Fundamentos de Persistência e ORM

Fundamentos de Persistência e ORM

- Persistir é o processo de armazenar dados de maneira a que, mesmo após a interrupção do serviço de armazenamento ou do processo que o armazenou, o dado ainda existe
- Dados persistidos devem ser estáticos, a não ser que sejam alterados explicitamente
- Recuperáveis mesmo após falhas
- Importantes ao negócio

Fundamentos de Persistência e ORM

- Exemplos de fontes de dados persistentes:
  - **Bancos de dados relacionais:** SQL Server, PostgreSQL, MySQL
  - **Bancos de dados não-relacionais:** Neo4j, MongoDB, Redis, Couchbase, Cassandra
  - **Arquivos em disco**
  - **Containers / buckets de dados:** Amazon S3, Azure Storage, Google Cloud Storage

Fundamentos de Persistência e ORM

- ORM significa “Object-Relational Mapper” e representa uma ferramenta que realiza o “meio de campo” entre uma aplicação que utiliza o paradigma orientado a objetos, e um banco de dados relacional, que utiliza o modelo relacional
  - Aplicações geralmente entendem dados em formato JSON
  - Modelos relacionais trabalham com tabelas e colunas
- No caso do .NET, antigamente precisava-se utilizar o ADO.NET, que é um conjunto de recursos do .NET, para acesso a dados
  - Porém, ele não oferece uma alternativa ágil e simplificada, sendo passível de erros de escrita de nomes de colunas, entre outros problemas
- Diante disso, surgiram os primeiros ORMs, se destacando o Entity Framework e o LINQ-to-SQL
- Com o uso de ORMs, a produtividade é aumentada através de métodos que permite interagir com o BD sem necessitar de código SQL

Até a próxima aula!

Configurando o Ambiente

Configurando o Ambiente

- Vamos utilizar o SQL Server, sendo esta então uma parte importante de configurar
  - Mas não se preocupe se não conseguir, podemos utilizar o banco de dados em memória como será mostrado em breve
- Idealmente, instalar o SQL Express e o SQL Management Studio
  - Para sistemas que não sejam Windows, uma alternativa interessante é o Azure Data Studio
- Para o MacOS com chip M1 / M2 / e afins, é utilizada a imagem Docker do Azure SQL Edge
- É possível executar o SQL Server com Docker
  - docker run --name sqlserver -e "ACCEPT_EULA=Y" -e "MSSQL_SA_PASSWORD=MyPass@word" -p 1433:1433 -d mcr.microsoft.com/mssql/server
- Caso esteja utilizando o MacOS, utilizar a imagem do mcr.microsoft.com/azure-sql-edge

Até a próxima aula!

Fundamentos de Entity Framework Core

Fundamentos de Entity Framework Core

- É um ORM
  - ORM significa “Object-Relational Mapper” e representa uma ferramenta que realiza o “meio de campo” entre uma aplicação que utiliza o paradigma orientado a objetos, e um banco de dados
- Com isso, a produtividade é aumentada através de métodos que permite interagir com o BD sem necessitar de código SQL
- Versão multiplataforma, mais leve e performática do ORM popular Entity Framework
- Suporte a SQL Server, SQLite, PostgreSQL, MySQL, entre outros

Fundamentos de Entity Framework Core

- Principais conceitos
  - **Code First:** metodologia onde primeiro se escreve o código, e com base nele depois se atualiza a estrutura do banco de dados
  - **DbContext:** objeto cuja instância representa uma sessão de acesso ao banco de dados, sendo utilizado para operações
  - **DbSet:** representa a coleção das entidades de um certo tipo que podem ser consultadas a partir do banco de dados
  - **Migrations:** recurso que permite a evolução da estrutura do banco de dados ao longo do tempo e a partir das alterações no código, principalmente as entidades

Para garantir que a ferramenta CLI do EF esteja instalada, execute o comando abaixo

dotnet tool install --global dotnet-ef

Fundamentos de Entity Framework Core

- Algumas configurações de propriedades disponíveis:
  - **IsRequired:** especifica se a propriedade permite ou não permite valores nulos
  - **HasMaxLength:** especifica o tamanho máximo de uma propriedade de cadeia de caracteres
  - **HasColumnType:** especifica o tipo da coluna no banco de dados
  - HasColumnName: especifica o nome da coluna no banco de dados
  - **ToTable:** permite especificar um nome para a tabela

Fundamentos de Entity Framework Core

- Configurando relacionamentos entre entidades
  - Criar uma propriedade que representa a chave estrangeira
  - Criar propriedade de navegação
  - Configurar o relacionamento entre as entidades via Fluent API (preferível utilizar) ou Data Annotation, junto com configuração de chave estrangeira
- Possibilidade de uso de banco de dados em memória
- Apesar disso, conhecer SQL é uma habilidade importante para um Desenvolvedor .NET

Fundamentos de Entity Framework Core

- **Gerando e aplicando Migrations**
  - **dotnet ef migrations add NomeDaMigration**
    - Parâmetros opcionais comuns: -o, -s
  - **dotnet ef database update**
    - Parâmetro opcional comum: -s

Até a próxima aula!