Entity Framework

Was ist ein ORM

Ein ORM (Entity Relationship Mapper) ist ein Schnittstelle, die es erlaubt, objektorientiert im Quellcode Datenbank Interaktionen durchzuführen, ohne dabei tatsächliche SQL Statements schreiben zu müssen.
Wir arbeiten in diesem Workshop mit dem von Microsoft empfohlenen Entity Framework. Andere Programmiersprachen benutzen andere ORM Frameworks, die grundsätzliche Funktionsweise sind aber meist sehr ähnlich.
ORMs bestehen meist aus zwei großen Teilen:

- Die Entities, welche die Datenbankstruktur modellieren.
- Die Migrations, welche Änderungen an der Datenbankstruktur darstellen.

Entities

Ein Entity ist ein Modell einer Datenbanktabelle. Es stellt die einzelnen Spalten der Tabelle, sowie Verbindungen zwischen den Tabellen, als Attribute dar.
Es ist unser Weg, im Quellcode mit der Datenbank zu interagieren, indem wir neue Objekte der Entity Klasse erstellen oder bereits bestehende Objekte manipulieren.

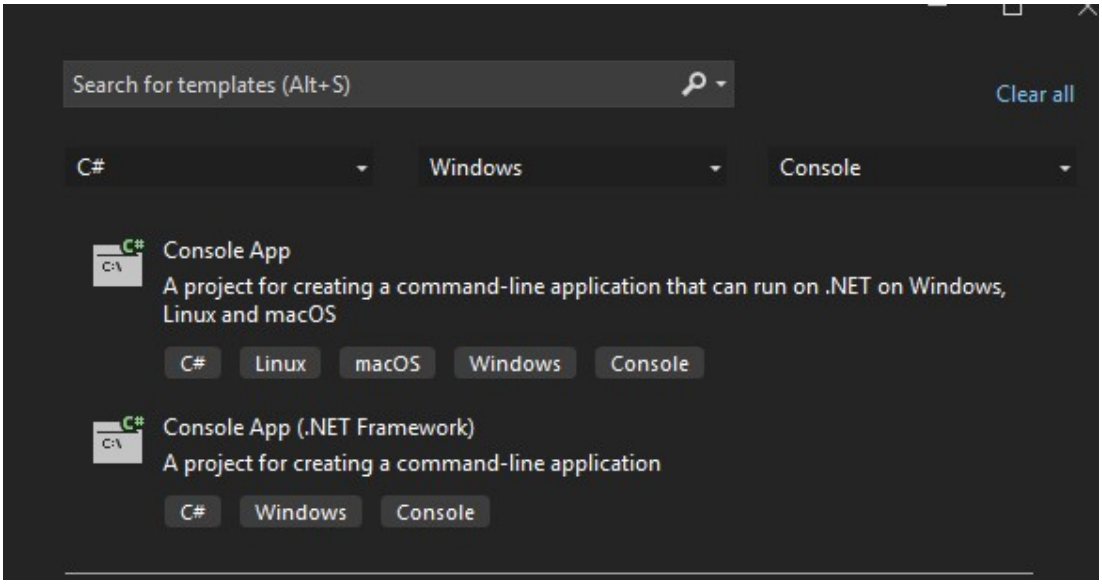
Migrations

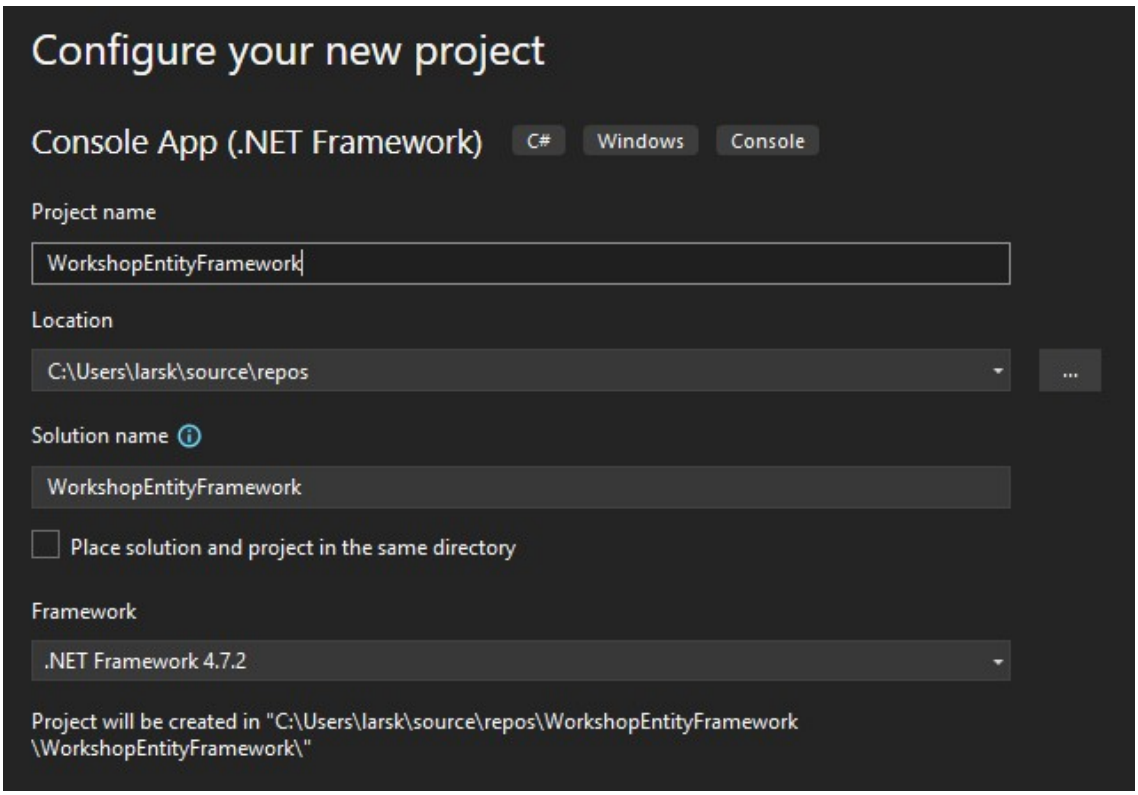
Migrations stellen SQL-Anweisungen in Quellcode dar. Sie werden meist automatisch generiert, indem die bestehende Struktur der Datenbank und der darin enthaltenen Tabellen mit der Struktur der Entities verglichen wird.
Hierbei ist immer das Ziel eine Migration zu generieren, welche die Datenbank auf den Stand der Entities bringt.
Zusätzlich dienen sie als Historie der Datenbankstruktur und sorgen dafür, dass jeder Stand der Datenbankstruktur wiede hergestellt werden kann, da nicht nur SQL Statements erstellt werden um die Datenbankstruktur zu aktualisieren sondern auch SQL Statements um die Änderung in der Migration wieder Rückgängig zu machen.

Einbinden des Entity Framework in ein Projekt

Erstellen eines Projekts

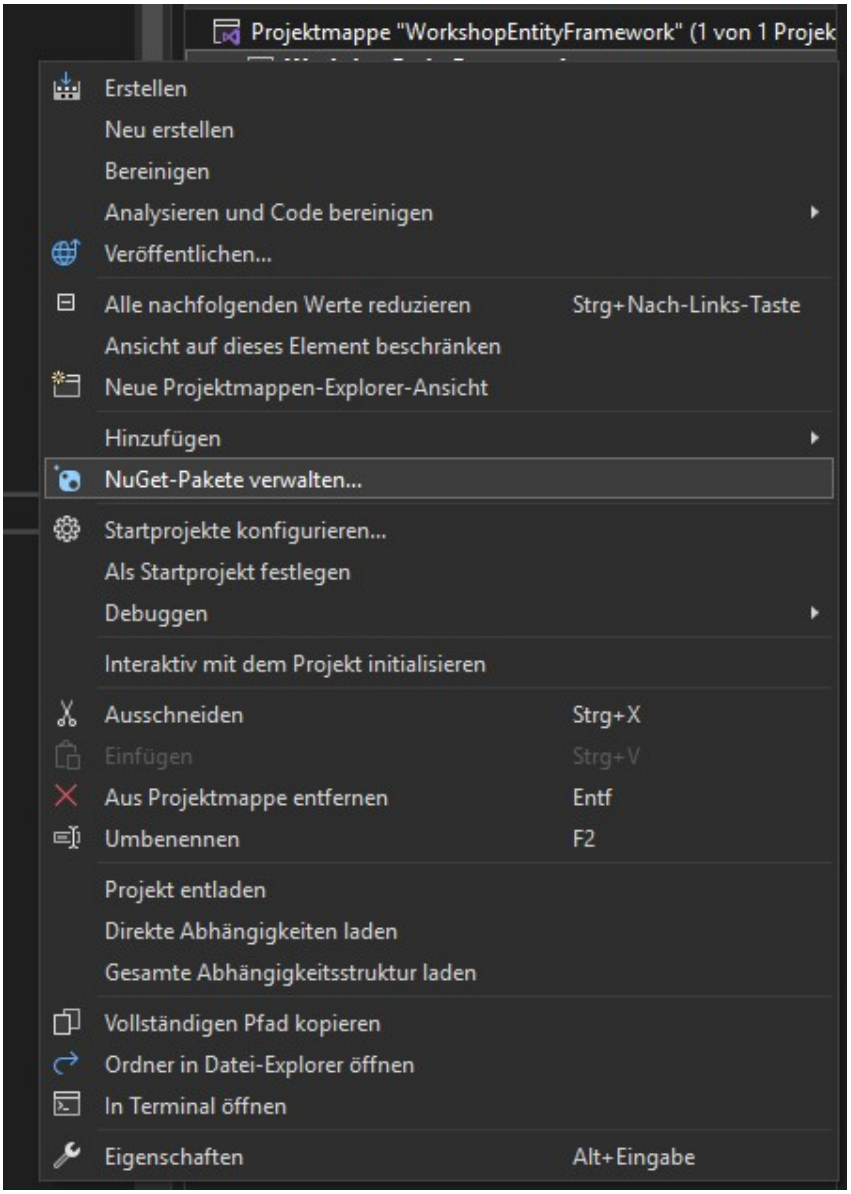
Wie gewohnt erstellen wir eine C# Konsolen Anwendung welche das .NET Framework benutzt:



