.EntityFrameworkCore.Design)

PowerShell in VS: installed as a dependency of Tools package

dotnet EF Core CLI: Make an explicit reference in project

Tooling and APIs for EF Core Migrations

	In Visual Studio	At the Command Line
Migrations commands	Add reference to Microsoft.EntityFrameworkCore.Tools to project	Install dotnet-ef tools on your dev machine*
Migrations APIs	Tools package will pull in Microsoft.EntityFrameworkCore.Design	Add reference to Microsoft.EntityFrameworkCore.Design to your project

```
* >dotnet tool install --global dotnet-ef
```

Runtime Needed to Run Migration Commands

What if DbContext is in a class library project?

Class Library
Add Tools package
Default PMC project

Executable Project
Make it startup project
Reference class library
Add Design package

Class Library
Default PMC project

Executable Project
Make it startup project
Reference class library
Add Tools package
(Design comes in via NuGet dependency)

Advanced Setup

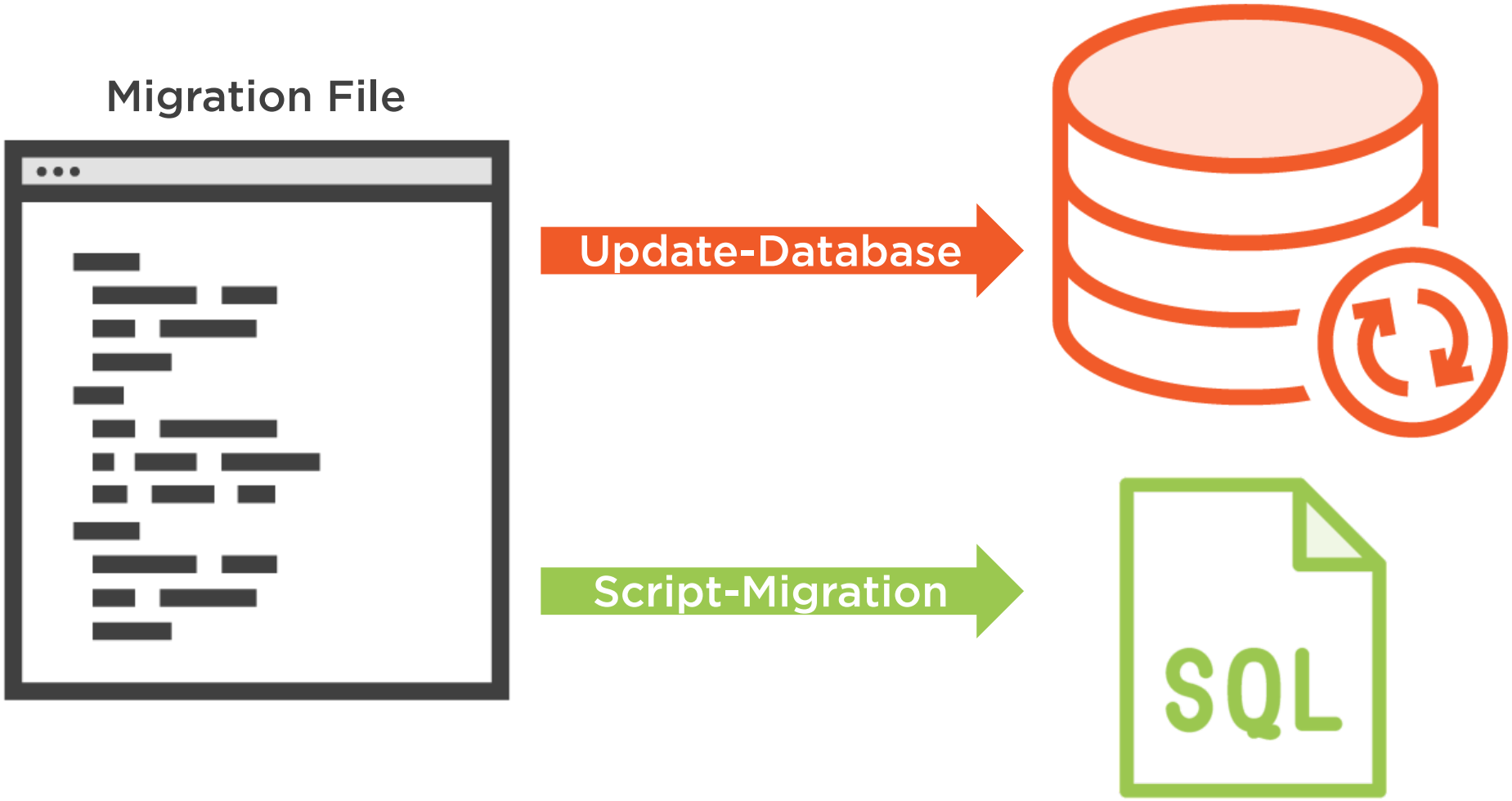
Class Library
Default PMC project
Mods to csproj file
Add Tools package
(Design comes in via NuGet dependency)

Executable Project
Not needed

Inspecting Your First Migration

Using Migrations to Script or Directly Create the Database

Applying Migrations



Migrations Recommendation

Development database

update-database

Production database

script-migration

What If Database Does Not Exist?

