Beyond Bash

Shell scripting in a typed, OO language

Scala by the Bay, 15 August 2015
Slides: http://tinyurl.com/beyondbash
0.1 Who am i

Li Haoyi

Paid $ to work on dev tools @ Dropbox

Not paid $ to work on Scala.js

Using Scala professionally since… never
0.2 Agenda

- 0.x: Agenda
- 1.x: Bash
- 2.x: Ammonite-Ops
- 3.x: Ammonite-REPL
- 4.x: Conclusion
- 5.x: Q&A
0.3 Problem Statement

“How can we stop using the worst languages in the world to build our most important infrastructure?”
1.1 Application Architecture
1.2 Application Architecture

Diagram showing the application architecture with labels such as HTML, Server, Client, Safety, and a question mark indicating maybe Safety?
1.3 Scala.js!

Javascript: Problem solved

Scala.js works

Check it out if you haven't

http://www.scala-js.org/
1.3 Scala.js!

- Casting is great
  - `elem.asInstanceOf[html.Input]`
  - In Javascript, *every expression is a cast!*

- Weird, unsound behavior is fine
  - As long as it’s less weird/unsound than Javascript

- Best-effort error-handling is outstanding
  - Javascript doesn’t put in effort at all
Bad when better than worse is excellent
1.4 Application Architecture

Bash, Python, Puppet, Ruby, Vagrant...

Maybe Safety?

DANGER
1.5 Danger Below!

High-performance, type-safe application code

High-performance, type-safe web front-end

Underpinned by a mix of Bash, Python, Ruby, Puppet, Vagrant, ...
1.5 Danger Below!

Hard to test!

Not typechecked!

Worst consequences for errors
1.5 Danger Below!

Bash, Python, Puppet, Ruby, Vagrant...
1.5 Danger Below!

HTML

Client

Server

Bash, Python, Puppet, Ruby, Vagrant...

Database

Ok
1.5 Danger Below!

Bash, Python, Puppet, Ruby, Vagrant...
1.6 What's wrong with Bash?

- Obscure syntax
  - Even though you use it every day for 10 yrs
- Everything is global
  - Everything is spooky!
- Everything is a String
- Even basic math/logic is incredibly difficult
1.7 What's wrong with Bash?

# Run a script on all files with some extension

```bash
find . -name '*.ext' | while IFS="\n" read -r FILE; do
    process "$(readlink -f "$FILE")" || echo "error processing: $FILE"
done
```

```bash
find . -name '*.ext' \
( -exec ./some_other_script "$PWD"/{} \; -o -print \)
```

```bash
find . -name '*.ext' -exec ./some_other_script "$PWD"/{} \
```


It seems to work

Incorrect

Correct Solution!

It seems to work
“It seems to work”

Such a high degree of confidence!
Why do people use Bash

Can we use something else?
Sample use case

- List the things in my current folder
- Look at my current git
- Make a folder with a file inside
- Delete the folder
Why do people use Bash

Can we use something else?
No
Bash is Better
1.9 Bash vs Scala

```
rm -rf folder/inner_dir
```

```scala
def removeAll(path: String) = {
  def rec(f: File): Seq[File] =
    f.listFiles
    .filter(_.isDirectory)
    .flatMap(rec)
    .++(f.listFiles)
  for(f <- rec(new File(path))){
    if (!f.delete())
      throw new RuntimeException()
  }
}
removeAll("folder/inner_dir")
```
1.10 Bash vs Python

rm -rf folder/inner_dir

import shutil
shutil.rmtree('folder/my_file.jpg')
1.11 Bash vs Python: Round 2

import subprocess
subprocess.check_call(['git', 'status'])

Important Bits

Dumb Noise
1.12 Bash is Better

Less syntactic ceremony

Common operations are short

Fewer keystrokes overall

Commands do what you want

Very Important!
Ammonite-Ops

Rock-solid filesystem ops in Scala

"com.lihaoyi" %% "ammonite-ops" % "0.4.5"
2.1 Ammonite-Ops

● Goals:
  ○ No more than 2x as verbose as Bash
  ○ Safer than working with Python or java.{io, nio}

● Non-Goals!
  ○ Monadic pure dependent-typed safety
  ○ Reactive manifesto accreditation
  ○ 50-year enterprise maintainability
2.2 Ammonite-Ops

git status

rm folder/my_file.jpg

%git 'status

rm! 'folder/"my_file.jpg"

1 line 10 chars

1 line 25 chars
2.3 A Taste of Ammonite

import ammonite.ops._

// Delete a file or folder
rm! cwd/'folder

// Make a folder named "folder"
mkdir! cwd/'folder

// Copy a file or folder
cp(cwd/'folder, cwd/'folder1)

// List the current directory
val listed = ls! cwd

Short commands that mirror Bash

That do what you want!

