Scala for Scripting

Implementing an OSGi enabled, JSR-223 compliant scripting engine for Scala
http://people.apache.org/~mduerig/scala4scripting/

Michael Dürig
michael.duerig@day.com

Day Software AG
http://www.day.com/

Scala Days 2010
April 15th

Agenda

- **Introduction**
  - Scripting with Scala
  - Scala for Apache Sling
- Requirements and goals
- Challenges and solutions
- Conclusion

Scripting with Scala

- (Illusion of) executing source code
- Concise and versatile*

```
var capitals = Map("Switzerland" -> "Bern", "Spain" -> "Madrid")
capitals += ("France" -> "Paris")

val c = capitals.find(_._1 == "France")
  .map(_._2)
  .getOrElse("NIL")
```

- DSL capabilities
- Type safe

*) Adapted from: Programming in Scala, Martin Odersky, Lex Spoon, Bill Venners. Artima, 2008.

Scripting with Scala (cont.)

- XML literals: type safe templating

```
<html>
  <head>
    <link rel="stylesheet" href="/apps/forum/static/blue.css" />
  </head>
  <body>
    <div id="Header">
      Welcome to the { node("name") } forum
      { Calendar.getInstance.getTime }
    </div>
    <div id="Content">
      { SearchBox.render(request) }
      { ThreadOverview.render(node) }
    </div>
  </body>
</html>
```

Apache Sling

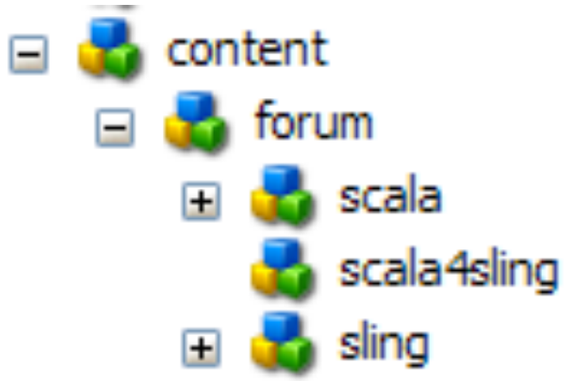
- Web application framework
  - Backed by a Java content repository (JSR-170/JSR-283): Apache Jackrabbit
  - Based on OSGi: Apache Felix
  - http://sling.apache.org/
- RESTful
  - Content resolution for mapping request URLs to resources (i.e. JCR nodes)
  - Servlet resolution for mapping resources to request handlers (i.e. scripts)
- Scriptable application layer
  - JSR-223: Scripting for the Java platform

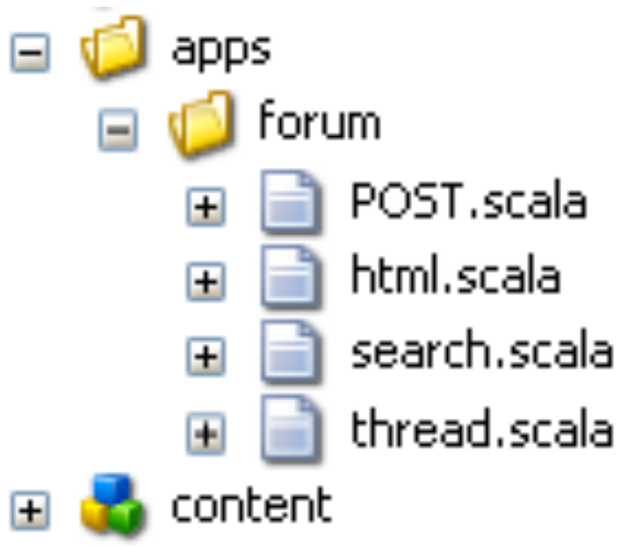
Sling URL decomposition


```
GET    /forum/scala4sling.html    HTTP/1.1
```

