

# Chapter 8: Basic Type Classes and Their Usage

The screenshot shows the Haskell documentation for the 'cats' package. At the top left, there is a circular logo with the letter 'p' and the text 'cats'. Below this, a header bar contains 'package cats'. A descriptive sentence reads: 'Symbolic aliases for various types are defined here.' Below this is a section titled 'Linear Supertypes' with a search bar containing the text 'Filter all members'. On the right side, a 'Packages' sidebar lists 'root' and 'cats' (highlighted), followed by sub-packages: 'arrow', 'data', 'evidence', 'instances', and 'syntax'. At the bottom of the sidebar, there is a list of type classes with circular icons: 'Alternative', 'Always', 'Applicative', and 'ApplicativeError'.

<input type="radio"/> <input checked="" type="checkbox"/> Alternative	<input type="radio"/> <input checked="" type="checkbox"/> Eval	<input checked="" type="checkbox"/> Now
<input type="radio"/> <input checked="" type="checkbox"/> Always	<input checked="" type="checkbox"/> EvalGroup	<input type="radio"/> <input checked="" type="checkbox"/> Parallel
<input type="radio"/> <input checked="" type="checkbox"/> Applicative	<input checked="" type="checkbox"/> EvalMonoid	<input checked="" type="checkbox"/> ParallelArityFunctions
<input type="radio"/> <input checked="" type="checkbox"/> ApplicativeError	<input checked="" type="checkbox"/> EvalSemigroup	<input checked="" type="checkbox"/> ParallelArityFunctions2
<input type="radio"/> <input checked="" type="checkbox"/> Apply	<input type="radio"/> <input checked="" type="checkbox"/> FlatMap	<input type="radio"/> <input checked="" type="checkbox"/> Reducible
<input checked="" type="checkbox"/> ApplyArityFunctions	<input type="radio"/> <input checked="" type="checkbox"/> Foldable	<input type="radio"/> <input checked="" type="checkbox"/> SemigroupK
<input type="radio"/> <input checked="" type="checkbox"/> Bifoldable	<input type="radio"/> <input checked="" type="checkbox"/> Functor	<input type="radio"/> <input checked="" type="checkbox"/> Semigroupal
<input type="radio"/> <input checked="" type="checkbox"/> Bifunctor	<input type="radio"/> <input checked="" type="checkbox"/> Inject	<input checked="" type="checkbox"/> SemigroupalArityFunctions
<input type="radio"/> <input checked="" type="checkbox"/> Bimonad	<input type="radio"/> <input checked="" type="checkbox"/> InjectK	<input type="radio"/> <input checked="" type="checkbox"/> Show
<input type="radio"/> <input checked="" type="checkbox"/> Bitraverse	<input type="radio"/> <input checked="" type="checkbox"/> Invariant	<input checked="" type="checkbox"/> StackSafeMonad
<input type="radio"/> <input checked="" type="checkbox"/> CoflatMap	<input type="radio"/> <input checked="" type="checkbox"/> InvariantMonoidal	<input type="radio"/> <input checked="" type="checkbox"/> Traverse
<input type="radio"/> <input checked="" type="checkbox"/> CommutativeApplicative	<input type="radio"/> <input checked="" type="checkbox"/> InvariantSemigroupal	<input type="radio"/> <input checked="" type="checkbox"/> UnorderedFoldable
<input type="radio"/> <input checked="" type="checkbox"/> CommutativeApply	<input type="radio"/> <input checked="" type="checkbox"/> Later	<input type="radio"/> <input checked="" type="checkbox"/> UnorderedTraverse
<input type="radio"/> <input checked="" type="checkbox"/> CommutativeFlatMap	<input type="radio"/> <input checked="" type="checkbox"/> Monad	<input type="radio"/> <input checked="" type="checkbox"/> implicits
<input type="radio"/> <input checked="" type="checkbox"/> CommutativeMonad	<input type="radio"/> <input checked="" type="checkbox"/> MonadError	
<input type="radio"/> <input checked="" type="checkbox"/> Comonad	<input type="radio"/> <input checked="" type="checkbox"/> MonoidK	
<input type="radio"/> <input checked="" type="checkbox"/> Contravariant	<input type="radio"/> <input checked="" type="checkbox"/> NonEmptyParallel	
<input type="radio"/> <input checked="" type="checkbox"/> ContravariantMonoidal	<input checked="" type="checkbox"/> NonEmptyReducible	
<input type="radio"/> <input checked="" type="checkbox"/> ContravariantSemigroupal	<input type="radio"/> <input checked="" type="checkbox"/> NonEmptyTraverse	
<input type="radio"/> <input checked="" type="checkbox"/> Distributive	<input type="radio"/> <input checked="" type="checkbox"/> NotNull	



# Monad

Companion [object](#) `Monad`

```
trait Monad[F[_]] extends FlatMap[F] with Applicative[F] with Serializable
```

Monad.

Allows composition of dependent effectful functions.

See: [Monads for functional programming](#)

Must obey the laws defined in `cats.laws.MonadLaws`.

### Linear Supertypes

[Applicative](#)[F], [InvariantMonoidal](#)[F], [FlatMap](#)[F], [Apply](#)[F], [ApplyArityFunctions](#)[F], [InvariantSemigroupal](#)[F], [Semigroupal](#)[F], [Functor](#)[F], [Invariant](#)[F], [Serializable](#), [Serializable](#), [AnyRef](#), [Any](#)

### Known Subclasses

[Bimonad](#), [CommutativeMonad](#), [MonadError](#), [StackSafeMonad](#)

all

alternative

applicative

applicativeError

apply

arrow

arrowChoice

bifoldable

bifunctor

bitraverse

cartesian

coflatMap

comonad

compose

contravariant

contravariantMonoidal

contravariantSemigroupal

distributive

either

eitherK

BitraverseOps

BitraverseSyntax

CoflatMapSyntax

ComonadSyntax

ComposeSyntax

ContravariantMonoidalOps

ContravariantMonoidalSyntax

[ContravariantSemigroupalSyntax](#)

ContravariantSyntax

DistributiveOps

DistributiveSyntax

EitherIdOps

EitherKOps

EitherKSyntax

EitherObjectOps

EitherOps

EitherSyntax

EqOps

EqSyntax

### syntax

AllSyntax

AllSyntaxBinCompat

AllSyntaxBinCompat0

AlternativeSyntax

ApplicativeErrorExtension

ApplicativeErrorExtensionOps

ApplicativeErrorIdOps

ApplicativeErrorOps

ApplicativeErrorSyntax

ApplicativeIdOps

ApplicativeOps

ApplicativeSyntax

ApplyOps

ApplySyntax

ArrowChoiceSyntax

ArrowSyntax

BifoldableSyntax

BifunctorSyntax



cats.syntax

# MonadOps

```
final class MonadOps[F[_], A] extends AnyVal
```

Linear Supertypes

▶ Filter all members

## Instance Constructors

```
new MonadOps(fa: F[A])
```

## Value Members

```
val fa: F[A]
```

```
def getClass(): Class[_ <: AnyVal]
```



```
def iterateUntil(p: (A) => Boolean)(implicit M: Monad[F]): F[A]
```

```
def iterateWhile(p: (A) => Boolean)(implicit M: Monad[F]): F[A]
```


```
def untilM[G[_]](p: F[Boolean])(implicit M: Monad[F], G: Alternative[G]): F[G[A]]
```

```
def untilM_(p: F[Boolean])(implicit M: Monad[F]): F[Unit]
```

```
def whileM[G[_]](p: F[Boolean])(implicit M: Monad[F], G: Alternative[G]): F[G[A]]
```



---



[cats.syntax](#)  
**MonadSyntax**

trait **MonadSyntax** extends [AnyRef](#)

Linear Supertypes


Known Subclasses

Filter all members

**Value Members**

```
implicit final def catsSyntaxMonad[F[_], A](fa: F[A]): MonadOps[F, A]
```

```
implicit final def catsSyntaxMonadIdOps[A](a: A): MonadIdOps[A]
```



[cats.syntax](#)  
**monad**

object **monad** extends [MonadSyntax](#)

Linear Supertypes

Filter all members

**Value Members**

```
implicit final def catsSyntaxMonad[F[_], A](fa: F[A]): MonadOps[F, A]
```

```
implicit final def catsSyntaxMonadIdOps[A](a: A): MonadIdOps[A]
```



cats.syntax

all

object **all** extends [AllSyntaxBinCompat](#)

#### Linear Supertypes

[AllSyntaxBinCompat](#), [AllSyntaxBinCompat0](#), [TrySyntax](#), [ApplicativeErrorExtension](#), [UnorderedTraverseSyntax](#), [ToUnorderedTraverseOps](#), [AllSyntax](#), [WriterSyntax](#), [VectorSyntax](#), [ValidatedSyntax](#), [NonEmptyTraverseSyntax](#), [ToNonEmptyTraverseOps](#), [TraverseSyntax](#), [ToTraverseOps](#), [StrongSyntax](#), [ToStrongOps](#), [ShowSyntax](#), [ToShowOps](#), [SemigroupKSyntax](#), [ToSemigroupKOps](#), [ReducibleSyntax](#), [ToReducibleOps](#), [ProfunctorSyntax](#), [ToProfunctorOps](#), [ParallelSyntax](#), [TupleParallelSyntax](#), [OrderSyntax](#), [PartialOrderSyntax](#), [OptionSyntax](#), [MonoidSyntax](#), [MonadSyntax](#), [MonadErrorSyntax](#), [ListSyntax](#), [IorSyntax](#), [InvariantSyntax](#), [ToInvariantOps](#), [HashSyntax](#), [GroupSyntax](#), [SemigroupSyntax](#), [FunctorSyntax](#), [ToFunctorOps](#), [FoldableSyntax](#), [ToUnorderedFoldableOps](#), [ToFoldableOps](#), [FlatMapSyntax](#), [ToFlatMapOps](#), [EqSyntax](#), [EitherSyntax](#), [EitherKSyntax](#), [ContravariantSemigroupalSyntax](#), [ContravariantMonoidalSyntax](#), [DistributiveSyntax](#), [ToDistributiveOps](#), [ContravariantSyntax](#), [ToContravariantOps](#), [ComposeSyntax](#), [ToComposeOps](#), [ComonadSyntax](#), [ToComonadOps](#), [CoflatMapSyntax](#), [ToCoflatMapOps](#), [SemigroupalSyntax](#), [BitraverseSyntax](#), [BitraverseSyntax1](#), [BifoldableSyntax](#), [ToBifoldableOps](#), [BifunctorSyntax](#), [ToBifunctorOps](#), [ArrowChoiceSyntax](#), [ToArrowChoiceOps](#), [ArrowSyntax](#), [ToArrowOps](#), [ApplySyntax](#), [TupleSemigroupalSyntax](#), [ApplicativeErrorSyntax](#), [ApplicativeSyntax](#), [AlternativeSyntax](#), [AnyRef](#), [Any](#).

## instances

<input type="radio"/> AllInstances	<input type="radio"/> InvariantMonoidalInstances	<input type="radio"/> all
<input type="radio"/> AnyValInstances	<input type="radio"/> ListInstances	<input type="radio"/> bigDecimal
<input type="radio"/> BigDecimalInstances	<input type="radio"/> LongInstances	<input type="radio"/> bigInt
<input type="radio"/> BigIntInstances	<input type="radio"/> MapInstances	<input type="radio"/> bitSet
<input type="radio"/> BitSetInstances	<input type="radio"/> OptionInstances	<input type="radio"/> boolean
<input type="radio"/> BooleanInstances	<input type="radio"/> OrderInstances	<input type="radio"/> byte
<input type="radio"/> ByteInstances	<input type="radio"/> OrderingInstances	<input type="radio"/> char
<input type="radio"/> CharInstances	<input type="radio"/> ParallelInstances	<input type="radio"/> double
<input type="radio"/> DoubleInstances	<input type="radio"/> PartialOrderInstances	<input type="radio"/> duration
<input type="radio"/> DurationInstances	<input type="radio"/> PartialOrderingInstances	<input type="radio"/> either
<input type="radio"/> EitherInstances	<input type="radio"/> QueueInstances	<input type="radio"/> eq
<input type="radio"/> EqInstances	<input type="radio"/> SetInstances	<input type="radio"/> equiv
<input type="radio"/> EquivInstances	<input type="radio"/> ShortInstances	<input type="radio"/> float
<input type="radio"/> FloatInstances	<input checked="" type="radio"/> SortedMapCommutativeMonoid	<input type="radio"/> function
<input type="radio"/> FunctionInstances	<input checked="" type="radio"/> SortedMapEq	<input type="radio"/> future
<input type="radio"/> FutureInstances	<input checked="" type="radio"/> SortedMapHash	<input type="radio"/> int
<input type="radio"/> HashInstances	<input type="radio"/> SortedMapInstances	<input type="radio"/> invariant
<input type="radio"/> IntInstances	<input type="radio"/> SortedMapInstances1	<input type="radio"/> list
	<input type="radio"/> SortedMapInstances2	<input type="radio"/> long
	<input checked="" type="radio"/> SortedMapMonoid	<input type="radio"/> map

---

```
trait FutureInstances extends FutureInstances1
```

Linear Supertypes

Known Subclasses

Filter all members

### Value Members

```
implicit def catsStdInstancesForFuture(implicit ec: ExecutionContext): MonadError[Future,  
  Throwable] with CoflatMap[Future] with Monad[Future]
```

```
implicit def catsStdMonoidForFuture[A](implicit arg0: Monoid[A], ec: ExecutionContext):  
  Monoid[Future[A]]
```

```
implicit def catsStdSemigroupForFuture[A](implicit arg0: Semigroup[A], ec: ExecutionContext):  
  Semigroup[Future[A]]
```



[cats.instances](#)

## future

```
object future extends FutureInstances
```

Linear Supertypes

Filter all members

### Value Members

```
implicit def catsStdInstancesForFuture(implicit ec: ExecutionContext): MonadError[Future,  
  Throwable] with CoflatMap[Future] with Monad[Future]
```

```
implicit def catsStdMonoidForFuture[A](implicit arg0: Monoid[A], ec: ExecutionContext):  
  Monoid[Future[A]]
```

```
implicit def catsStdSemigroupForFuture[A](implicit arg0: Semigroup[A], ec: ExecutionContext):
```



cats.instances

all

object **all** extends [AllInstances](#)

#### Linear Supertypes

[AllInstances](#), [VectorInstances](#), [kernel.instances.VectorInstances](#), [VectorInstances1](#), [VectorInstances2](#), [UUIDInstances](#), [kernel.instances.UUIDInstances](#), [TryInstances](#), [TryInstances1](#), [TryInstances2](#), [SymbolInstances](#), [kernel.instances.SymbolInstances](#), [StringInstances](#), [kernel.instances.StringInstances](#), [StreamInstances](#), [kernel.instances.StreamInstances](#), [StreamInstances1](#), [StreamInstances2](#), [SortedSetInstances](#), [SortedSetInstances1](#), [SortedMapInstances](#), [SortedMapInstances2](#), [SortedMapInstances1](#), [SetInstances](#), [kernel.instances.SetInstances](#), [SetInstances1](#), [QueueInstances](#), [kernel.instances.QueueInstances](#), [QueueInstances1](#), [QueueInstances2](#), [PartialOrderingInstances](#), [PartialOrderInstances](#), [kernel.instances.PartialOrderInstances](#), [PartialOrderToPartialOrderingConversion](#), [ParallelInstances](#), [ParallelInstances1](#), [OrderingInstances](#), [OrderInstances](#), [kernel.instances.OrderInstances](#), [OrderToOrderingConversion](#), [OptionInstances](#), [kernel.instances.OptionInstances](#), [OptionInstances0](#), [OptionInstances1](#), [OptionInstances2](#), [MapInstances](#), [kernel.instances.MapInstances](#), [MapInstances1](#), [ListInstances](#), [kernel.instances.ListInstances](#), [ListInstances1](#), [ListInstances2](#), [InvariantMonoidalInstances](#), [HashInstances](#), [kernel.instances.HashInstances](#), [HashToHashingConversion](#), [FutureInstances](#), [FutureInstances1](#), [FutureInstances2](#), [FunctionInstances](#), [Function1Instances](#), [Function1Instances0](#), [Function0Instances](#), [Function0Instances0](#), [kernel.instances.FunctionInstances](#), [FunctionInstances0](#), [FunctionInstances1](#), [FunctionInstances2](#), [FunctionInstances3](#), [FunctionInstances4](#), [EqInstances](#), [kernel.instances.EqInstances](#), [EqToEqivConversion](#), [EitherInstances](#), [kernel.instances.EitherInstances](#), [EitherInstances0](#), [EitherInstances1](#), [BitSetInstances](#), [kernel.instances.BitSetInstances](#), [BigDecimalInstances](#), [kernel.instances.BigDecimalInstances](#), [BigIntInstances](#), [kernel.instances.BigIntInstances](#), [AnyValInstances](#), [TupleInstances](#), [kernel.instances.TupleInstances](#), [TupleInstances1](#), [TupleInstances2](#), [TupleInstances3](#), [Tuple2Instances](#), [Tuple2Instances1](#), [Tuple2Instances2](#), [static.javadoc.io/ora.tv/elevel/cats-core-2.12/1.1.0/cats/index.html](#)

---

## data

- C AppFunc
- C Cokleisli
- C Const
- C EitherK
- C EitherT
- C Func
- C IdT
- C IndexedReaderWriterStateT
- C IndexedStateT
- C Ior
- C IorT
- C Kleisli
- C Nested
- C NonEmptyList
  - C NonEmptyMapOps
  - C NonEmptySetOps
- C NonEmptyVector
- C OneAnd
- C OptionT
- Reader
- ReaderWriterState
- ReaderWriterStateT
- State
- StateT
- C Tuple2K
- C Validated
- Writer
- C WriterT
- C ZipList
- C ZipStream
- C ZipVector

---

object **Writer**

Linear Supertypes

▶ Filter all members

### Value Members

```
def apply[L, V](l: L, v: V): WriterT\[Id, L, V\]
```

```
def tell[L](l: L): Writer\[L, Unit\]
```

```
def value[L, V](v: V)(implicit arg0: Monoid\[L\]): Writer\[L, V\]
```

### Abstract Value Members

```
abstract def flatMap[A, B](fa: F[A])(f: (A) => F[B]): F[B]
```

```
abstract def pure[A](x: A): F[A]  
pure lifts any value into the Applicative Functor.
```

```
abstract def tailRecM[A, B](a: A)(f: (A) => F[Either[A, B]]): F[B]  
Keeps calling f until a scala.util.Right[B] is returned.
```



cats

# Functor

Companion object `Functor`

```
trait Functor[F[_]] extends Invariant[F] with Serializable
```

Functor.

