07

DAOs para acesso a dados

No vídeo, criamos um arquivo `utils` para realizar a conversão dos dados para JSON e para Pickle.

Pensando em uma abordagem mais orientada a objetos, podemos utilizar um objeto de acesso a dados que fica sendo responsável pela comunicação do mundo Python com o mundo dos arquivos. Este objeto é conhecido como **DAO**, ou Objeto de Acesso a Dados (*Data Access Object*, em inglês).

Quem já trabalhou com banco de dados provavelmente conhece esse padrão de persistência. O DAO é um padrão de projeto muito utilizado por quem busca um meio de acessar seus dados. Popularmente, ele é muito utilizado para acessar o banco de dados e realizar as operações de criação, busca, exclusão e atualização. Além disso, ele pode ser utilizado para salvar e recuperar dados em arquivos, por exemplo.

Como podemos definir um DAO no projeto?

No nosso caso, queremos ler os contatos de um CSV e salvá-los em dois formatos, pickle e JSON. Pensando em isolar as responsabilidades, podemos definir três DAOs diferentes, um para cada tipo de arquivo. A fim de manter todos os objetos de acesso a dados com as mesmas assinaturas, podemos ainda definir uma classe abstrata que será estendida pelas classes que implementam de fato o acesso a dados.

Para criar uma classe abstrata, precisamos importar do pacote `abc` a classe `ABC` – que será herdada pela classe que padronizará os DAOs – e o *decorator* `abstractmethod` – que será usado para definir os métodos abstratos que serão implementados pelas classes filhas, portanto:

```
from abc import ABC, abstractmethod
```

A classe que definirá as assinaturas das funções se chamará `ContatoDao` , e sua definição será parecida com esta:

```
class ContatoDao(ABC):

    @abstractmethod
    def buscar_todos(self, caminho):
        pass

    @abstractmethod
    def salvar(self, contatos, caminho):
        pass
```

Agora, basta definirmos as implementações para essa classe, por exemplo, a classe que salva os dados em JSON pode ficar parecida com esta:

```
class ContatoDaoJSON(ContatoDao):

    @abstractmethod
    def buscar_todos(self, caminho):
        contatos = []
        with open(caminho, mode='r') as arquivo:
```

```
            contatos_json = json.load(arquivo)
            for contato in contatos_json:
                        c = Contato(**contato)
                        contatos.append(c)

        return contatos

    @abstractmethod
    def salvar(self, contatos, caminho):
        with open(caminho, mode='w') as arquivo:
    json.dump(contatos, arquivo, default=lambda objeto: objeto.__dict__)
```

De uma forma análoga podemos ter as classes `ContatoDaoPickle` , `ContatoDaoCSV` , `ContatoDaoSQL` e as mais diversas implementações da classe `ContatoDao` , cada uma isolando a lógica de acesso em seu domínio