Instituto Superior Engenharia de Lisboa

Relatório

Série 3

Ambientes Virtuais de Execução

Licenciatura em Engenharia Informática e Computadores

Semestre de Inverno 2016/2017

Lisboa, 15 de Janeiro de 2017

Fernando Miguel Carvalho

Alunos:

João Rodrigues, 35392

Nick Laert, 35466

Ricardo Ramos, 36896

INTRODUÇÃO

Na sequência do trabalho desenvolvido nas séries anteriores, foi pedido para complementar a *API (Application Programming Interface)* de forma a ser possível indicar quais os campos a serem ignorados pela *API,* bem como a utilização de funções para a inicialização dos mesmos. Foi também pedido para fazer várias verificações sobre o tipo de dados utilizado de forma a fornecer robustez à aplicação.

IMPLEMENTAÇÃO

A *API* foi desenvolvida tendo como base as séries anteriores. Assim sendo adicionou-se um novo projecto à solução denominado AutoFixture. Manteve-se o código separado em duas classes, nomeadamente **AutoFixture** e **Fixture**.

Em AutoFixture adicionou-se um método com o nome *Ignore* que retorna a interface *IFixture.* Este método permite à interface, via reflexão, fazer a chamada ao método ignore, passando o nome do field ou property e um tipo.

Na classe Fixture definiram-se novos métodos ou alteraram-se os existentes de forma a suportar novos requisitos. Segue uma breve explicação da forma como cada um dos requisitos foi realizado

Não inicialização das propriedades/campos pelo nome

Definiu-se um método ignore que adiciona a uma lista o nome da propriedade ou campo. Esta lista vai ser consultada aquando a realização da atribuição do valor, quer a propriedades quer a campos, e em caso de o nome existir na lista, a atribuição não é realizada.

Não inicialização através de anotações

Implementou-se o método *Ignore<A> where A : Attribute*. Este método procura as propriedades da classe e verifica o atributo. No caso do atributo existir para essa propriedade, ou seja, a propriedade está anotada com esse tipo, então adiciona à mesma lista do método anterior o nome da propriedade.

Utilização de Funções na inicialização dos campos ou propriedades

Implementou-se um método member que recebe um nome e uma função que retorna <*R*>. De forma a fornecer robustez à aplicação, é feita uma verificação o tipo de retorno da função, sem chamar o método *Invoke*, com o tipo de retorno da propriedade ou campo. Se forem compatíveis, é adicionado ao dicionário o nome da propriedade e a função. No método *New*, é chamado o método *DynamicInvoke* do delegate (função), afectando assim o resultado na propriedade ou campo.

Implementação para propriedades do tipo []R ou IEnumerable<R>

Foi alterada a assinatura do método *T[] Fill* para *IEnumerable<T> Fill*. No método *New* é feito a verificação para saber se o tipo recebido é do tipo *ICollection* ou Array, forma a suportar todo o tipo de Collections.

CONCLUSÃO

Este trabalho serviu para consolidar e aplicar os conceitos dados nas aulas (reflexão, lambdas, delegates e custom attributes).

Foi interessante desenvolver uma API mais fluida e perceber o quanto é importante o uso de Interfaces e genéricos para ter um programa modular e adaptável.

No posto de vista programável, foi complicado perceber como tratar os lambdas, estávamos a ter algumas dificuldades em obter o tipo de retorno do lamba sem chamar o método *Invoke* em runtime, contudo conseguimos saber o tipo de retorno via reflexão.