The name is short for "covariant functor".

Must obey the laws defined in `cats.laws.FunctorLaws`.

*Self Type*      [Functor](#)[F]

Linear Supertypes

[Invariant](#)[F], [Serializable](#), [Serializable](#), [AnyRef](#), [Any](#)

Known Subclasses

[Alternative](#), [Applicative](#), [ApplicativeError](#), [Apply](#), [Bimonad](#), [CoflatMap](#), [CommutativeApplicative](#), [CommutativeApply](#), [CommutativeFlatMap](#), [CommutativeMonad](#), [Comonad](#), [Distributive](#), [FlatMap](#), [Monad](#), [MonadError](#), [NonEmptyTraverse](#), [StackSafeMonad](#), [Traverse](#)

## Abstract Value Members



```
abstract def map[A, B](fa: F[A])(f: (A) => B): F[B]
```

## Concrete Value Members

```
def as[A, B](fa: F[A], b: B): F[B]
```

Replaces the A value in `F[A]` with the supplied value.

```
def compose[G[_]](implicit arg0: Functor[G]): Functor[[ $\alpha$ ]F[G[ $\alpha$ ]]]
```

```
def compose[G[_]](implicit arg0: Invariant[G]): Invariant[[ $\alpha$ ]F[G[ $\alpha$ ]]]
```

```
def composeContravariant[G[_]](implicit arg0: Contravariant[G]): Contravariant[[ $\alpha$ ]F[G[ $\alpha$ ]]]
```

```
def composeFunctor[G[_]](implicit arg0: Functor[G]): Invariant[[ $\alpha$ ]F[G[ $\alpha$ ]]]
```

```
final def fmap[A, B](fa: F[A])(f: (A) => B): F[B]
```

Alias for `map`, since `map` can't be injected as syntax if the implementing type already had a built-in `.map` method.



---

```
trait Apply[F[_]] extends Functor[F] with InvariantSemigroupal[F] with ApplyArityFunctions[F] with Serializable
```

Weaker version of `Applicative`[F]; has `apply` but not `pure`.

Must obey the laws defined in `cats.laws.ApplyLaws`.

*Self Type*          `Apply`[F]

Linear Supertypes

Known Subclasses

Filter all members

#### Abstract Value Members

```
abstract def ap[A, B](ff: F[(A) => B])(fa: F[A]): F[B]
```

Given a value and a function in the `Apply` context, applies the function to the value.

```
abstract def map[A, B](fa: F[A])(f: (A) => B): F[B]
```

```
trait Monoid[A] extends Semigroup[A]
```

A monoid is a semigroup with an identity. A monoid is a specialization of a semigroup, so its operation must be associative. Additionally, `combine(x, empty) == combine(empty, x) == x`. For example, if we have `Monoid[String]`, with `combine` as string concatenation, then `empty = ""`.

Linear Supertypes

Known Subclasses

Filter all members

#### Abstract Value Members

```
abstract def combine(x: A, y: A): A
```

Associative operation which combines two values.

```
abstract def empty: A
```

Return the identity element for this monoid.



cats.kernel

# Semigroup

Companion [object Semigroup](#)

trait **Semigroup**[A] extends [Serializable](#)

A semigroup is any set A with an associative operation (combine).

Linear Supertypes

Known Subclasses

Filter all members

## Abstract Value Members

```
abstract def combine(x: A, y: A): A
    Associative operation which combines two values.
```



cats

# MonoidK

Companion [object MonoidK](#)

## Abstract Value Members

```
abstract def combineK[A](x: F[A], y: F[A]): F[A]
    Combine two F[A] values.
```

```
abstract def empty[A]: F[A]
    Given a type A, create an "empty" F[A] value.
```

## Concrete Value Members

```
def algebra[A]: Monoid[F[A]]
    Given a type A, create a concrete Monoid[F[A]].
```

```
def compose[G[_]]: MonoidK[[ $\alpha$ ]F[G[ $\alpha$ ]]]
    "Compose" with a G[_] type to form a SemigroupK for  $\lambda[\alpha] \Rightarrow F[G[\alpha]]$ .
```



cats

# Traverse

Companion [object Traverse](#)

trait **Traverse**[F[\_]] extends [Functor](#)[F] with [Foldable](#)[F] with [UnorderedTraverse](#)[F] with [Serializable](#)

## Abstract Value Members

```
abstract def foldLeft[A, B](fa: F[A], b: B)(f: (B, A) => B): B  
Left associative fold on 'F' using the function 'f'.
```

```
abstract def foldRight[A, B](fa: F[A], lb: Eval[B])(f: (A, Eval[B]) => Eval[B]): Eval[B]  
Right associative lazy fold on F using the folding function 'f'.
```

```
abstract def traverse[G[_], A, B](fa: F[A])(f: (A) => G[B])(implicit arg0: Applicative[G]): G[F[B]]  
Given a function which returns a G effect, thread this effect through the running of this function on all the values in F, returning an F[B] in a G context.
```

## Concrete Value Members

```
def as[A, B](fa: F[A], b: B): F[B]  
Replaces the A value in F[A] with the supplied value.
```

```
def collectFirst[A, B](fa: F[A])(pf: PartialFunction[A, B]): Option[B]
```

```
def collectFirstSome[A, B](fa: F[A])(f: (A) => Option[B]): Option[B]  
Like collectFirst from scala.collection.Traversable but takes A => Option[B] instead of PartialFunctions.
```

```
def combineAll[A](fa: F[A])(implicit arg0: Monoid[A]): A  
Alias for fold.
```

---

## Chapter 9: Libraries for Pure Functional Programming

```
def apply[A](body: ⇒ A): IO[A]
```

Suspends a synchronous side effect in IO.

Any exceptions thrown by the effect will be caught and sequenced into the IO.



scala.concurrent

### ExecutionContext

Companion object `ExecutionContext`

```
trait ExecutionContext extends AnyRef
```

```
abstract def execute(runnable: Runnable): Unit
```

Runs a block of code on this execution context.

---

**runnable**    the task to execute

```
def fromExecutor(e: Executor): ExecutionContextExecutor
```

Creates an **ExecutionContext** from the given **Executor** with the default reporter.

---

**e**            the **Executor** to use. If **null**, a new **Executor** is created with default configuration.

**returns**    the **ExecutionContext** using the given **Executor**

```
final def start: IO[Fiber[IO, A]]
```

Start execution of the source suspended in the IO context.

This can be used for non-deterministic / concurrent execution. The following code is more or less equivalent with `parMap2` (minus the behavior on error handling and cancellation):

```
def par2[A, B](ioa: IO[A], iob: IO[B]): IO[(A, B)] =  
  for {  
    fa <- ioa.start  
    fb <- iob.start  
    a <- fa.join  
    b <- fb.join  
  } yield (a, b)
```

Note in such a case usage of `parMapN` (via `cats.Parallel`) is still recommended because of behavior on error and cancellation — consider in the example above what would happen if the first task finishes in error. In that case the second task doesn't get cancelled, which creates a potential memory leak.

IMPORTANT — this operation does not start with an asynchronous boundary. But you can use `IO.shift` to force an async boundary just before start.



cats.effect

## Fiber

Companion [object](#) `Fiber`

```
trait Fiber[F[_], A] extends AnyRef
```

`Fiber` represents the (pure) result of an `Async` data type (e.g. `IO`) being started concurrently and that can be either joined or cancelled.

You can think of fibers as being lightweight threads, a fiber being a concurrency primitive for doing cooperative multi-tasking.

### Abstract Value Members

```
abstract def cancel: F[Unit]
```

Triggers the cancellation of the fiber.

Returns a new task that will complete when the cancellation is sent (but not when it is observed or acted upon).

Note that if the background process that's evaluating the result of the underlying fiber is already complete, then there's nothing to cancel.

```
abstract def join: F[A]
```

Returns a new task that will await for the completion of the underlying fiber, (asynchronously) blocking the current run-loop until that result is available.

---

```
def shift(implicit timer: Timer[IO]): IO[Unit]
```

Asynchronous boundary described as an effectful IO, managed by the provided [Timer](#).

This operation can be used in **flatMap** chains to "shift" the continuation of the run-loop to another thread or call stack.

For example we can introduce an asynchronous boundary in the **flatMap** chain before a certain task:

```
IO.shift.flatMap(_ => task)
```

Or using Cats syntax:

```
import cats.syntax.all._
```

```
IO.shift *> task
```

Or we can specify an asynchronous boundary *after* the evaluation of a certain task:

```
task.flatMap(a => IO.shift.map(_ => a))
```

Or using Cats syntax:

```
task <* IO.shift
```

```
implicit def timer(implicit ec: ExecutionContext): Timer[IO]
```

Returns a [Timer](#) instance for IO, built from a Scala **ExecutionContext**.


N.B. this is the JVM-specific version. On top of JavaScript the implementation needs no **ExecutionContext**.

**ec** is the execution context used for actual execution tasks (e.g. bind continuations)

*Definition Classes* [IOTimerRef](#)

```
1. bash
[info] Application root not yet started
[info] Starting application root in the background ...
root Starting jvm.AsynchronyHeavy.main()
[success] Total time: 3 s, completed Aug 4, 2018 4:27:40 PM
sbt:jvm> root pool-1-thread-2; Task 1: 0
root pool-1-thread-1; Task 2: 0
root pool-1-thread-2; Task 1: 1
root pool-1-thread-1; Task 2: 1
root pool-1-thread-2; Task 1: 2
root pool-1-thread-1; Task 2: 2
root pool-1-thread-2; Task 1: 3
root pool-1-thread-1; Task 2: 3
root pool-1-thread-2; Task 1: 4
root pool-1-thread-1; Task 2: 4
root pool-1-thread-2; Task 1: 5
root pool-1-thread-1; Task 2: 5
root pool-1-thread-2; Task 1: 6
root pool-1-thread-1; Task 2: 6
root pool-1-thread-2; Task 1: 7
root pool-1-thread-1; Task 2: 7
root pool-1-thread-2; Task 1: 8
root pool-1-thread-1; Task 2: 8
root pool-1-thread-2; Task 1: 9
root pool-1-thread-1; Task 2: 9
root pool-1-thread-2; Task 1: 10
root pool-1-thread-1; Task 2: 10
root pool-1-thread-2; Task 1: 11
root pool-1-thread-1; Task 2: 11
root ... killing ...
MacBook-Pro-Anatolii:jvm anatolii$
```

```
1. bash
root pool-1-thread-2; Task 974: 2
root pool-1-thread-1; Task 975: 2
root pool-1-thread-2; Task 976: 2
root pool-1-thread-1; Task 977: 2
root pool-1-thread-2; Task 978: 2
root pool-1-thread-1; Task 979: 2
root pool-1-thread-2; Task 980: 2
root pool-1-thread-1; Task 982: 2
root pool-1-thread-2; Task 981: 2
root pool-1-thread-1; Task 983: 2
root pool-1-thread-2; Task 972: 2
root pool-1-thread-1; Task 984: 2
root pool-1-thread-2; Task 985: 2
root pool-1-thread-1; Task 986: 2
root pool-1-thread-2; Task 987: 2
root pool-1-thread-1; Task 988: 2
root pool-1-thread-2; Task 989: 2
root pool-1-thread-1; Task 990: 2
root pool-1-thread-2; Task 991: 2
root pool-1-thread-1; Task 992: 2
root pool-1-thread-2; Task 993: 2
root pool-1-thread-1; Task 994: 2
root pool-1-thread-2; Task 995: 2
root pool-1-thread-1; Task 996: 2
root pool-1-thread-2; Task 997: 2
root pool-1-thread-1; Task 999: 2
root pool-1-thread-2; Task 998: 2
root pool-1-thread-1; Task 1000: 2
root ... killing ...
MacBook-Pro-Anatolii:jvm anatolii$
```



cats.effect  
**Timer**

Companion [object Timer](#)

```
trait Timer[F[_]] extends AnyRef
```

Timer is a scheduler of tasks.

This is the purely functional equivalent of:

- Java's [ScheduledExecutorService](#)
- JavaScript's [setTimeout](#).

It provides:

1. the ability to get the current time
2. thread / call-stack shifting
3. ability to delay the execution of a task with a specified time duration

It does all of that in an F monadic context that can suspend side effects and is capable of asynchronous execution (e.g. [IO](#)).

This is NOT a type class, as it does not have the coherence requirement.

**Annotations**     @implicitNotFound( ... )

**Source**             [Timer.scala](#)



## Abstract Value Members

- abstract def **clockMonotonic**(unit: TimeUnit): F[Long]  
Returns a monotonic clock measurement, if supported by the underlying platform.
- abstract def **clockRealTime**(unit: TimeUnit): F[Long]  
Returns the current time, as a Unix timestamp (number of time units since the Unix epoch), suspended in F[\_].
- abstract def **shift**: F[Unit]  
Asynchronous boundary described as an effectful F[\_] that can be used in **flatMap** chains to "shift" the continuation of the run-loop to another thread or call stack.
- abstract def **sleep**(duration: FiniteDuration): F[Unit]  
Creates a new task that will sleep for the given duration, emitting a tick when that time span is over.

```
1. java
[1] jvm.AsynchronyHeavy
[2] jvm.AsynchronyLight
[3] jvm.Bracket
[4] jvm.FibersCancelled
[5] jvm.FibersParallel
[6] jvm.FibersSequential
[7] jvm.HelloWorld
[8] jvm.ProductRight
[9] jvm.SequentialTraverse
[10] jvm.SyncVsAsyncAsync
[11] jvm.SyncVsAsyncSync

Enter number: 11

[info] Application root not yet started
[info] Starting application root in the background ...
root Starting jvm.SyncVsAsyncSync.main()
[success] Total time: 7 s, completed Aug 4, 2018 4:30:30 PM
sbt:jvm> root pool-1-thread-2: Task 1: Computed!
root pool-1-thread-1: Task 2: Computed!
root pool-1-thread-1: Task 4: Computed!
root pool-1-thread-2: Task 3: Computed!
root pool-1-thread-1: Task 5: Computed!
root pool-1-thread-2: Task 6: Computed!
root pool-1-thread-1: Task 7: Computed!
root pool-1-thread-2: Task 8: Computed!
root pool-1-thread-1: Task 9: Computed!
root pool-1-thread-2: Task 10: Computed!
root Computed result List(42, 42, 42, 42, 42, 42, 42, 42, 42, 42) in 5 seconds
```

```
def async[A](k: ((Either[Throwable, A]) => Unit) => Unit): IO[A]
```

Suspends an asynchronous side effect in IO.