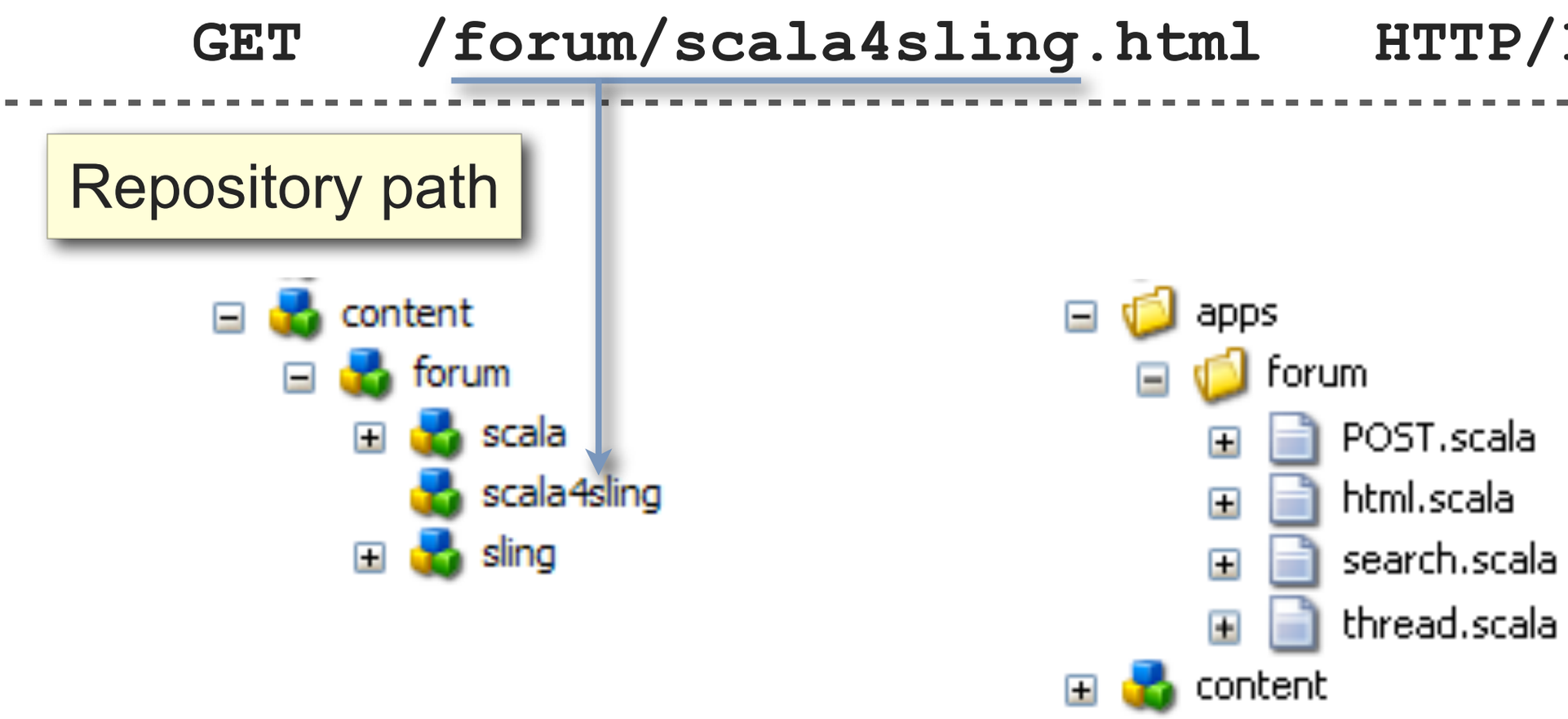
Sling URL decomposition


```
GET    /forum/scala4sling.html    HTTP/1.1
```

