Simplify Enterprise Development With Scripting

Guillaume Laforge

Software Architect
OCTO Technology
http://www.octo.com

Tim Gleason/Tugdual Grall

OracleAS Development
Oracle Corporation
http://www.oracle.com

TS-1246

Groovy Goal of This Talk

Simplifying Enterprise Development with Scripting

Learn how **scripting** can increase your productivity and help you build and test solutions faster on the **Java™ Platform, Enterprise Edition**

Agenda

Scripting back on rails

What is Groovy, and why it matters?

Groovy syntax, APIs and beyond

SOA scripting

JSR 223, an API to rule them all

Conclusion

Agenda

Scripting back on rails

What is Groovy, and why it matters?

Groovy syntax, APIs and beyond

SOA scripting

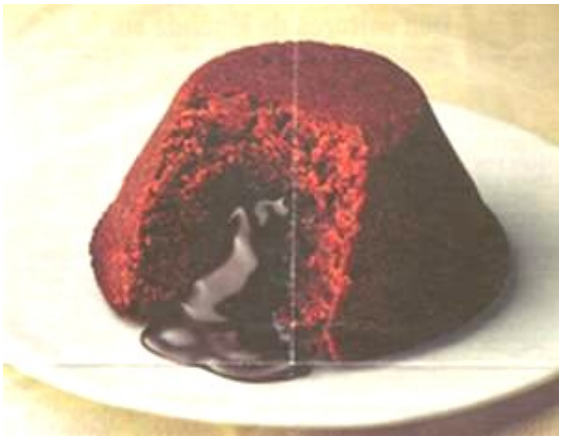
JSR 223, an API to rule them all

Conclusion

Full Steam Ahead on Scripting!

- Scripting languages are in fashion (again?)
  - **AJAX** with JavaScript™ technology, and **Ruby on Rails** with Ruby
- **Java SE 6 integrates JSR 223** with Rhino
- JCP℠ agreed to standardize Groovy (JSR 241) and BeanShell (JSR 274)
- Upcoming bytecode **invokeDynamic** (JSR 292)
- Microsoft also believe in dynamic languages and hire IronPython lead
- Need simplicity to overcome enterprise development complexity → **Groovy can help!**

Scripting Integration Patterns



“Super-glue” Pattern

- Glue together isolated components or applications to make them interact and bring value through the combination

“Liquid Center” Pattern

- Customize, externalize business rules, presentation logic to make applications evolve with more agility

Three Main Use Cases of Integration

- **Prototyping/testing/scripting**
  - Shell or build scripting, data manipulation, unit testing, code generation, driving native applications
- **Building standalone applications**
  - Small to mid-sized non-critical applications
- **Integrating scripting in Java EE applications**
  - Programmatic configuration (less XML)
  - Business rules externalization
  - UI or application customizations

Java Technology and Scripting

- **Leverage Java Platform**
  - Use scripts inside Java SE and Java EE
  - Reuse existing Java technology skills and components
- Many existing languages on the Java VM
  - Groovy, Rhino (JavaScript), BeanShell, Jython, JRuby, Pnuts, Scala…
- Current **standardization effort** around scripting languages and Java technology
  - JSR 223, JSR 241, JSR 274, JSR 292
- Let's dive in Groovy as an example…

Agenda

Scripting back on rails

What is Groovy, and why it matters?

