Reflexão

Márcio Saraiva
PED 2016.1 - MC302

17 / 05 / 2016

Contexto

Os **computadores antigos** foram programados em sua **linguagem de máquina nativa**, que era **inerentemente reflexiva**, pois suas arquiteturas só podiam ser programadas através da definição de instruções como dados e utilizando código modificado para elas mesmas.

Como a programação mudou para linguagens de alto nível como C e JAVA, essa “capacidade reflexiva desapareceu”.

Problema / Pergunta

Problema: Algumas aplicações precisam utilizar códigos desconhecidos e tratar ações que só são conhecidas quando a aplicação já está rodando.

Pergunta:

Como verificar e/ou modificar em tempo de execução o comportamento de aplicativos em execução na JVM?

Solução

Em 1982, a tese de doutorado de Brian Cantwell Smith introduziu a noção de **reflexão computacional** em linguagens de programação.

Agenda

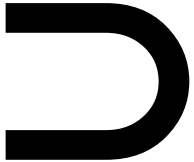
- O que é Reflexão?
- Quando deve ser utilizada?
- Quando NÃO deve ser utilizada?
- Como utilizar? Exemplos

O que é Reflexão?

A reflexão é comumente usada por programas que exigem a capacidade de examinar ou modificar em tempo de execução o comportamento de aplicativos em execução na máquina virtual Java.

Reflexão

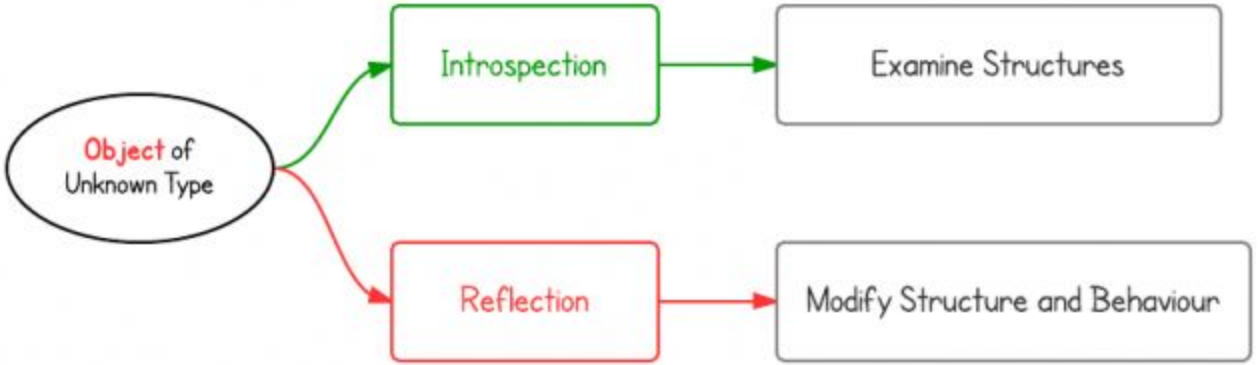
capacidade de um programa para examinar e modificar a estrutura e o comportamento de um objeto em tempo de execução



Introspecção

capacidade de um programa para examinar o tipo ou propriedades de um objecto em tempo de execução

O que é Reflexão?



O que é Reflexão?

Exemplo de Introspecção

```
if(obj instanceof Dog){
   Dog d = (Dog) obj;
   d.bark();
}
```

O que é Reflexão?

Exemplo de Reflexão

```
Class<?> c = Class.forName("classpath.and.classname");
Object dog = c.newInstance();
Method m = c.getDeclaredMethod("bark", new Class<?>[0]);
m.invoke(dog);
```

O que é Reflexão? - Java

Em Java é possível para inspecionar classes, interfaces, campos e métodos em tempo de execução, sem saber os nomes das classes, métodos, etc. em tempo de compilação. Também é possível instanciar novos objetos, chamar métodos e obter / definir valores de campo usando a reflexão.

O que é Reflexão? - Java

- Em Java, a reflexão é mais introspecção, porque você não pode alterar a estrutura de um objeto.

- Há algumas APIs para mudar acessibilidades de métodos e campos, mas não estruturas.

Quando deve ser utilizada?

