01

Gravando Cadastro no Banco de Dados

Transcrição

Para gravamos o cadastro no banco de dados, precisaremos passar por algumas etapas: primeiramente preencheremos na view de cadastro as informações, depois os dados serão recebidos na action `Resumo`, local em que iremos validar os dados no lado do servidor. Depois que o modelo for validado, chamaremos o método `UpdateCadastro()`, que está em `PedidoRepository.cs`.

O método `UpdateCadastro()` obterá `pedido` a partir do método `GetPedido()`, depois, navegará para `Cadastro` coletará o `Id` para passar para o método `Update()` do repositório.

```
public Pedido UpdateCadastro(Cadastro cadastro)
{
    var pedido = GetPedido();
    cadastroRepository.Update(pedido.Cadastro.Id, cadastro);
    return pedido;
}
```

Como `pedido` saberá qual é o `Id` de `Cadastro`? Primeiramente, faremos uma consulta para averiguar a informação. Navegaremos para `GetPedido()`, em `PedidoRepository.cs` para estudar melhor essa questão.

Obtemos o `pedidoId` que está na sessão utilizando o método `GetPedidoId()`. Uma vez que esta informação for coletava, faremos uma consulta Linq, que procurará `pedidoId` e retornará algumas entidades, como `p.Itens` e `i.Produto`, além do `Id` em si.

```
public Pedido GetPedido()
        {
            var pedidoId = GetPedidoId();
            var pedido = dbSet
                .Include(p => p.Itens)
                    .ThenInclude(i => i.Produto)
                .Where(p => p.Id == pedidoId)
                .SingleOrDefault();

            if (pedido == null)
            {
                pedido = new Pedido();
                dbSet.Add(pedido);
                contexto.SaveChanges();
                SetPedidoId(pedido.Id);
            }

            return pedido;
        }
```

Não especificamos em nenhum momento do código que `Cadastro` também deveria ser consultado. Utilizaremos o método `Include()` para que os dados de `Cadastro`, inclusive o `Id`, sejam verificados.

```
public Pedido GetPedido()
        {
            var pedidoId = GetPedidoId();
            var pedido = dbSet
                .Include(p => p.Itens)
                    .ThenInclude(i => i.Produto)
                                .Inclue(p =>p.Cadastro)
                .Where(p => p.Id == pedidoId)
                .SingleOrDefault();

            if (pedido == null)
            {
                pedido = new Pedido();
                dbSet.Add(pedido);
                contexto.SaveChanges();
                SetPedidoId(pedido.Id);
            }

            return pedido;
        }
```

Começaremos a implementar a gravação de `Cadastro` no banco de dados. Em `CadastroRepository.cs` teremos o método `Updade()`.

```
public Cadastro Update(int cadastroId, Cadastro novoCadastro)
{
    throw new NotImplementedException();
}
```

O primeiro passo é armazenarmos em uma variável local o cadastro proveniente do banco de dados, chamaremos esta variável de `cadastroDB`, que será um objeto de cadastro a partir de `dbSet`. Contudo, devemos realizar um filtro a partir do `cadastroId`. Para isso, escreveremos o método `Where()` com uma expressão lambda.

Em outra linha adicionaremos o método `SingeOrDefault()`, para que seja trazido um cadastro ou nenhum, e caso a informação não seja encontrada, quer dizer que o objeto é nulo e precisamos tratá-lo de alguma maneira.

Inseriremos a condicional `If` para tratarmos do objeto com valor nulo. Declararemos uma exceção `ArgumentNullException()`,que receberá como parâmetro `"cadastro"`

```
public Cadastro Update(int cadastroId, Cadastro novoCadastro)
{
    var cadastroDB = dbSet.Where(c => c.Id == cadastroId)
        .SingleOrDefault();

    if (cadastroDB == null)
    {
        throw new ArgumentNullException("cadastro");
    }
}
```

Prosseguiremos coletando o `novoCadastro`, recebido como parâmetro, para atualizar o banco de dados. Escreveremos ao final do código `cadastroDB.Update()`, e passaremos como parâmetro `novoCadastro`, que possui os novos dados. Em seguida, usaremos o método `SaveChanges()` para gravarmos as informações no banco. Ao final, retornaremos `CadastroDB`.

```
public Cadastro Update(int cadastroId, Cadastro novoCadastro)
{
    var cadastroDB = dbSet.Where(c => c.Id == cadastroId)
        .SingleOrDefault();

    if (cadastroDB == null)
    {
        throw new ArgumentNullException("cadastro");
    }
}

cadastroDB.Update(novoCadastro);
contexto.SaveChanges();
return CadastroDB;
```

Contudo, não possuímos o método `Update()` no modelo de cadastro, isto é, não possuímos um método para atualizar informações a partir de um segundo cadastro. Iremos implementar o método `Update()`: com ele selecionado,pressionaremos o atalho "Ctrl + ." e escolheremos a opção "Gerar método 'Cadastro.Update'".

Dessa forma, o método `Updade()` já foi gerado na classe `Cadastro`, em `modelo.cs`.

```
internal void Update(Cadastro novoCadastro)
{
    throw new NotImplementedException();
}
```

Iremos implementar o método `Update()`. Inseriremos `this.bairro` - a primeira propriedade - como sendo igual a `novoCadastro.Bairro`. Faremos esse mesmo procedimento para todas as outras propriedades.

```
internal void Update(Cadastro novoCadastro)
        {
            this.Bairro = novoCadastro.Bairro;
            this.CEP = novoCadastro.CEP;
            this.Complemento = novoCadastro.Complemento;
            this.Email = novoCadastro.Email;
            this.Endereco = novoCadastro.Endereco;
            this.Municipio = novoCadastro.Municipio;
            this.Nome = novoCadastro.Nome;
            this.Telefone = novoCadastro.Telefone;
            this.UF = novoCadastro.UF;
        }
    }
```

Nas próximas aulas executaremos a aplicação para verificar se conseguimos gravar as informações no banco de dados.