



# Fundamentos de Desenvolvimento Web com Banco de Dados

## Aula 06: Entity Framework

**Professor:** Rinaldo Ferreira Junior

**E-mail:** [rinaldo.fjunior@prof.infnet.edu.br](mailto:rinaldo.fjunior@prof.infnet.edu.br)

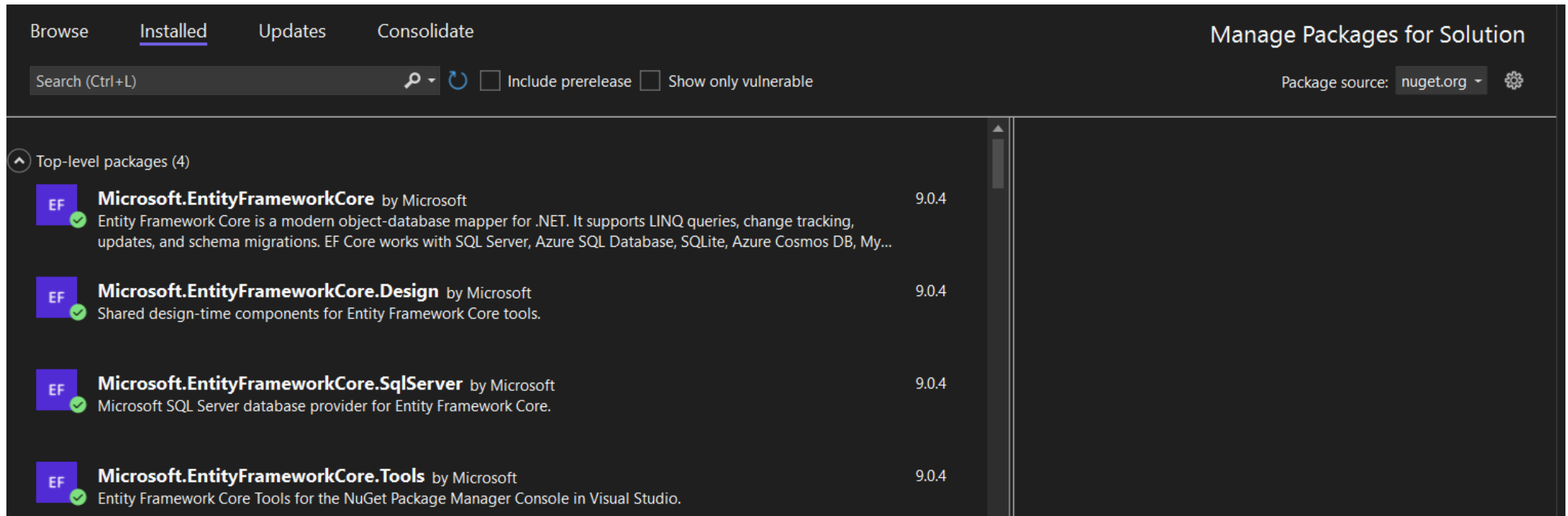


- **Professor:** Rinaldo Ferreira Junior
- **Graduação:** Pós-graduado em Arquitetura de Softwares
- **Atuação:** .Net | C# | SQL | NoSQL | Engenheiro de Software
- **E-mail:** [rinaldo.fjunior@prof.infnet.edu.br](mailto:rinaldo.fjunior@prof.infnet.edu.br)
- **Linkedin:** <https://www.linkedin.com/in/rinaldo-ferreira-junior-787326a>



- Entity Framework


- EF Core é um ORM (object-relational mapper).
- ORMs trabalham fazendo o mapeamento entre bases de dados e as aplicações que manipulam essas bases.
- ORMs permitem ao desenvolvedor, programar o acesso aos dados através da linguagem da aplicação, C# por exemplo, abstraindo o conhecimento em SQL.
- EF Core pode ser trabalhado de duas formas:
  - Code-First: O modelo de dados é criado primeiro em código, e o EF Core gera o esquema baseado no modelo programado.
  - Database-First: Quando a base de dados já existe, o EF Core gera o esquema baseado no esquema da base existente.

- O **EF Core** precisa ser instalado, assim como os providers adequados para o banco de dados que você utiliza..
- O meio ideal de instalação, é o **Nuget Package Manager**.











Browse Installed Updates Consolidate

Search (Ctrl+L)   ☐ Include prerelease ☐ Show only vulnerable

Package source: **nuget.org** 

Manage Packages for Solution

Top-level packages (4)

 	<b>Microsoft.EntityFrameworkCore</b> by Microsoft Entity Framework Core is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations. EF Core works with SQL Server, Azure SQL Database, SQLite, Azure Cosmos DB, My...	9.0.4
 	<b>Microsoft.EntityFrameworkCore.Design</b> by Microsoft Shared design-time components for Entity Framework Core tools.	9.0.4
 	<b>Microsoft.EntityFrameworkCore.SqlServer</b> by Microsoft Microsoft SQL Server database provider for Entity Framework Core.	9.0.4
 	<b>Microsoft.EntityFrameworkCore.Tools</b> by Microsoft Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.	9.0.4

- O acesso às bases se dá por meio de conexões. Elas devem ser estabelecidas de acordo com a base de dados que vai ser usada, e os parâmetros de configuração de acesso.
- O local ideal para estabelecer essas conexões, é o arquivo `appSettings.json`. O .Net já possui um elemento padrão para fazer a leitura das conexões a partir dali.

