UNIVERSIDAD COMPLUTENSE DE MADRID

ntic master School

Scala

Control de errores

Enero de 2023

1 Try y su uso

Excepciones


```
def toInt(s: String): Int =
 try {
   s.toInt
 } catch {
   case _: NumberFormatException => 0
 }
```

- En Scala, se puede usar las expresiones *try-catch*, similar a otros lenguajes.
- Para situaciones donde no sea necesario un bloque finally, Scala nos provee una alternativa muy útil: *Try*.

Uso de Try


```
scala> import scala.util.{Try, Success, Failure}
import scala.util.{Try, Success, Failure}

scala> Try("100".toInt)
res29: scala.util.Try[Int] = Success(100)

scala> Try("Scala".toInt)
res30: scala.util.Try[Int] = Failure(java.lang.NumberFormatException: For input string: "Scala")
```

- *Try* evalúa la expresión que se le proporciona y devuelve el resultado envuelto en un objeto *Success* o *Failure.*
- No nos importa qué tipo de excepción puede generar la expresión a evaluar, sólo nos importa detectar si la evaluación fue satisfactoria o fallida.

Pattern Matching y Try


```scala
def makeInt(s: String): Int = Try(s.toInt) match {
 case Success(n) => n
 case Failure(_) => 0
}
```

```
scala> makeInt("24")
res31: Int = 24

scala> makeInt("Python")
res32: Int = 0
```

- *Try* se puede usar de la misma manera que *Option*.
- Una de esas maneras es con pattern matching.

Funciones de orden superior y Try

```
def makeInt(s: String): Int = Try(s.toInt).getOrElse(0)
```

```
scala> Success("Scala").map(_.reverse)
res33: scala.util.Try[String] = Success(alacS)

scala> for {
     | language <- Success("Scala")
     | behavior <- Success("mola")
     | } yield s"$language $behavior"
res34: scala.util.Try[String] = Success(Scala mola)
```

- *getOrElse*, al igual que en *Option*, devuelve el valor que contenga (cuando es *Success*) o un valor por defecto (cuando es *Failure*).
- *Try* también ofrece funciones de orden superior como las colecciones.

Ventajas de Try

- Nos permite hacer llamadas consecutivas a funciones y capturar las excepciones que puedan aparecer y aún así terminar todo el flujo de operaciones.

- *flatMap* y *map* se pueden invocar incluso cuando se obtenga un *Failure*.

- *Try* captura las excepciones que no son fatales (*scala.util.control.NonFatal*). Los errores de sistema sí son lanzados y no capturadas por Try.