

Monad Transformers



Monad Transformers

Higher-kinded types for convenience over nested monadic values

wrapper over List[Option[Int]]

```
import cats.data.OptionT
import cats.instances.list._ // for a Monad[List]

val listOfNumberOptions: OptionT[List, Int] = OptionT(List(Option(1), Option(2)))
val listOfCharOptions: OptionT[List, Char] = OptionT(List(Option('a'), Option('b')))
val listOfTuples: OptionT[List, (Int, Char)] = for {
  char <- listOfCharOptions
  number <- listOfNumberOptions
} yield (number, char)
```

can access map/flatMap with an
implicit Monad[List] in scope

EitherT

Transformer for monadic values of Either instances

```
import cats.data.EitherT

// apply factory method
val listEithers: EitherT[List, String, Int] = EitherT(List(Left("something wrong"), Right(43), Right(2)))

// alternative: `left` and `right` methods
val futureEither: EitherT[Future, String, Int] = EitherT.right(Future(45)) // wrap Future(Right(45))

// change the deeply nested Either
val transformedEither = futureOfEither.transform {
  case Left(undesirable) => // another Either
  case Right(desirable) => // another Either
}
```

Cats rock