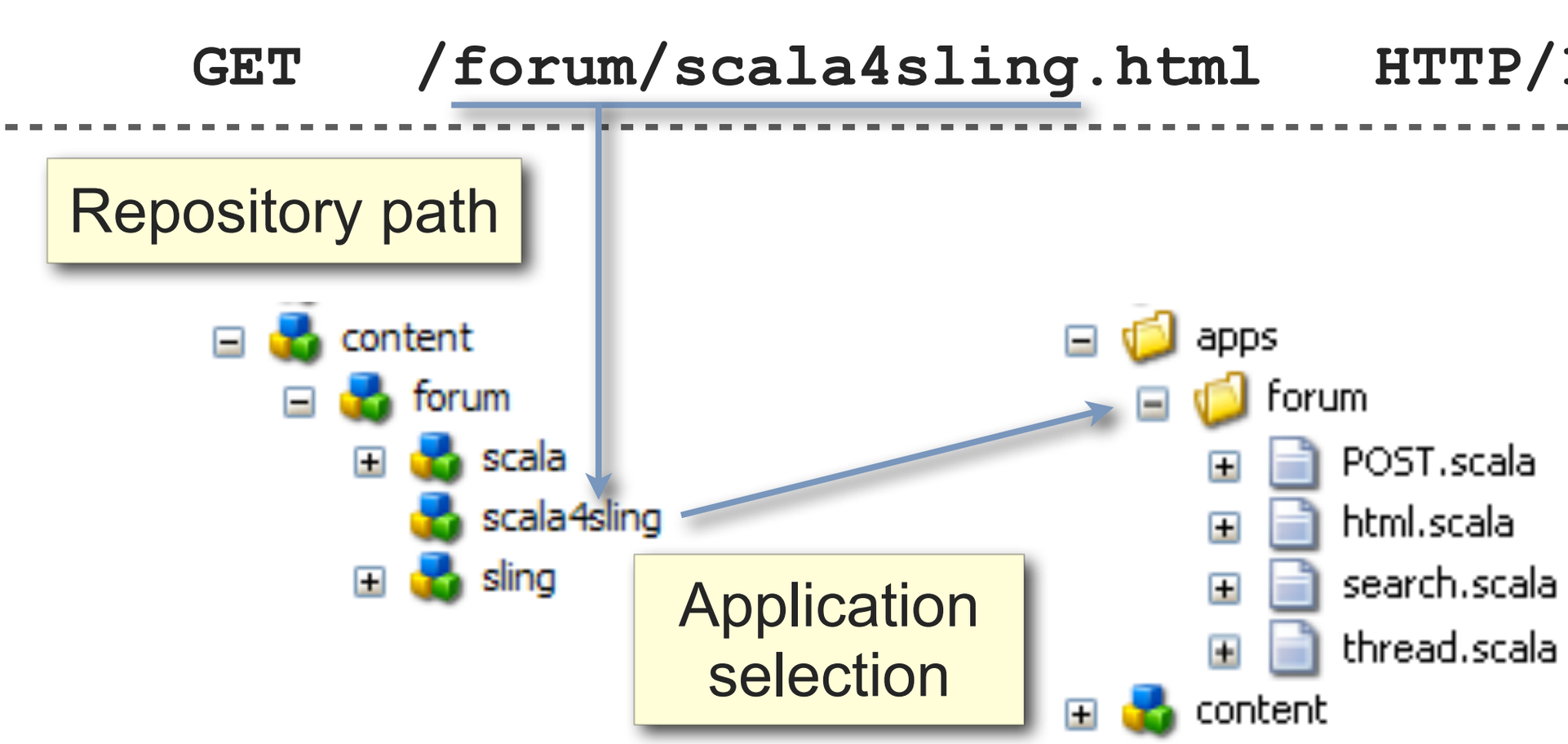




Sling URL decomposition



Sling URL decomposition

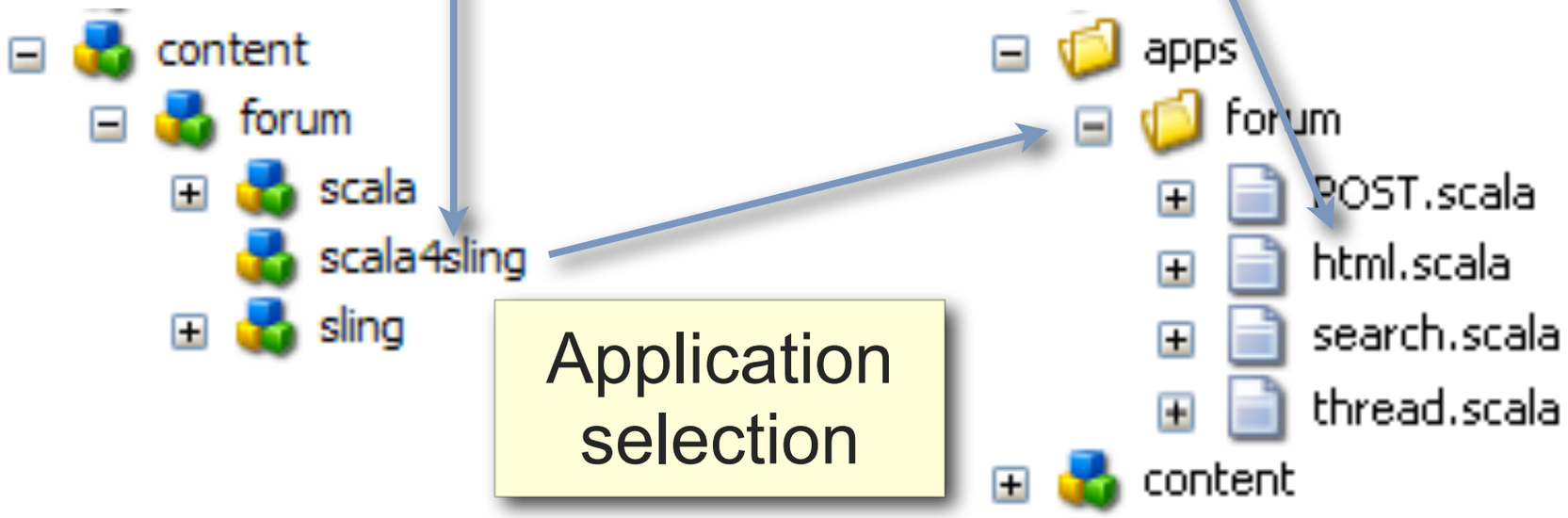


Sling URL decomposition

```
GET    /forum/scala4sling.html    HTTP/1.1
```

Repository path

Script selection



Application selection

Scala for Sling


```
package forum

class html(args: htmlArgs) {
  import args._
  // further imports omitted

  println {
    <html>
      <body>
        <div id="Header">
          Welcome to the { node("name") } forum
          { Calendar.getInstance.getTime }
        </div>
        <div id="Content">
          { SearchBox.render(request) }
          { ThreadOverview.render(node) }
        </div>
      </body>
    </html>
  }
}
```

Scala for Sling


```
package forum

class html(args: htmlArgs) {
  import args._
  // further imports omitted

  println {
    <html>
      <body>
        <div id="Header">
          Welcome to the { node("name") } forum
          { Calendar.getInstance.getTime }
        </div>
        <div id="Content">
          { SearchBox.render(request) }
          { ThreadOverview.render(node) }
        </div>
      </body>
    </html>
  }
}
```

Import bindings

Scala for Sling

```
package forum

class html(args: htmlArgs) {
  import args._
