Módulo 19

BackEnd Java

Rodrigo Pires

1 Reflection / Reflexão

O que são?

Reflexão é um recurso da API Java que possibilita aos aplicativos o acesso e a modificação do comportamento de aplicações que estão rodando na Java Virtual Machine. Uma classe pode acessar outras classes em tempo de execução, sem conhecer sua definição no momento da compilação. Informações relativas à esta definição, como seus construtores, métodos e atributos, podem ser facilmente acessados através de métodos de reflexão da API Java.

Classes

```
Class c1 = boolean.class;
System.out.println(c1);

Class c2 = java.io.PrintStream.class;
System.out.println(c2);

Class c = Class.forName("br.com.rpires.reflextions.ReflectionsClasses");
System.out.println(c);

System.out.println(ReflectionsClasses.class);
```

Classes

```
boolean
class java.io.PrintStream
class br.com.rpires.reflextions.ReflectionsClasses
class br.com.rpires.reflextions.ReflectionsClasses
ReflectionsClasses
br.com.rpires.reflextions.ReflectionsClasses
```

Acesso a partir das classes

Membro	Class	Lista dos Membros?	Membros herdados?	Membros privados?
Field	`getDeclaredField()`	Não	Não	Sim
	`getField()`	Não	Sim	Não
	`getDeclaredFields()`	Sim	Não	Sim
	`getFields()`	Sim	Sim	Não
Method	`getDeclaredMethod()`	Não	Não	Sim
	`getMethod()`	Não	Sim	Não
	`getDeclaredMethods()`	Sim	Não	Sim
	`getMethods()`	Sim	Sim	Não
Constructor	`getDeclaredConstructor()`	Não	N/A1	Sim
	`getConstructor()`	Não	N/A[1]	Não
	`getDeclaredConstructors()`	Sim	N/A[1]	Sim
	`getConstructors()`	Sim	N/A[1]	Não

Construtores

```
System.out.println("**** Construtores ****");
Class prodC = ProdutoReflextion.class;
System.out.println(prodC);

Constructor con = prodC.getConstructor();
System.out.println(con);

ProdutoReflextion prod = (ProdutoReflextion) con.newInstance();
System.out.println(prod);

Constructor con1 = prodC.getConstructor(Long.class);
System.out.println(con1);
ProdutoReflextion prod1 = (ProdutoReflextion) con1.newInstance( ...initargs: 10L);
System.out.println(prod1 + " tem o valor: " + prod1.getCodigo());

Constructor[] constructos = prodC.getDeclaredConstructors();
System.out.println("Construtores declarados");
for (Constructor cons : constructos) {
    System.out.println(cons);
}
```

Propriedades

```
System.out.println("**** Propriedades ****");
ProdutoReflextion prod = new ProdutoReflextion();
Field[] fields = prod.getClass().getDeclaredFields();
for (Field field : fields) {
    System.out.println("Nome completo: " + field);
    System.out.println("Nome simples: " + field.getName());
    System.out.println("Tipo da propriedade: " + field.getType());
    System.out.println();
}
```

Métodos

```
System.out.println("**** Métodos ****");
ProdutoReflextion prod = new ProdutoReflextion();
Method[] methods = prod.getClass().getDeclaredMethods();
for (Method m : methods) {
    System.out.println("Nome completo: " + m);
    System.out.println("Nome simples: " + m.getName());
    System.out.println("Tipo de retorno: " + m.getReturnType());
    System.out.println();
}

Method method = prod.getClass().getMethod( name: "getNome");
System.out.println("Pegando método pelo nome: " + method.getName());

Method method1 = prod.getClass().getMethod( name: "setNome", String.class);
System.out.println("Pegando método pelo nome: " + method1.getName());

method1.invoke(prod,  ...args: "Rodrigo");
System.out.println("Pegando valor do Nome: " + method.invoke(prod));
```

Referências

Exemplos disponíveis no meu github:

https://github.com/digaomilleniun/backend-java-ebac