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "LibraryConnection": "Server=.;Database=Library;Trusted_Connection=True;TrustServerCertificate=True"
  }
}
```

- A conexão aqui é para um banco [SQL Server](#). Como os parâmetros variam de acordo com o banco, consulte a documentação do banco, ou o site: [The Connection Strings Reference](#)

- Para criar a base a partir do código, é preciso criar as classes que se tornarão tabelas e suas características. Deve ser criada também a classe de contexto ([DbContext](#)).

```
public class Book
{
    0 references
    public int BookId { get; set; }
    0 references
    public string Title { get; set; }
    0 references
    public string Description { get; set; }
    0 references
    public int AuthorId { get; set; }
    0 references
    public Author Author { get; set; }
}
```

```
public class Author
{
    0 references
    public int AuthorId { get; set; }
    0 references
    public string Name { get; set; }
    0 references
    public string WebUrl { get; set; }
}
```

```
using Microsoft.EntityFrameworkCore;

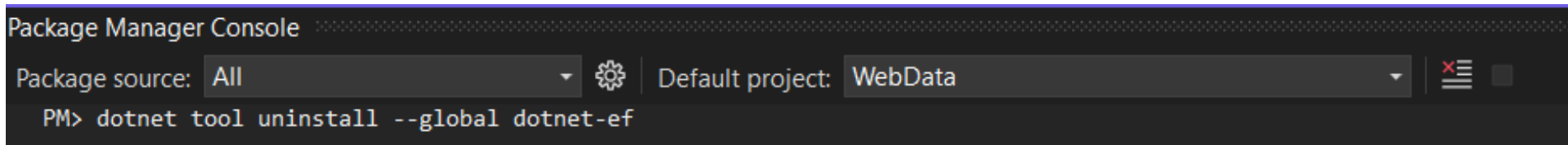
namespace WebData.Models
{
    3 references
    public class LibraryContext : DbContext
    {
        0 references
        public LibraryContext()
        {
        }

        0 references
        public LibraryContext(DbContextOptions<LibraryContext> options) : base(options) { }

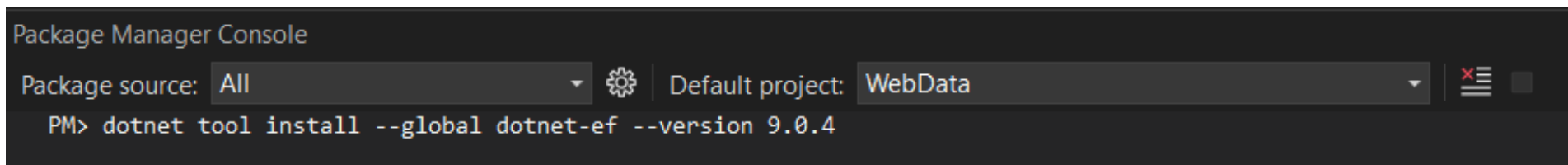
        0 references
        public DbSet<Book> Books { get; set; }
        0 references
        public DbSet<Author> Authors { get; set; }
    }
}
```

O DbContext representa uma sessão aberta com o banco de dados. É através dele, que os comando e consultas são executados.

- Para realizar a criação da base, é usado o recurso chamado [migrations](#). Deve ser executado pelo terminal do VS.
- Clique no menu [Tools](#) -> [NuGet Package Manager](#) e selecione [Package Manager Console](#).
- Para evitar erros sobrenaturais, desinstale o utilitário [dotnet ef](#) e em seguida instale-o com a mesma versão do [Nuget Package Manager](#):



```
Package Manager Console
Package source: All | Default project: WebData
PM> dotnet tool uninstall --global dotnet-ef
```



```
Package Manager Console
Package source: All | Default project: WebData
PM> dotnet tool install --global dotnet-ef --version 9.0.4
```



- Com o `dotnet ef` instalado, é hora de gerar os `migrations`. Os `migrations` são as rotinas de criação/atualização de modelo, executados pelo Entity para alteração da estrutura das bases de dados.

Package Manager Console

Package source: All | Default project: WebData

