Lazy Initialization - Padrão de Projeto

Alunos: Ruan Donato Reis
Victor Henrique Fernandes

Data: 14/09/2016

Problema

“ Criação de objetos ou processos, podem ter um custo muito caro, até que realmente seja necessário utilizar determinado recurso.“

Motivação

- Evitar que classes sejam instanciadas a qualquer momento.
- Economizar recursos.

Solução

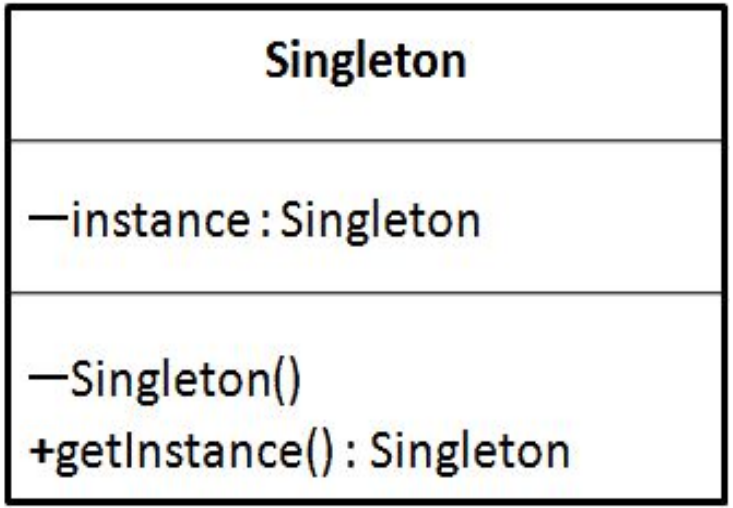
“ Adiar a execução de determinado processo do meu sistema, até que ocorra a primeira chamada ou acesso, através de um padrão de projeto. “

Solução utilizando o singleton

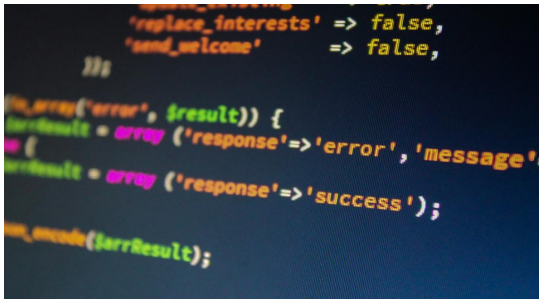
- Singleton: A intenção do pattern singleton consiste em basicamente garantir que uma classe possua somente uma instância durante todo o ciclo de vida de uma aplicação assim como somente um ponto de acesso a essa instância.

- Através de um método de instância, o Singleton usa o Lazy Initialization, fazendo com que o valor que o método retorna não seja criado e armazenado até ser acessado pela primeira vez.

Modelagem


Singleton
—instance : Singleton
—Singleton()
+getInstance() : Singleton

Solução em código



Código 1

In [1]:
```
class Singleton:

    _instance = None

    def __init__(self):
        self.some_attribute = None

    @classmethod
    def instance(cls):
        if cls._instance is None:
            cls._instance = cls()
        return cls._instance
```

In [2]:
```
a = Singleton.instance()
```

In [3]:
```
b = Singleton.instance()
```

In [4]:
```
a is b
```

Código 2

In [15]:
```
import time

def do_some_big_calculation():
    """Simulating some ginormous calculation going on somewhere..."""
    print("{}: Calculations started...".format(time.time))
    time.sleep(5)
    print("{}: Calculations complete!".format(time.time))
    return 42
```

In [16]:
```
import lazy_property

class YetAnotherClass(object):

    @lazy_property.LazyProperty
    def calculation_value(self):

        return do_some_big_calculation()
```

In [17]:
```
yac = YetAnotherClass()
```

In [18]:
```
yac.calculation_value
```

Take home message

Em quais tipos de sistema você utilizaria o Lazy Initialization? Por quê?

De qual forma o Lazy Initialization pode influenciar no desempenho do sistema?

Referências

[1]

http://martinfowler.com/bliki/LazyInitialization.html

[2]

http://python-3-patterns-idioms-test.readthedocs.io/en/latest/Singleton.html

[3]

http://design-patterns-ebook.readthedocs.io/en/latest/creational/singleton/

[4]

https://pypi.python.org/pypi/lazy-property

FIM