Walkthrough 5 - Entity Framework Introduction

Setup

This walkthrough will add database persistence to the MVC movie tracking application.

1. Start SQL Server.
2. Open MovieTracker from the end of the previous walkthrough.

MoviesController.cs

1. Copy the three movies inside the static list. Open Notepad and paste them in. They will be needed later.
2.
```
private static List<Movie> movies = new List<Movie>
{
    new Movie
    {
        Id = 1,
        Title = "Birds of Prey",
        DateSeen = DateTime.Now.AddDays(-50),
        Genre = "Action",
        Rating = 6
    },
    new Movie
    {
        Id = 2,
        Title = "Palm Springs",
        DateSeen = DateTime.Now.AddDays(-25),
        Rating = 7
    },
    new Movie
    {
        Id = 3,
        Title = "Hamilton",
        Genre = "Drama"
    }
};
```

EF Scaffolding

1. Delete MoviesController.cs and the Views / Movies folder.
2. Right-click the project in Solution Explorer and select Manage NuGet Packages..., notice that there is only one package installed.
3. Right-click the Controllers folder and select Add / Controller...
4. Select the MVC Controller with views, using Entity Framework, click Add.
5. Set Model class to **Movie (MovieTracker.Models)**.
6. Click the plus button of Data context class. Accept the default name **MovieTracker.Data.MovieTrackerContext**, click Add.
7. Accept default Controller name **MoviesController**, click Add.
8. Intermittently, the scaffolder will fail and suggest installing Microsoft.VisualStudio.Web.CodeGeneration.Design. This package should already be installed. If this error occurs, select Rebuild Solution from the Build menu. If that doesn't succeed, close the solution and re-open it. If that still doesn't succeed, close Visual Studio and re-open. If that still doesn't solve it, uninstall and reinstall the package.
9. The following things will happen.
    - Microsoft.EntityFrameworkCore.SqlServer package will be installed.
    - Microsoft.EntityFrameworkCore.Tools package will be installed.
    - A folder named Data will be created with the file MovieTrackerContext.cs.
    - MoviesController.cs will be created.
    - Views / Movies folder will be created with the following views.
        - Index.cshtml
        - Details.cshtml
        - Create.cshtml
        - Edit.cshtml
        - Delete.cshtml
    - appsettings.json will be updated with a connection string for the database.
    - The ConfigureServices method of Startup.cs will updated to register the database.

Packages

1. Right-click the MovieTracker project and select Manage NuGet Packages...
2. Notice the two new packages that have been installed.

Startup.cs

1. Open Startup.cs and note the services.AddDbContext line in the ConfigureServices method. This is the dependency injection; also known as registering the database context.

2.
```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();

    services.AddDbContext<MovieTrackerContext>(options =>
        options.UseSqlServer(Configuration.GetConnectionString("MovieTrackerContext")));
}
```

appsettings.json

1. Update the connection string to use SQL Server Express and rename the database to **movie_tracker**.
2.
```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "MovieTrackerContext": "Server=(localdb)\\mssqllocaldblocalhost\\sqlexpress;Database=MovieTrackerContext-a7217986-e755-4fb3-9316-dac758a5bcfdmovie_tracker;Trusted_Connection=True;MultipleActiveResultSets=true"
  }
}
```
3. Save the file.
4. Run the site. Click the Movies link, an error will occur because the database doesn't exist yet.

MovieTrackerContext.cs

1. Notice the DbSet Movie property.
2. Update the constructor to ensure the database is created.
3.
```
public MovieTrackerContext (DbContextOptions<MovieTrackerContext> options)
    : base(options)
{
    Database.EnsureCreated();
}
```
4. Save the file.
5. Run the site. Click the Movies link. Add a movie.
6. Close the browser.
7. Implement the OnModelCreating method to seed the database on first creation. Paste the three movies from Notepad.
8.
```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Movie>().HasData(
        new Movie
        {
            Id = 1,
            Title = "Birds of Prey",
            DateSeen = DateTime.Now.AddDays(-50),
            Genre = "Action",
            Rating = 6
        },
        new Movie
        {
            Id = 2,
            Title = "Palm Springs",
            DateSeen = DateTime.Now.AddDays(-25),
            Rating = 7
        },
        new Movie
        {
            Id = 3,
            Title = "Hamilton",
            Genre = "Drama"
        }
    );
}
```
9. Run the site. Click the Movies link. The seeded movies will not be here; because the database already existed.
10. From the View menu, select SQL Server Object Explorer (SSOE).
11. Expand the SQL Server node, localhost\sqlexpress may already be present. If it isn't, click the Add SQL Server button. Set Server Name to **localhost\sqlexpress**, click Connect.
12. Expand localhost\sqlexpress / Databases and right-click movie_tracker and select Delete. On the delete confirmation dialog, select Close existing connections.
13. Run the site. Click the Movies link, the database will be created again and seeded with these records this time.
14. Close the browser.