The given function will be invoked during evaluation of the IO to "schedule" the asynchronous callback, where the callback is the parameter passed to that function. Only the *first* invocation of the callback will be effective! All subsequent invocations will be silently dropped.

```
1. java
[2] jvm.AsynchronyLight
[3] jvm.Bracket
[4] jvm.FibersCancelled
[5] jvm.FibersParallel
[6] jvm.FibersSequential
[7] jvm.HelloWorld
[8] jvm.ProductRight
[9] jvm.SequentialTraverse
[10] jvm.SyncVsAsyncAsync
[11] jvm.SyncVsAsyncSync

Enter number: 10

[info] Stopping application root (by killing the forked JVM) ...
[info] Starting application root in the background ...
root Starting jvm.SyncVsAsyncAsync.main()
root .. finished with exit code 143
[success] Total time: 3 s, completed Aug 4, 2018 4:38:45 PM
sbt:Jvm> root Thread-0: Task 1: Computed!
root Thread-3: Task 4: Computed!
root Thread-1: Task 2: Computed!
root Thread-2: Task 3: Computed!
root Thread-4: Task 5: Computed!
root Thread-5: Task 6: Computed!
root Thread-6: Task 7: Computed!
root Thread-8: Task 9: Computed!
root Thread-9: Task 10: Computed!
root Thread-7: Task 8: Computed!
root Computed result List(42, 42, 42, 42, 42, 42, 42, 42, 42, 42) in 2 seconds
```

```
def pure[A](a: A): IO[A]
```

Suspends a pure value in IO.

This should *only* be used if the value in question has "already" been computed! In other words, something like `IO.pure(readLine)` is most definitely not the right thing to do! However, `IO.pure(42)` is correct and will be more efficient (when evaluated) than `IO(42)`, due to avoiding the allocation of extra thunks.

```
1. java
[11] jvm.SyncVsAsyncSync
Enter number: 6

[info] Stopping application root (by killing the forked JVM) ...
[info] Starting application root in the background ...
root Starting jvm.FibersSequential.main()
root ... finished with exit code 143
[success] Total time: 5 s, completed Aug 4, 2018 4:32:31 PM
sbt:jvm> root main: Running total from 1 to 10, currently at 1: 0
root pool-1-thread-1: Running total from 1 to 10, currently at 2: 1
root pool-1-thread-2: Running total from 1 to 10, currently at 3: 3
root pool-1-thread-1: Running total from 1 to 10, currently at 4: 6
root pool-1-thread-2: Running total from 1 to 10, currently at 5: 10
root pool-1-thread-1: Running total from 1 to 10, currently at 6: 15
root pool-1-thread-2: Running total from 1 to 10, currently at 7: 21
root pool-1-thread-1: Running total from 1 to 10, currently at 8: 28
root pool-1-thread-2: Running total from 1 to 10, currently at 9: 36
root pool-1-thread-1: Running total from 10 to 20, currently at 10: 0
root pool-1-thread-2: Running total from 10 to 20, currently at 11: 10
root pool-1-thread-1: Running total from 10 to 20, currently at 12: 21
root pool-1-thread-2: Running total from 10 to 20, currently at 13: 33
root pool-1-thread-1: Running total from 10 to 20, currently at 14: 46
root pool-1-thread-2: Running total from 10 to 20, currently at 15: 60
root pool-1-thread-1: Running total from 10 to 20, currently at 16: 75
root pool-1-thread-2: Running total from 10 to 20, currently at 17: 91
root pool-1-thread-1: Running total from 10 to 20, currently at 18: 108
root pool-1-thread-2: Running total from 10 to 20, currently at 19: 126
root Computed result 220 in 9 seconds
```

```
1. java
[11] jvm.SyncVsAsyncSync
Enter number: 5

[info] Stopping application root (by killing the forked JVM) ...
[info] Starting application root in the background ...
root ... finished with exit code 143
root Starting jvm.FibersParallel.main()
[success] Total time: 3 s, completed Aug 4, 2018 4:33:43 PM
sbt:jvm> root pool-1-thread-1: Running total from 1 to 10, currently at 1: 0
root pool-1-thread-2: Running total from 10 to 20, currently at 10: 0
root pool-1-thread-1: Running total from 1 to 10, currently at 2: 1
root pool-1-thread-2: Running total from 10 to 20, currently at 11: 10
root pool-1-thread-1: Running total from 1 to 10, currently at 3: 3
root pool-1-thread-2: Running total from 10 to 20, currently at 12: 21
root pool-1-thread-2: Running total from 10 to 20, currently at 13: 33
root pool-1-thread-1: Running total from 1 to 10, currently at 4: 6
root pool-1-thread-2: Running total from 10 to 20, currently at 14: 46
root pool-1-thread-1: Running total from 1 to 10, currently at 5: 10
root pool-1-thread-2: Running total from 10 to 20, currently at 15: 60
root pool-1-thread-1: Running total from 1 to 10, currently at 6: 15
root pool-1-thread-2: Running total from 10 to 20, currently at 16: 75
root pool-1-thread-1: Running total from 1 to 10, currently at 7: 21
root pool-1-thread-2: Running total from 10 to 20, currently at 17: 91
root pool-1-thread-1: Running total from 1 to 10, currently at 8: 28
root pool-1-thread-2: Running total from 10 to 20, currently at 18: 108
root pool-1-thread-1: Running total from 1 to 10, currently at 9: 36
root pool-1-thread-1: Running total from 10 to 20, currently at 19: 126
root Computed result 220 in 5 seconds
```

```
1. java
[1] jvm.AsynchronyHeavy
[2] jvm.AsynchronyLight
[3] jvm.Bracket
[4] jvm.FibersCancelled
[5] jvm.FibersParallel
[6] jvm.FibersSequential
[7] jvm.HelloWorld
[8] jvm.ProductRight
[9] jvm.SequentialTraverse
[10] jvm.SyncVsAsyncAsync
[11] jvm.SyncVsAsyncSync

Enter number: 4

[info] Stopping application root (by killing the forked JVM) ...
root ... finished with exit code 143
[info] Starting application root in the background ...
root Starting jvm.FibersCancelled.main()
[success] Total time: 3 s, completed Aug 4, 2018 4:34:52 PM
sbt:jvm> root pool-1-thread-1: Running total from 1 to 5, currently at 1: 0
root pool-1-thread-2: Running total from 10 to 20, currently at 10: 0
root pool-1-thread-2: Running total from 1 to 5, currently at 2: 1
root pool-1-thread-1: Running total from 10 to 20, currently at 11: 10
root pool-1-thread-2: Running total from 10 to 20, currently at 12: 21
root pool-1-thread-1: Running total from 1 to 5, currently at 3: 3
root pool-1-thread-2: Running total from 10 to 20, currently at 13: 33
root pool-1-thread-1: Running total from 1 to 5, currently at 4: 6
root pool-1-thread-2: Running total from 10 to 20, currently at 14: 46
root Computed result 15 in 2 seconds
```

```
final def bracket[B](use: (A) => IO[B])(release: (A) => IO[Unit]): IO[B]
```

Returns an IO action that treats the source task as the acquisition of a resource, which is then exploited by the **use** function and then **released**.

The **bracket** operation is the equivalent of the `try {} catch {} finally {}` statements from mainstream languages.

The **bracket** operation installs the necessary exception handler to release the resource in the event of an exception being raised during the computation, or in case of cancellation.

If an exception is raised, then **bracket** will re-raise the exception *after* performing the **release**. If the resulting task gets cancelled, then **bracket** will still perform the **release**, but the yielded task will be non-terminating (equivalent with `IO.never`).

```
def raiseError[A](e: Throwable): IO[A]
```

Constructs an IO which sequences the specified exception.

If this IO is run using `unsafeRunSync` or `unsafeRunTimed`, the exception will be thrown. This exception can be "caught" (or rather, materialized into value-space) using the `attempt` method.

---

*See also*      [IO#attempt](#)

```
1.java
[warn] Multiple main classes detected. Run 'show discoveredMainClasses' to see the list

Multiple main classes detected, select one to run:

[1] jvm.AsynchronyHeavy
[2] jvm.AsynchronyLight
[3] jvm.Bracket
[4] jvm.FibersCancelled
[5] jvm.FibersParallel
[6] jvm.FibersSequential
[7] jvm.HelloWorld
[8] jvm.ProductRight
[9] jvm.SequentialTraverse
[10] jvm.SyncVsAsyncAsync
[11] jvm.SyncVsAsyncSync

Enter number: 3

[info] Application root not yet started
root Starting jvm.Bracket.main()
[info] Starting application root in the background ...
[success] Total time: 10 s, completed Aug 4, 2018 4:38:00 PM
sbt:jvm> root Session intercept before execution: null
root Users:
root John
root Ann
root Session intercept after execution: jvm.Bracket$DBSession@5c5a1b69
root Session intercept closed status: true
root ... finished with exit code 0
```

`final val sql: fr0.type`

object `fr0` extends `ProductArgs`

Interpolator for a statement fragment that can contain interpolated values. Unlike `fr` no attempt is made to be helpful with respect to whitespace.

```
macro def applyDynamic(method: String)(args: Any*): Any

def applyProduct[A](a: A)(implicit arg0: util.param.Param\[A\]): util.fragment.Fragment
```



doobie.util.fragment

## Fragment

Companion object **Fragment**

```
sealed trait Fragment extends AnyRef
```

A statement fragment, which may include interpolated values. Fragments can be composed by concatenation, which maintains the correct offset and mappings for interpolated values. Once constructed a **Fragment** is opaque; it has no externally observable properties. Fragments are eventually used to construct a [Query0](#) or [Update0](#).

```
def ++(fb: Fragment): Fragment
```

Concatenate this fragment with another, yielding a larger fragment.

```
def execWith[B](fa: doobie.PreparedStatementIO[B]): doobie.ConnectionIO[B]
```

Construct a program in ConnectionIO that constructs and prepares a PreparedStatement, with further handling delegated to the provided program.

```
def query[B](implicit arg0: Read[B], h: doobie.LogHandler = LogHandler.nop): doobie.Query0[B]
```

Construct a [Query0](#) from this fragment, with asserted row type **B**.

```
def queryWithLogHandler[B](h: doobie.LogHandler)(implicit cb: Read[B]): doobie.Query0[B]
```

Construct a [Query0](#) from this fragment, with asserted row type **B** and the given **LogHandler**.

```
def stripMargin: Fragment
```

```
def stripMargin(marginChar: Char): Fragment
```

```
def toString(): String
```

```
def update(implicit h: doobie.LogHandler = LogHandler.nop): doobie.Update0
```

Construct an [Update0](#) from this fragment.

```
def updateWithLogHandler(h: doobie.LogHandler): doobie.Update0
```

Construct an [Update0](#) from this fragment with the given **LogHandler**.

```
def query[B](implicit arg0: Read[B], h: doobie.LogHandler = LogHandler.nop): doobie.Query0[B]
```

Construct a [Query0](#) from this fragment, with asserted row type **B**.

```
def update(implicit h: doobie.LogHandler = LogHandler.nop): doobie.Update0
```

Construct an [Update0](#) from this fragment.



doobie.util.update

## Update0

Companion [object Update0](#)

trait **Update0** extends AnyRef

### Diagnostics

abstract def **analysis**: [doobie.ConnectionIO](#)[[Analysis](#)]  
Program to construct an analysis of this query's SQL statement and asserted parameter types.

abstract val **pos**: [Option](#)[[Pos](#)]  
An optional **Pos** indicating the source location where this [Query](#) was constructed.

abstract val **sql**: [String](#)  
The SQL string.

### Execution

abstract def **run**: [doobie.ConnectionIO](#)[[Int](#)]  
Program to execute the update and yield a count of affected rows.

abstract def **withGeneratedKeysWithChunkSize**[[K](#)](columns: [String](#)\*)(chunkSize: [Int](#))(*implicit* arg0: [Read](#)[[K](#)]): [Stream](#)[[doobie.ConnectionIO](#), [K](#)]  
Construct a stream that performs the update, yielding generated keys of readable type **K**, identified by the specified columns, given a **chunkSize**.

abstract def **withUniqueGeneratedKeys**[[K](#)](columns: [String](#)\*)(*implicit* arg0: [Read](#)[[K](#)]): [doobie.ConnectionIO](#)[[K](#)]  
Construct a program that performs the update, yielding a single set of generated keys of readable type **K**, identified by the specified columns.

def **withGeneratedKeys**[[K](#)](columns: [String](#)\*)(*implicit* arg0: [Read](#)[[K](#)]): [Stream](#)[[doobie.ConnectionIO](#), [K](#)]  
Construct a stream that performs the update, yielding generated keys of readable type **K**, identified by the specified columns.

type **ConnectionIO**[[A](#)] = [Free](#)[[ConnectionOp](#), [A](#)]





[doobie.syntax](#)

# ConnectionIOOps

```
class ConnectionIOOps[A] extends AnyRef
```

## Instance Constructors

```
new ConnectionIOOps(ma: free.connection.ConnectionIO[A])
```

```
def transact[M[_]](xa: util.transactor.Transactor[M])(implicit arg0: Monad[M]): M[A]
```

```
final val sql: fr0.type
```

Alternative name for the **fr0** interpolator.

```
object fr extends ProductArgs
```

Interpolator for a statement fragment that can contain interpolated values. When inserted into the final SQL statement this fragment will be followed by a space. This is normally what you want, and it makes it easier to concatenate fragments because you don't need to think about intervening whitespace. If you do not want this behavior, use **fr0**.

```
object fr0 extends ProductArgs
```

Interpolator for a statement fragment that can contain interpolated values. Unlike **fr** no attempt is made to be helpful with respect to whitespace.



[doobie.util.query](#)

## Query0

Companion [object Query0](#)

```
trait Query0[B] extends AnyRef
```

An abstract query closed over its input arguments and yielding values of type **B**, without a specified disposition. Methods provided on [Query0](#) allow the query to be interpreted as a stream or program in [CollectionIO](#).

---

## Diagnostics

```
abstract def analysis: doobie.ConnectionIO[Analysis]
  Program to construct an analysis of this query's SQL statement and asserted parameter and column types.
```

```
abstract def outputAnalysis: doobie.ConnectionIO[Analysis]
  Program to construct an analysis of this query's SQL statement and result set column types.
```

```
abstract def pos: Option[Pos]
  An optional Pos indicating the source location where this Query was constructed.
```

```
abstract def sql: String
  The SQL string.
```

## Results

```
abstract def accumulate[F[_]](implicit arg0: Alternative[F]): doobie.ConnectionIO[F[B]]
  Program in ConnectionIO yielding an F[B] accumulated via MonadPlus append.
```

```
abstract def nel: doobie.ConnectionIO[NonEmptyList[B]]
  Program in ConnectionIO yielding a NonEmptyList[B] and raising an exception if the resultset does not have at least one row.
```

```
abstract def option: doobie.ConnectionIO[Option[B]]
  Program in ConnectionIO yielding an optional B and raising an exception if the resultset has more than one row.
```

```
abstract def streamWithChunkSize(n: Int): Stream[doobie.ConnectionIO, B]
  Stream with given chunk factor, with effect type ConnectionIO yielding elements of type B.
```

```
abstract def to[F[_]](implicit cbf: CanBuildFrom[Nothing, B, F[B]]): doobie.ConnectionIO[F[B]]
  Program in ConnectionIO yielding an F[B] accumulated via the provided CanBuildFrom.
```

```
abstract def unique: doobie.ConnectionIO[B]
  Program in ConnectionIO yielding a unique B and raising an exception if the resultset does not have exactly one row.
```

```
def stream: Stream[doobie.ConnectionIO, B]
  Stream with default chunk factor, with effect type ConnectionIO yielding elements of type B.
```

```
def sink(f: (B) => doobie.ConnectionIO[Unit]): doobie.ConnectionIO[Unit]
  Convenience method, equivalent to stream.evalMap(f).compile.drain.
```

```
def ++(fb: Fragment): Fragment
```

Concatenate this fragment with another, yielding a larger fragment.

```

1. docker
x docker-compose %1 x docker %2 x docker %3 x bash %4
[12] jvm.server.Server
[13] jvm.server.db.CustomerDB

Enter number: 13

[info] Stopping application root (by killing the forked JVM) ...
root Starting jvm.server.db.CustomerDB.main()
[info] Starting application root in the background ...
root .. finished with exit code 143
[success] Total time: 6 s, completed Aug 7, 2018 6:50:34 PM
sbt:jvm> root Looking up customers by name
root Some(Customer(Some(1),John Smith))
root None
root
root All customers
root Customer(Some(1),John Smith)
root Customer(Some(2),Ann Watson)
root
root Customer with id 1
root Customer(Some(1),John Smith)
root
root Update customer with id 1
root Rows affected: 1
root Updated customer: Customer(Some(1),Bob)
root
root Clean-up: remove all customers
root Customers table after clean-up: List()
root .. finished with exit code 0

sbt:jvm>


```

```

def apply[F[_]](pf: PartialFunction[Request[F], F[Response[F]]])(implicit F:
  Applicative[F]): HttpService[F]

Lifts a partial function to an HttpService. Responds with OptionT.none for any request where pf is not
defined.