Reflexão permite-nos:

- Examinar a classe de um objeto em tempo de execução;
- Construir um objeto para uma classe em tempo de execução;
- Examinar campo e método de uma classe em tempo de execução;
- Invocar qualquer método de um objeto em tempo de execução;
- Alterar a acessibilidade do construtor, método e campo;

etc.

Quando deve ser utilizada?

Exemplos

1. **Recursos de extensibilidade**

   Um aplicativo pode fazer uso de classes externas, que necessitam “**aprimoramento**”.

2. **Browsers de classes e ambientes de desenvolvimento Visual**

Uma Classe navegador precisa ser capaz de **enumerar classes**. **Ambientes de desenvolvimento visual podem se beneficiar** ao fazer uso de **informações sobre os tipos** disponíveis para ajudar o desenvolvedor a escrever o código correto.

Quando deve ser utilizada?

Exemplos

3. **Depuradores e ferramentas de teste**

Depuradores precisa ser capaz de examinar as partes privadas de uma classes. Ferramentas de teste podem fazer uso de reflexão para chamar sistematicamente um **conjunto de APIs** definidos em uma classe, para garantir um **alto nível de cobertura** de código em um **conjunto de testes**.

4. **Metaprogramação**

Soluções para desenvolvimento de software; Refatoração.

Quando NÃO deve ser utilizada?

Exemplos

1. **Sobrecarga de desempenho**

Porque reflexão envolve tipos que são resolvidos de forma dinâmica; **determinadas otimizações da JVM podem não ser executadas**. Consequentemente, as operações reflexivas têm um desempenho mais lento do que suas contrapartes não-reflexivas, e deve ser evitado em seções de código que são chamados com frequência em aplicações sensíveis ao desempenho.

2. **Restrições de segurança**

Reflexão **requer permissões** em tempo de execução, **que podem não estar presente quando o código é executado**. Isto é uma consideração importante para o código que tem de ser executado num contexto de segurança restrito, tal como em um Applet.

Quando NÃO deve ser utilizada?

Exemplos

3. Exposição de Internals

Desde reflexão permite que o código para executar **operações que seriam ilegais** em código não-reflexivo, como acessar campos privados e métodos. O uso de reflexão pode resultar em efeitos colaterais inesperados, que podem tornar o **código infuncional**. Código reflexivo **quebra abstrações** e, portanto, pode mudar o comportamento com atualizações da plataforma.

4. Manutenção de código

As **classes e métodos não estão diretamente expostos** no código e podem variar de forma dinâmica. Por exemplo, pode ficar difícil de alterar o número de parâmetros de um método que espera o código de um outro método invocado de forma reflexiva.

Como utilizar? Exemplos

Exemplo 1: Receber o nome da classe de um objeto;

Exemplo 2: Invocar o método de um objeto desconhecido;

Exemplo 3: Criar objetos de uma instância de Classe;

Exemplo 4: Obter construtor e criar instância;

Exemplo 5: Alterar o tamanho de um Array através da reflexão;

Exemplo 6: Acessar atributos privados.

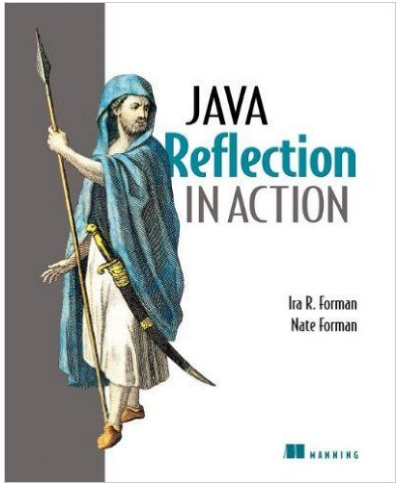
O que vimos hoje?

- O que é Reflexão?
  - Contexto, Problema, Origem, Solução e Significado (≠ Introspecção)
- Quando deve ser utilizada?
- Quando NÃO deve ser utilizada?
- Como utilizar? Exemplos

Material Extra

Livro:

Java Reflection in Action (In Action series) - Ira R. Forman, Nate Forman



Tutoriais:

- http://tutorials.jenkov.com/java-reflection/index.html
- https://www.javacodegeeks.com/2014/11/java-reflection-api-tutorial.html

Reflexão

Márcio Saraiva
marcio.saraiva@ic.unicamp.br