Groovy syntax, APIs and beyond

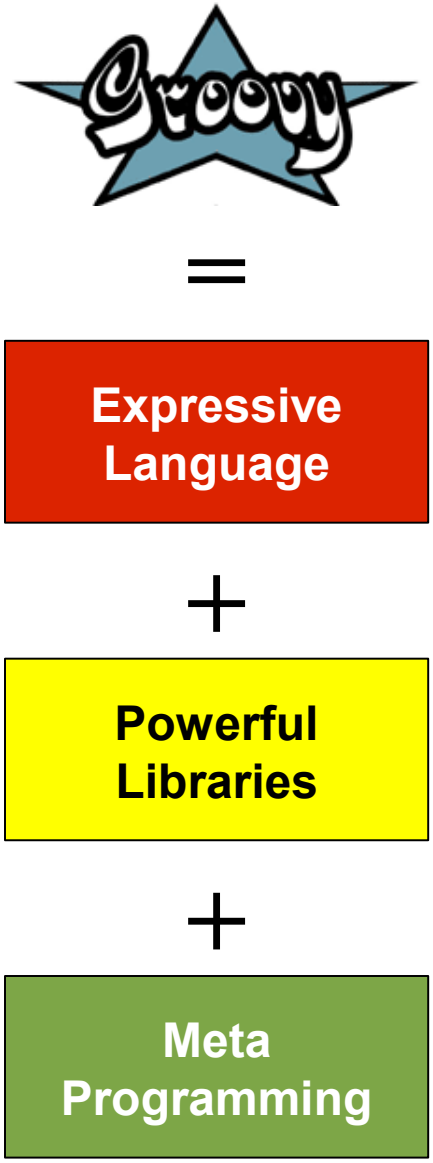
SOA scripting

JSR 223, an API to rule them all

Conclusion

Why Groovy Is Relevant?

- Groovy is a dynamic and agile scripting language for the Java VM
- OSS project hosted by **Codehaus**
- Inspired by Ruby and Python
- **Generates bytecode** for the Java VM: integrates well with Java libraries
- However, Groovy differentiates itself
  - **GDK**: additional libraries to simplify complex APIs
  - **MOP**: advanced meta-programming features



Groovy in Production

- **Finance/Insurance**: risk or rate computation
  - Mutual of Omaha (Fortune 500)
  - Aspect Capital (Finance, London)
- **Collaborative software**:
  - eXo Platform (eXo SARL), XWiki (XpertNet)
- **Web sites**: PepsiCo, UUZone.com
- **Frameworks**:
  - Spring, RIFE, NanoContainer, Drools, ServiceMix
- **E-Learning**: Maxim Learning (Grails' lead)
- **Testing**: Canoo Web Test

Agenda

Scripting back on rails

What is Groovy, and why it matters?

Groovy syntax, APIs and beyond

SOA scripting

JSR 223, an API to rule them all

Conclusion

Groovy at a Glance

- Expressive Java language-like syntax
- Same OO model as Java technology
- Can be used in a shell or embedded
- Supports “**duck typing**”, and strong typing
- Native syntax constructs for Lists, Maps, regex
- Permits operator overloading
- GStrings: interpolated strings
- **Closures**: reusable and assignable code blocks

Groovy Syntax Basics

```
class Speaker {
    @Property String name
    @Property age
    String toString() {
      "My name is $name and I'm $age"
    }
}

def speakers = [
    new Speaker(name:'Tim Gleason', age:30),
    new Speaker(name:'Guillaume Laforge', age:28)
]

def upper = { it.toString().toUpperCase() }
speakers.findAll { name -> name =~ /.*Tim.*/ }
        .collect(upper).each { println it }
```

Groovy version of a JavaBean

Getter/setters generated

GString

Weakly typed

Native syntax for lists: [a, b, c]
For maps: [a:1, b:2, c:3]

Default ctor with call of setters

Closure with implicit parameter

Regex and matcher

Named closure

Inline closure

Groovy Development Kit

- **JDK™ software class extensions**, and write your own!
  - String → each(), execute(), center(), reverse()
  - Collection → collect(), join(), findAll(), inject()
  - File → eachFile(), eachLine(), getText()
- Powerful APIs
  - Easy to play with JDBC™ software
  - GString powered template engine
  - Handy XML parsers and GPath
  - Manipulating JMX™ MBeans API as local objects
  - Additional modules: SOAP, scripting ActiveX/COM

