ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Combinators on Futures (2/2)

Principles of Reactive Programming

Erik Meijer

Better recovery with less matching

```
def sendToSafe(packet: Array[Byte]): Future[Array[Byte]] =
  sendTo(mailServer.europe, packet) recoverWith {
    case europeError =>
     sendTo(mailServer.usa, packet) recover {
      case usaError  => usaError.getMessage.toByteArray
    }
  }

def fallbackTo(that: =>Future[T]): Future[T] = {
  … if this future fails take the successful result
    of that future …
  … if that future fails too, take the error of
    this future …
}
```

Better recovery with less matching

```
def sendToSafe(packet: Array[Byte]):Future[Array[Byte]]=
  sendTo(mailServer.europe, packet) fallbackTo {
    sendTo(mailServer.usa, packet)
  } recover {
    case europeError =>
          europeError.getMessage.toByteArray
  }
def fallbackTo(that: =>Future[T]): Future[T] = {
  … if this future fails take the succcessful result
    of that future …
  … if that future fails too, take the error of
    this future …
}
```

Fallback implementation

```
def fallbackTo(that: =>Future[T]): Future[T] = {
  this recoverWith {
    case _ => that recoverWith { case _ => this }
  }
}
```

Asynchronous where possible, blocking where necessary

```
trait Awaitable[T] extends AnyRef {
  abstract def ready(atMost: Duration): Unit
  abstract def result(atMost: Duration): T
}

trait Future[T] extends Awaitable[T] {
   def filter(p: T⇒Boolean): Future[T]
   def flatMap[S](f: T⇒ Future[S]): Future[U]
   def map[S](f: T⇒S): Future[S]
   def recoverWith(f: PartialFunction[Throwable,
Future[T]]): Future[T]
}
```

Asynchronous where possible, blocking where necessary

```
val socket = Socket()
val packet: Future[Array[Byte]] =
  socket.readFromMemory()
val confirmation: Future[Array[Byte]] =
  packet.flatMap(socket.sendToSafe(_))

val c = Await.result(confirmation, 2 seconds)
println(c.toText)
```

Duration

```
import scala.language.postfixOps

object Duration {
  def apply(length: Long, unit: TimeUnit):
Duration
}


val fiveYears = 1826 minutes
```

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

End of Combinators on Futures (2/2)

Principles of Reactive Programming

Erik Meijer