Haxe

Paulo Sendas, Thiago Guerra, Thiago Utsch

April 9, 2022

1 Link Git Hub com o código

https://github.com/thiagoutsch/HaxeCompiladoPython.git

2 Link Vídeo explicativo YouTube

https://youtu.be/3cQgauqJMPE

3 Introdução

Haxe é uma linguagem de programação e compilador de plataforma cruzada de alto nível de código aberto que pode produzir aplicativos e código-fonte para muitas plataformas de computação diferentes a partir de uma base de código. É um software livre e de código aberto, lançado sob a licença MIT.

É uma linguagem baseada na web que provê suporte a C++, C Sharp, SWF/ActionScript, Java e Neko byte code (também desenvolvida pelo autor de Haxe).

4 Aplicações Linguagem Haxe

Haxe é uma linguagem de programação e compilador de plataforma cruzada de alto nível de código aberto que pode produzir aplicativos e código-fonte para muitas plataformas de computação diferentes a partir de uma base de código.

5 Características Linguagem Haxe

Haxe é uma linguagem de programação de plataforma cruzada de alto nível de código aberto e compilador que pode produzir aplicativos e código-fonte, para muitas plataformas de computação diferentes a partir de um código-base. É um software gratuito e de código aberto , lançado sob a licença MIT . O compilador, escrito em OCaml , é lançado sob a GNU General Public License (GPL) versão 2.

Haxe inclui um conjunto de recursos e uma biblioteca padrão com suporte em todas as plataformas , como tipos de dados numéricos , strings , arrays , mapas , binário , reflexão , matemática, http, sistema de arquivos e formatos de arquivo comuns . O Haxe também inclui APIs específicas da plataforma para cada destino do compilador. Kha , OpenFL e Heaps.io são frameworks Haxe populares que permitem a criação de conteúdo multiplataforma a partir de uma base de código.

Haxe originou com a ideia de apoiar do lado do cliente e do lado do servidor de programação em um idioma, e simplificando a lógica comunicação entre eles. O código escrito na linguagem Haxe podem ser compilados em JavaScript , C ++ , Java , JVM , PHP , C , Python , Lua e Node.js . O Haxe também pode compilar diretamente o bytecode SWF , HashLink e Neko e também é executado no modo interpretado.

5.1 Alvos

No Haxe, as plataformas suportadas são conhecidas como "destinos", que consistem nos seguintes módulos:

Os back-ends do compilador responsáveis por gerar o respectivo código. As APIs específicas de tempo de execução que vão além do suporte ao idioma principal (destinos de plataforma).

A tabela a seguir documenta a plataforma e o suporte ao idioma no Haxe. A linguagem Haxe permite que os desenvolvedores obtenham acesso a muitos recursos da plataforma, mas o Haxe não é um mecanismo completo, eles podem precisar de estruturas que permitem a criação de conteúdo para determinadas plataformas.

Alvo do compilador	Resultado	Plataforma	Usar	Desde a versão Haxe
JavaScript [3]	fonte	HTML5 , NodeJS , PhoneGap	Servidor, desktop, navegador, celular	2006
C ++	fonte	Windows , Linux , MacOS , Android , iOS , Palm , WebOS	Servidor, desktop, celular, CLI, consoles de jogos	2009 (2.04)
PHP [3]	fonte	PHP	Servidor	2008 (2,0)
C # [3]	fonte	.NET Framework	Servidor, desktop, celular	2012 (2,10)
Java [3]	fonte	Java	Servidor, desktop	2012 (2,10)
JVM [3]	bytecode	Máquina Virtual JAVA	Servidor, desktop	2019 (4,0)
Python [3]	fonte	Pitão	CLI, web, desktop	2014 (3,2)
Lua [3]	fonte	Lua	CLI, web, desktop, mobile	2016 (3,3)
Neko [3]	código de byte	NekoVM	Servidor, desktop, CLI	2005
Flash / SWF [3]	código de byte	Adobe Flash Player 9+, Adobe AIR , Tamarin	Desktop, navegador, servidor	2005
HashLink [3]	código de byte	HashLink VM ou HL / C (compilar para arquivo C)	Servidor, desktop, celular, consoles de jogos (exportação C)	2016 (3,4)

Desde a versão 1.12 do Haxe (2007), havia um destino de origem do ActionScript 3 (para Adobe FlashPlayer), que foi removido do Haxe na versão 4.0.

