02

A plataforma Java

Transcrição

Antes de mais nada, vamos ver um pouco do que é o Java, o qual te trouxe até aqui: há cerca de vinte anos, quando a linguagem Java nasceu, ela chamava a atenção por conta das seguintes características:

- Orientado a Objeto (O.O.)
- Muitas bibliotecas
- Parece com C++ (hoje em dia isso pode até ser uma desvantagem)
- Roda em vários sistemas operacionais

Você pode estar pensando "poxa, mas a linguagem que uso no dia a dia, atualmente, já possui estas características!". É verdade. É por isto que queremos focar na **plataforma Java**, e não especificamente na linguagem em si, algo que ficará mais claro no decorrer do curso, e até mesmo nesta aula!

A plataforma Java traz:

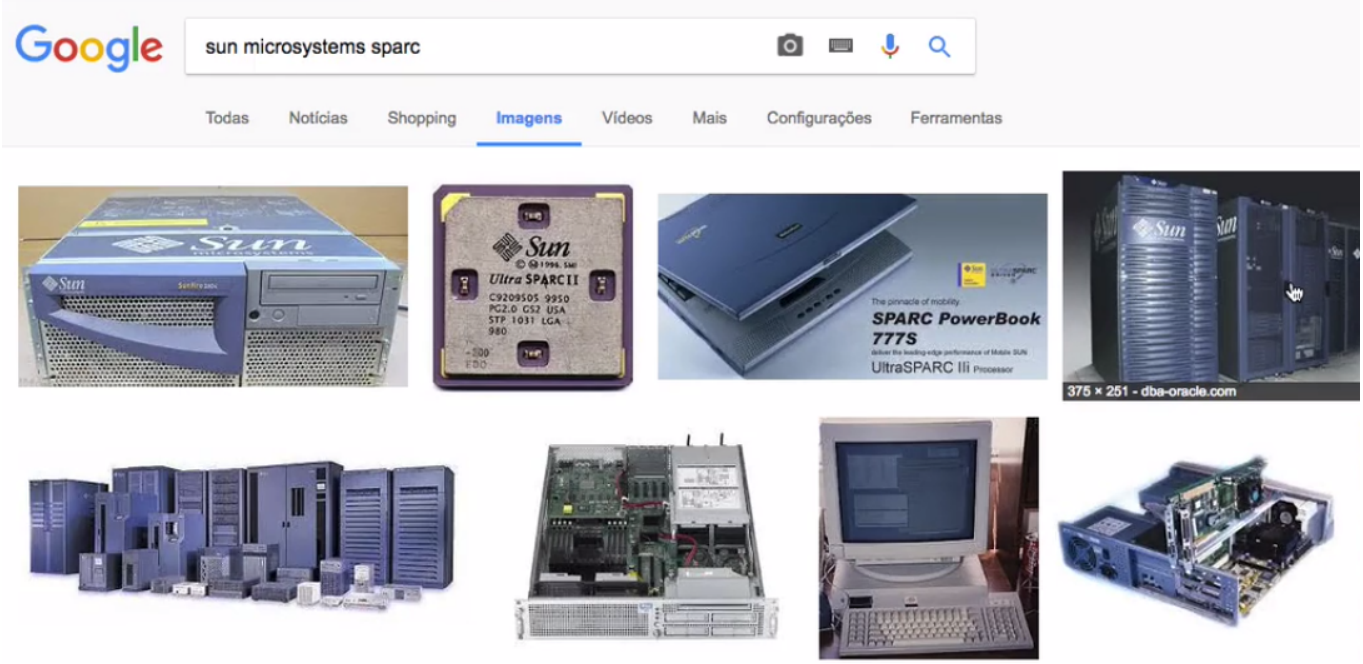
- Portabilidade
- Fácil acesso e desenvolvimento
- Segurança
- Onipresença

> Você pode dar uma olhada no site oficial (https://java.com), porém ele ajuda mais o usuário do Java, do que aqueles que irão compilar e escrever programas.

Falando sobre a história da linguagem: James Gosling é considerado um dos "gênios da computação", sendo considerado o "pai do Java", apesar da linguagem ter sido criada por um grupo, normalmente considerado de quatro pessoas.

Em 1992, o James Gosling trabalhava em uma empresa atualmente inexistente chamada Sun Microsystems (sendo que Sun é acrônimo para *Stanford University Network*), uma dessas *startups* da década de 60, 70, para lidar mais com hardware, que é o que estava dando mais dinheiro.

Eles possuíam um microcomputador, o ***Sun Microsystems SPARC***, que hoje em dia já não aparecem em lugar algum, grandes servidores denominados "micro":



Sendo a Sun uma empresa mais focada em hardware, naquela época, a IBM e a Microsoft começaram a crescer vendendo softwares. Os softwares que a Sun utilizava no sistema deles, o UNIX (o tal de Solaris), eram disponibilizados gratuitamente.

Um dia, esses executivos, dentre os quais o próprio James Gosling, se perguntaram como poderiam lucrar com softwares, já que eles o disponibilizavam de graça, e fizeram um retiro de um mês para tentarem chegar a uma conclusão.

A ideia que eles tiveram envolvia um problema de eletrônicos da década de 90: havia muitos deles sendo criados naquela época, como o VHS que, para quem não sabe, é o videocassete. Era a época de surgimento de TVs, videogames, liquidificadores e geladeira.