Create.cshtml

1. Add an autofocus attribute to the input tag for Title.
2.
```
...
<div class="form-group">
    <label> asp-for="Title" class="control-label"></label>
    <input asp-for="Title" class="form-control" autofocus />
    <span> asp-validation-for="Title" class="text-danger"></span>
```

```
</div>
...
```

3. Save the file.

Edit.cshtml

1. Notice that Id is a hidden field. Add an autofocus attribute to the input tag for Title.
2.
```
...
<input type="hidden" asp-for="Id" />
<div class="form-group">
    <label> asp-for="Title" class="control-label"></label>
    <input asp-for="Title" class="form-control" autofocus />
    <span> asp-validation-for="Title" class="text-danger"></span>
</div>
...
```

3. Save the file.

MoviesController.cs

1. Add a breakpoint (F9) to every public method.
2. Press F5 to run in debug mode. Click the Movies link.

Constructor

1. This is where the database context dependency is requested.
2. Hover over context, expand context / Movie / Results View / [0] to see the first movie record.
3. Press F5 to continue.

Index

1. Note that the Index method just returns the list of movies.
2. Press F5 to continue.
3. Click the Details link of the first movie.

Details

1. Note that the constructor is called again; as it will be each time. Remove the constructor breakpoint (F9).
2. Press F5 to continue.
3. Hover over id, note that it is 1.
4. Press F10 to step through the method inspecting it, until reaching a return.
5. Press F5 to continue.
6. Change the URL to **http://localhost:12345/Movies/Details/10**.
7. Step through the code and continue when reaching a return.
8. Change the URL to **http://localhost:12345/Movies/Details/1a**.
9. Step through the code and continue when reaching a return.
10. Change the URL to **http://localhost:12345/Movies/Details/1**.
11. Step through the code and continue when reaching a return.
12. Click the Edit link.

Edit

1. Step through the code and continue when reaching a return.
2. Change the movie title, click Save.
3. Step through the code to the ModelState check.
4. Hover over ModelState, expand ModelState / Values / Results View / [2] to see new title.
5. Step through the code and continue when reaching a return.
6. Remove the Index breakpoint, continue.
7. Click the Create New link.

Create

1. Step through the code and continue when reaching a return.
2. Enter only a new movie title, click Create.
3. Notice that the ModelState is valid even though the movie only has a title.
4. Step through the code and continue when reaching a return.
5. Click the Delete link of the new movie.

Delete

1. Step through the code and continue when reaching a return.
2. Click Delete
3. Step through the code and continue when reaching a return.
4. Close the browser.
5. End the application by pressing Shift+F5 if necessary.

6. From the Debug menu, select Windows / Breakpoints.
7. Remove all breakpoints.

SQLite

1. Open the Tools menu, select NuGet Package Manager / Pack Manager Console. Issue the command **Install-Package Microsoft.EntityFrameWorkCore.Sqlite**.

Startup.cs

1. In the ConfigureServices method, comment out the existing SQL Server database dependency injection and add an SQLite injection with a different connection string.
2. ```
   public void ConfigureServices(IServiceCollection services)
   {
       services.AddControllersWithViews();

       //services.AddDbContext<MovieTrackerContext>(options =>
       //    options.UseSqlServer(Configuration.GetConnectionString("MovieTrackerContext")));
       services.AddDbContext<MovieTrackerContext>(options =>
           options.UseSqlite(Configuration.GetConnectionString("MovieTrackerContextLite")));
   }
   ```
3. Save the file.

appsettings.json

1. Add a connection string for SQLite.
2. ```
   {
     "Logging": {
       "LogLevel": {
         "Default": "Information",
         "Microsoft": "Warning",
         "Microsoft.Hosting.Lifetime": "Information"
       }
     },
     "AllowedHosts": "*",
     "ConnectionStrings": {
       "MovieTrackerContext": "Server=localhost\\sqlexpress;Database=movie_tracker;Trusted_Connection=True;MultipleActiveResultSets=true",
       "MovieTrackerContextLite": "filename=movie_tracker.db"
     }
   }
   ```
3. Save the file.
4. Run the site and add a movie.
5. Notice in Solution Explorer, there is now a movie_tracker.db file.
6. If you select it, you will notice that it is a binary file. Scroll through it to see recognizable data, but you shouldn't edit it directly.

SQLite/SQL Server Compact Toolbox

1. From the Extensions menu, select Manage Extensions.
2. On the left, select Online.
3. Search for **sqlite**.
4. Select **SQLite/SQL Server Compact Toolbox**, click the Download button.
5. Once the download completes, exit Visual Studio and restart it. Open the project.
6. From the Tools menu, select SQLite/SQL Server Compact Toolbox.
7. Click the Add SQLite and SQL Compact connections from current Solution button.
8. Expand MovieTracker.db / Tables. Right-click Movies and select Edit Top 200 Rows to see the data.

Startup.cs

1. Change the project back to SQL Server by commenting out the SQLite injection in favour of SQL Server in ConfigureServices.
2. ```
   public void ConfigureServices(IServiceCollection services)
   {
       services.AddControllersWithViews();

       //services.AddDbContext<MovieTrackerContext>(options =>
       //    options.UseSqlServer(Configuration.GetConnectionString("MovieTrackerContext")));
       //services.AddDbContext<MovieTrackerContext>(options =>
       //    options.UseSqlite(Configuration.GetConnectionString("MovieTrackerContextLite")));
   }
   ```
3. Save the file.
   ```