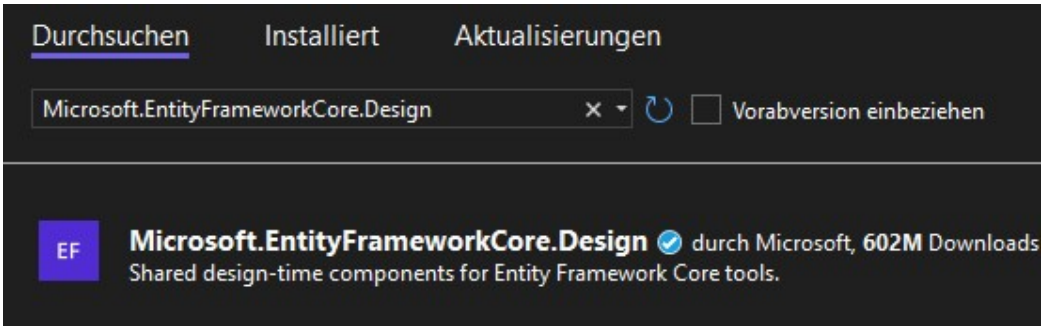


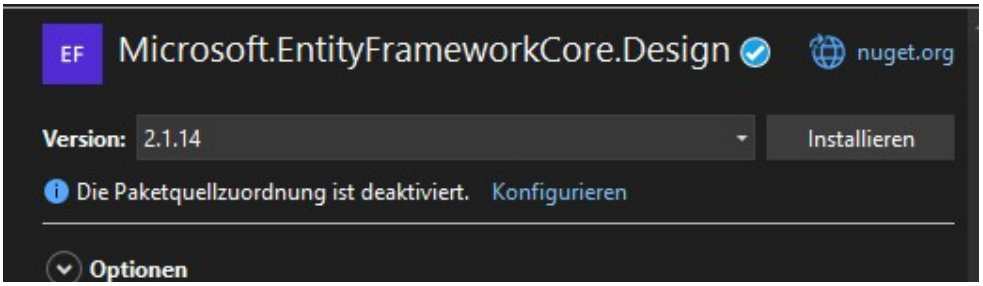
Hinzufügen der Pakete

Sobald das Projekt erstellt wurde, müssen wir die nötigen Entity Framework Pakete installieren. **Achtet dabei auf die Version!**
Rechtsklicke auf das Projekt in der Dateiübersicht und wähle **NuGet-Pakete verwalten**



Im Fenster was sich daraufhin öffnet, suche im Reiter **Durchsuchen** nach **Microsoft.EntityFrameworkCore.Design** und installiere die Version **2.1.14**



