Simple Data Validation in Scala

WRITTEN BY DAVE GURNELL

For the past couple of weeks I've been toying with a nice little DSL for data validation in Scala. I came up with the idea when writing my macros talk for ScalaDays and have since fleshed it out into its own library.

Grab the code from Github and check out the unit tests for a full guide. Here's a synopsis for the impatient.

First we define some data that we'd like to validate:

```
case class Address(house: Int, street: String)
case class Person(name: String, age: Int, address: Address)
case class Business(name: String, addresses: Seq[Address])
```

Then we import the validation library and define some `Validator` objects:

```
import io.underscore.validation._

implicit val addressValidator: Validator[Address] =
  validate[Address].
  field(_.house)(warn(gte(1))).
  field(_.street)(warn(nonEmpty))

implicit val personValidator: Validator[Person] =
  validate[Person].
  field(_.name)(nonEmpty).
  field(_.age)(gte(1)).
  field(_.address)

implicit val businessValidator: Validator[Business] =
  validate[Business].
  field(_.name)(nonEmpty).
  seqField(_.addresses)
```

Each validator has an `apply()` method that accepts a case class object of the relevant type. The validator checks the fields of the case class and returns a list of failures and warnings, each annotated with the location of the error:

```
Person("", 0, Address(0, "")).validate.prettyPrint

// ==> Validated Person(,0,Address(0,)):
//       - Error: name - Must not be empty
//       - Error: age - Must be 1 or higher
//       - Warning: address.house - Must be 1 or higher
//       - Warning: address.street - Must not be empty
```

There are a couple of nice features in the library, including:

- a simple set of combinators for building validators;
- a DSL for specifying data paths involving field-based and index-based access;
- a macro that inspects the names of accessors used to locate validated data, injects their names into the relevant error

paths automatically.

« Return home