Lenguaje Scala

Scalable Language

Equipo:
Alexandra Saavedra Sánchez
Victoria Saavedra Sánchez
Eleani Consuelo Moo Sosa

Creador

- Científico informático y profesor de métodos de programación en École Polytechnique Fédérale de Lausanne(EPFL) en Suiza
- Presidente de la empresa Lightbend
- Cocreador del lenguaje de programación Funnel genérico de Java, javac y EPFL

Historia del lenguaje

Sucesos que llevaron a la creación del lenguaje

1955

Martin Odersky aprende de Java y escribe un lenguaje

1999

Martin se une a EPFL y trata de diseñar un lenguaje

EPFL

2001

Del lenguaje Funnel se creó Scala

2003

Scala se lanza al público

Scala

Paradigmas de programación

Martin Odersky considera a dos paradigmas de programación como dos caras de una misma moneda, los cuales deben estar lo más unificados posible, estos son:

Funcional

Cada función tiene valor y este lenguaje permite:

- Definir funciones anónimas
- Admitir funciones de orden superior
- Anidar funciones
- Admitir **curry**.
- Agregar las clases de casos y su soporte integrado para la coincidencia de patrones
- Agrupar funciones que no son miembros de una clase a través de objetos singleton.

Orientado A Objetos

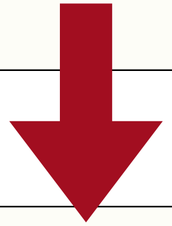
Cada valor es un objeto, cuyo tipo y comportamiento es descrito a partir de clases y atributos. De manera que nos permite

- Extender las clases mediante subclases y utilizando un mecanismo de composición flexible basado en **mixin** que funciona como un reemplazo limpio de la herencia múltiple.

Curry

Los métodos pueden tener múltiples listas de parámetros.

```
trait Iterable[A] {
  ...
  def foldLeft[B](z: B)(op: (B, A) => B): B
  ...
}
```



```
val numbers = List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
val res = numbers.foldLeft(0)((m, n) => m + n)
println(res) // 55
```

Mixins

Los mixins son rasgos que se utilizan para componer una clase.

```
abstract class A {
  val message: String
}
class B extends A {
  val message = "I'm an instance of class B"
}
trait C extends A {
  def loudMessage = message.toUpperCase()
}
class D extends B with C

val d = new D
println(d.message)  // I'm an instance of class B
println(d.loudMessage)  // I'M AN INSTANCE OF CLASS B
```

Áreas de aplicación

Como lenguaje de propósito general de alto nivel, cuenta con una amplia gama de posibles aplicaciones.

Análisis de datos

Los ingenieros de datos lo utilizan junto con Apache Spark para procesar grandes cantidades de datos de forma fluida y eficiente.

Desarrollo Web

Para crear aplicaciones web utilizando framework como Play, Scalatra y Akka HTTP.

Computación distribuida y concurrencia

Scala es ideal para la construcción de sistemas distribuidos y altamente concurrentes. Frameworks como Akka proporcionan herramientas para la creación de sistemas distribuidos.

Ventajas

Sintaxis precisa y reducción de código redundante: Scala ofrece una sintaxis precisa que elimina la necesidad de escribir código innecesario o redundante

Hello World in Java:

```java
public class HelloJava {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Hello World in Scala:

```scala
object HelloScala {
    def main(args: Array[String]): Unit = {
        println("Hello World!")
    }
}
```

Java:

```java
List<String> list = new ArrayList<String>();
list.add("1");
list.add("2");
list.add("3");
```

Scala:

```scala
val list = List("1", "2", "3")
```

Ventajas

- Útil para construir sistemas concurrentes y tolerantes a fallas.
- Útil para construir sistemas distribuidos
- Es un lenguaje orientado a objetos y funcional
- Scala se ejecuta en la Máquina Virtual de Java (JVM) y puede interoperar sin problemas con el código Java existente.
- Tipado estático. Tiene un fuerte sistema de tipo estático. Ayuda a prevenir errores de programación comunes durante la compilación del código.

Desventajas

Difícil de dominar: en comparación con lenguajes como Java y Ruby, ya que requiere competencia en programación orientada a objetos (OOP), programación funcional (FP) y tipado estático.

Comunidad: La comunidad de Scala es relativamente pequeña en comparación con otras comunidades de lenguajes de programación.

Referencias

- https://javarevisited.blogspot.com/2018/01/10-reasons-to-learn-scala-programming.html#axzz8S3mtKwre
- https://docs.scala-lang.org/tour/tour-of-scala.html
- https://medium.com/@kodegasm.id/the-rise-of-functional-programming-languages-scala-haskell-and-clojure-4e3dadac22a0
- https://data-flair.training/blogs/scala-advantages/
- https://www.toptal.com/scala/why-should-i-learn-scala
- https://emeritus.org/blog/coding-scala-programming-language/
- https://blog.insightdatascience.com/a-newbies-guide-to-scala-and-why-it-s-used-for-distributed-computing-979070a9f8
- https://learning.acm.org/techtalks/distributedsystems
- https://www.sitech.me/blog/10-use-cases-for-scala-what-is-scala-used-for
- https://www.linkedin.com/advice/1/how-do-programming-languages-handle-distributed-data-fvgef
- https://www.lihaoyi.com/post/StrategicScalaStylePrincipleofLeastPower.html
-