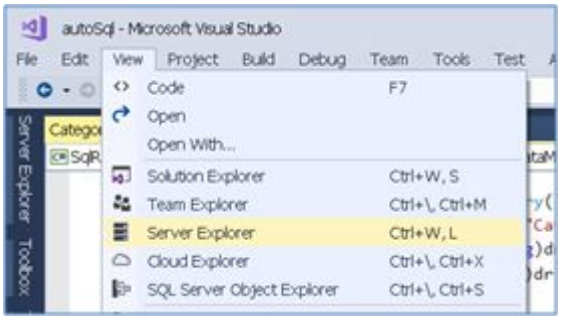
Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores
Ambientes Virtuais de Execução – Semestre de Verão – 2017/2018

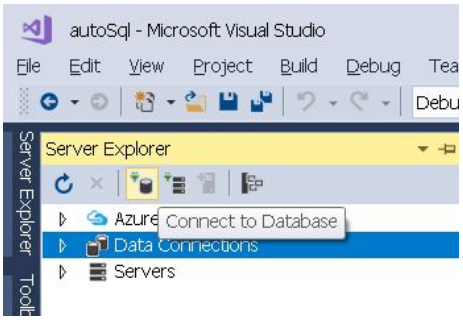
Primeiro Trabalho

Objectivos: Prática com a API de reflexão
Data limite de entrega: 29 de Março de 2018

Este trabalho deve ser desenvolvido usando como base a solução Visual Studio 2017 - `autoSql.sln` - disponibilizada em https://github.com/isel-leic-ave/autosql.
Copie toda a solução **incluindo o ficheiro .gitignore** para o repositório Github do grupo de AVE.
Faça "*Rebuild Solution*" e execute os testes unitários, dos quais 7 devem ter sucesso e outros 7 devem falhar.
Os testes com o sufixo `Reflect` devem corresponder aos 7 testes falhados.

Neste trabalho pretende-se desenvolver a biblioteca **`SqlReflect`** que permite criar instâncias de um *data mapper* (especificado pela interface `IDataMapper`) para qualquer tipo de entidade de domínio.
Como exemplo de utilização será usada a base de dados de SQL Server **Northwind**, que já está incluída nos projectos **App** e **`SqlReflectTest`** da solução `autoSql.sln`.
Pode visualizar o conteúdo da Northwind no VS2017 através da *view* Server Explorer e adicionando uma ligação para o ficheiro da BD SQL Server localizado em: **data\NORTHWND.MDF**.





Um *data mapper* é uma forma de organizar as operações de acesso a dados CRUD (*Create*, *Read*, *Update* e *Delete*) por entidade de domínio (e.g. `Product`, `Employee`, `Region`, etc.). Para cada entidade de domínio existe um *data mapper* cuja classe tem o nome **`<Entidade>DataMapper`** e disponibiliza os métodos correspondentes às operações CRUD.
A interface `IDataMapper`, presente na biblioteca **`SqlReflect`** tem a seguinte definição:

```
public interface IDataMapper
{
    /// <summary>
    /// Returns a domain object with the given id.
    /// </summary>
    object GetById(object id);
    /// <summary>
    /// Returns all rows as domain objects from the corresponding table.
    /// </summary>
    IEnumerable GetAll();
    /// <summary>
    /// Inserts the target domain object into the corresponding table.
    /// </summary>
    /// <returns>The identity value of the primary key column.</returns>
    object Insert(object target);
    /// <summary>
    /// Updates the corresponding table row with the values of the target domain object.
    /// </summary>
    void Update(object target);
    /// <summary>
    /// Removes the row of the table corresponding to the target domain object.
    /// </summary>
    void Delete(object target);
}
```

A título de exemplo o projecto **`SqlReflectTest`** inclui três implementações de *data mappers*: `ProductDataMapper`, `CategoryDataMapper` e `SupplierDataMapper` (este último incompleto). O objectivo da biblioteca **`SqlReflect`** é substituir as várias classes *data mapper* por uma única classe designada de `ReflectDataMapper`.

Assim podem ser obtidas novas instâncias de **`IDataMapper`** para qualquer entidade de domínio **sem ser necessário implementar o código** da classe *data mapper* para essa entidade. Cada instância de `ReflectDataMapper` representa um *data mapper* para o tipo passado no seu construtor. Exemplo:

```
IDataMapper categories =  new ReflectDataMapper(typeof(Category), NORTHWIND)
```

Complete a implementação da biblioteca **`SqlReflect`** seguindo a abordagem proposta:

1. ["*Warm Up*"] Implemente uma classe de domínio e respectivo *data mapper* para uma outra entidade da **Northwind** diferente das já incluídas no projecto de testes unitários (`Product`, `Category` e `Supplier`).
   Defina a nova entidade de domínio sem associações para outras entidades.
   Baseie-se em `CategoryDataMapper` para fazer a sua implementação do *data mapper* para a nova entidade.
   Implemente os testes unitários do novo *data mapper* com base na implementação de `CategoryDataMapperTest`.

2. Implemente `ReflectDataMapper` de modo a satisfazer a criação de *data mappers* para entidades **sem associações** para outras entidades (e.g. `Category`). A tabela correspondente a uma entidade de domínio é especificada por um novo *custom attribute* de classe, `TableAttribute`, e a propriedade correspondente à chave primária por um novo *custom attribute* `PKAttribute`.
   A implementação de `ReflectDataMapper` deve passar os testes unitários `CategoryDataMapperReflectTest` e uma nova versão dos testes unitários da alínea 1) usando uma instância `ReflectDataMapper` para a entidade criada em 1).

3. Adicione a `ReflectDataMapper` suporte para relações entre entidades e *data mappers*. Se uma propriedade de uma entidade é referência para outro tipo complexo (e.g. `Product` tem uma propriedade do tipo `Category`), então o seu *data mapper* depende do *data mapper* da entidade referida (e.g. `ProductDataMapper` depende de `CategoryDataMapper`).
   Por simplificação, ADMITA que não existem referências cíclicas entre entidades.

Tenha cuidados de eficiência na implementação de `ReflectDataMapper` de modo a não repetir a execução de trabalho de reflexão redundante, como seja pesquisa e processamento da informação sobre propriedades para tipos que já foram analisados uma vez.