

# Erlang

---

## Aspectos Básicos

I

# ¿Qué es Erlang/OTP?

Erlang is a programming language used to build **massively scalable soft real-time systems** with requirements on **high availability**. Some of its uses are in telecoms, banking, e-commerce, computer telephony and instant messaging. Erlang's runtime system has built-in support for **concurrency, distribution and fault tolerance**. Originally developed at Ericsson, it was released as open source in 1998.

OTP is a set of Erlang **libraries and design principles** providing middle-ware to develop these systems.

**soft real-time systems:** Sistemas en tiempo real en los que no se pueden hacer garantías sobre el tiempo exacto que va a llevar una acción. Estas garantías solo las pueden hacer dispositivos hardware muy específicos, eso sería hard real-time.

conurrencia, distribución y tolerancia a fallos van unidas, si quieres un sistema que realice un gran número de tareas y que además sea capaz de gestionar errores tanto de software como de hardware (que antes o después se producirán), necesitas que ese sistema esté distribuido en varias máquinas.

# Lenguaje Funcional

- No OO
- Funciones agrupadas en módulos
- Estado inmutable
- Funciones sin side-effects (casi)

# Variables Invariables

```
Eshell V5.8.3 (abort with ^G)
1> A = 1.
1
2> A = 2.
** exception error: no match of right hand side value 2
```

# Pattern Matching (I)

```
Eshell V5.8.3 (abort with ^G)
1> [H | T] = ["uno", "dos", "tres", "cuatro", "cinco"].
["uno", "dos", "tres", "cuatro", "cinco"]
2> H.
"uno"
3> T.
["dos", "tres", "cuatro", "cinco"]
```

# Pattern Matching (II)

```
Eshell V5.8.3 (abort with ^G)
```

```
1> {_, String, [Primero | Resto]} = {hola, "abcde", [1,2,3,4,5,6]}.
```

```
{hola, "abcde", [1,2,3,4,5,6]}
```

```
2> String.
```

```
"abcde"
```

```
3> Primero.
```

```
1
```

```
4> Resto.
```

```
[2,3,4,5,6]
```

# Pattern Matching (III)

```
-module(greet).  
  
-export([greet/2]).  
  
greet(male, Name) ->  
    io:format("Hello, Mr. ~s!~n", [Name]);  
greet(female, Name) ->  
    io:format("Hello, Mrs. ~s!~n", [Name]);  
greet(_, Name) ->  
    io:format("Hello, ~s!~n", [Name]).
```

# Pattern Matching (y IV)

```
Eshell V5.8.3 (abort with ^G)
1> greet:greet(male, "Bob").
Hello, Mr. Bob!
ok
2> greet:greet(female, "Alice").
Hello, Mrs. Alice!
ok
3> greet:greet(XXX, "Alice").
Hello, Alice!
ok
```

# Recursividad

```
-module(recursion).

-export([map/2]).

map(F, List) ->
    map(F, [], List).

map(_F, Result, []) ->
    Result;
map(F, Result, [Head | Tail]) ->
    map(F, Result ++ [F(Head)], Tail).
```

```
Eshell V5.8.3 (abort with ^G)
1> recursion:map(fun(E) -> 2*E end, [1,2,3,4,5]).
[2,4,6,8,10]
```

# Higher Order Functions

```
2> PorDos = fun(E) -> 2*E end.  
#Fun<erl_eval.6.13229925>  
3> recursion:map(PorDos, [1,2,3,4,5]).  
[2,4,6,8,10]
```

IO

# List Comprehensions

```
Eshell V5.8.3 (abort with ^G)
1> [ 2*X || X <- [1,2,3,4,5]].
[2,4,6,8,10]
```

II

# Bit Syntax (I)

```
Eshell V5.8.3 (abort with ^G)
1> Color = 16#F09A29.
15768105
2> <<R:8, G:8, B:8>> = <<Color:24>>.
<<240,154,41>>
3> R.
240
4> G.
154
5> B.
41
```

# Bit Syntax (II)

Bit offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																	
0	Source port																								Destination port																								
32	Sequence number																																																
64	Acknowledgment number (if ACK set)																																																
96	Data offset	Reserved	N S	C W	E C	U R	A C	P S	R S	S Y	F I	Window Size																																					
128	Checksum										Urgent pointer (if URG set)																																						
160	Options (if Data Offset > 5)																padding																																
...	...																																																

```
<<SourcePort:16, DestinationPort:16,
SequenceNumber:32,
AckNumber:32,
DataOffset:4, _Reserved:4, Flags:8, WindowSize:16,
CheckSum: 16, UrgentPointer:16,
Payload/binary>> = SomeBinary.
```

# Otras Características

- Hot Code Loading
- Operaciones IO asíncronas

# Erlang

---

## Concurrencia

# Concurrencia VS Paralelismo

I6

Concurrencia: Se produce cuando varias tareas se inician ejecutan y finalizan superpuestas en el mismo periodo de tiempo. No significa necesariamente que en un instante determinado se estén ejecutando a la vez.

Paralelismo: Las tareas se ejecutan exactamente al mismo tiempo, por ejemplo en un procesador multicore.

Erlang tiene concurrencia desde sus inicios en los años 80, el paralelismo ha sido añadido recientemente (SMP).

# Escalabilidad

I7

Erlang está diseñado desde el principio para soportar un gran número de procesos ligeros. El objetivo de la escalabilidad es superar las limitaciones del hardware. Hay dos formas de hacer esto, añadiendo mejor hardware (más RAM, discos más rápidos, etc...) o añadiendo más máquinas. La primera opción es útil sólo hasta cierto punto.

# Tolerancia a fallos (Let it Crash)

I8

Puedes intentar evitar que el software tenga bugs, pero es muy difícil evitar todos los posibles errores. Aún así no hay forma de evitar que se produzcan fallos hardware.

Los errores ocurren, Erlang proporciona formas de gestionar los errores cuando ocurren en vez de evitar que se produzcan.

# Modelo de Actores

- Procesos ligeros
- Memoria independiente
- Envío de mensajes

I9

Procesos ligeros: Los procesos se pueden crear y destruir muy rápidamente y la máquina virtual puede cambiar entre un proceso y otro muy rápidamente también.

Memoria independiente: Los procesos no comparten memoria, de manera que se evita la necesidad de locks y que un proceso en un estado inconsistente pueda afectar a otro. También facilita la recolección de basura una vez un proceso ha terminado. Esto ayuda en la implementación de soft real-time systems ya que las pausas de GC son muy pequeñas.

Envío de mensajes: La única forma de pasar datos de un proceso a otro es enviando un mensaje, lo cual implica copiar datos de una zona de memoria a otra. Es más lento pero más seguro.

# Recursos

- <http://www.erlang.org>
- <http://learnyousomeerlang.com>
- <http://trapexit.org/>
- <http://www.scribd.com/doc/44221/Thinking-in-Erlang>

# Recursos (y II)