  // further imports omitted

  println {
    <html>
      <body>
        <div id="Header">
          Welcome to the { node("name") } forum
          { Calendar.getInstance.getTime }
        </div>
        <div id="Content">
          { SearchBox.render(request) }
          { ThreadOverview.render(node) }
        </div>
      </body>
    </html>
  }
}
```

Import bindings

Use arguments

Agenda

- Introduction
- **Requirements and goals**
  - JSR-223 compliance
  - OSGi tolerant
  - Further design goals
- Challenges and solutions
- Conclusion

JSR-223

- Scripting for the Java language
  - Allow scripts access to the Java Platform
  - Scripting for Java server-side applications
  - Executing scripts: `javax.script.{ScriptEngine, ScriptEngineFactory}`
  - Binding application objects into scripts: `javax.script.Bindings`

JSR-223: usage

```
val factories = ServiceRegistry.lookupProviders(classOf[ScriptEngineFactory])
val factory = factories.find(_.getEngineName == "Scala Scripting Engine")

val engine = factory.map(_.getScriptEngine).getOrElse {
  throw new Error("Cannot locate Scala scripting engine")
}

val bindings = engine.getBindings(ScriptContext.ENGINE_SCOPE)
bindings.put("request", servletRequest)
engine.eval(script, bindings)
```