JDBC Software

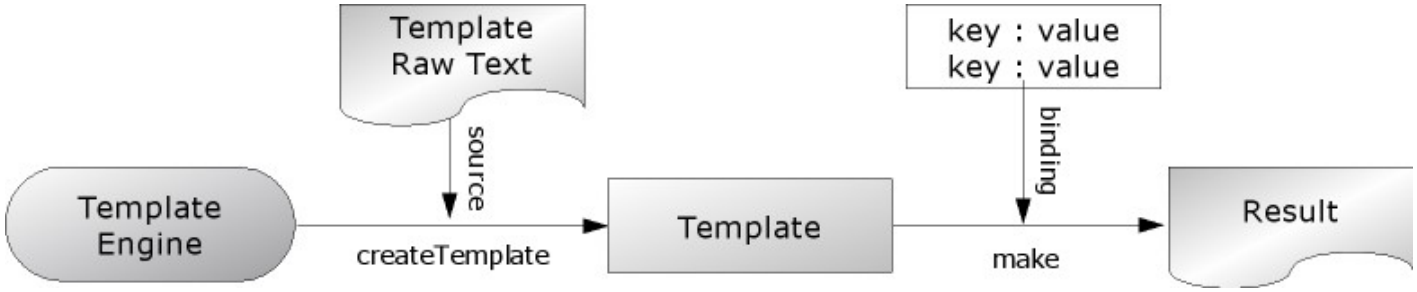
- Easy to use JDBC software thanks to closures

```
def sql = Sql.newInstance(url, usr, pwd, driver)
sql.execute("insert into table values ($foo, $bar)")
sql.execute("insert into table values(?,?)", [a, b])
sql.eachRow("select * from USER") { print it.name }
def list = sql.rows("select * from USER")
```

- DataSet notion: poor-man ORM

```
def set = sql.dataSet("USER")
set.add(name: "Johnny", age: 33)
set.each { user -> println user.name }
set.findAll { it.age > 22 && it.age < 42 } // LINQ ☺
```

Template Engine



- Dear $fname ${lname.toUpperCase()},
  nice to meet you in <%= city %>.
  ${signed}

```
def binding = [fname:"Joe",   lname: "Bar",
               city: "Paris", signed:"MrG"]
def engine = new SimpleTemplateEngine()
def template =
    engine.createTemplate(text).make(binding)
println template.toString()
```

Parsing XML and GPath Expressions

- XML's never been that fun to parse!

```
def xml = """
<languages>
   <language name="Groovy">
      <feature coolness="low">SQL</feature>
      <feature coolness="high">Template</feature>
   </language>
   <language name="Perl"/>
</languages>"""
```

- XML nodes are mere properties!

