+

@sourcedelica

Typesafe Config on Steroids

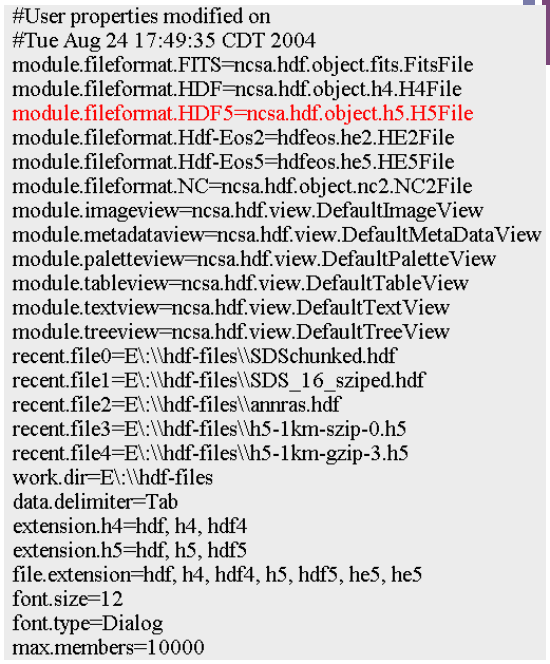
+

Agenda

- Introduction to Typesafe Config
- Scopes – taking it to the next level

+
Property Files

- The old standby
  - On JVM
- Simple
- Hard to scale



```
#User properties modified on
#Tue Aug 24 17:49:35 CDT 2004
module.fileformat.FITS=ncsa.hdf.object.fits.FitsFile
module.fileformat.HDF=ncsa.hdf.object.h4.H4File
module.fileformat.HDF5=ncsa.hdf.object.h5.H5File
module.fileformat.Hdf-Eos2=hdfeos.he2.HE2File
module.fileformat.Hdf-Eos5=hdfeos.he5.HE5File
module.fileformat.NC=ncsa.hdf.object.nc2.NC2File
module.imageview=ncsa.hdf.view.DefaultImageView
module.metadataview=ncsa.hdf.view.DefaultMetaDataView
module.paletteview=ncsa.hdf.view.DefaultPaletteView
module.tableview=ncsa.hdf.view.DefaultTableView
module.textview=ncsa.hdf.view.DefaultTextView
module.treeview=ncsa.hdf.view.DefaultTreeView
recent.file0=E\:\\hdf-files\\SDSchunked.hdf
recent.file1=E\:\\hdf-files\\SDS_16_sziped.hdf
recent.file2=E\:\\hdf-files\\annras.hdf
recent.file3=E\:\\hdf-files\\h5-1km-szip-0.h5
recent.file4=E\:\\hdf-files\\h5-1km-gzip-3.h5
work.dir=E\:\\hdf-files
data.delimiter=Tab
extension.h4=hdf, h4, hdf4
extension.h5=hdf, h5, hdf5
file.extension=hdf, h4, hdf4, h5, hdf5, he5, he5
font.size=12
font.type=Dialog
max.members=10000
```

+
Apache Commons Configuration

- Typed API
- Substitutions
- Include other files
- Downsides
  - Limited by property file format
  - Composition is limited

Typesafe Config

- Used by Play Framework and Akka
  - But is a standalone project with no dependencies
- JSON-like format
- Java API
  - Typed
  - Immutable
  - Power features

Typesafe Config Format

- HOCON
  - Human-Optimized Config Object Notation

```
demo {
  # Our development DB
  database {
    username = "scott"
    password = "tiger"
    connections = 20
    timeout = 5 seconds
  }
  jms.brokers =
    ["nyamq10", "sfamq20", "jpamq54" ]
```

External Values

- System properties are used by default
  - Easy to override configuration values in scripts

- Reference environment variables

```
basedir = "/foo/bar/baz"
basedir = ${?BASEDIR_OVERRIDE}
```

  - Would override `basedir` if set

API examples

```
# Loads all application.conf in the classpath
Config config = ConfigFactory.load()

String username =
  config.getString("demo.database.username")

int connections =
  config.getInt("demo.database.connections")
```

API Examples

```
# Config values are like "15s", "5 minutes"
Duration timeout =
  config.getDuration("demo.database.timeout")

# Config values are like "100m", "2 gigs"
long size = config.getBytes("demo.cache.size")

List<String> brokers =
  config.getStringList("demo.jms.brokers")
```

Create Configs at Runtime

```
 val config = ConfigFactory.parseString(
      s"""
        |akka {
        |  remote {
        |    netty.tcp {
        |      hostname = "$hostname"
        |      port = $port
        |    }
        |  }
        |}
      """.stripMargin)

val system = ActorSystem("demo", config)
```

Config Objects

```
demo {
  services = [
    { name = "user",
      host = "nycdev101", port = 8080
    }
    { name = "search",
      host = "nycdev102", port = 9201 }
    # etc…
  ]
}

List<Config> configList =
  config.getConfigList("demo.services")
```

Merging Configs

```
Config appConfig=
  ConfigFactory.parseFile("application.conf")

Config userConfig =
  ConfigFactory.parseFile(username + ".conf")

# Merge configs and resolve substitutions
Config finalConfig =
  ConfigFactory.systemProperties().
    withFallback(userConfig).
    withFallback(appConfig).
    resolve()
```

Scopes Library

- A library built on top of Typesafe Config
- Adds
  - Multiple scopes of configuration
    - aka Configurable config merging
  - Pluggable resource handling
  - and more

Scopes Configuration

```
scopes = [
   {
     name = application
     path = "classpath://config.conf"
   }
   {
     name = environment
     path = "classpath://config-${environment}.conf"
   }
   {
     name = shared
     path = "zk://config/shared"
   }
   {
     name = shared
     path = "zk://config/shared/${environment}"
   }
]
```

Scopes API

- Superset of Typesafe Config API
- Adds
  - Scala API
  - Resource values
  - Defaults
  - Dynamic updates
  - and more

Extended API

```
demo {
  jms.brokers =
    ["nyamq10", "sfamq20", "jpamq54" ]
}


val config = Scopes.config()

# Scala collections are used by default
config.getStringList("demo.jms.brokers").
  foreach(println)
```

Extended API

```
demo {
  database.username = "scott"
  database.password = "tiger"
}

val username =
  config.getString("demo.username", "admin")

# Returns Some[String] or None if not set
val password =
  config.optString("demo.password").
    getOrElse("admin")
```

Resource Properties

```
demo {
  # Currently supported: classpath, file, zk
  helpText = "classpath://help/help.txt"
}


val helpResource =
  config.getResource("demo.helpText")

val helpText = helpResource.asString

val helpStream = helpResource.asStream
```

Dynamic updates

- Watches config locations defined in scopes
- If changes are made, callbacks get new values

Interested?

- Interested in Scopes?
  - I’m thinking of open-sourcing
  - Contact me @sourcedelica

+ More Info

- Github
  - https://github.com/typesafehub/config
- Javadoc
  - http://typesafehub.github.io/config/latest/api/
- Intellij
  - Scala Plugin has editing support
    - Syntax checking, folding