02

Fábricas e o problema de criação de objetos

Instanciar objetos pode dar dor de cabeça

Em Python, quando trabalhamos com o paradigma orientado à objetos, precisamos instanciar objetos para que possamos utilizá-los, mas nem sempre esse processo de construção de objetos é tão simples assim. Por exemplo, veja o código abaixo, no qual nosso programa principal captura uma conexão com o banco de dados fazendo uso do *MySQLdb-python connector* para logo em seguida utilizá-la para realizar uma consulta:

```
# -*- coding: utf-8 -*-
import MySQLdb

# cria uma conexão com o banco
# tratamento de erro omitido
connection=MySQLdb.connect(host="localhost", user='root', passwd='',db='alura')

cursor = connection.cursor()

# executa a query
cursor.execute('SELECT * from cursos')

# itera sobre o resultado
for linha in cursor:
    print linha

# fecha a conexão
connection.close()
```

Veja a linha que cria uma conexão: `MySQLdb.connect` . Repare também que só nessa linha temos muitas informações importantes: endereço do banco (localhost), nome do banco (alura), usuário (root) e senha (vazia).

Agora imagine que devemos usar essa linha em todo lugar que precisa de uma conexão com o banco de dados. O que aconteceria se precisássemos trocar, por exemplo, o endereço do banco de dados? Precisaríamos propagar a mudança para todos os pontos que pedem uma conexão. Isso significa um trabalho grande demais, e que com certeza, poderíamos esquecer de alterar alguma parte do nosso código.

Para resolver esse problema, podemos isolar esse processo de criação do banco de dados em uma classe específica, que só faz isso. Por exemplo:

```
# -*- coding: utf-8 -*-
import MySQLdb

class Connection_factory(object):

    def get_connection(self):
        # tratamento de erro omitido
        return MySQLdb.connect(host="localhost",
            user='root',
            passwd='',
            db='alura')
```

Veja que a classe acima agora cria a Conexão e retorna o objeto criado. Agora, todos que quiserem fazer uso de uma conexão devem instanciar uma `Connection_Factory` e invocar o método `get_connection`:

```
# -*- coding: utf-8 -*-
from connection_factory import Connection_factory

# escondeu os detalhes de criação do banco
# tratamento de erro omitido
connection=Connection_factory().get_connection()

cursor = connection.cursor()
cursor.execute('SELECT * from cursos')

for linha in cursor:
    print linha

connection.close()
```

Agora, se precisarmos mudar a string de conexão, basta agora mudar na classe que escrevemos, e a mudança será automaticamente propagada. Muito mais fácil!

Quando precisamos isolar o processo de criação de um objeto, para facilitar a troca dele no futuro, levamos o processo de instanciação dessa classe para uma **Factory**.

Alternativa ao uso de classes

Se nossa fábrica de conexões só existe para chamarmos seu método `get_connection`, que tal transformar nossa fábrica em uma simples função? Vamos alterar `connection_factory.py`:

```
# -*- coding: utf-8 -*-
import MySQLdb

def get_connection():
    # tratamento de erro omitido
    return MySQLdb.connect(host="localhost",
        user='root',
        passwd='',
        db='alura')
```

E claro, alterar `app.py` para usar diretamente a função:

```
# -*- coding: utf-8 -*-
from connection_factory import get_connection

# escondeu os detalhes de criação do banco
# tratamento de erro omitido
connection= get_connection()

cursor = connection.cursor()
cursor.execute('SELECT * from cursos')

for linha in cursor:
```

```
        print linha

    connection.close()
```

Factory Vs Builder

No primeiro curso, tínhamos também um exemplo de um objeto que é difícil de ser criado. Demos o exemplo da classe `NotaFiscal`. Lá, uma nota fiscal era composta por nome da pessoa, ítens da nota, valor e etc. Tudo isso tornava o objeto difícil de ser criado e utilizamos recursos do Python que nos ajudou bastante neste processo de criação. Também vimos como aplicar o padrão Builder na íntegra para resolver o mesmo problema de criação de objetos. Mas qual a diferença entre Factory e Builder, já que o Factory também ajuda na construção? No Builder, ainda estamos no controle da criação do objeto, porém com o auxílio do Builder e podemos construir um objetos de diferentes maneiras. Já o Factory, não participamos do processo de criação do objeto, isto é, já recebemos o objeto pronto!