Handling Options

Lesson Objectives

- After completing this lesson, you should be able to:
  - Describe how to pattern match on an Option
  - Outline how to use higher order functions on an option to avoid null-checking
  - Illustrate how to use for comprehensions to work with Option

Typesafe

Pattern Matching an Option

```
def getMiddleName(value: Option[String]): String = {
  value match {
    case Some(middleName) => middleName
    case None => "No middle name"
  }
}
```

Typesafe

Pattern Matching an Option

```
scala> case class Customer(first: String = "",
                           middle: Option[String] = None,
                           last: String = "")

scala> val martin = Customer("Martin", last = "Odersky")
martin: Customer = Customer(Martin,None,Odersky)

scala> getMiddleName(martin.middle)
res0: String = No middle name
```

Typesafe

Pattern Matching an Option

```
scala> case class Customer(first: String = "",
                           middle: Option[String] = None,
                           last: String = "")

scala> val jane = Customer("Jane", Option("D."), "Doe")
jane: Customer = Customer(Jane,Some(D.),Doe)

scala> getMiddleName(jane.middle)
res1: String = D.
```

Typesafe

HOFs and Option

```
scala> Option("Martin")
res0: Option[String] = Some(Martin)

scala> res0.map(name => println("Yay, " + name))
Yay, Martin
res1: Option[Unit] = Some(())

scala> res0.foreach(name => println("Yay, " + name))
Yay, Martin

scala> None.foreach(name => println("Yay, " + name))
```

For Expressions and Option

```
scala> val martin = Option("Martin")
martin: Some[String] = Some(Martin)

scala> val jane = Option("Jane")
jane: Some[String] = Some(Jane)

scala> for {
     |   m <- martin
     |   j <- jane
     | } yield (m + " is friends with " + j)
res1: Option[String] = Some(Martin is friends with Jane)
```

For Expressions and Option

```
scala> val martin = Option("Martin")
martin: Some[String] = Some(Martin)

scala> val noValue = None
noValue: None.type = None

scala> for {
     |   m <- martin
     |   n <- noValue
     | } yield (m + " is friends with " + n)
res2: Option[String] = None
```

Typesafe

Lesson Summary

- Having completing this lesson, you should be able to:
  - Describe how to pattern match on an Option
  - Outline how to use higher order functions on an option to avoid null-checking
  - Illustrate how to use for comprehensions to work with Option