Speaker Notes: Ammonite: Succinct Scala Shell Scripting

Markus Dale, medale@asymmetrik.com

October 2019

Setup

- Open Ammonite site https://ammonite.io
- Download page: https://github.com/lihaoyi/Ammonite/releases
- Requests lib: https://github.com/lihaoyi/requests-scala
- CSV data: https://catalog.data.gov/dataset?res_format=CSV&page=2, https://data.ny.gov/api/views/e8ky-4vqe/rows.csv?accessType=DOWNLOAD

Intro, Slides And Code

- Bio:
  - mostly Java, big data with Hadoop
  - big data with Spark, Databricks, Scala
  - Now Asymmetrik - Scala, Spark, Elasticsearch, Akka...
  - Data Engineer
  - Slides: https://github.com/medale/prez-ammonite-scala-shell/blob/master/presentation/SparkDataEngineering.pdf
  - Code Examples: https://github.com/medale/prez-ammonite-scala-shell/code

Examples of Shell Scripting

- Automate a small but labor-intensive task
- Run clean-up jobs with cron
- Refresh all git repositories (git pull)
- build and publish all projects under a common dir

That Bash Syntax

- hard to remember if not used all the time
- each command has its own set of options
- Can pipe things together - very useful
- No types - mostly treated as strings (other than exit code)
- Dr. Google - but fragile to maintain

Scala: file exists?

- Use java import
- More verbose

Scala: find and delete

- from 1 liner to 8 (and don’t forget close)

Ammonite: file exists?

- Domain-specific language for Scala shell scripting
- Make common tasks as terse as possible (but typed)
- imports Scala commands and implicits to convert strings to paths
- spawn subprocesses and pipes
- additional ease of use to remove need for full project/build system

Installing Ammonite

- Open https://github.com/lihaoyi/Ammonite/releases

Other ways of installing Ammonite

- `curl`
- `brew install ammonite-repl`

Configure the shell ~/.ammonite/predef.sc

- Download from https://github.com/lihaoyi/Ammonite/shell/src/main/resources/ammonite/shell/example-predef.sc

Major improvements over Scala REPL

- open Scala REPL and Ammonite
- Syntax highlighting input/output
- output valid Scala code: `Seq.fill(10)("Hello, Ammonite")`
- multi-line editing (up arrow and search)
- Classpath search: `StandardCharsets<TAB>` (if not imported yet but on classpath)

Magic import `$file`: Importing scripts

```
import $file.CommonImports
CommonImports.printCwd //accesses Paths OK
Paths.get("foo") //no imports - error
CommonImports.printWithNewlines(List(1,2,3))

import $file.scripts.Utils
Utils.du()
import Utils._
du()

import $file.^.print
print.message()
```

Magic import $exec: Bring in the defs AND imports

- all definitions AND import statements

```
//contains: import java.nio.file._

import $exec.CommonImports
Paths.get("foo")
import CommonImports._
printWithNewlines(List(1,2,3))
```

Magic import $ivy: Download libraries and their dependencies

```
import $ivy.`com.univocity:<TAB>`

import $ivy.`com.univocity:univocity-parsers:<TAB>`

import $ivy.`com.univocity:univocity-parsers:2.8.3`

CsvParser<TAB>
import com.univocity.parsers.csv.CsvParser
...
```

repl object

```
repl.imports        //user imports only
repl.fullImports    //also Ammonite imports

repl.clipboard.read/write

repl.clipboard.write("Hello from Ammonite")

repl.sess.save("clean")
repl.sess.load("clean")
```

interp object

```
interp.load.cp('lib/”baz.jar”)

import foo.bar.Baz
Baz.VeryImportantNumber

source(Baz)

import java.net.URL

source(URL) //no - has to be object

source(new URL(”http://foo.bar”))
```

Paths - Path vs. RelPath

```
Path("/tmp")
RelPath("lib")
Path("temp")
RelPath("/lib")

val dir = "/tmp/rest"
ls! dir //no
Path("/tmp/rest")
root/'tmp/'rest
```

Special path variables

- `home` - `$HOME or user.home`
- `root` - `ls! root` == Bash: `ls /`
- `pwd` - `sys.props("user.dir")` //does not change
- `wd` - changes with cd! command
- `up` - `..` in path, e.g. `root/'tmp/up` == `root`

Ops on paths

- `ls!` == `ls! wd` == `ls(wd)`
- `rm! dir` - Bash: `rm -R dir`
- `cp(src,dest)`
- `mv(src,dest)`
- `mkdir! newDirPath` - Bash: `mkdir -p newDir`
- `stat! filePath`
- `ls!, cd! root <TAB>`
- `os.symlink/hardlink`

read and write

- `write` https://github.com/lihaoyi/os-lib/blob/master/os/src/os/Source.scala

```
val imports = read("CommonImports.sc")
val lines = read.lines("CommonImports.sc")
val in = read.getInputStream("CommonImports.sc")

import $ivy.`com.typesafe.akka::akka-http:10.1.9`
val app = read(resource / "reference.conf")
write("defaults.conf", app) //also input stream, byte array
//write.over
```

Spawning subprocesses

```
%git 'status
%git("status", "--help")
%%git("status", "--help")
```

Pipes and grep!

```
ls! wd | grep! ".*\\.sc".r //error two implicits
def isFile(f: Path) = f.isFile
val allFiles = ls.rec! wd |? isFile | read
browse(allFiles)
//flatMap - Seq[GrepResult]
ls! wd || grep! ".*\\.sc".r
//filter - Seq[Path] implicit to Boolean
ls! wd |? grep! ".*\\.sc".r

def sum(a: Int, b: Int) = a + b
List(1,2,3) |& sum

ls! wd |? grep! "foo|bar".r
```

Built-in libraries: upickle and requests-scala

```
val lines =
  read.lines('input / "ids.json", StandardCharsets.UTF_8)
val jsons = lines.map { l =>
   ujson.read(l)
}
jsons.foreach { obj =>
  obj("source") = ujson.Str("asymmetrik")
  obj("dest") = ujson.Num(42)
}
jsons.foreach { obj =>
   obj.obj.remove("dest")
}
val response = requests.post("http://httpbin.org/post",
   headers = Map("accept"->"application/json"),
   data = ujson.write(ujson.Obj("fp" -> "map")))
```

Demos

- WikipediaVocab.sc
- Spreadsheet.sc