```



org.http4s

# Request

Companion [object Request](#)

---

sealed abstract case class **Request**[F[\_]] extends [Message](#)[F] with [RequestOps](#)[F] with Product with Serializable

## Type Members

```
type Self = Request[F]
```

## Value Members

```
final def addCookie(name: String, content: String, expires: Option[HttpDate] = None)(implicit F: Functor[F]): Self
```

Add a Cookie header with the provided values

```
final def addCookie(cookie: Cookie)(implicit F: Functor[F]): Self
```

Add a Cookie header for the provided [Cookie](#)

```
final def as[T](implicit F: FlatMap[F], decoder: EntityDecoder[F, T]): F[T]
```

Decode the [Message](#) to the specified type

```
def attemptAs[T](implicit F: FlatMap[F], decoder: EntityDecoder[F, T]): DecodeResult[F, T]
```

Decode the [Message](#) to the specified type

```
val attributes: AttributeMap
```

```
lazy val authType: Option[AuthScheme]
```

```
val body: EntityBody[F]
```

```
def apply[F[_]](pf: PartialFunction[Request[F], F[Response[F]]))(implicit F: Applicative[F]): HttpService[F]
```

Lifts a partial function to an **HttpService**.



cats.data

## Kleisli

Companion [object](#) **Kleisli**

```
final case class Kleisli[F[_], A, B](run: (A => F[B])) extends Product with Serializable
```

Represents a function **A => F[B]**.

**Self Type**

[Kleisli](#)[F, A, B]

```
val Ok: Status
```



[org.http4s.dsl.impl.Responses](http://org.http4s.dsl.impl.Responses)

## OkOps

```
final class OkOps[F[_]] extends AnyVal with EntityResponseGenerator[F]
```

### Instance Constructors

```
new OkOps(status: Ok.type)
```

### Value Members

```
def apply[A](body: A, headers: Header*)(implicit F: Monad[F], w: EntityEncoder[F, A]):  
  F[Response[F]]
```

```
def apply(headers: Header*)(implicit F: Applicative[F]): F[Response[F]]
```

```
def getClass(): Class[_ <: AnyVal]
```

```
val status: Ok.type
```

```
1. docker
x docker-compose %1 x docker %2 x docker %3 x bash %4
Usage: ./client.sh command [arguments]
Possible commands:
* cc - create customer
* po - place order
* lo - list orders
* lg - list goods
root@26d8b94a0e92:~/examples/Chapter9/bash# ./client.sh cc
Usage: ./client.sh cc customer_name
root@26d8b94a0e92:~/examples/Chapter9/bash# ./client.sh cc "Bob"
Note: Unnecessary use of -X or --request, POST is already inferred.
* Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 8888 (#0)
> POST /customer HTTP/1.1
> Host: localhost:8888
> User-Agent: curl/7.52.1
> Accept: */*
> Content-Length: 15
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 15 out of 15 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json
< Date: Tue, 07 Aug 2018 18:55:50 GMT
< Content-Length: 15
<
* Curl_http_done: called premature == 0
* Connection #0 to host localhost left intact
{"success": "4"}
root@26d8b94a0e92:~/examples/Chapter9/bash#
```

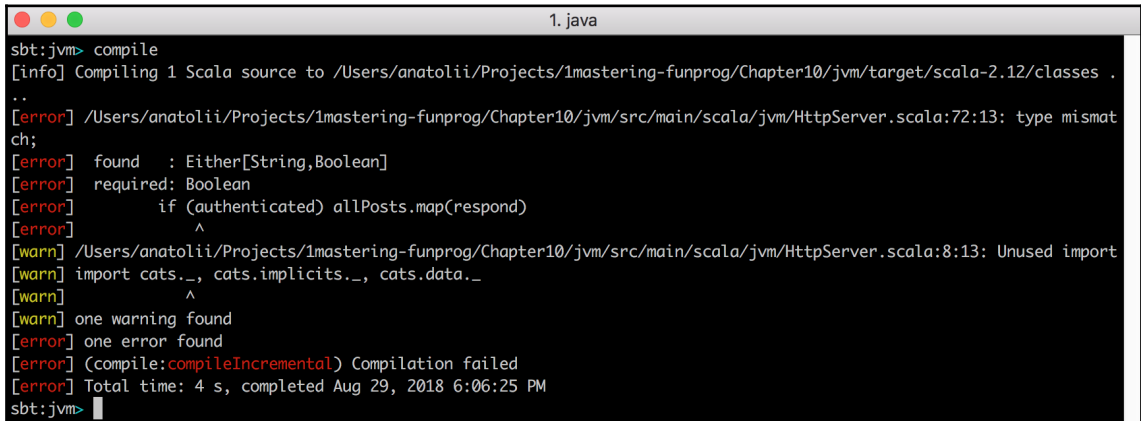
```
1. docker
x docker-compose %1 x docker %2 x docker %3 x bash %4
root@26d8b94a0e92:~/examples/Chapter9/bash# ./client.sh lg
Note: Unnecessary use of -X or --request, GET is already inferred.
* Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 8888 (#0)
> GET /good HTTP/1.1
> Host: localhost:8888
> User-Agent: curl/7.52.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Date: Tue, 07 Aug 2018 18:56:26 GMT
< Content-Length: 289
<
* Curl_http_done: called premature == 0
* Connection #0 to host localhost left intact
[{"id":1,"name":"MacBook Pro 15","price":2500.0,"stock":15},{"id":2,"name":"iPhone 10","price":1000.0,"stock":10},{"id":3,"name":"MacBook Air","price":900.0,"stock":3},{"id":4,"name":"Samsung Galaxy S5","price":500.0,"stock":8},{"id":5,"name":"Panasonic Camera","price":120.0,"stock":34}]
root@26d8b94a0e92:~/examples/Chapter9/bash#
```

```
1. docker
docker-compose  %1  docker  %2  docker  %3  bash  %4
root@26d8b94a0e92:~/examples/Chapter9/bash# ./client.sh po 6 1
Note: Unnecessary use of -X or --request, POST is already inferred.
* Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 8888 (#0)
> POST /order HTTP/1.1
> Host: localhost:8888
> User-Agent: curl/7.52.1
> Accept: */*
> Cookie: shop_customer_id=6
> Content-Length: 11
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 11 out of 11 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json
< Date: Tue, 07 Aug 2018 18:57:48 GMT
< Content-Length: 42
<
* Curl_http_done: called premature == 0
* Connection #0 to host localhost left intact
{"success":{"id":3,"customer":6,"good":1}}
root@26d8b94a0e92:~/examples/Chapter9/bash#
```

```
1. docker
docker-compose  %1  docker  %2  docker  %3  bash  %4
root@26d8b94a0e92:~/examples/Chapter9/bash# ./client.sh lo
Note: Unnecessary use of -X or --request, GET is already inferred.
* Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 8888 (#0)
> GET /order HTTP/1.1
> Host: localhost:8888
> User-Agent: curl/7.52.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Date: Tue, 07 Aug 2018 18:58:09 GMT
< Content-Length: 32
<
* Curl_http_done: called premature == 0
* Connection #0 to host localhost left intact
[{"id":3,"customer":6,"good":1}]
root@26d8b94a0e92:~/examples/Chapter9/bash#
```

---

# Chapter 10: Patterns of Advanced Functional Programming



```
1. java
sbt:jvm> compile
[info] Compiling 1 Scala source to /Users/anatolii/Projects/1mastering-funprog/Chapter10/jvm/target/scala-2.12/classes .
..
[error] /Users/anatolii/Projects/1mastering-funprog/Chapter10/jvm/src/main/scala/jvm/HttpServer.scala:72:13: type mismatch;
ch;
[error] found   : Either[String,Boolean]
[error] required: Boolean
[error]     if (authenticated) allPosts.map(respond)
[error]     ^
[warn] /Users/anatolii/Projects/1mastering-funprog/Chapter10/jvm/src/main/scala/jvm/HttpServer.scala:8:13: Unused import
[warn] import cats._, cats.implicits._, cats.data._
[warn]     ^
[warn] one warning found
[error] one error found
[error] (compile:compileIncremental) Compilation failed
[error] Total time: 4 s, completed Aug 29, 2018 6:06:25 PM
sbt:jvm> |
```



```
def flatMap[S](f: (T) => Future[S])(implicit executor: ExecutionContext):  
Future[S]
```

Creates a new future by applying a function to the successful result of this future, and returns the result of the function as the new future. If this future is completed with an exception then the new future will also contain this exception.

Example:

```
val f = Future { 5 }  
val g = Future { 3 }  
val h = for {  
  x: Int <- f // returns Future(5)  
  y: Int <- g // returns Future(3)  
} yield x + y
```

is translated to:

```
f flatMap { (x: Int) => g map { (y: Int) => x + y } }
```

**S** the type of the returned **Future**  
**f** the function which will be applied to the successful result of this **Future**  
**returns** a **Future** which will be completed with the result of the application of the function



cats.data

# EitherT

Companion object **EitherT**

```
final case class EitherT[F[_], A, B](value: F[Either[A, B]]) extends Product with  
Serializable
```

Transformer for **Either**, allowing the effect of an arbitrary type constructor **F** to be combined with the fail-fast effect of **Either**.

**EitherT[F, A, B]** wraps a value of type **F[Either[A, B]]**. An **F[C]** can be lifted in to **EitherT[F, A, C]** via **EitherT.right**, and lifted in to a **EitherT[F, C, B]** via **EitherT.left**.

```
def flatMap[AA >: A, D](f: (B) => EitherT[F, AA, D])(implicit F: Monad[F]):  
EitherT[F, AA, D]
```

```
1.java
[warn] Multiple main classes detected. Run 'show discoveredMainClasses' to see the list

Multiple main classes detected, select one to run:

[1] jvm.HListSum
[2] jvm.ListSum
[3] jvm.TaglessFinalExample

Enter number: 3

[info] Application root not yet started
[info] Starting application root in the background ...
root Starting jvm.TaglessFinalExample.main()
[success] Total time: 2 s, completed Jul 26, 2018 4:25:37 PM
sbt:jvm> root Trace of the Future's result: Name,Price
root Bread,2
root Butter,4
root Cabbage,3
root Water,2
root
root Trace of the Future's result: (Name,Price
root Bread,2
root Butter,4
root Cabbage,3
root Water,2
root ,11.0)
root Notifying admin@shop.com: Total income made today: 11.0
root Trace of the Future's result: ()
root ... finished with exit code 0
```

```
1.java
x java %1 x bash %2
sbt:jvm> reStart
[info] Compiling 1 Scala source to /Users/anatolii/Projects/1mastering-funprog/Chapter10/jvm/target/scala-2.12/classes .
..
[info] Done compiling.
[warn] Multiple main classes detected. Run 'show discoveredMainClasses' to see the list

Multiple main classes detected, select one to run:

[1] jvm.HListSum
[2] jvm.ListSum
[3] jvm.TaglessFinalExample

Enter number: 2

[info] Application root not yet started
[info] Starting application root in the background ...
root Starting jvm.ListSum.main()
root 7.0
root ... finished with exit code 0
[success] Total time: 8 s, completed Jul 26, 2018 4:39:38 PM
sbt:jvm>
```

```
final def map[B](f: (A) => B): List[B]
```

[use case]

Builds a new collection by applying a function to all elements of this list.

**B** the element type of the returned collection.

**f** the function to apply to each element.

**returns** a new list resulting from applying the given function **f** to each element of this list and collecting the results.

```
1. java
x java %1 x bash %2
Multiple main classes detected, select one to run:
[1] jvm.HListSum
[2] jvm.ListSum
[3] jvm.TaglessFinalExample
Enter number: 2
[info] Application root not yet started
[info] Starting application root in the background ...
root Starting jvm.ListSum.main()
[success] Total time: 6 s, completed Jul 26, 2018 4:40:43 PM
sbt:jvm> root [ERROR] Exception in thread "main" scala.MatchError: Fraction(4,2) (of class jvm.Fraction)
root [ERROR] at jvm.ListSum$.anonfun$sum$1(HListSum.scala:10)
root [ERROR] at jvm.ListSum$.anonfun$sum$1$adapted(HListSum.scala:7)
root [ERROR] at scala.collection.immutable.List.map(List.scala:287)
root [ERROR] at jvm.ListSum$.delayedEndpoint$jvm$ListSum$1(HListSum.scala:7)
root [ERROR] at jvm.ListSum$delayedInit$body.apply(HListSum.scala:5)
root [ERROR] at scala.Function0.apply$mcV$sp(Function0.scala:34)
root [ERROR] at scala.Function0.apply$mcV$sp$(Function0.scala:34)
root [ERROR] at scala.runtime.AbstractFunction0.apply$mcV$sp(AbstractFunction0.scala:12)
root [ERROR] at scala.App$.anonfun$main$1$adapted(App.scala:76)
root [ERROR] at scala.collection.immutable.List.foreach(List.scala:389)
root [ERROR] at scala.App.main(App.scala:76)
root [ERROR] at scala.App.main$(App.scala:74)
root [ERROR] at jvm.ListSum$.main(HListSum.scala:5)
root [ERROR] at jvm.ListSum.main(HListSum.scala)
root ... finished with exit code 1
```

```
def implicitly[T](implicit e: T): T
```

**Annotations**

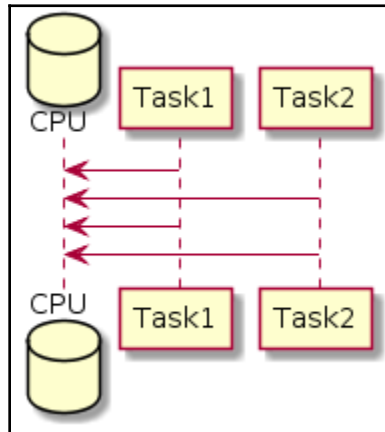
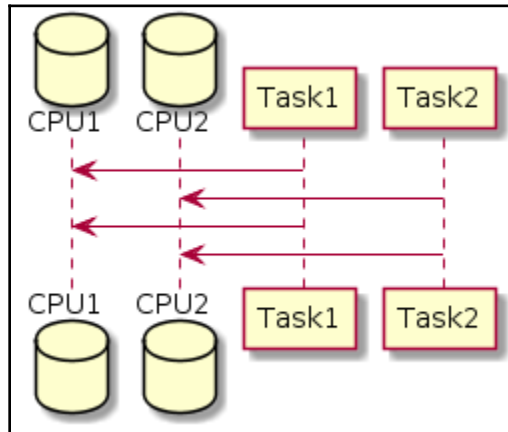
**@inline()**

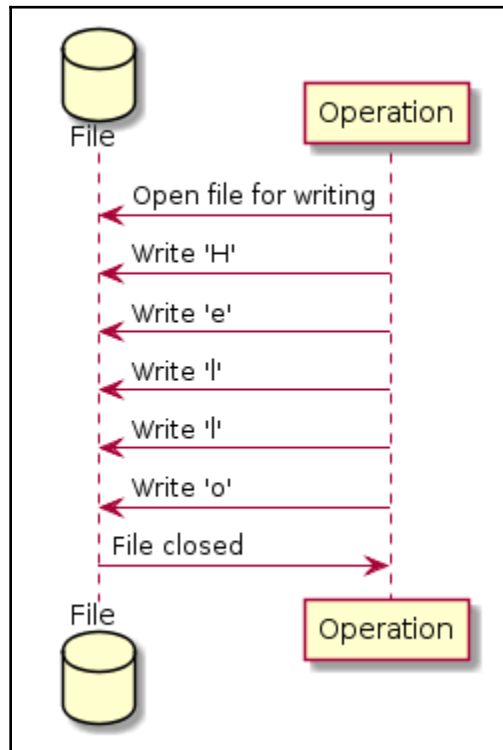
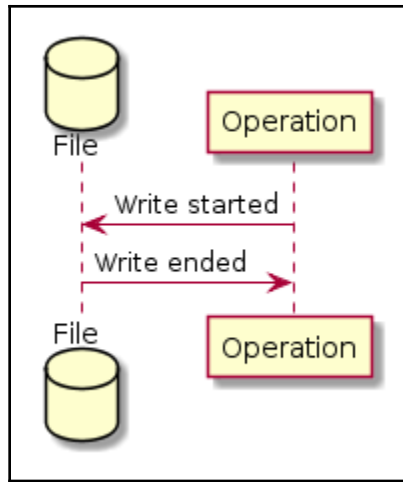
```
1.java
x java %1 x bash %2
sbt:jvm> reStart
[info] Compiling 1 Scala source to /Users/anatolii/Projects/1mastering-funprog/Chapter10/jvm/target/scala-2.12/classes .
...
[error] /Users/anatolii/Projects/1mastering-funprog/Chapter10/jvm/src/main/scala/jvm/HListSum.scala:32:14: could not find
implicit value for parameter m: jvm.MapToDouble.Aux[String :: Int :: jvm.Fraction :: jvm.HNil,LR]
[error]   val s = sum(hlist)
[error]         ^
[error] one error found
[error] (compile:compileIncremental) Compilation failed
[error] Total time: 1 s, completed Jul 26, 2018 5:11:10 PM
sbt:jvm>
```

