**Ambiente Virtuais de Execução**

**LI42D – José Simão**

**Relatório da Terceira Série de Exercícios**

****

**Carlos Pereira nº41375**

**João Gameiro nº41893**

**Paulo Alexandre nº40639**

**Objetivo**

A terceira série de exercícios tem como objetivo principal a prática com ***Delegates*** e ***Genéricos***, dando seguimento ao trabalho desenvolvido em ***MapperEmit*** de modo a complementar a API de ***Mapper*** com novos métodos, mantendo todas as funcionalidades já implementadas. As novas funcionalidades serão construídas sobre a estrutura existente.

**Desenvolvimento**

Antes da realização dos exercícios propostos no enunciado, procedemos à correção de erros presente na cache feita anteriormente.

Anteriormente, ao mapearmos um ***Student*** com uma ***Person***, verificávamos se o objeto a ser mapeado já estava presente na cache, tal procedimento estava incorreto, sendo que alterámos a implementação da cache de modo a guardar a instância de ***IMapper***.

1. Acrescentar um novo método ***Build***, genérico em dois tipos parâmetros, o qual retorna uma instância compatível com a definição da *interface IMapper​<​TSrc​, ​TDest​>​*

Acrescentámos a classe genérica *Mapper* que implementa a *interface* genérica *IMapper<TSrc,TDest>*, também adicionada. Essa classe redefine os métodos declarados na *interface*.

1. A instância obtida de ***AutoMapper.Build()*** deve permitir especificar num método ***Map*** o tipo de coleção destino

Este método terá as mesmas funcionalidades que o método ***Map*** já implementado anteriormente, recebendo o tipo fonte e retornando o tipo destino, posto isto reutilizámos o ***Map*** já existente passando-lhe os dois ***Types*** como parâmetro.

Implementámos um novo método ***Map*** que retorna um ***ICollection*** que tem como argumento tipo o ***Type*** do destino, para mapear cada tipo fonte, teremos que percorrer o respetivo ***array***, chamar o método ***Map*** que recebe apenas um tipo fonte e, por fim, adicionar o tipo destino retornado pelo ***Map*** à instância de ***ICollection***.

1. Adicionar o método ***MapLazy***,a ***IMapper***, que permite mapear sequências de elementos ***lazy***

Este método recebe um ***IEnumerable*** cujo o argumento tipo é o ***Type*** da fonte e retornará um ***IEnumerable*** cujo o argumento tipo é o ***Type*** do destino. Utilizando a instrução ***foreach***, permite-nos mapear os elementos de forma ***lazy***, através a instrução ***yield return***.

1. Possibilidade de passar uma função ***Func​<​R>*** que deve ser usada na inicialização de um determinado membro.

Para a realização deste exercício, criámos um novo dicionário cujo o ***value*** é do tipo ***string*** e a ***key*** é do tipo ***func***.A implementação do método ***For<R>*** consiste em verificar a compatibilidade entre a propriedade com o nome da ***string*** passada como parâmetrocom o retorno da ***Func***, ou seja ***R***,também passada como parâmetro, caso não se verifique a compatibilidade, é lançada uma exceção, neste caso ***MismatchTypeException***.

Aquando o mapeamento, verificamos se o dicionário ***mapFor*** contém elementos, em caso afirmativo, vamos afetar a propriedade do objeto fonte cujo nome é especificado pela ***string*** com o valor retornado pela ***Func***, ficando assim o objeto fonte preparado para ser mapeado.