08

Para saber mais: java.util.Properties

Em projetos mais complexos (que você verá ainda em outros cursos) temos muitas configurações a fazer para nossa aplicação funcionar. Por exemplo, é preciso configurar usuários, senhas, endereços ou portas para acessar outros aplicativos e serviços. Um exemplo clássico é o acesso ao banco de dados, que precisa do login/senha, etc.

Essas configurações podem ficar dentro código fonte, mas isso exige a recompilação do código fonte assim que uma configuração muda. O melhor seria externalizá-las e colocá-las em um arquivo separado, por exemplo um arquivo de texto, que não exige de compilação. Dessa forma só precisamos alterar esse texto, sem mexer no código fonte Java.

Ótimo, e exatamente isso é feito em milhares de projetos Java e para facilitar e padronizar mais ainda, foi criado um mini-padrão para esse tipo de arquivo. Eles são chamados de arquivos de propriedade ou simplesmente ***properties***.

Um arquivo *properties* associa o nome da configuração com o seu valor. Veja o exemplo:

```
login = alura
senha = alurapass
endereco = www.alura.com.br
```

A configuração `login` tem o seu valor `alura`, `senha` tem o valor `alurapass` e assim em diante. Sempre tem uma **chave** (*key*) e um **valor** (*value*) associados. E como isso é tão comum, já foi criado a classe específica `java.util.Properties`, para trabalhar com esses pares de chave/valor, mas claro, poderíamos usar um *Scanner* também!

O uso dessa classe é muito simples. Veja a representação dos valores acima, através de um objeto da classe `Properties`:

```
//import deve ser java.util.Proporties
Properties props = new Properties();
props.setProperty("login", "alura"); //chave, valor
props.setProperty("senha", "alurapass");
props.setProperty("endereco", "www.alura.com.br");
```

Com o objeto criado podemos ver como escrever esses dados no HD.

Escrita de properties

Agora só falta gravar um arquivo para realmente externalizar as configurações. Para tal, você usa o método `store`, da classe `Properties`, que recebe um *stream* ou *writer*, além dos comentários desejados:

```
props.store(new FileWriter("conf.properties"), "algum comentário");
```

Isso cria um arquivo **conf.properties**, com os dados do objeto acima:

```
#algum comentário
#Thu May 10 14:29:38 BRT 2018
senha=alurapass
```

```
login=alura
endereco=www.alura.com.br
```

Leitura de properties

Para ler esse arquivo de *properties*, basta usar o método `load` :

```
Properties props = new Properties();
props.load(new FileReader("conf.properties"));

String login = props.getProperty("login");
String senha = props.getProperty("senha");
String endereco = props.getProperty("endereco");

System.out.println(login + ", " + senha  + ", " +  endereco);
```

Repara que, uma vez lido o arquivo, podemos usar o método `getProperty(key)` , da classe `Properties` , para recuperar o seu valor.