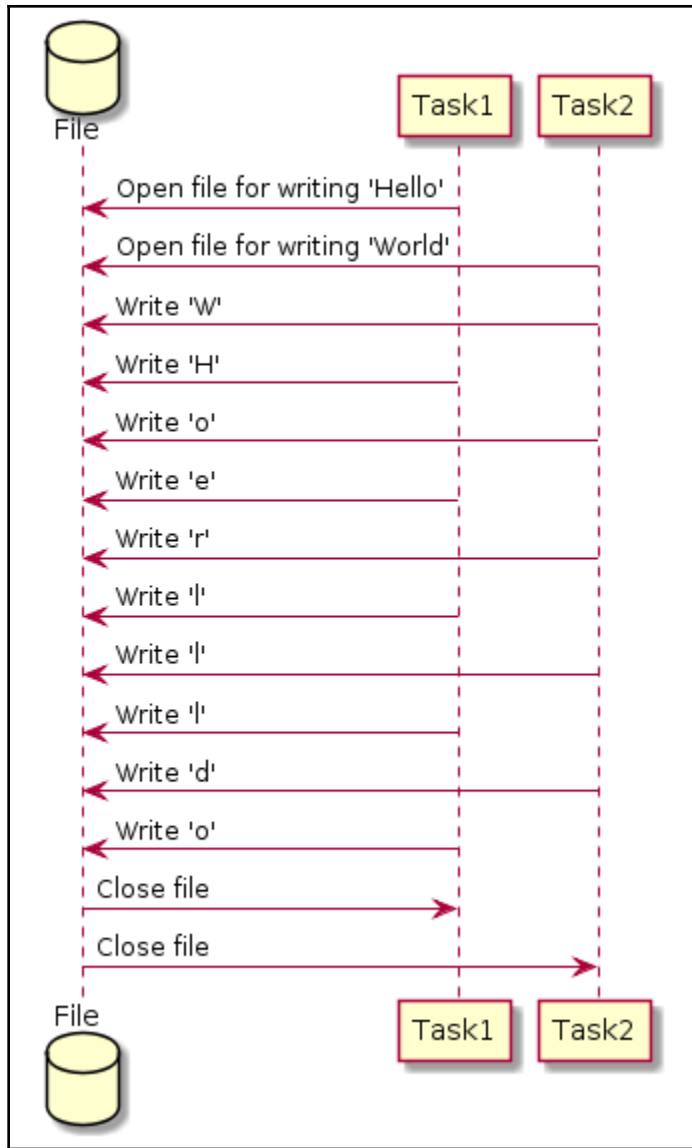
```
1.java
x java %1 x bash %2
sbt:jvm> reStart
[info] Compiling 1 Scala source to /Users/anatolii/Projects/1mastering-funprog/Chapter10/jvm/target/scala-2.12/classes .
...
[error] /Users/anatolii/Projects/1mastering-funprog/Chapter10/jvm/src/main/scala/jvm/HListSum.scala:32:13: could not find
implicit value for parameter e: jvm.MapToDouble[String :: Int :: jvm.Fraction :: jvm.HNil]
[error]   implicitly[MapToDouble[String :: Int :: Fraction :: HNil]]
[error]         ^
[error] /Users/anatolii/Projects/1mastering-funprog/Chapter10/jvm/src/main/scala/jvm/HListSum.scala:33:13: could not find
implicit value for parameter e: jvm.MapToDouble[Int :: jvm.Fraction :: jvm.HNil]
[error]   implicitly[MapToDouble[Int :: Fraction :: HNil]]
[error]         ^
[error] /Users/anatolii/Projects/1mastering-funprog/Chapter10/jvm/src/main/scala/jvm/HListSum.scala:34:13: could not find
implicit value for parameter e: jvm.MapToDouble[jvm.Fraction :: jvm.HNil]
[error]   implicitly[MapToDouble[Fraction :: HNil]]
[error]         ^
[error] /Users/anatolii/Projects/1mastering-funprog/Chapter10/jvm/src/main/scala/jvm/HListSum.scala:35:13: could not find
implicit value for parameter e: jvm.ToDouble[jvm.Fraction]
[error]   implicitly[ToDouble[Fraction]]
[error]         ^
[error] four errors found
[error] (compile:compileIncremental) Compilation failed
[error] Total time: 0 s, completed Jul 26, 2018 5:13:54 PM
sbt:jvm>
```

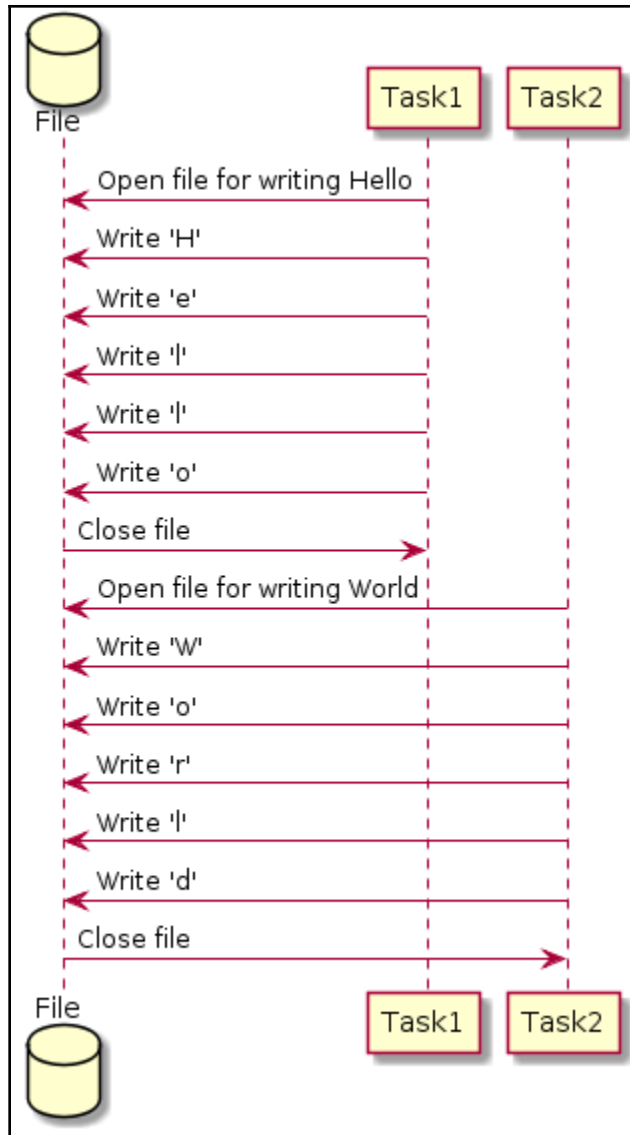
---

# Chapter 11: Introduction to the Actor Model

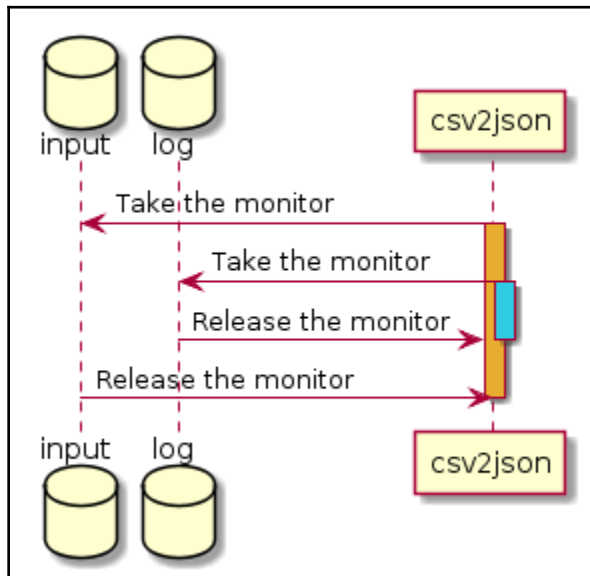
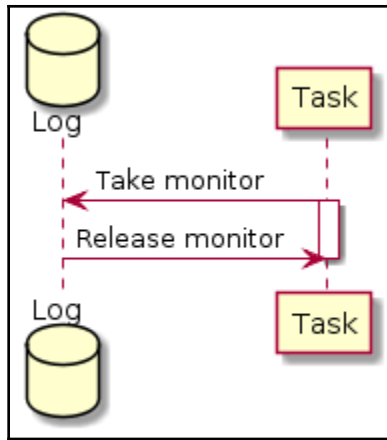


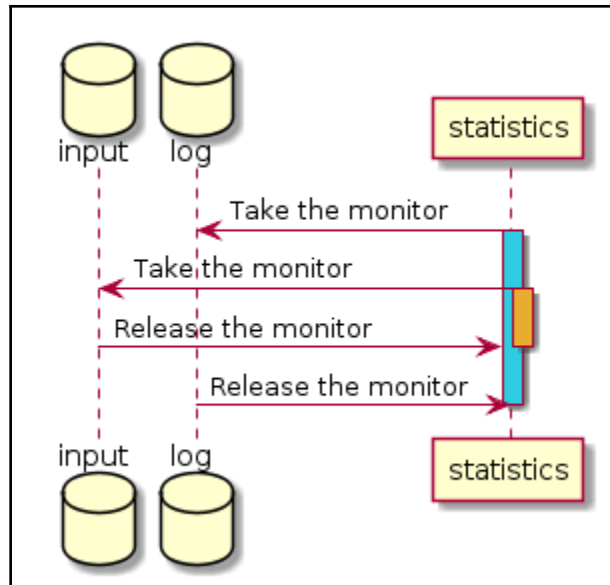


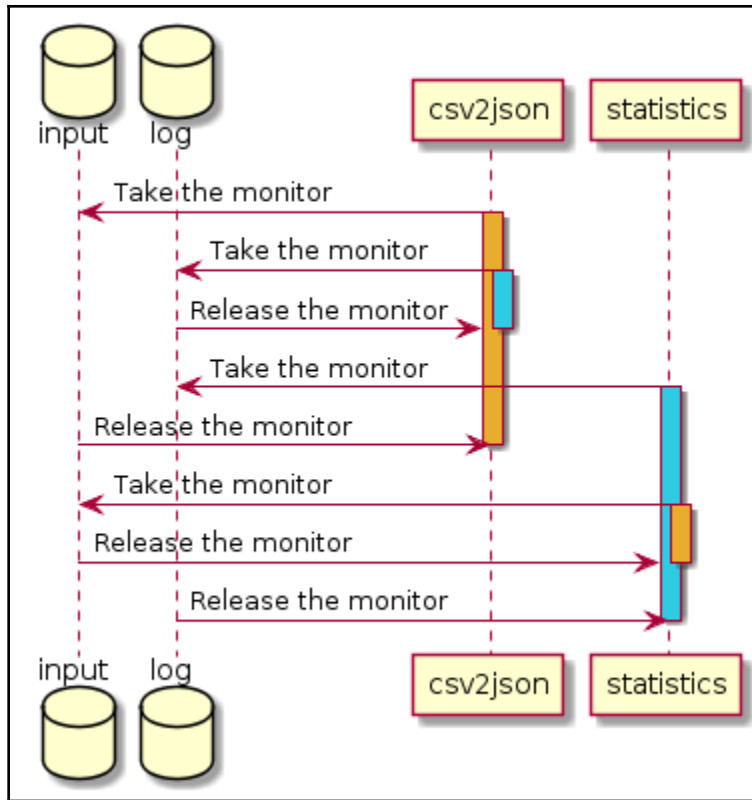


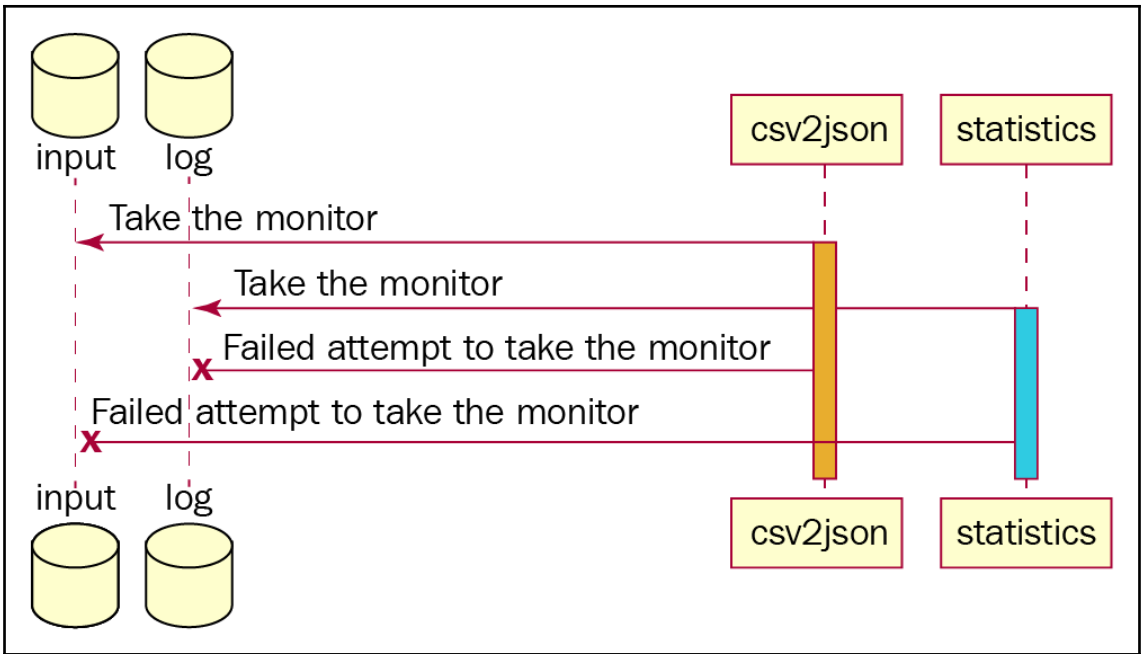


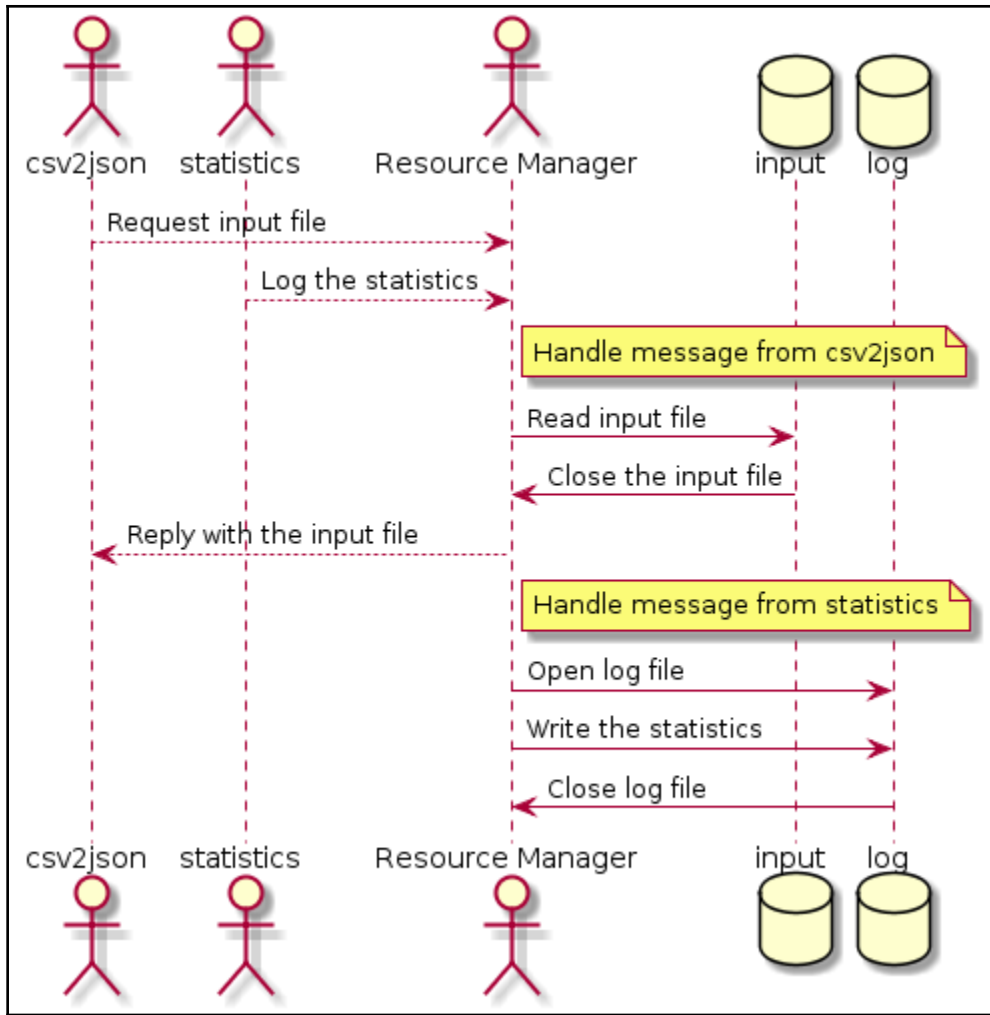












---

# Chapter 12: The Actor Model in Practice

## Type Members

```
type Receive = PartialFunction[Any, Unit]
```

## Abstract Value Members

```
abstract def receive: Actor.Receive
```

Scala API: This defines the initial actor behavior, it must return a partial function with the actor logic.

```
def apply[T](system: ActorSystem, logSource: T)(implicit arg0: LogSource[T]):  
LoggingAdapter
```

Obtain `LoggingAdapter` for the given actor system and source object. This will use the system's event stream and include the system's address in the log source string.

