Scala => lift

David Pollak

May 21, 2008

David Pollak

- Not strict, but pretty lazy
- Lead developer for lift web framework
- Scala since November 2006, Ruby/Rails before that, Java/J2EE before that
- Spreadsheet junky (writing more than using)

Today

- Scala is a great language for building web frameworks and web apps
- Scala has a ton of features that make the job super-simple
- So... here we go...

Scala Functions

- Functions are Objects
- Objects can be passed as parameters
- Functions are syntactically easy to create
- var name = “”
  SHtml.text(name, name = _)
- They bind to variables/values (e.g. name)

Partial Functions

- PartialFunction[A,B] extends Function1[A,B]
- isDefinedAt(x: A)

- Better known as pattern matching:

```
{
  case Foo(bar) => bar
  case Baz(dog) => dog
}
```

Composing Partial Functions

```
{ case Foo(bar) => bar
  case Baz(dog) => dog
} orElse { // compose
  case Moo(cow) => cow
  case Meow(cat) => cat
}
```

Extractors and Guards

- Extract data while matching other parts in a pattern:

```
{ case "Foo" :: id :: Nil => doIt(id) }
```

- Guards:

```
{ case "Foo" :: id :: Nil
  if isValid(id) && loggedIn_? => doIt(id) }
```

Remembering Functions

- Functions are Objects
- Map[String, String => XML]
  Map[String, PartialFunction[String, XML]]
- GET /ajax?OPAQUE_ID=someValue
  Map[OPAQUE_ID](someValue)

XML literals and manipulation

- In Scala, XML is like String: supported at the language level and immutable
- <foo>{(1 to 10).map(i => <val>{i}</val>)}</foo>
- (xml \ "val").map(_.text.toInt).
  .foldLeft(0)(_ + _) == 55

Actors and Partial Functions

- Threadless, stackless units of execution
- React to events and otherwise consume nothing but memory
- react(PartialFunction[Any, Any]) ->
  react {case Foo(bar) => doSomething(bar)
  case Baz(dog) => doElse(dog) }
- react(primaryHndlr orElse secondaryHndler)

lift REST APIs

- LiftRules.addDispatchBefore {
  case RequestMatcher(RequestState(
  "showstates":: xs, _),_) =>
  XmlServer.showStates(xs) }

- def showStates(...) = XmlResponse(
  <states renderedAt={timeNow.toString}>
  ... </states>)

lift and HTML forms

- var name = ""
  text(name, name = _)

- def setLocale(loc: String) ...

  select(Locale.getAvailableLocales.toList.
  map(lo => (lo.toString, lo.getDisplayName)),
  setLocale)

lift & AJAX

- AJAX elements are bound to functions:

  a(() => {cnt = cnt + 1; SetHtml("cnt_id", Text
  (cnt.toString))}, “click me”)

- ajaxSelect(opts,

    v => DisplayMessage("You selected "+v))

lift CometActors

- lift deals with all the plumbing
- def render = bind("time" -> timeSpan)
- override def lowPriority = {
  case Tick => reRender(false)
  }

Questions

?