No ambiguity in parsing arguments
2.4 A Taste of Ammonite

// List the current directory
val listed: Seq[Path] = ls! cwd

// Commands return normal values
// you can process normally
for(path <- listed){
  println(path)
  // paths are proper data-structures
  // with attributes, methods, etc.
  if (path.ext == "tmp") rm! path
}

Values are typed, structured data

No string munging trying to do simple tasks!
2.5 Piping

- `things | f` -> `things map f`
- `things || f` -> `things flatMap f`
- `things |? f` -> `things filter f`
- `things |& f` -> `things reduce f`
- `things |! f` -> `things foreach f`
- `things |> f` -> `f(things)`
- `f! thing` -> `f(thing)`

Traversable

Any

T => V
2.6 Putting it Together

- Concise filesystem operations
  - `ls! cwd`

- Structured, concise path operations
  - `ls! cwd/src/main`

- Pipes as aliases for collection methods
  - `ls! cwd/src/main | (? (_ext == "scala") | (_size) sum)`
2.7 Putting it Together

# Recursive line count of Javascript files
find ./dir -name '*.js' | xargs wc -l

38 chars

ls.rec! cwd/'dir |? (_.ext == "js") | read.lines | (_size) sum

64 chars
2.8 Putting it Together

# List dot-files *only*

```
ls -a | grep "^\.
```

19 chars

```
ls! cwd |? (_last(0) == '.')
```

30 chars
2.9 Putting it Together

# Largest 7 files in the current directory

```
fine . -ls | sort -nrk 7 | head -7
```

35 chars

```
ls.rec! cwd | (x => x.size => x) sortBy (-_.1) take 7
```

55 chars
2.10 Ammonite-Ops

- Easy, convenient filesystem ops in Scala!
- (Almost) as concise as Bash
  - Definitely less typing than java.io/nio
- Clean, structured data-model
  - Paths. Are. Not. Strings!
  - Results from commands aren’t strings either
2.11 This begs the question...

Can we use Ammonite-Ops + Scala-REPL as our default shell?

Let’s try contributing some changes to https://github.com/lihaoyi/demo
No
2.12 No

- Echo-ed output is unreadable
- Ctrl-C kills everything; bye bye work!
- Can’t subprocess out w/o borking JLine
- [http://lihaoyi.github.io/Ammonite/#OtherFixes](http://lihaoyi.github.io/Ammonite/#OtherFixes)
Ammonite-REPL

Re-inventing the Scala REPL
3.1 Ammonite-REPL

● Goal
  ○ You should not need to exit the REPL

● How often do you need to restart Bash?
3.2 Using the Ammonite REPL

# Standalone Executable

curl -L -o amm https://git.io/v3E3V; chmod +x amm; ./amm

// SBT project

libraryDependencies += (  
  "com.lihaoyi" % "ammonite-repl" % "0.4.5" % "test"  
  cross CrossVersion.full
)

initialCommands in (Test, console) :=  
  """ammonite.repl.Repl.run("""")"""" // sbt test/console
Live Demo

Whee!
3.3 Fun Features

- Great pretty-printing
- Syntax-highlighted everything!
- Ctrl-C Interruptible
- Live-loading modules from maven central
- Multi-line editing!
3.4 Ammonite-REPL

- A strictly-better Scala REPL
- Usable in any SBT project
- Or standalone
3.5 This begs the question...

Can we use Ammonite-Ops + Ammonite-REPL as our default shell?