**Do not use this if you want to supply a log category string (like "com.example.app.whatever") unaltered, supply `system.eventStream` in this case or use**

```
Logging(system, this.getClass)
```

The source is used to identify the source of this logging channel and must have a corresponding implicit `LogSource[T]` instance in scope; by default these are provided for `Class[_]`, `Actor`, `ActorRef` and `String` types. See the companion object of [akka.event.LogSource](#) for details.

You can add your own rules quite easily, see [akka.event.LogSource](#).



akka.event

## Logging

object **Logging**

Main entry point for Akka logging: log levels and message types (aka channels) defined for the main transport medium, the main event bus. The recommended use is to obtain an implementation of the `Logging` trait with suitable and efficient methods for generating log events:

```
val log = Logging(<bus>, <source object>)  
...  
log.info("hello world!")
```

```
def postRestart(reason: Throwable): Unit
```

User overridable callback: By default it calls **preStart()**.

```
def postStop(): Unit
```

User overridable callback.

```
def preRestart(reason: Throwable, message: Option[Any]): Unit
```

Scala API: User overridable callback: **By default it disposes of all children and then calls postStop()**.

```
def preStart(): Unit
```

User overridable callback.

```
def unhandled(message: Any): Unit
```

User overridable callback.

Is called when a message isn't handled by the current behavior of the actor by default it fails with either a [akka.actor.DeathPactException](#) (in case of an unhandled [akka.actor.Terminated](#) message) or publishes an [akka.actor.UnhandledMessage](#) to the actor's system's [akka.event.EventStream](#)

```
def supervisorStrategy: SupervisorStrategy
```

User overridable definition the strategy to use for supervising child actors.



akka.actor

## SupervisorStrategy

Companion object [SupervisorStrategy](#)

```
abstract class SupervisorStrategy extends AnyRef
```

An Akka SupervisorStrategy is the policy to apply for crashing children.

### IMPORTANT:

You should not normally need to create new subclasses, instead use the existing [akka.actor.OneForOneStrategy](#) or [akka.actor.AllForOneStrategy](#), but if you do, please read the docs of the methods below carefully, as incorrect implementations may lead to "blocked" actor systems (i.e. permanently suspended actors).

**Source** [FaultHandling.scala](#)

Linear Supertypes

Known Subclasses

[AllForOneStrategy](#), [OneForOneStrategy](#)



akka.actor

## AllForOneStrategy

```
case class AllForOneStrategy(maxNrOfRetries: Int = -1, withinTimeRange: Duration = Duration.Inf,  
loggingEnabled: Boolean = true)(decider: Decider) extends SupervisorStrategy with Product with  
Serializable
```

Applies the fault handling **Directive** (Resume, Restart, Stop) specified in the **Decider** to all children when one fails, as opposed to `akka.actor.OneForOneStrategy` that applies it only to the child actor that failed.



akka.actor

## OneForOneStrategy

```
case class OneForOneStrategy(maxNrOfRetries: Int = -1, withinTimeRange: Duration = Duration.Inf,  
loggingEnabled: Boolean = true)(decider: Decider) extends SupervisorStrategy with Product with  
Serializable
```

Applies the fault handling **Directive** (Resume, Restart, Stop) specified in the **Decider** to the child actor that failed, as opposed to `akka.actor.AllForOneStrategy` that applies it to all children.

```
type Decider = PartialFunction[Throwable, Directive]
```





`akka.actor.SupervisorStrategy`

# Directive

sealed trait **Directive** extends [AnyRef](#)

**Source**

[FaultHandling.scala](#)

Linear Supertypes

Known Subclasses

[Escalate](#), [Restart](#), [Resume](#), [Stop](#)



`akka.actor.SupervisorStrategy`

## Escalate

object **Escalate** extends [Directive](#) with [Product](#) with [Serializable](#)

Escalates the failure to the supervisor of the supervisor, by rethrowing the cause of the failure, i.e. the supervisor fails with the same exception as the child.



[akka.actor.SupervisorStrategy](#)

## Restart

object **Restart** extends [Directive](#) with [Product](#) with [Serializable](#)

Discards the old Actor instance and replaces it with a new, then resumes message processing.



[akka.actor.SupervisorStrategy](#)

## Resume

object **Resume** extends [Directive](#) with [Product](#) with [Serializable](#)

Resumes message processing for the failed Actor



[akka.actor.SupervisorStrategy](#)

## Stop

object **Stop** extends [Directive](#) with [Product](#) with [Serializable](#)

Stops the Actor

---

```
implicit final val self: ActorRef
```

The 'self' field holds the ActorRef for this actor.

Can be used to send messages to itself:

```
self ! message
```

---

```
final def sender(): ActorRef
```

The reference sender Actor of the last received message. Is defined if the message was sent from another Actor, else **deadLetters** in [akka.actor.ActorSystem](#).

WARNING: Only valid within the Actor itself, so do not close over it and publish it to other threads!

```
implicit val context: ActorContext
```

Scala API: Stores the context for this actor, including self, and sender. It is implicit to support operations such as **forward**.

WARNING: Only valid within the Actor itself, so do not close over it and publish it to other threads!

[akka.actor.ActorContext](#) is the Scala API. **getContext** returns a [akka.actor.AbstractActor.ActorContext](#), which is the Java API of the actor context.



[akka.actor](#)

## ActorContext

```
trait ActorContext extends ActorRefFactory
```

The actor context - the view of the actor cell from the actor. Exposes contextual information for the actor and the current message.

```
abstract def actorOf(props: Props, name: String): ActorRef
```

Create new actor as child of this context with the given name, which must not be null, empty or start with "\$".

```
abstract def actorOf(props: Props): ActorRef
```

Create new actor as child of this context and give it an automatically generated name (currently similar to base64-encoded integer count, reversed and with "\$" prepended, may change in the future).

---

```
abstract def child(name: String): Option[ActorRef]
```

Get the child with the given name if it exists.

**\*Warning\***: This method is not thread-safe and must not be accessed from threads other than the ordinary actor message processing thread, such as `java.util.concurrent.CompletionStage` and `scala.concurrent.Future` callbacks.

```
abstract def children: Iterable[ActorRef]
```

Returns all supervised children; this method returns a view (i.e. a lazy collection) onto the internal collection of children. Targeted lookups should be using `child` instead for performance reasons:

```
val badLookup = context.children find (_.path.name == "kid")  
// should better be expressed as:  
val goodLookup = context.child("kid")
```

**\*Warning\***: This method is not thread-safe and must not be accessed from threads other than the ordinary actor message processing thread, such as `java.util.concurrent.CompletionStage` and `scala.concurrent.Future` callbacks.

```
abstract def parent: ActorRef
```

Returns the supervising parent `ActorRef`.

This method is thread-safe and can be called from other threads than the ordinary actor message processing thread, such as `java.util.concurrent.CompletionStage` and `scala.concurrent.Future` callbacks.

```
implicit abstract def system: ActorSystem
```

The system that the actor belongs to. Importing this member will place an implicit `ActorSystem` in scope.

This method is thread-safe and can be called from other threads than the ordinary actor message processing thread, such as `java.util.concurrent.CompletionStage` and `scala.concurrent.Future` callbacks.

```
abstract def stop(actor: ActorRef): Unit
```

Stop the actor pointed to by the given `akka.actor.ActorRef`; this is an asynchronous operation, i.e. involves a message send. If this method is applied to the `self` reference from inside an Actor then that Actor is guaranteed to not process any further messages after this call; please note that the processing of the current message will continue, this method does not immediately terminate this actor.

**Definition Classes** [ActorRefFactory](#)

---

```
abstract def become(behavior: Receive, discardOld: Boolean): Unit
```

Changes the Actor's behavior to become the new 'Receive' (PartialFunction[Any, Unit]) handler. This method acts upon the behavior stack as follows:

- if **discardOld = true** it will replace the top element (i.e. the current behavior)
- if **discardOld = false** it will keep the current behavior and push the given one atop

The default of replacing the current behavior on the stack has been chosen to avoid memory leaks in case client code is written without consulting this documentation first (i.e. always pushing new behaviors and never issuing an **unbecome()**)

**\*Warning\***: This method is not thread-safe and must not be accessed from threads other than the ordinary actor message processing thread, such as `java.util.concurrent.CompletionStage` and `scala.concurrent.Future` callbacks.

```
abstract def unbecome(): Unit
```

Reverts the Actor behavior to the previous one on the behavior stack.

**\*Warning\***: This method is not thread-safe and must not be accessed from threads other than the ordinary actor message processing thread, such as `java.util.concurrent.CompletionStage` and `scala.concurrent.Future` callbacks.

```
abstract def watch(subject: ActorRef): ActorRef
```

Registers this actor as a Monitor for the provided ActorRef. This actor will receive a Terminated(subject) message when watched actor is terminated.

**watch** is idempotent if it is not mixed with **watchWith**.

It will fail with an `IllegalStateException` if the same subject was watched before using **watchWith**. To clear the termination message, `unwatch` first.

**\*Warning\***: This method is not thread-safe and must not be accessed from threads other than the ordinary actor message processing thread, such as `java.util.concurrent.CompletionStage` and `scala.concurrent.Future` callbacks.

**returns** the provided ActorRef



`akka.actor`

## ActorSystem

Companion [object ActorSystem](#)

```
abstract class ActorSystem extends ActorRefFactory
```

An actor system is a hierarchical group of actors which share common configuration, e.g. dispatchers, deployments, remote capabilities and addresses. It is also the entry point for creating or looking up actors.

---

```
abstract def actorOf(props: Props, name: String): ActorRef
```

Create new actor as child of this context with the given name, which must not be null, empty or start with "\$".

```
abstract def actorOf(props: Props): ActorRef
```

Create new actor as child of this context and give it an automatically generated name (currently similar to base64-encoded integer count, reversed and with "\$" prepended, may change in the future).

```
def apply(name: String, config: Option[Config] = None, classLoader: Option[ClassLoader] = None, defaultExecutionContext: Option[ExecutionContext] = None): ActorSystem
```

Creates a new ActorSystem with the specified name, the specified ClassLoader if given, otherwise obtains the current ClassLoader by first inspecting the current threads' getContextClassLoader, then tries to walk the stack to find the callers class loader, then falls back to the ClassLoader associated with the ActorSystem class.



akka.actor

## Props

Companion [object](#) Props

```
final case class Props(deploy: Deploy, clazz: Class[_], args: Seq[Any]) extends Product with Serializable
```

Props is a configuration object used in creating an [Actor](#); it is immutable, so it is thread-safe and fully shareable.

```
def apply(clazz: Class[_], args: Any*): Props
```

Scala API: create a Props given a class and its constructor arguments.

```
def apply[T <: Actor](creator: => T)(implicit arg0: ClassTag[T]): Props
```

Scala API: Returns a Props that has default values except for "creator" which will be a function that creates an instance using the supplied thunk.

```
def apply[T <: Actor]() (implicit arg0: ClassTag[T]): Props
```

Scala API: Returns a Props that has default values except for "creator" which will be a function that creates an instance of the supplied type using the default constructor.

```
1. docker
x docker-compose %1 x docker %2 x bash %3
reStart
[warn] Multiple main classes detected. Run 'show discoveredMainClasses' to see the list
Multiple main classes detected, select one to run:

[1] jvm.ActorProperties
[2] jvm.HelloWorld
[3] jvm.HierarchiesDemo
[4] jvm.HierarchiesResolve

Enter number: 1

[info] Stopping application root (by killing the forked JVM) ...
root ... finished with exit code 143
[info] Starting application root in the background ...
root Starting jvm.ActorProperties.main()
[success] Total time: 4 s, completed Aug 12, 2018 6:41:35 PM
sbt:jvm> root [INFO] [08/12/2018 18:41:37.798] [hello-custom-akka.actor.default-dispatcher-3] [akka://hello-custom/user/hello-aliens] Hello, Alien Invaders
root [INFO] [08/12/2018 18:41:37.798] [hello-custom-akka.actor.default-dispatcher-2] [akka://hello-custom/user/hello-name] Hello, Awesome Person

sbt:jvm> |
```



[akka.actor](#)

## PoisonPill

object **PoisonPill** extends [PoisonPill](#) with [Product](#) with [Serializable](#)

A message all Actors will understand, that when processed will terminate the Actor permanently.


```
def actorSelection(path: ActorPath): ActorSelection
```

Construct an [akka.actor.ActorSelection](#) from the given path, which is parsed for wildcards (these are replaced by regular expressions internally).

```
def actorSelection(path: String): ActorSelection
```

Construct an [akka.actor.ActorSelection](#) from the given path, which is parsed for wildcards (these are replaced by regular expressions internally).

```
1. docker
docker-compose %1 docker %2 bash %3
reStart
[warn] Multiple main classes detected. Run 'show discoveredMainClasses' to see the list
Multiple main classes detected, select one to run:
[1] jvm.ActorProperties
[2] jvm.HelloWorld
[3] jvm.HierarchiesDemo
[4] jvm.HierarchiesResolve
Enter number: 2
[info] Stopping application root (by killing the forked JVM) ...
root .. finished with exit code 143
[info] Starting application root in the background ...
root Starting jvm.HelloWorld.main()
[success] Total time: 3 s, completed Aug 12, 2018 6:29:29 PM
sbt:jvm> root [INFO] [08/12/2018 18:29:32.123] [default-akka.actor.default-dispatcher-3] [akka://default/user/hello-world] Hello World
```



akka.pattern.PipeToSupport  
**PipeableFuture**

```
final class PipeableFuture[T] extends AnyRef
```



#### Instance Constructors

```
new PipeableFuture(future: Future[T])(implicit executionContext: ExecutionContext)
```

#### Value Members

```
val future: Future[T]
```

```
def pipeTo(recipient: ActorRef)(implicit sender: ActorRef = Actor.noSender): Future[T]
```

```
def pipeToSelection(recipient: ActorSelection)(implicit sender: ActorRef = Actor.noSender): Future[T]
```

```
def to(recipient: ActorSelection, sender: ActorRef): PipeableFuture[T]
```



[akka.pattern](#)

# AskableActorRef

```
final class AskableActorRef extends AnyVal
```

#### Instance Constructors

```
new AskableActorRef(actorRef: ActorRef)
```

#### Value Members

```
def ?(message: Any)(implicit timeout: Timeout, sender: ActorRef = Actor.noSender): Future[Any]
```

```
val actorRef: ActorRef
```

