TOUR OF SCALA

TRAITS

Traits are used to share interfaces and fields between classes. They are similar to Java 8's interfaces. Classes and objects can extend traits, but traits cannot be instantiated and therefore have no parameters.

Defining a trait

A minimal trait is simply the keyword `trait` and an identifier:

```
trait HairColor
```

Traits become especially useful as generic types and with abstract methods.

```
trait Iterator[A] {
  def hasNext: Boolean
  def next(): A
}
```

Extending the `trait Iterator[A]` requires a type `A` and implementations of the methods `hasNext` and `next`.

Using traits

Use the `extends` keyword to extend a trait. Then implement any abstract members of the trait using the `override` keyword:

```
trait Iterator[A] {
  def hasNext: Boolean
  def next(): A
}

class IntIterator(to: Int) extends Iterator[Int] {
  private var current = 0
  override def hasNext: Boolean = current < to
  override def next(): Int = {
    if (hasNext) {
      val t = current
      current += 1
      t
    } else 0
  }
}


val iterator = new IntIterator(10)
iterator.next()  // returns 0
iterator.next()  // returns 1
```

This `IntIterator` class takes a parameter `to` as an upper bound. It `extends Iterator[Int]` which means that the `next` method must return an Int.

Subtyping

Where a given trait is required, a subtype of the trait can be used instead.

```
import scala.collection.mutable.ArrayBuffer

trait Pet {
  val name: String
}

class Cat(val name: String) extends Pet
class Dog(val name: String) extends Pet

val dog = new Dog("Harry")
val cat = new Cat("Sally")

val animals = ArrayBuffer.empty[Pet]
animals.append(dog)
animals.append(cat)
animals.foreach(pet => println(pet.name))  // Prints Harry Sally
```

The `trait Pet` has an abstract field `name` that gets implemented by Cat and Dog in their constructors. On the last line, we call `pet.name`, which must be implemented in any subtype of the trait `Pet`.

More resources

- Learn more about traits in the Scala Book
- Use traits to define Enum

← **previous** **next** →

Contributors to this page:

ckipp01 mlachkar rhumbertgz ashawley asakaev mghildiy bradfordlynch
Sal Borrelli sake92 voidance heathermiller