OSGi

- OSGi Service Platform
  - Runtime environment for Java applications (bundles)
  - Module system
  - Life cycle management
  - Service registry
- Mostly transparent
  - Bundle information in manifest file
  - Relies on class loading

Further design goals

- Leverage existing tools
  - Compilers
  - IDEs
  - Test suites
  - Documentation and reporting tools
- Independent from Sling and OSGi
  - Versatile
  - Configurable
  - Embeddable

Agenda

- Introduction
- Requirements and goals
- **Challenges and solutions**
  - Leverage existing tools
  - Passing arguments to scripts
  - Scalac and OSGi
  - Performance
- Conclusion

Leverage existing tools

- Scripts are valid Scala source entities
- Tradeoff
  - ceremony

```
package forum

class html(args: htmlArgs) {
  import args._
  // ...
}
```

- Advantages
  - Write, refactor and debug with Scala IDEs
  - Compile with Scala compiler
  - Unit testing

Unit testing

```
class html(args: htmlArgs) {
  import args._
  // further imports omitted

  println {
    <html>
      <body>
        <div id="Header">
          Welcome to the { node("name") } forum
          { Calendar.getInstance.getTime }
        </div>
        <div id="Content">
          { SearchBox.render(request) }
          { ThreadOverview.render(node) }
        </div>
      </body>
    </html>
  }
}
```

Unit testing

```
class html(args: htmlArgs) {
  import args._
  // further imports omitted

  println {
    <html>
      <body>
        <div id="Header">
          Welcome to the { node("name") } forum
          { Calendar.getInstance.getTime }
        </div>
        <div id="Content">
          { SearchBox.render(request) }
          { ThreadOverview.render(node) }
        </div>
      </body>
    </html>
  }
}
```

```
class htmlArgs {
  val node = new {
    def apply(name: String) = "Scala scripting"
  }
  val request = new { /* ... */ }
}

new html(new htmlArgs)
```

Unit testing

```
class html(args: htmlArgs) {
  import args._
  // further imports omitted

  println {
    <html>
      <body>
        <div id="Header">
          Welcome to the { node("name") } forum
          { Calendar.getInstance.getTime }
        </div>
        <div id="Content">
          { SearchBox.render(request) }
          { ThreadOverview.render(node) }
        </div>
      </body>
    </html>
  }
}
```

```
class htmlArgs {
  val node = new {
    def apply(name: String) = "Scala scripting"
  }
  val request = new { /* ... */ }
}

new html(new htmlArgs)
```

```
<html>
  <body>
    <div id="Header">
      Welcome to the Scala scripting forum
      Tue Mar 16 19:46:38 CET 2010
    </div>
    <div id="Content">
      <!-- output omitted -->
    </div>
  </body>
</html>
```

Passing arguments to Scripts

```
class ResettableOutStream extends OutputStream implements Resettable {
  public void write(int b) throws IOException { /* ... */ }
  public void reset() { /* ... */ }
}

OutputStream getResetableOutputStream() { /* ... */ }
```

```
bindings.put("output", getResetableOutputStream)
engine.eval(script, bindings)
```

```
class html(args: htmlArgs) {
  import args._

  output.write('c')
  output.reset()
}
```

Passing arguments to Scripts

```
class ResettableOutStream extends OutputStream implements Resettable {
  public void write(int b) throws IOException { /* ... */ }
  public void reset() { /* ... */ }
}

OutputStream getResetableOutputStream() { /* ... */ }
```

```
bindings.put("output", getResetableOutputStream)
engine.eval(script, bindings)
```

Where are the types?

```
class html(args: htmlArgs) {
  import args._

  output.write('c')
  output.reset()
}
```

Passing arguments to Scripts

```
class ResettableOutStream extends OutputStream implements Resettable {
  public void write(int b) throws IOException { /* ... */ }
  public void reset() { /* ... */ }
}

OutputStream getResetableOutputStream() { /* ... */ }
```

```
bindings.put("output", getResetableOutputStream)
engine.eval(script, bindings)
```