```
PM> dotnet ef migrations add InitialCreate --project WebData
```

Package Manager Console

Package source: All | Default project: WebData

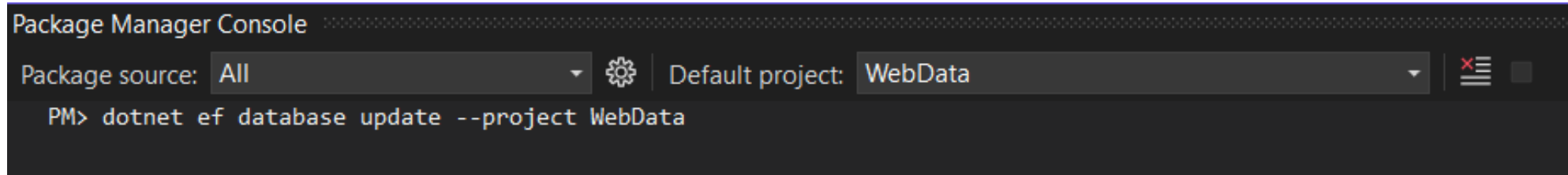
```
PM> dotnet ef migrations add InitialCreate --project WebData
Build started...
Build succeeded.
Done. To undo this action, use 'ef migrations remove'
PM>
```

Solution 'RazorEntity' (1 of 1 project)

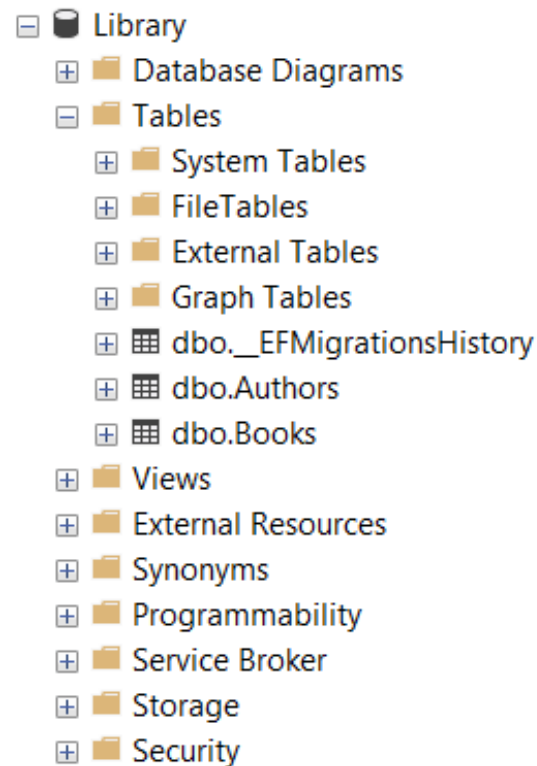
**WebData**

- Connected Services
- Dependencies
- Properties
- wwwroot
- Migrations
  - 20250509182816\_InitialCreate.cs
  - LibraryContextModelSnapshot.cs
- Models
  - Author.cs
  - Book.cs
  - LibraryContext.cs
- Pages
- appsettings.json
- Program.cs

- Os [migrations](#) são gerados, mas não executados imediatamente. É preciso rodar a sua execução.



```
Package Manager Console
Package source: All
Default project: WebData
PM> dotnet ef database update --project WebData
```



- A tabela `__EFMigrationHistory` mantém um histórico das atualizações do modelo, realizadas através de migrations.
- É possível alterar o nome e o esquema dessa tabela, caso você queira por exemplo, usar algum padrão definido por um DBA ou por regras da empresa/cliente.

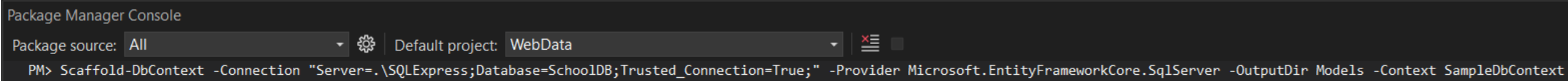
```
builder.Services.AddDbContext<LibraryContext>(options =>
{
    options.UseSqlServer(
        builder.Configuration.GetConnectionString("LibraryConnection"),
        x => x.MigrationsHistoryTable("HistoricoMigrations", "dbo")
    );
});
```

- Em casos de alteração do modelo diretamente na base, o entity pode fazer a leitura reversa e atualizar o modelo no código.

```
Package Manager Console
Package source: All [v] [g] Default project: WebData [v] [x] [m] [■]
PM> dotnet ef dbcontext scaffold "Name=ConnectionStrings:LibraryConnection" Microsoft.EntityFrameworkCore.SqlServer -o Models --force --project WebData
```

- O comando depende de qual SGBD estiver sendo utilizado, e de uma conexão configurada no [appSettings.json](#).

- Para criar um contexto a partir de um banco de dados já existente, execute um **scaffold** da base.



```
Package Manager Console
Package source: All Default project: WebData
PM> Scaffold-DbContext -Connection "Server=.\SQLExpress;Database=SchoolDB;Trusted_Connection=True;" -Provider Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models -Context SampleDbContext
```

- O parâmetro **-Provider** depende do banco de dados sendo utilizado
- O parâmetro **-Context** recebe o nome da classe de contexto que será criada.