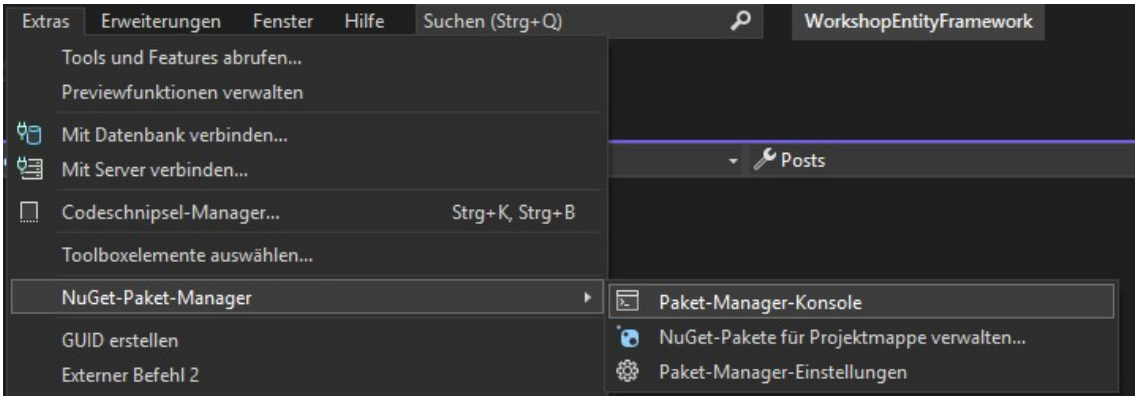


Installiere danach auch die Pakete
Microsoft.EntityFrameworkCore.Sqlite.Core
und **Microsoft.EntityFrameworkCore.Tools** ebenfalls in der Version **2.1.14**

Erstellen der Datenbank

Um die Datenbank zu erstellen, benötigen wir zuerst den DbContext und die Entities. Den Quellcode für beides findest du am Ende dieses Handouts. Kopiere die Code Snippets in dein Projekt. Achte hierbei darauf, dass die Klassen **außerhalb der Program Klasse aber innerhalb deines Namespace sind!**

Sobald die Entities in deinem Projekt sind und der DbContext diese an eine Datenbank Konfiguration bindet, kannst du eine Migration generieren und diese ausführen.
Öffne hierfür die Paket Manager Konsole unter **Extras -> NuGet-Paket-Manager -> Paket-Manager-Konsole**



In dieser Konsole können wir nun die Migration generieren mit dem Befehl

```
Unset
Add-Migration InitialCreate
```

```
PM> Add-Migration InitialCreate
To undo this action, use Remove-Migration.
PM>
```

Und diese dann ausführen mit

```
Unset
Update-Database
```

```
PM> Update-Database
Applying migration '20250317211849_InitialCreate'.
Done.
PM>
```

Testen der Datenbank

Die Datenbank sollte nun komplett funktionsfähig sein. Um dies kurz zu testen, kopiere die Code Snippets für Datenbank Interaktionen und führe diese aus.

Code Snippets

DbContext

```
public class BloggingContext : DbContext
{
     public DbSet<Blog> Blogs { get; set; }
     public DbSet<Post> Posts { get; set; }

     public string DbPath { get; }

     public BloggingContext()
     {
     var folder =
Environment.SpecialFolder.LocalApplicationData;
     var path = Environment.GetFolderPath(folder);
     DbPath = System.IO.Path.Combine(path, "blogging.db");
     }

     protected override void
OnConfiguring(DbContextOptionsBuilder options)
     {
     options.UseSqlite($"Data Source={DbPath}");
     }
}
```

Entities

```
public class Blog
{
     public int BlogId { get; set; }
     public string Url { get; set; }

     public List<Post> Posts { get; } = new List<Post>();
}
```

```
public class Post
{
     public int PostId { get; set; }
     public string Title { get; set; }
     public string Content { get; set; }

     public int BlogId { get; set; }
     public Blog Blog { get; set; }
}
```

Interaktionen mit der Datenbank

```
BloggingContext db = new BloggingContext();

db.Add(new Blog { Url = "https://lms.bbs1-kl.de/" });
db.SaveChanges();

// Create
db.Add(new Blog { Url = "https://lms.bbs1-kl.de/" });
db.SaveChanges();

// Read
Blog blog = db.Blogs
     .OrderBy(b => b.BlogId)
     .First();

var blogs = db.Blogs
     .OrderBy(b => b.BlogId)
     .Where(x => true);

// Update
blog.Url = "https://www.google.com";
blog.Posts.Add(
     new Post { Title = "Hello World", Content = "Entity
Framework Workshop Test!" });
db.SaveChanges();

// Delete
db.Remove(blog);
db.SaveChanges();
```