Entity Framework

Profa. Denise Bandeira

Apresentação

Apresentação

- Alunas!!

Por que Precisamos de uma Ferramenta de ORM?

Introdução ao Object-Relational Mapping (ORM) e sua importância no desenvolvimento de aplicações.

O Problema do Mapeamento Manual

- O mapeamento manual requer escrita de SQL dentro do código da aplicação.
- Desvantagens:
  - Complexidade do código
  - Maior chance de erros
  - Dificuldade de manutenção e refatoração
  - Exemplo: Código SQL para operações CRUD.

O Que é um ORM?

- Um ORM é uma ferramenta que converte dados entre sistemas orientados a objetos e bancos de dados relacionais.
- Benefícios:
  - Abstração de consultas SQL
  - Simplificação do código
  - Suporte a mudanças no esquema de banco de dados.

Benefícios de Usar um ORM

- Redução de Código Repetitivo:
  - ORM reduz a necessidade de escrever SQL para cada operação.
- Segurança: Prevenção de injeção de SQL, já que consultas são geradas pelo framework.
- Manutenção Simplificada:
  - Mudanças no modelo afetam apenas as classes do modelo.

Exemplo Prático - CRUD com SQL Puro

- Código de exemplo em SQL puro para adicionar, atualizar, excluir e ler registros de um banco de dados de produtos:

```sql
INSERT INTO Produtos (Nome, Preco) VALUES ('Produto A', 100);
SELECT * FROM Produtos WHERE Preco > 50;
UPDATE Produtos SET Preco = 120 WHERE Nome = 'Produto A';
DELETE FROM Produtos WHERE Nome = 'Produto A';
```

Exemplo Prático - CRUD com ORM (Entity Framework em C#)

- Código equivalente usando Entity Framework:

```
var produto = new Produto { Nome = "Produto A", Preco = 100 };
dbContext.Produtos.Add(produto);
dbContext.SaveChanges();

var produtos = dbContext.Produtos.Where(p => p.Preco > 50).ToList();

var produtoAtual = dbContext.Produtos.FirstOrDefault(p => p.Nome == "Prod
produtoAtual.Preco = 120;
dbContext.SaveChanges();

dbContext.Produtos.Remove(produtoAtual);
dbContext.SaveChanges();
```

Comparação do Exemplo

- Menos linhas de código com ORM.
- Código mais legível e próximo à linguagem da aplicação.
- Melhor suporte à refatoração e manutenção.

Outros Recursos do ORM

- Suporte a Relacionamentos:
  - Mapear relações 1:1, 1:N, N:N entre entidades.
- Migrações de Banco de Dados:
  - Criar, atualizar e reverter o esquema do banco de dados diretamente do código.
- Cache e Otimização de Consultas:
  - ORM pode implementar cache e otimização de queries.

Quando Usar um ORM?

- Ideal para aplicações com interação frequente e complexa com o banco de dados.
- Menos adequado para operações de leitura massiva (ETL), em que consultas SQL otimizadas são preferíveis.

Conclusão

- ORMs não eliminam a necessidade de entender SQL, mas tornam o desenvolvimento mais eficiente e menos propenso a erros.
- Usar um ORM pode acelerar o desenvolvimento de aplicações de médio a grande porte, oferecendo um código mais limpo e de fácil manutenção.