```scala
/**
  * Created by psc on 9/6/17.
  */

package au.com.carringbushsw.Akka.ResponderManager

import MessageTag._

import akka.actor._
import akka.event._
import akka.actor.OneForOneStrategy
import akka.actor.SupervisorStrategy._
import scala.concurrent.duration._

// messages
sealed case class Response(msg: MessageTag)
object Response { // companion object
  val name = "Response"
}
// duplicate message??
sealed case class Request(msg: MessageTag)
object Request { // companion object
  val name = "Request"
}
```

// duplicate msg?

```scala
object ResponderManager { // companion object
  val name = "ResponderManager"
}

class ResponderManager extends Actor {
  val name = "ResponderManager"
  val log = Logging(context.system, this)

  override val supervisorStrategy =
    OneForOneStrategy(
      maxNrOfRetries = 5, withinTimeRange = Duration.create("1 minute")) {
      case _: NullPointerException          => Restart
      case _: ArithmeticException           => Resume
      case _: IllegalArgumentException      => Stop
      case _: UnsupportedOperationException => Stop
      case _: Exception                     => Escalate
    }
```

common method

```scala
  override def preStart: Unit = {
    log.info(s"${ResponderManager.name}.preStart")
    context.become(response)
  }

  override def postStop: Unit = log.info(s"${ResponderManager.name}.postStop")

  def receive = response

  def response: Receive = {
    // request-and-response
    case "A REQUEST" => log.info("received A REQUEST")
    case msg: String => log.info(s"${msg}")

    case Response(MessageTag.request) => {
      val s = sender  // get sender before any context change

      log.info(s"${Response.name}('received ${MessageTag.request}')")
      log.info(s"${Response.name}('${MessageTag.response}...')")
      // send response back to requester
      s ! Request(MessageTag.response)
    }

    case _ => log.info(s"${ResponderManager.name}.${Response.name}.help!!")

  } // response: Receive
```

```
} // ResponderManager
```