Scala | Either

In Scala Either, functions exactly similar to an Option. The only dissimilarity is that with Either it is practicable to return a string which can explicate the instructions about the error that appeared. The Either has two children which are named as Right and Left where, Right is similar to the Some class and Left is same as None class. Left is utilized for the failure where, we can return the error occurred inside the child Left of the Either and Right is utilized for Success.

Example:

 Either[String, Int]

Here, the String is utilized for the Left child of Either as its the left argument of an Either and Int is utilized for the Right child as its the right argument of an Either. Now, let's discuss it in details with the help of some examples.

Example :

```
// Scala program of Either



// Creating object and inheriting

// main method of the trait App

object GfG extends App

{

   // Defining a method and applying

   // Either

   def Name(name: String): Either[String, String] =

   {

      if (name.isEmpty)

         // Left child for failure

         Left("There is no name.")



      else

         // Right child for success
```

```
        Right(name)
    }


    // Displays this if name is
    // not empty
    println(Name("GeeksforGeeks"))


    // Displays the String present
    // in the Left child
    println(Name(""))
}
```

Output:

```
Right(GeeksforGeeks)
Left(There is no name.)
```

Here, isEmpty method checks if the field of name is empty or filled, if its empty then Left child will return the String inside itself and if this field is not empty then the Right child will return the name stated.

Example :

```
// Scala program of Either with
// Pattern matching
// Creating object and inheriting
// main method of the trait App
object either extends App
{
    // Defining a method and applying
    // Either
```

```
  def Division(q: Int, r: Int): Either[String, Int] =
  {
    if (q == 0)
      // Left child for failure
      Left("Division not possible.")
    else
      // Right child for success
      Right(q / r)
  }
  // Assigning values
  val x = Division(4, 2)

  // Applying pattern matching
  x match
  {
    case Left(l) =>

    // Displays this if the division
    // is not possible
    println("Left: " + l)
    case Right(r) =>
    // Displays this if division
    // is possible
    println("Right: " + r)
  }
}
```

Output:

```
Right: 2
```

Here, the division is possible which implies success so, Right returns 2. Here, we have utilized Pattern Matching in this example of Either.