Where are the types?

```
class html(args: htmlArgs) {
  import args._

  output.write('c')
  output.reset()
}
```

Bindings wrapper provides static types

Bindings wrapper

- No type information in `javax.script.Bindings`
  - Generate bindings wrapper exposing static types
  - Arguments appear to be of all accessible types
  - Implicit conversions to the rescue
- Least accessible types
  - *v*: *C*, I_1, ..., I_n
  - Expose *v* with static type *D* of least accessible super type of *C*
  - Expose *v* through implicit conversion to I_k if neither *D* nor any I_m ($m \neq k$) implements I_k.

Bindings wrapper (cont.)

```
class ResettableOutStream extends OutputStream implements Resettable {
  public void write(int b) throws IOException { /* ... */ }
  public void reset() { /* ... */ }
}

OutputStream getResetableOutputStream() { /* ... */ }
```

```
bindings.put("output", new ResettableOutStream)
engine.eval(script, bindings)
```

Bindings wrapper (cont.)

```
class ResettableOutStream extends OutputStream implements Resettable {
  public void write(int b) throws IOException { /* ... */ }
  public void reset() { /* ... */ }
}

OutputStream getResetableOutputStream() { /* ... */ }
```

```
bindings.put("output", new ResettableOutStream)
engine.eval(script, bindings)
```

Bindings wrapper

```
class htmlArgs(bindings: Bindings) {
  lazy val output = bindings.getValue("output").asInstanceOf[OutputStream]
  implicit def outputStream2Resettable(x: OutputStream): Resettable =
                x.asInstanceOf[Resettable]
}
```

Bindings wrapper (cont.)

```
class ResettableOutStream extends OutputStream implements Resettable {
  public void write(int b) throws IOException { /* ... */ }
  public void reset() { /* ... */ }
}

OutputStream getResetableOutputStream() { /* ... */ }
```

```
bindings.put("output", new ResettableOutStream)
engine.eval(script, bindings)
```

Bindings wrapper

```
class htmlArgs(bindings: Bindings) {
  lazy val output = bindings.getValue("output").asInstanceOf[OutputStream]
  implicit def outputStream2Resettable(x: OutputStream): Resettable =
                x.asInstanceOf[Resettable]
}
```

Bindings wrapper (cont.)

- Limitations
  - Scala's visibility modifiers not exposed through reflection
  - Not yet working with parametrized types
  - Ambiguities in generated implicit conversion

Scalac and OSGi

- Scalac
  - Requires compiler class path
  - File system based
- OSGi
  - Provides class loaders
  - Bundle based
- Bridging the gap
  - File system abstraction over OSGi bundles
  - Implement `scala.tools.nsc.io.AbstractFile`
  - Limitation: wiring probably not fully respected

Independent through configuration

- Abstract implementation details into configuration class
  - Default implementation for standalone usage
  - Specialized implementations for Sling
  - Set through `ScriptEngineFactory` or injected via OSGi service lookup

Configuration

- Scala compiler
  - Per script engine
  - `scripting.scala.SettingsProvider`
  - Class and output paths: `scala.tools.nsc.io.AbstractFile`
  - Settings: `scala.tools.nsc.Settings`
  - Reporting: `scala.tools.nsc.reporters.Reporter`
- Script entry point
  - Per invocation
  - `scripting.scala.ScriptInfo`

Performance

- On the fly compilation is slow
  - Cache pre compiled scripts
  - Dependency on bindings: include bindings wrapper
  - Work in progress

Conclusion

- Scala is attractive for scripting
  - Concise and versatile
  - DSL capabilities
  - Type safe
  - XML literals for type safe templating
- Impedance mismatches
  - JSR-223: dynamic vs. static typing
  - OSGi: runtime environment vs. compile time utility
  - Performance: *illusion of* executing source code

References

- Scala for Scripting
  http://people.apache.org/~mduerig/scala4scripting/
- Apache Sling
  http://sling.apache.org/

- Michael Dürig
  michael.duerig@day.com
- Day Software AG
  http://www.day.com/