Let’s try contributing some changes to [https://github.com/lihaoyi/wootjs](https://github.com/lihaoyi/wootjs)
3.6 Ammonite-REPL

- Scala-REPL is not a plausible systems shell
- Ammonite-REPL is!
- (Possibly)
- You can do real work in it
3.7 Work In Progress

- Extensible Autocomplete
  - *Already* autocomplete properties, names in scope
  - *Need* to autocomplete filesystem paths
  - *Nice to have* autocomplete for ivy coordinates, etc.

- Fetch scaladoc, source to show in-terminal

- Windows support for Ammonite-REPL
  - Ammonite-Ops already works
Conclusion

WTF did we just do?
4.1 Ammonite...

Ammonite-Ops
Really-nice Filesystem Library

Ammonite-REPL
Really-nice Scala REPL

Bash Replacement?
4.2 Ammonite...

- Re-implemented much of Bash’s functionality in Scala
- Twisted Scala’s syntax into a weird, bash-like form
- Re-implemented the Scala REPL to make this work
Why?

Did we need to do so many things?
4.3 Why Not...

● Make Bash less unsafe?

● Make Python less verbose?

● Improve on java.io or java.nio?
4.4 Space of Possible Systems APIs

Cliffs of Insanity

Danger

FP, Inferred Types,

Ammonite

Bash

Python

java.io

java.nio

Plains of Diminished Returns
4.5 Problems w/ Scala as your Shell

- JVM takes time to boot up!
  - 3-4s startup time
  - Not just JVM boot but classloading, etc.

- 3-4s first command compile
  - 0.2-0.3s compile overhead after warmup

- Bash takes ~0.004s to boot, Python ~0.03s

- Jar is 30mb, jar + JVM is >100mb
4.6 Hopefully free improvements

- Java 9 w/ modules will help JDK size/speed
  - Can bundle minimal JVM for smaller executable
  - Fewer classes to load on boot

- Dotty would (hopefully) speed compilation
  - At least it can’t get much slower, right? Right?...

- Dotty Linker would help overall
  - Should cut down the amount of stuff to load/JIT
5.0 Application Architecture

Server

Client

Client

Database

Bash, Python, Puppet, Vagrant...

Maybe Safety?
5.1 Application Architecture

Bash, Bash, Bash, Puppet, Puppet, Vagrant...

Safety

DANGER

Maybe Safety?
5.2 Application Architecture
5.3 Beyond Bash

- [http://lihaoyi.github.io/Ammonite/](http://lihaoyi.github.io/Ammonite/)
- "com.lihaoyi" %% "ammonite-ops" % "0.4.5"
- curl -L -o amm https://git.io/v3E3V; chmod +x amm; ./amm
- Questions?
Additional Slides
2.5 Absolute Paths & RelPaths

```scala
case class Path(segments: Seq[String])
```

Absolute

```scala
case class RelPath(segments: Seq[String], ups: Int)
```

Any ..s at the start of the path
2.6 Constructing Paths

Paths are constructed using / and and...

- Segments
  - Strings
  - Symbols

- Builtins
  - root: Path
  - cwd: Path
  - up: RelPath

> root
/ 

> root/"usr/"bin
/usr/bin

> "src/"main
src/main

> up/up/"src/"main
../../src/main
2.7 Combining Paths

> val rel = 'src/\'main
src/main

> val wd = root/\'Users/\'lihaoyi
/Users/lihaoyi

> wd/rel
/Users/lihaoyi/src/main

> wd/rel/up
/Users/lihaoyi/src

Paths can be stitched together using /

Paths are normalized at every step!

not /Users/lihaoyi/src/..
2.8 Invalid Paths

```scala
> val rel: RelPath = 'src/'main
> val abs: Path = root/'usr/'bin

> abs/rel
/usr/bin/src/main
> rel/abs
<console>:15: error: type mismatch;
> rel/rel
src/main/src/main
> abs/abs
<console>:14: error: type mismatch;
```

Combining Paths & RelPaths improperly is a **compilation error**
2.9 Invalid Paths

> val rel: RelPath = 'src/'main
> val abs: Path = root/'usr/'bin

> abs/rel
/usr/bin/src/main
> rel/abs
> rel/rel
src/main/src/main
> abs/abs
<console>:14: error: type mismatch;
> abs/abs
<console>:15: error: type mismatch;
> os.path.join(abs, rel)