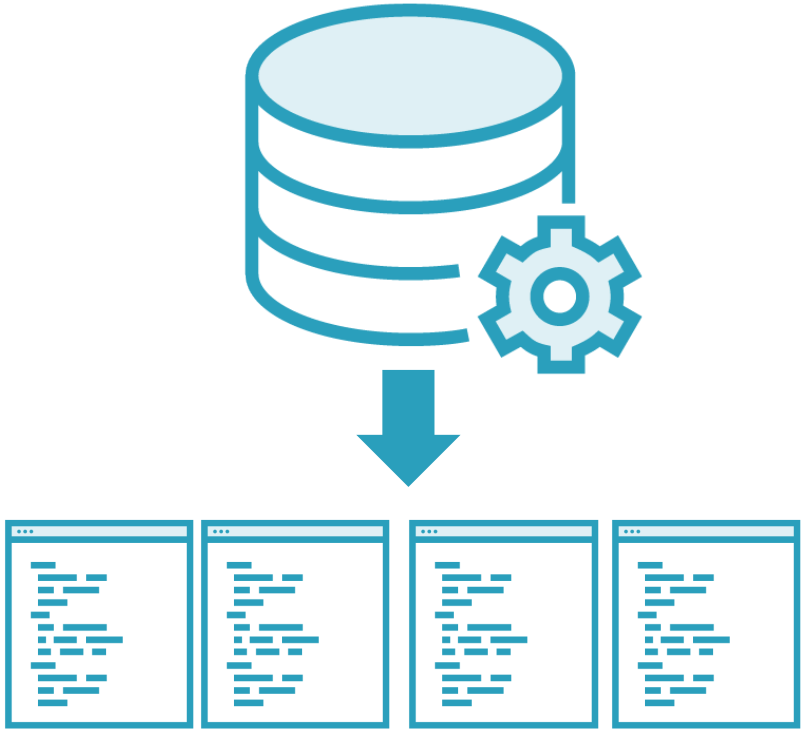
update-database

API's internal code will create the database

script-migration

You must create the database before running the script

Reverse Engineering an Existing Database



Create DbContext & classes from database

Updating model is not currently supported

Transition to migrations is not pretty ... look for helpful link in resources

PowerShell command: scaffold-dbcontext

Reverse Engineer with the Scaffold Command

PowerShell Command

```
Scaffold-DbContext
```

EF CLI

```
dotnet ef dbcontext scaffold
```

The Many Parameters of scaffold-dbcontext

```
-Connection <String>
    The connection string to the database.

-Provider <String>
    The provider to use. (E.g. Microsoft.EntityFrameworkCore.SqlServer)

-OutputDir <String>
    The directory to put files in. Paths are relative to the project
    directory.

-ContextDir <String>
    The directory to put the DbContext file in. Paths are relative to the
    project directory.

-Context <String>
    The name of the DbContext. Defaults to the database name.

-Schemas <String[]>
    The schemas of tables to generate entity types for.

-Tables <String[]>
    The tables to generate entity types for.

-DataAnnotations [<SwitchParameter>]
    Use attributes to configure the model (where possible). If omitted,
    only the fluent API is
```

```
-UseDatabaseNames [<SwitchParameter>]
    Use table and column names directly from the database.

-Force [<SwitchParameter>]
    Overwrite existing files.

-NoOnConfiguring [<SwitchParameter>]
    Don't generate DbContext.OnConfiguring.

-Project <String>
    The project to use.

-StartupProject <String>
    The startup project to use. Defaults to the solution's startup
    project.

-Namespace <String>
    The namespace to use. Matches the directory by default.

-ContextNamespace <String>
    The namespace of the DbContext class. Matches the directory by
    default.

-NoPluralize [<SwitchParameter>]
    Don't use the pluralizer.
```

How EF Core Determines Mappings to DB

Conventions

Default assumptions

```
property name=column name
```

Override with Fluent Mappings

Apply in DbContext using Fluent API

```
modelBuilder.Entity<Quotes>()
  .Property(q => q.Text)
  .HasColumnName("Line");
```

Override with Data Annotations

Apply in entity

```
[Column("Line")]
public string Text{get;set;}
```

Review

- Workflow of how EF Core determines database schema
- Where Migrations API fits in
- PowerShell or CLI commands for creating and executing migrations
- Explored API used in a migrations file
- Used migrations commands to generate script or create a new database directly
- Reverse engineer existing database into code

Resources

Entity Framework Core on GitHub github.com/aspnet/entityframework

EF Core Documentation docs.microsoft.com/ef

EF Core Power Tools Extension (model visualizer and more): github.com/ErikEJ/SqlCeToolbox/wiki/EF-Core-Power-Tools

EF Core migrations with existing database schema cmatskas.com/ef-core-migrations-with-existing-database-schema-and-data

Entity Framework Magic Unicorn (and much more!) is now open source with take backs hanselman.com/blog/entity-framework-magic-unicorn-and-much-more-is-now-open-source-with-take-backs

Controlling Database Creation and Schema Changes with Migrations

Julie Lerman
MOST TRUSTED AUTHORITY ON ENTITY FRAMEWORK

@julielerman thedatafarm.com