CURSO DE PROGRAMACION SCALA
Sesión 8

Sergio Couto Catoira

Índice

- Alternativas a excepciones
  - Más TDA: Implementación de Option
- Lifting
- For – comprehension

Alternativas a excepciones

- Valor por defecto
  - No siempre es posible
  - Permite que errores se propaguen silenciosamente si alguien se olvida de chequearlo
  - Obliga a tener mucho código de control
  - No es válido con funciones genéricas
  - Obliga a los que usan la función a saber qué puede devolver en lugar de únicamente usarla
- Valor por defecto indicado en la llamada
  - Si se llama dentro de un proceso complejo es difícil de tratar
  - Difícil bifurcar el código si es necesario

Implementación de Option

- Al igual que lista se crea como sealed trait
  - ¿Por qué?
  - Cuando sea posible, se implementarán los métodos dentro del propio trait
  - De esa forma se puede llamar con la forma clásica objeto.metodo

Ejercicios con Option

- Implementa los siguientes métodos definidos en el trait
  - Map: aplica f si es un valor válido
  - flatMap: aplica f si es valor válido. F puede devolver None
  - fetOrElse: devuelve el valor válido o el por defecto
  - orElse: devuelve el option si es válido o el valor por defecto en caso contrario
  - filter: devuelve el option si cumple la función, None en caso contrario

Ejercicios con Option

- Define una función mean que calcule la media de una secuencia.
  - def mean (xs :Seq[Double]): Option[Double]

Lifting de funciones

- ¿Es neceario reescribir todas las funciones para operar con option?
- Técnica para emplear Option en funciones conocidas

```
def lift[A,B](f: A => B): Option[A] => Option[B] = {
    _ map f
}
```

- Ejemplo: valor absoluto con Option
  - val abs0: Option[Double] => Option[Double] = lift(math.abs)

Lifting de funciones

- Ejemplo con calcularCuota.
- Recibe dos enteros, pero el usuario lo mete como String a través de un formulario
  - ¿Qué pasa si introduce un string que no es un número?
  - No se puede reescribir la función y tampoco debería hacer falta. ¿Qué más le da si falló algo antes? **No es su problema**
- Se podría resolver con Try y viendo explícitamente si alguno es Failure. Pero, ¿Y si hay muchos valores?

Lifting de funciones

- Define una función map2 que reciba dos valores opcionales y una funcion. Debe devolver None si alguno de ellos es None.
  - def map2[A,B,C](a: Option[A], b: Option[B])(f:(A,B) => C): Option©
- Úsala para llamar a calcularCuota sin riesgo.

For comprehension

- Syntactic sugar para combinar maps y flatMaps
- Legible y entendible
- Sintaxis:

```
for {
    elem ← estructura [if clause]
    elem2 ← estructura2
    .
    .
    .
    [if clause]
    } yield retorno
```

For comprehension

- Cada línea se correspodne con un flatMap
- La última se corresponde con un map
- Sentencia if se corresponde con un filter
- Tipo de estructura de salida viene dado por el tipo de entrada
- Tipo interno de salida viene dado por lo que haga el yield

Ejercicio

- Aplica for comprehension a una lista de Strings de forma que obtengas una lista de enteros con la longitud de cada String
- Introduce en el for anterior una sentencia para devolver sólo los resultados que sean impares y devuelve la palabra en formato (palabra, tamaño)
- Haz lo mismo siendo la entrada una lista de listas y descartando las listas que tengan menos de 2 elementos.
- Redefine el método map2 usando for comprehension

Ejercicio

- Define una función secuencia que combine una lista de Options en un Option con una lista de los valores. Si alguno de los valores es None, debe devolver None.
  - def sequence[A](a: List[Option[A]]): Option[List[A]]
- Esta función es útil para recorrer una lista transformando cada elemento y dar error si algún valor no es correcto. Pero antes de poder usarla debería convertir toda la lista a lista de Options.
- Se recorre la lista dos veces

Ejercicio

- Define una función traverse que evite lo anterior.
  - def traverse[A, B](a: List[A])(f: A => Option[B]): Option[List[B]]

- Implementa la función sequence en base a traverse

Ejercicios (difícil)

- Define una función variance que calcule la varianza de una secuencia.
  - Si la media de una secuencia es m
  - La varianza es la media de math.pow(x-m, 2) para cada elemento x
  - def variance (xs :Seq[Double]): Option[Double]