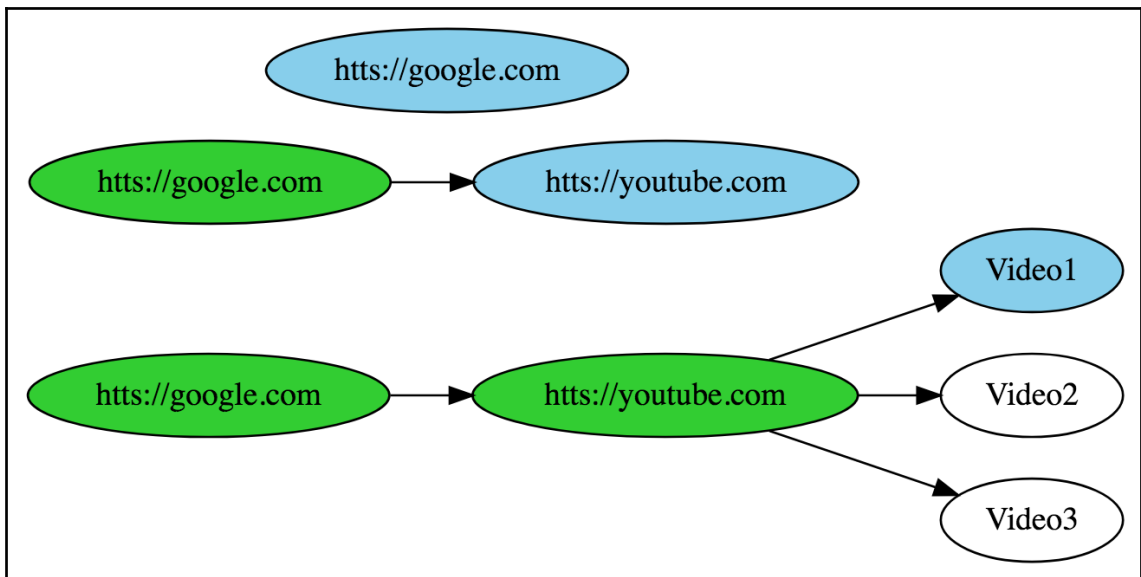
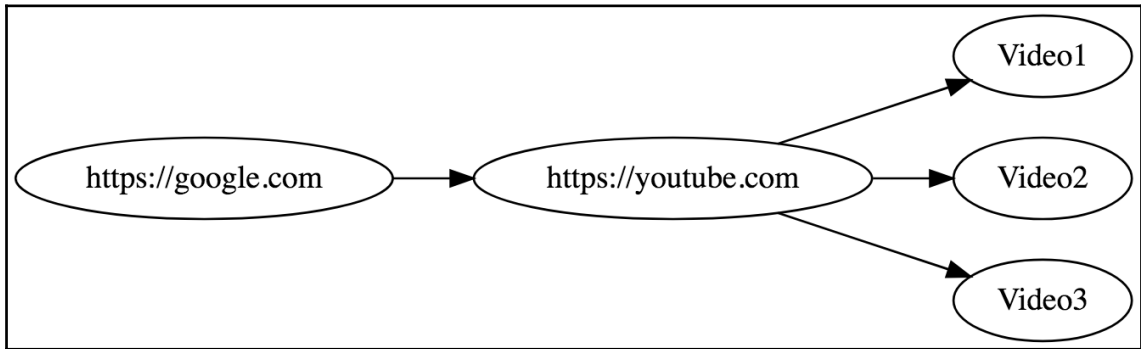
```
def ask(message: Any)(implicit timeout: Timeout, sender: ActorRef = Actor.noSender): Future[Any]
```

```
1. docker
x docker-compose #1 x docker #2 x bash #3
sbt:jvm> root [INFO] [08/12/2018 18:33:36.254] [scala-execution-context-global-14] [akka://hierarchy-demo/user/greetings-manager] Selection: ActorSelection[Anchor(akka://hierarchy-demo/user/greetings-manager#-335253773), Path(/greeter-*)]; Res: Failure(akka.actor.ActorNotFound: Actor not found for: ActorSelection[Anchor(akka://hierarchy-demo/user/greetings-manager#-335253773), Path(/greeter-*)])
root [INFO] [08/12/2018 18:33:36.254] [hierarchy-demo-akka.actor.default-dispatcher-3] [akka://hierarchy-demo/user/greetings-manager] Spawning 10 greeters
root [INFO] [08/12/2018 18:33:36.310] [hierarchy-demo-akka.actor.default-dispatcher-3] [akka://hierarchy-demo/user/greetings-manager] Created 10 greeters
root [INFO] [08/12/2018 18:33:37.093] [scala-execution-context-global-8] [akka://hierarchy-demo/user/greetings-manager] Selection: ActorSelection[Anchor(akka://hierarchy-demo/user/greetings-manager#-335253773), Path(/greeter-*)]; Res: Success(Actor[akka://hierarchy-demo/user/greetings-manager/greeter-6#1542315700])
root [INFO] [08/12/2018 18:33:37.103] [scala-execution-context-global-8] [akka://hierarchy-demo/user/greetings-manager] Selection: ActorSelection[Anchor(akka://hierarchy-demo/user/greetings-manager#-335253773), Path(/greeter-*)]; Res: Success(Actor[akka://hierarchy-demo/user/greetings-manager/greeter-1#-449866333])
root [INFO] [08/12/2018 18:33:37.123] [scala-execution-context-global-8] [akka://hierarchy-demo/user/greetings-manager] Selection: ActorSelection[Anchor(akka://hierarchy-demo/user/greetings-manager#-335253773), Path(/greeter-*)]; Res: Success(Actor[akka://hierarchy-demo/user/greetings-manager/greeter-1#-449866333])
root [INFO] [08/12/2018 18:33:37.126] [scala-execution-context-global-8] [akka://hierarchy-demo/user/greetings-manager] Selection: ActorSelection[Anchor(akka://hierarchy-demo/user/greetings-manager#-335253773), Path(/greeter-*)]; Res: Success(Actor[akka://hierarchy-demo/user/greetings-manager/greeter-1#-449866333])
root [INFO] [08/12/2018 18:33:37.128] [scala-execution-context-global-8] [akka://hierarchy-demo/user/greetings-manager] Selection: ActorSelection[Anchor(akka://hierarchy-demo/user/greetings-manager#-335253773), Path(/greeter-*)]; Res: Success(Actor[akka://hierarchy-demo/user/greetings-manager/greeter-1#-449866333])
root [INFO] [08/12/2018 18:33:37.136] [scala-execution-context-global-8] [akka://hierarchy-demo/user/greetings-manager] Selection: ActorSelection[Anchor(akka://hierarchy-demo/user/greetings-manager#-335253773), Path(/greeter-*)]; Res: Success(Actor[akka://hierarchy-demo/user/greetings-manager/greeter-1#-449866333])
root [INFO] [08/12/2018 18:33:37.136] [scala-execution-context-global-14] [akka://hierarchy-demo/user/greetings-manager] Selection: ActorSelection[Anchor(akka://hierarchy-demo/user/greetings-manager#-335253773), Path(/greeter-*)]; Res: Success(Actor[akka://hierarchy-demo/user/greetings-manager/greeter-1#-449866333])
root [INFO] [08/12/2018 18:33:37.136] [scala-execution-context-global-14] [akka://hierarchy-demo/user/greetings-manager]
```

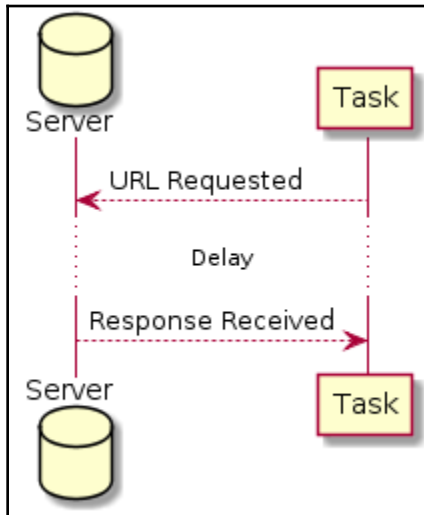
```
1. docker
x docker-compose %1 x docker %2 x bash %3
root == Children: empty, Message: SayHello
root [ERROR] [08/12/2018 18:36:15.393] [hierarchy-demo-akka.actor.default-dispatcher-2] [akka://hierarchy-demo/user/greetings-manager] There are no greeters. Please create some first with SpawnGreeters message.
root
root == Children: empty, Message: SpawnGreeters
root [INFO] [08/12/2018 18:36:15.434] [hierarchy-demo-akka.actor.default-dispatcher-2] [akka://hierarchy-demo/user/greetings-manager] Spawning 3 greeters
root
root == Children: present, Message: SpawnGreeters
root [INFO] [08/12/2018 18:36:15.486] [hierarchy-demo-akka.actor.default-dispatcher-2] [akka://hierarchy-demo/user/greetings-manager] Created 3 greeters
root [INFO] [08/12/2018 18:36:15.493] [hierarchy-demo-akka.actor.default-dispatcher-2] [akka://hierarchy-demo/user/greetings-manager] Spawning 3 greeters
root [WARN] [08/12/2018 18:36:15.525] [hierarchy-demo-akka.actor.default-dispatcher-5] [akka://hierarchy-demo/user/greetings-manager] Greeters already exist. Killing them and creating the new ones.
root [INFO] [08/12/2018 18:36:16.222] [hierarchy-demo-akka.actor.default-dispatcher-5] [akka://hierarchy-demo/user/greetings-manager] All greeters terminated, report: List(Dead, Dead, Dead). Creating the new ones now.
root [INFO] [08/12/2018 18:36:16.225] [hierarchy-demo-akka.actor.default-dispatcher-5] [akka://hierarchy-demo/user/greetings-manager] Created 3 greeters
root
root == Children: present, Message: SayHello
root [INFO] [08/12/2018 18:36:16.234] [hierarchy-demo-akka.actor.default-dispatcher-5] [akka://hierarchy-demo/user/greetings-manager] Dispatching message SayHello(me) to greeters
root [INFO] [08/12/2018 18:36:16.236] [hierarchy-demo-akka.actor.default-dispatcher-4] [akka://hierarchy-demo/user/greetings-manager/greeter-2] Greetings to me
root [INFO] [08/12/2018 18:36:16.235] [hierarchy-demo-akka.actor.default-dispatcher-3] [akka://hierarchy-demo/user/greetings-manager/greeter-1] Greetings to me
root [INFO] [08/12/2018 18:36:16.237] [hierarchy-demo-akka.actor.default-dispatcher-3] [akka://hierarchy-demo/user/greetings-manager/greeter-3] Greetings to me
```

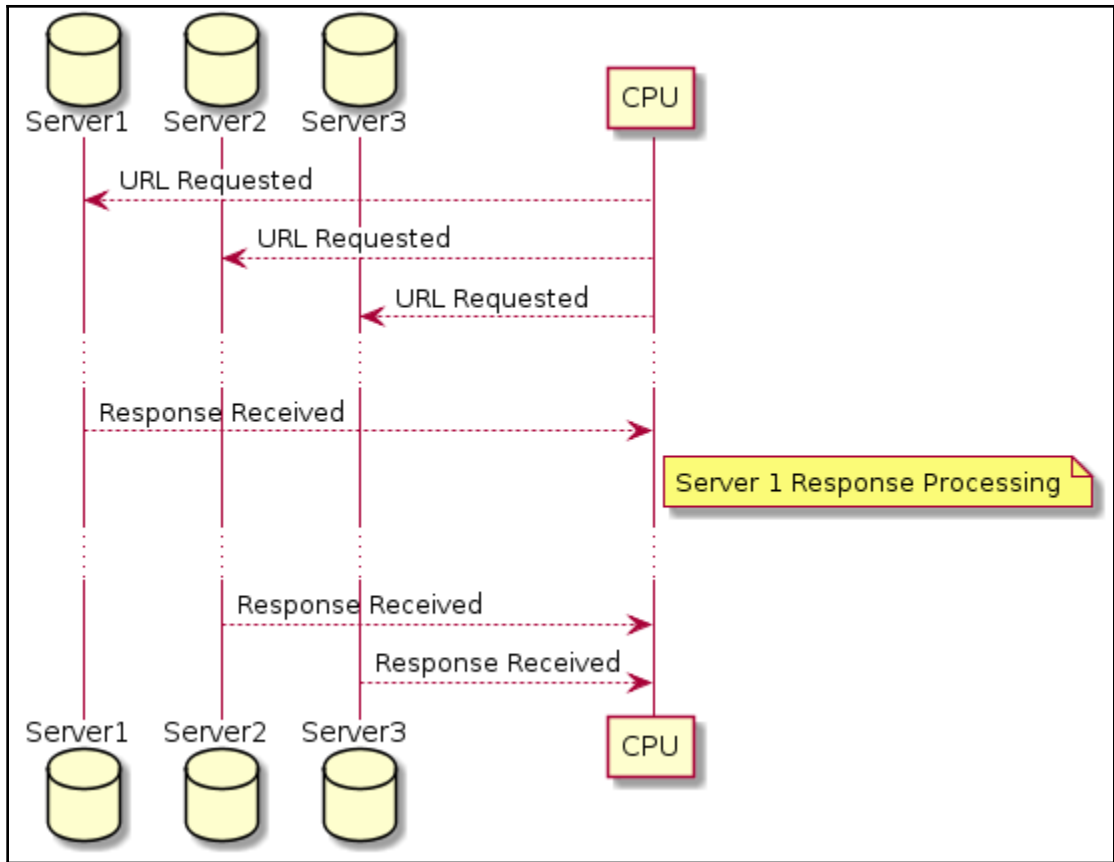
---

## Chapter 13: Use Case - A Parallel Web Crawler

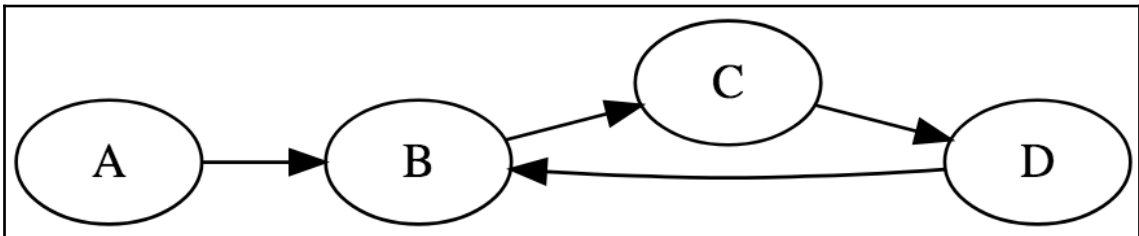


```
1. java
Progress for depth 1: 153 of 165
Progress for depth 1: 154 of 165
Progress for depth 1: 155 of 165
Progress for depth 1: 156 of 165
Progress for depth 1: 157 of 165
Progress for depth 1: 158 of 165
Progress for depth 1: 159 of 165
Progress for depth 1: 160 of 165
Progress for depth 1: 161 of 165
Progress for depth 1: 162 of 165
Progress for depth 1: 163 of 165
Progress for depth 1: 164 of 165
http://mvnrepository.com/artifact/com.github.paulwarren
http://mvnrepository.com/artifact/io.micrometer/micrometer-registry-influx/1.0.5
http://central.maven.org/maven2/com.nexion/discovery-plugin-config-center/4.0.0
http://mvnrepository.com/artifact/org.webpieces/http1_1-parser/1.9.45
http://mvnrepository.com/artifact/net.nemerosa.ontrack/ontrack-extension-git/3.35.8#ivy
http://mvnrepository.com/artifact/net.nemerosa.ontrack/ontrack-json/3.35.8#sbt
http://mvnrepository.com/artifact/org.webpieces/core-channelmanager2/1.9.35/usage
http://mvnrepository.com/artifact/io.github.lukehutch/fast-classpath-scanner/2.12.2
http://mvnrepository.com/artifact/org.webpieces/core-channelmanager2
http://mvnrepository.com/artifact/com.github.paulwarren/spring-content-solr
1664
background log: debug: Thread run-main-0 exited.
background log: debug: Interrupting remaining threads (should be all daemons).
background log: debug: Not interrupting system thread Thread[Keep-Alive-Timer,8,system]
background log: debug: Sandboxed run complete..
background log: debug: Exited with code 0
[success] Total time: 22 s, completed Jul 16, 2018 2:02:01 PM
>
```





```
1. java
Progress for depth 1: 153 of 165
Progress for depth 1: 154 of 165
Progress for depth 1: 155 of 165
Progress for depth 1: 156 of 165
Progress for depth 1: 157 of 165
Progress for depth 1: 158 of 165
Progress for depth 1: 159 of 165
Progress for depth 1: 160 of 165
Progress for depth 1: 161 of 165
Progress for depth 1: 162 of 165
Progress for depth 1: 163 of 165
Progress for depth 1: 164 of 165
http://mvnrepository.com/artifact/com.github.paulcwarren
http://mvnrepository.com/artifact/io.micrometer/micrometer-registry-influx/1.0.5
http://central.maven.org/maven2/com.nepxion/discovery-plugin-config-center/4.0.0
http://mvnrepository.com/artifact/org.webpieces/http1_1-parser/1.9.45
http://mvnrepository.com/artifact/net.nemerosa.ontrack/ontrack-extension-git/3.35.8#ivy
http://mvnrepository.com/artifact/net.nemerosa.ontrack/ontrack-json/3.35.8#sbt
http://mvnrepository.com/artifact/org.webpieces/core-channelmanager2/1.9.35/usage
http://mvnrepository.com/artifact/io.github.lukehutch/fast-classpath-scanner/2.12.2
http://mvnrepository.com/artifact/org.webpieces/core-channelmanager2
http://mvnrepository.com/artifact/com.github.paulcwarren/spring-content-solr
1664
background log: debug: Thread run-main-0 exited.
background log: debug: Interrupting remaining threads (should be all daemons).
background log: debug: Not interrupting system thread Thread[Keep-Alive-Timer,8,system]
background log: debug: Sandboxed run complete..
background log: debug: Exited with code 0
[success] Total time: 22 s, completed Jul 16, 2018 2:02:01 PM
>
```



```
final def getOrElse[B >: A](default: ⇒ B): B

  Returns the option's value if the option is nonempty, otherwise return the result of evaluating default.

  default    the default expression.

  Annotations    @inline()
```

---

```
def zipWithIndex: List[(A, Int)]
```

[use case]

Zips this list with its indices.