5.2 Compilador

O compilador Haxe é dividido em um front-end e vários back-ends. O frontend cria uma árvore de sintaxe abstrata (AST) a partir do código-fonte e executa a verificação de tipo, expansão de macro e otimização no AST . Os vários back-ends traduzem o AST processado em código-fonte ou geram bytecode , dependendo de seu destino.

O compilador é escrito em OCaml . Ele pode ser executado no modo de servidor para fornecer autocompletar código para ambientes de desenvolvimento integrado (IDEs) e manter um cache, para acelerar ainda mais a compilação.

6 Tipos

Haxe possui um sistema de tipologia sofisticado e flexível. Os tipos de tipo que ele oferece são classes, interfaces, tipos de método de função, tipos anônimos, tipos de dados algébricos (ADTs, chamados de enum em Haxe) e tipos abstratos. O polimorfismo paramétrico é possível com classes, ADTs e tipos de função, dando suporte à linguagem para programação genérica baseada na eliminação de tipo. Isso inclui suporte para variação em funções polimórficas , embora não em construtores de tipo .

O sistema de tipo é estático, a menos que anotações para tipagem dinâmica estejam presentes, para uso com destinos que as suportam. A verificação de tipo segue a digitação nominal, com exceção dos tipos anônimos, onde a tipagem estrutural é usada. Finalmente, a inferência de tipo é suportada, permitindo declarações de variáveis sem anotações de tipo .

Exemplo: Abstract Kilometer (Float) public function new (v : Float) this = v ; Abstract Milha (Float) função pública nova (v : Float) this = v ; @: para função inline pública toKilometer (): Retorno de quilômetro (novo quilômetro (este / 0,62137)); classe Teste var km estático : Quilômetro ; função estática main () var one100Miles = new Mile (100); km = one100 milhas ; traço (km); // 160,935

7 Tipagem

A tipagem da Linguagem Haxe é estrutural. Em muitas linguagens de programação funcional, a tipagem estrutural desempenha um papel importante. Haxe o emprega na presença de tipos anônimos, usando a tipagem nominativa da programação orientada a objetos , quando apenas tipos nomeados estão envolvidos. Os tipos anônimos em Haxe são análogos às interfaces implícitas da linguagem Vá para a digitação. Em contraste com as interfaces Go, é possível construir um valor usando um tipo anônimo.

Exemplo: classe FooBar public var foo : Int ; barra de var pública : String ; função pública new () foo = 1 ; bar = ”2” ; função anyFooBar (v : foo : Int , bar : String) trace (v . foo); teste de função estática () var fb = new FooBar (); fb . anyFooBar (fb); fb . anyFooBar (foo : 123 , bar : ”456”);

8 Vantegens da Linguagem

Capacidade de direcionar várias plataformas e dispositivos usando o mesmo idioma, capacidade de usar código estritamente digitado, capacidade de usar macros (transformação de sintaxe), o que pode ser feito com a linguagem Haxe, adicionados recursos de linguagem, como métodos de extensão e programação funcional, o desempenho de tempo de execução dos programas Haxe está em velocidade comparável ao de fontes escritas à mão.

9 Módulos

Todo o código Haxe é organizado em módulos, que são endereçados por caminhos. Em essência, cada arquivo .hx representa um módulo que pode conter vários tipos. Por exemplo, para criar o tipo Ano pacote my.pack conforme mostrado, a estrutura da pasta deve ser my pack e o arquivo pode ser A.hx no pacote de pastas .

// arquivo my / pack / A.hx package my . pack ;classe A

Em outros módulos, outros tipos podem ser importados colocando importinstruções abaixo da definição do pacote, por exemplo, import my.pack.A; Um módulo pode conter vários tipos, como

o seguinte. É possível importar um tipo de cada vez desse módulo, usando import my.pack2.A;. Um tipo pode ser e private, nesse caso, apenas o módulo que o contém pode acessá-lo.

pacote meu . pack2 ;typedef A = a : String typedef privado B = b : String

10 Concorrentes da Haxe

As principais linguagens alternativas para Haxe são: Java, Python, PHP, C, C++.

