

Scalacheck

Pepe García

2020-04-20

Scalacheck is a library for property based testing. Property based testing libraries are very useful to detect corner cases in our domain logic.

we use it with the import:

```
import org.scalacheck._
```

Properties

Properties are the name we give to assertions in this kind of testing. The difference with assertions is that instead of testing them with just one value, we will pass a bunch of randomly generated values.

Generating values

In order to generate values for a type, we need a generator (called `Gen` in Scalacheck).

Generating values

Basic generators (**see the following in the console**):

- `Gen.alphaUpperStr`
- `Gen.posNum[Int]`
- `Gen.posNum[Double]`
- `Gen.negNum[Int]`
- `Gen.oneOf`
- `Gen.listOf`
- `Gen.const`

Generating values

We can construct more interesting generators using Gen's functional combinators.

We will create a generator for the following datatype:

```
case class Car(age: Int, model: String, brand: String)
```


Generating values

```
val carGenUsingFlatmap: Gen[Car] =  
  Gen.posNum[Int].flatMap { age =>  
    Gen.alphaLowerStr.flatMap { model =>  
      Gen.alphaLowerStr.map { brand =>  
        Car(age, model, brand)  
      }  
    }  
  }  
}  
  
// carGenUsingFlatmap: Gen[Car] = org.scalacheck.Gen$$anon$3@4be37e22
```

Generating values

Something neat we can do is refactor this example and use a for comprehension instead.

```
val carGen: Gen[Car] = for {  
  age <- Gen.posNum[Int]  
  model <- Gen.alphaLowerStr  
  brand <- Gen.alphaLowerStr  
} yield Car(age, model, brand)  
  
// carGen: Gen[Car] = org.scalacheck.Gen$$anon$3@4196f52  
  
carGen.sample  
// res0: Option[Car] = Some(  
//   Car(  
//     7,  
//     "zkuvbff",  
//     "ywshuzlnpqcvqtqajyayrvvpaviglckdcidtslkmormwwglqjlujlywgcxskzhkake"
```

Properties

Imagine that we want to ensure a simple thing about our model, we don't want to generate instances of `Car` with `age < 0`, and with `brand` or `model` `== ""`.

```
import org.scalacheck.Prop.forAll

val prop = forAll(carGen) { car =>
  car.age > 0 && car.model != "" && car.brand != ""
}

// prop: Prop = Prop

prop.check()

// ! Falsified after 0 passed tests.
// > ARG_0: Car(1,,)
```