**returns** A new list containing pairs consisting of all elements of this list paired with their index. Indices start at **0**.

**Definition Classes** [IterableLike](#) → [GenIterableLike](#)

Full Signature

Example:

```
List("a", "b", "c").zipWithIndex = List(("a", 0), ("b", 1), ("c", 2))
```

```
def foldLeft[B](z: B)(op: (B, A) => B): B
```

Applies a binary operator to a start value and all elements of this sequence, going left to right.

Note: will not terminate for infinite-sized collections.

**B** the result type of the binary operator.

**z** the start value.

**op** the binary operator.

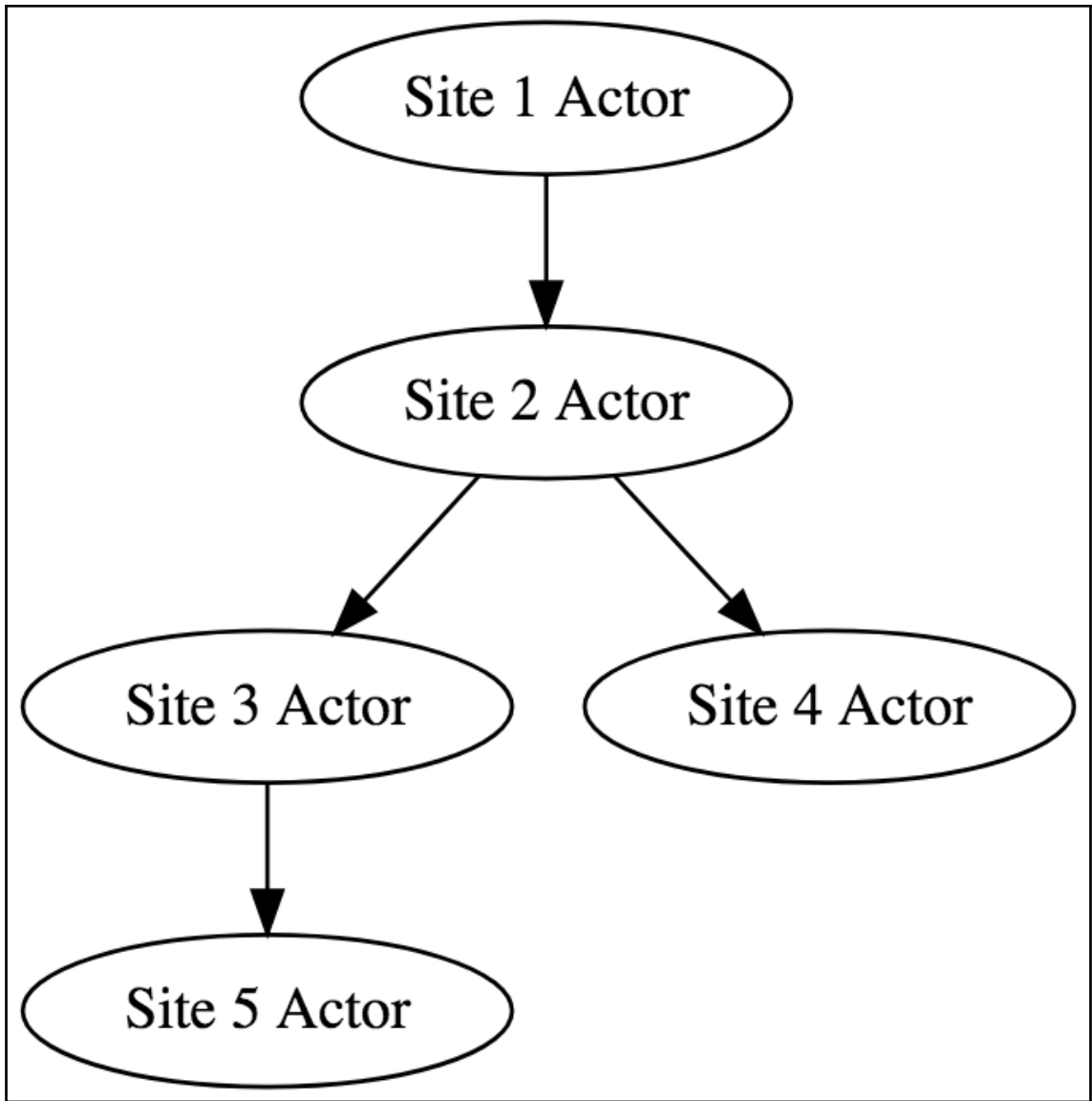
**returns** the result of inserting **op** between consecutive elements of this sequence, going left to right with the start value **z** on the left:

```
op(...op(z, x_1), x_2, ..., x_n)
```

where **x<sub>1</sub>, ..., x<sub>n</sub>** are the elements of this sequence. Returns **z** if this sequence is empty.

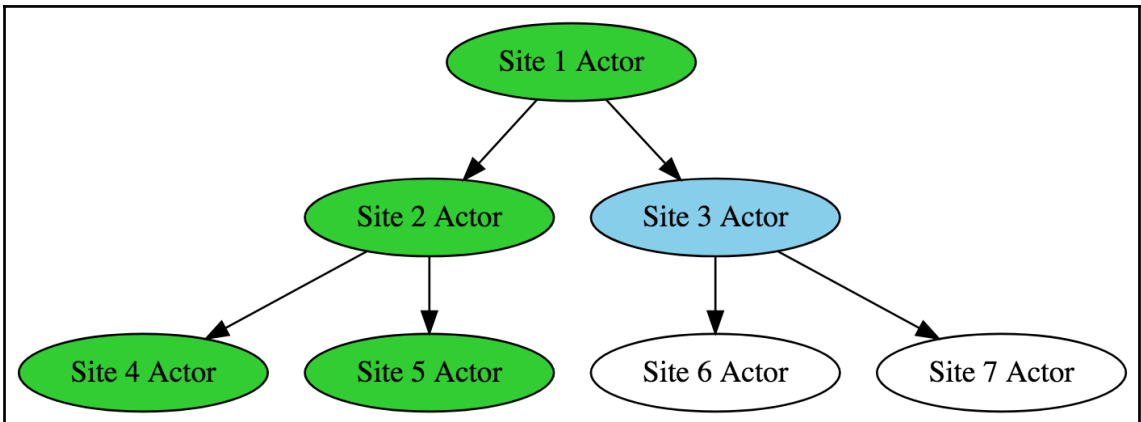
**Definition Classes** [LinearSeqOptimized](#) → [TraversableOnce](#) → [GenTraversableOnce](#)

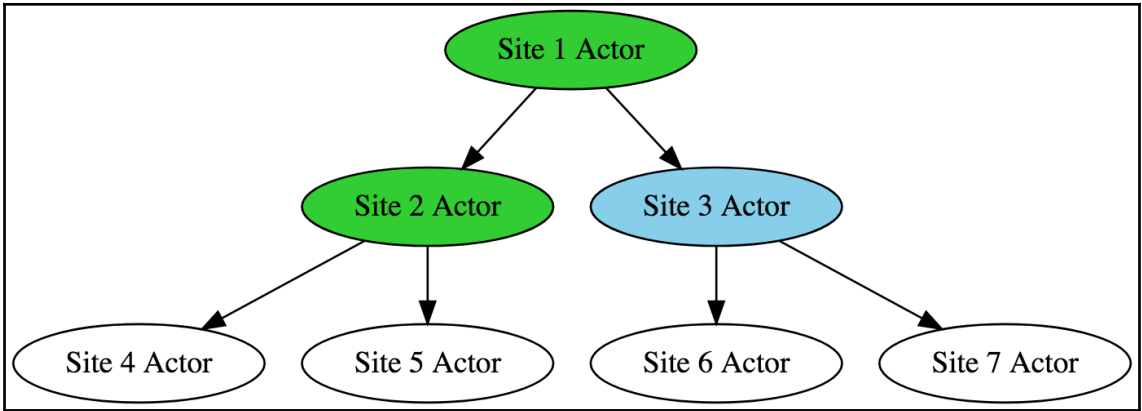




```
def ! (message: Any)(implicit sender: ActorRef = Actor.noSender): Unit  
  Sends a one-way asynchronous message.  
def ? (message: Any)(implicit timeout: Timeout, sender: ActorRef = Actor.noSender):  
  Future[Any]
```

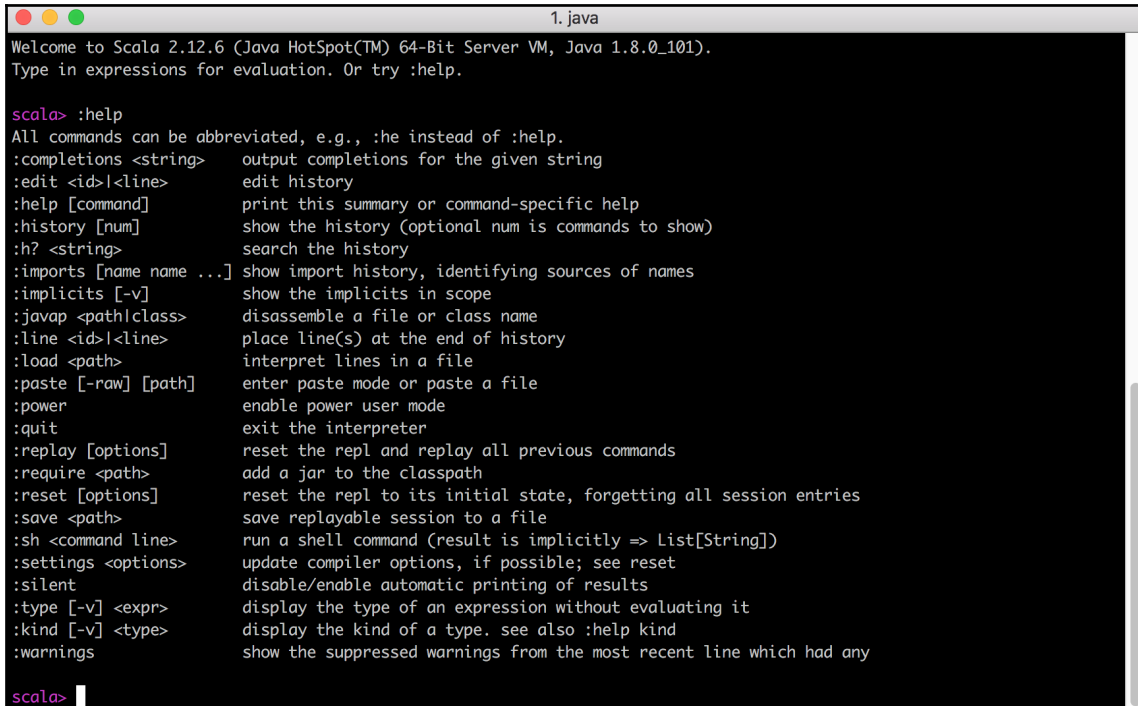
```
1.java
Actor[akka://PiSystem/user/$a#300673380]: 158 actors responded of 165
Reached maximal depth on http://mvnrepository.com/open-source/core-utilities - returning its links only
Actor[akka://PiSystem/user/$a#300673380]: 159 actors responded of 165
Reached maximal depth on http://mvnrepository.com/artifact/org.mini2Dx/mini2Dx-core - returning its links only
Actor[akka://PiSystem/user/$a#300673380]: 160 actors responded of 165
Reached maximal depth on http://mvnrepository.com/artifact/com.nepxion - returning its links only
Actor[akka://PiSystem/user/$a#300673380]: 161 actors responded of 165
Reached maximal depth on http://mvnrepository.com/artifact/net.nemerosa.ontrack/ontrack-ui-support/3.35.8 - returning its links only
Actor[akka://PiSystem/user/$a#300673380]: 162 actors responded of 165
Reached maximal depth on http://mvnrepository.com/repos/apache-releases - returning its links only
Actor[akka://PiSystem/user/$a#300673380]: 163 actors responded of 165
Reached maximal depth on http://mvnrepository.com/artifact/net.nemerosa.ontrack/ontrack-model/usages - returning its links only
Actor[akka://PiSystem/user/$a#300673380]: 164 actors responded of 165
Reached maximal depth on https://mvnrepository.com/ - returning its links only
Actor[akka://PiSystem/user/$a#300673380]: 165 actors responded of 165
Crawling finished successfully
http://mvnrepository.com/artifact/net.nemerosa.ontrack/ontrack-it-utils/2.24.1/usages
http://mvnrepository.com/artifact/com.github.paulwarren
http://mvnrepository.com/artifact/net.nemerosa.ontrack/ontrack-it-utils/2.29.3/usages
http://mvnrepository.com/artifact/com.google.appengine/appengine-api-1.0-sdk
http://mvnrepository.com/artifact/com.android.support/appcompat-v7/usages
http://mvnrepository.com/artifact/net.nemerosa.ontrack/ontrack-test-utils/3.1.5/usages
http://mvnrepository.com/artifact/io.micrometer/micrometer-registry-influx/1.0.5
http://mvnrepository.com/artifact/net.nemerosa.ontrack/ontrack-json/3.1.3
http://mvnrepository.com/artifact/net.nemerosa.ontrack/ontrack-service/3.0-beta.5/usages
http://mvnrepository.com/artifact/net.nemerosa.ontrack/ontrack-tx/3.35.8#maven
10086
```





---

# Appendix - Introduction to Scala



```
1. java
Welcome to Scala 2.12.6 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_101).
Type in expressions for evaluation. Or try :help.

scala> :help
All commands can be abbreviated, e.g., :he instead of :help.
:completions <string>    output completions for the given string
:edit <id>|<line>       edit history
:help [command]         print this summary or command-specific help
:history [num]          show the history (optional num is commands to show)
:h? <string>            search the history
:imports [name name ...] show import history, identifying sources of names
:implicits [-v]         show the implicits in scope
:javap <pathclass>     disassemble a file or class name
:line <id>|<line>       place line(s) at the end of history
:load <path>           interpret lines in a file
:paste [-raw] [path]   enter paste mode or paste a file
:power                 enable power user mode
:quit                 exit the interpreter
:replay [options]      reset the repl and replay all previous commands
:require <path>        add a jar to the classpath
:reset [options]       reset the repl to its initial state, forgetting all session entries
:save <path>           save replayable session to a file
:sh <command line>    run a shell command (result is implicitly => List[String])
:settings <options>   update compiler options, if possible; see reset
:silent               disable/enable automatic printing of results
:type [-v] <expr>     display the type of an expression without evaluating it
:kind [-v] <type>     display the kind of a type. see also :help kind
:warnings             show the suppressed warnings from the most recent line which had any

scala> |
```