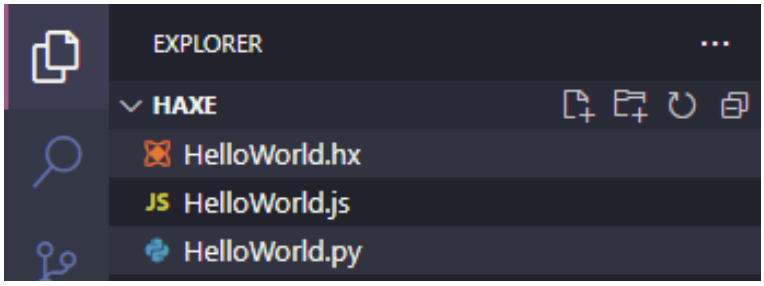
11 Paradigmas da linguagem

Haxe é uma linguagem de programação multiparadigma, o que significa dizer que ela suporta mais de um padrão e pode ser usada para resolver uma infinidade de problemas. A linguagem Haxe foi desenvolvida para suportar os paradigmas tanto estático como orientado a objetos

11.1 Paradigma Escreva uma vez e execute onde quiser

Essa estratégia de compilar em várias linguagens de código-fonte é inspirada no paradigma escrever uma vez, executar em qualquer lugar . Também permite que o programador escolha a melhor plataforma para o trabalho.

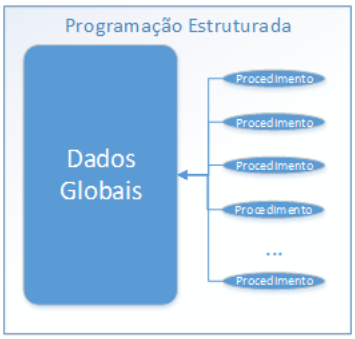
Aqui por exemplo, é um código Haxe que compilamos em JavaScript e também em Python.



11.2 Paradigma Orientado a Objetos

O desenvolvimento de software é extremamente amplo. Nesse mercado, existem diversas linguagens de programação, que seguem diferentes paradigmas. Um desses paradigmas é a Orientação a Objetos, que atualmente é o mais difundido entre todos. Isso acontece porque se trata de um padrão que tem evoluído muito, principalmente em questões voltadas para segurança e reaproveitamento de código, o que é muito importante no desenvolvimento de qualquer aplicação moderna.

A programação orientada a objetos surgiu como uma alternativa as características da programação estruturada. O intuito da sua criação também foi o de aproximar o manuseio das estruturas de um programa ao manuseio das coisas do mundo real, daí o nome "objeto" como uma algo genérico, que pode representar qualquer coisa tangível.



12 Recursividade

A recursividade de Haxe é a partir de tipos enumerados, os tipos enumerados são uma característica importante da linguagem; eles podem ter parâmetros de tipo e ser recursivos. Fornecem suporte básico para tipos de dados algébricos , permitindo a inclusão de tipos de produtos , de forma semelhante a Haskell e ML.

13 Alocação de Memória

Haxe é implementado como um tipo abstrato sobre a implementação nativa do array para um dado target e pode ser mais rápido para coleções de tamanho fixo, porque a memória para armazenamento de seus elementos é pré-alocada.

14 Exemplos de código

14.1 Hello World em Haxe

```
class HelloWorld {

    static public function main() {

        trace("Hello World");

    }

}
```

14.2 Operações Matemáticas em Haxe

```
// exemplo de operações basicas
var x = 10;
var y = 5;
//soma
var soma = x+y;
//subtração
var subtracao= x-y;
//multiplicação
var multiplicacao = x*y;
//divisão
var divisao = x/y;

//resultado soma
trace("Soma de " +x+ "+" +y+ "=" +soma);
//resultado subtração
trace("Subtração de " +x+ "+" +y+ "=" +subtracao);
//resultado multiplicação
trace("Multiplicação de " +x+ "+" +y+ "=" +multiplicacao);
//resultado divisão
trace("Divisao de " +x+ "+" +y+ "=" +divisao);
```

15 Bônus

Instalação e compilação do Haxe no Vs code.

https://github.com/ThiagoGuerra09/Haxe

References

https://stringfixer.com/pt/Haxe

https://github.com/aszasz/manual-do-haxe-bruto/blob/master/Manual$_{d}o_{H}axe.txt$

https://docplayer.com.br/62308710-Desenvolvimento-de-um-jogo-multiplataforma-utilizando-cross-platform-toolkit-haxe.html

https://pt.slideshare.net/adrianots/paradigmas-de-linguagens-de-programao-escopo-estaticodinamico

https://learnxinyminutes.com/docs/pt-br/haxe-pt/

https://melhores-alternativas.com/tools/haxe