Cada um deles possui seu código fonte, necessitando de uma linguagem própria para funcionar, e escrever o código para cada um, reescrevendo-o quando tivessem que passar por uma troca de chip, por exemplo, não fazia muito sentido! A linguagem utilizada neles, ***Assembly***, que hoje em dia é raramente usada, precisava ser reescrito várias vezes, imagine o trabalho!

O James Gosling e sua equipe pensaram em escrever um único código que gerasse um "executável" - entre aspas porque após a compilação ele estará em um formato não exatamente compreensível pelo aparelho em si, mas por um intermediário, no caso, um processador ou uma placa de hardware, para que, aí sim, passe o código aos aparelhos.

Trata-se de algo que realmente simula um computador bem simples e traduz esta linguagem "executável" de acordo com o aparelho em questão. Isto é, esta "máquina de mentira" traduzirá tudo, como se fosse um sistema operacional.

É por isto que surgiu o nome **máquina virtual**, pois veio da ***virtual machine***!

A ideia deles foi, então, criar uma placa pequena, um hardware, que é uma máquina real e compõe todo liquidificador, computador, videocassete, e por aí vai. Desta forma, as pessoas poderão escrever em apenas uma linguagem, que na época se chamava ***Oak*** e depois se tornou Java.

Isso pareceu muito bom, mas acabou fracassando de maneira retumbante, pois era muito caro produzir chips distintos para cada aparelho, cada qual adaptado a uma determinada linguagem.

Então, em 1995, com o *boom* da Web e o surgimento de mais navegadores, como Mosaic, Netscape e posteriormente Internet Explorer, a ideia de máquina virtual foi visualizada como um problema interessante pelo Gosling.

Assim como na atualidade, existia uma variedade relevante de navegadores e sistemas operacionais. E, para escrever um código para Windows, utilizava-se a linguagem no Microsoft Visual Basic, que por sua vez era compilado por um executável

(um EXE, no caso do Windows).

Isto é, ele só funciona neste sistema operacional, com determinadas DLLs na máquina, e assim por diante. O executável e o código fonte ficavam atrelados a uma plataforma específica, um conjunto de sistema operacional, hardware e outros detalhes.

Para tentar resolver este problema, que geraria um código e um executável diferentes para cada sistema operacional existente, o Gosling desengavetou a ideia da máquina de verdade, do chip, que eles haviam criado anteriormente.

Com um código fonte único, teríamos um intermediário que soubesse traduzir ou instruir o sistema operacional acerca dos comandos a serem enviados e recebidos. Este meio de campo seria realizado pela **Máquina Virtual Java (JVM)**, que não é meramente um interpretador por conta de alguns detalhes internos que vão além da interpretação.

O código, então, seria a linguagem Java, e o código "executável", quando compilado, não geraria um `.exe` (pois este seria lido apenas pelo Windows), e sim um formato chamado ***bytecode Java***, de extensão `.class`, lido pela Máquina Virtual Java, que passaria a informação aos sistemas operacionais.

Um exemplo deste formato entendido pela *virtual machine* (JVM), o *bytecode*, é o seguinte:

```
Compiled from "Onibus.java"
class Teste {

    public static void main(java.lang.String);
      Code:
       0: new              #2  // class Onibus
       3: dup
       4: invokespecial    #3 Onibus."<init>":()V
       7: astore_1
       8: aload_1
       9: ldc              #4 // String Jabaquara...
      11: putfield         #5
        // Field Onibus.linha:Ljava/lang/String;
      14: return
}
```

Quem conhece a linguagem de ***Assembly*** talvez identifique a semelhança, mas este código não parece ser de fácil leitura e compreensão. Para meios de comparação, segue um exemplo de um arquivo `.java`, a ser compilado e traduzido para `.class`, o tal do bytecode:

```
public class Onibus {
  String nome;
  String linha;
}

class Teste {
  public static void main(String args) {
    Onibus o = new Onibus();
    o.linha = "Jabaquara-Liberdade";
  }
}
```

Então, em 1995 surgiu o Java, capaz de rodar em vários dispositivos e sistemas operacionais, com foco de criar *applets*, quando ainda tínhamos que instalar o Java para rodá-lo dentro do navegador.

O Java nasceu com um propósito, mas acabou se fortalecendo em ***server-side***, pois quando escrevemos uma aplicação, um site web ou sistema grande, não queremos ficar dependendo de diferentes sistemas operacionais, em implantações e *deploys*.

O Java traz liberdade, quebrando nossa dependência em relação às versões de sistema operacional e navegadores. Empresas grandes, como bancos e o governo, não querem ficar engessados - o que é conhecido por *Vendor lock-in*.

As principais características do conceito de Máquina Virtual Java são:

- Multiplataforma
- Gerenciamento de memória
- Segurança
- Sandbox
- Otimizações
- JIT Compiler

Hoje, mais do que na linguagem Java em si, o enfoque está na plataforma, no **ecossistema Java**! A *virtual machine* é interessante para as empresas pois elas não dependem do que se encontra abaixo da sua *stack*, ou pilha de tecnologia, além do acesso a uma grande variedade de bibliotecas, e as linguagens Java que rodam nesta plataforma.

Não é à toa que há programas que lidam com linguagens Ruby, Clojure ou Scala, por exemplo, e geram o bytecode Java. Depois, basta a Máquina Virtual Java, JVM, trabalhar de acordo com o sistema operacional desejado.