ISEL
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

1ª Série de Exercícios

Licenciatura em Engenharia Informática e de Computadores

Ambientes Virtuais de Execução

Prof. Eng. José Manuel Simão

Grupo 5:
Rui Lima A42200

Bruno Lourenço A42400

Lisboa, 15 de Abril de 2017

Introdução

Este trabalho tem como objetivo implementar uma biblioteca *MapperReflect* que permita realizar o mapeamento entre membros (campos, propriedades, etc.) de objetos de tipos diferentes (tipos valor ou referência).

Descrição Sucinta

A classe *AutoMapper* contém um campo estático do tipo *Dictionary* que representa a cache e o método *Build* que consoante os tipos recebidos nos parâmetros, verifica se existe um *Mapper* desses tipos na cache, caso não exista cria um *Mapper* novo, adiciona a cache e retorna-o.

A classe *Mapper* contém quatro campos, dois do tipo *Type* que guardam os tipos que vão ser mapeados, um *Mapping* que vai conter os métodos para mapear e um *Dictionary* que guarda se há correspondência de membros com nomes diferentes.

Esta classe tem o método *Bind* que afeta o campo *Mapping* com o objeto do tipo *Mapping* que receber, o método *Match* que vai introduzir no dicionário correspondências de membros com nomes diferentes, e o métodos *Map* que recebe como parâmetro um array de objetos a serem mapeado, que o que faz é chamar para cada objeto, o método *Map* que mapeia apenas um objeto. Este verifica se é possível mapear e cria um objeto do tipo que vai ser mapeado e chama o método *Map* do seu *Mapping* passando como parâmetro, o objeto destino, o objeto origem e o dicionário de correspondências.

Para implementar a hierarquia dos tipos de mapeamento foi criada uma classe *Mapping* que contém três campos, um do tipo *MappingProperties* para representar o mapeamento de propriedades, um do tipo *MappingFields* para representar o mapeamento de campos, outro do tipo *Type* que representa o *custom attribute*.

Esta classe contém dois métodos *Map* que diferem na quantidade de parâmetros que recebem. O método *Map* que não recebe o tipo que representa o atributo como parâmetro possibilita fazer o mapeamento do *custom attribute* ou fazer o mapeamento por definição de propriedades, ou seja se o campo que representa o atributo não estiver com o valor nulo, chama o método *Map,* que recebe como parâmetro o *custom attribute*, de *MappingProperties* e *MappingFields,* senão apenas chama o *Map* de *MappingProperties* . Este é posteriormente redefinido em ambas as classes.

As classes *MappingProperties* e *MappingFields* contém a redefinição dos métodos *Map*, em que o que não recebe o tipo que representa o *custom attribute* chama o método *Map* que recebe, com o valor nulo nesse parâmetro. O método *Map* que recebe o tipo que representa o *custom attribute* contém o código para mapear ou propriedades ou campos consoante o *custom attribute* ou não.

Para os testes foram criadas várias classes para fazer o mapeamento, sendo essas as classes *Student*, *School*, *Person*, *Organization* e *Teacher*.

A seguir apresenta-se um diagrama que ilustra todas as classes que existem na biblioteca.