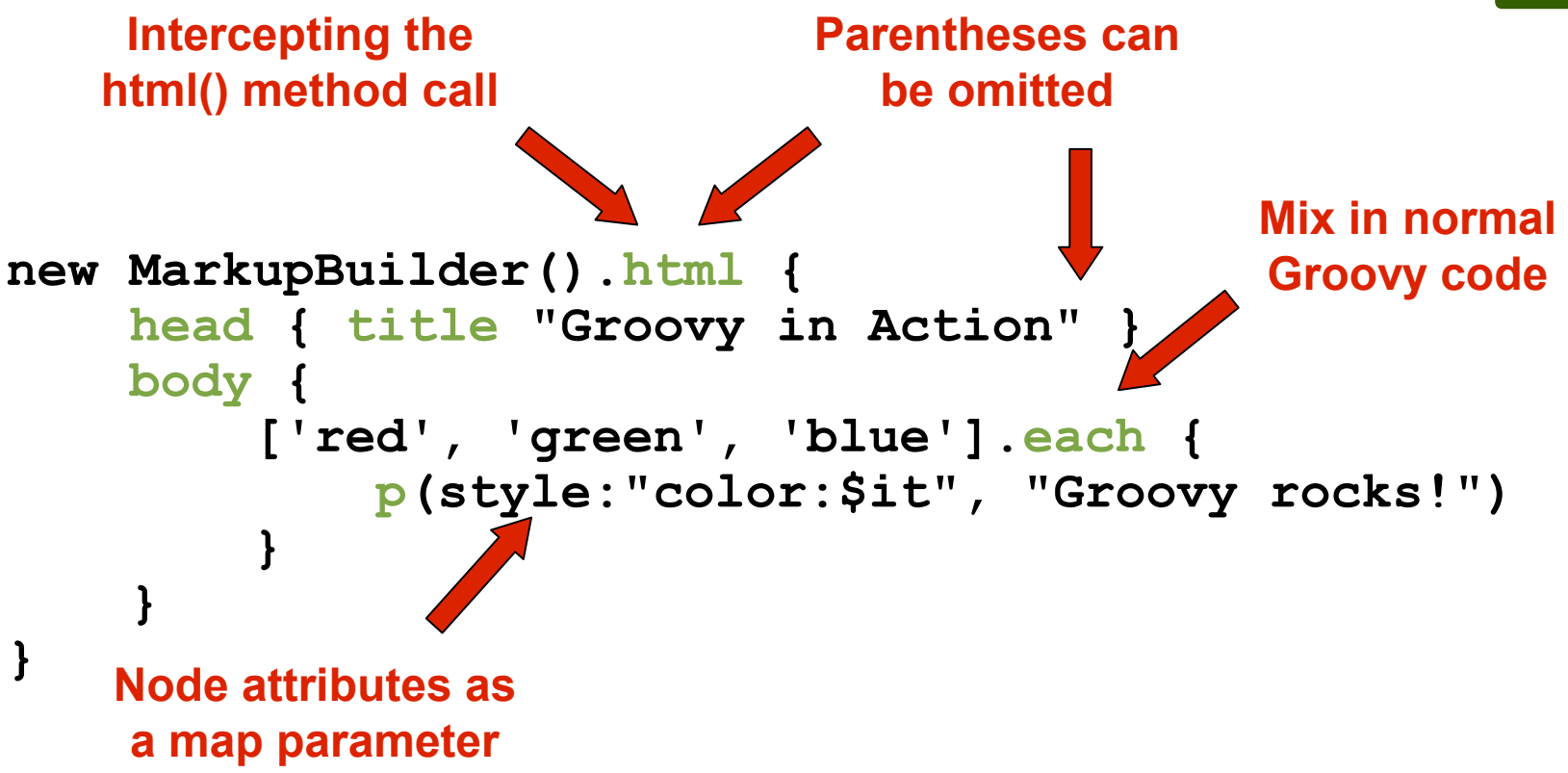
```
def root = new XmlParser().parseText(xml)
println root.language.feature[1].text()

root.language.feature
    .findAll{ it['@coolness'] == "low" }
    .each{ println it.text() }
```

Groovy's MOP

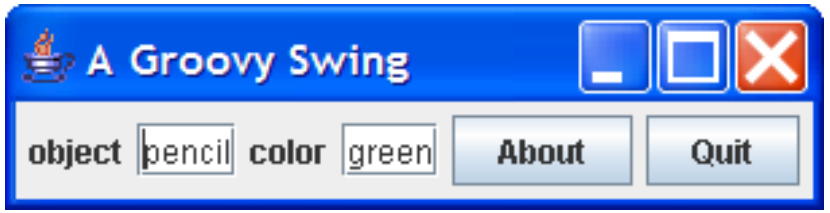
- What's that MOP $#@??
  → **Meta Object Protocol**
- At runtime, allows you to
  - Intercept method calls (even non-existing ones)
  - Intercept property accesses
  - Inject new methods or new properties
- You can create **Domain Specific Languages**
- With the MOP, come the…builders
  - **Builder pattern at the syntax level!**
  - MarkupBuilder, SwingBuilder, AntBuilder…

MarkupBuilder

- Creating XHTML markup programmatically



SwingBuilder



- Composition and nesting of Swing components

```
def theMap = [color: "green", object: "pencil"]
def swing = new SwingBuilder()
def frame = swing.frame(
            title: 'A Groovy Swing', location: [240,240],
            defaultCloseOperation:WindowConstants.EXIT_ON_CLOSE) {
  panel {
    for (entry in theMap) {
      label(text: entry.key)
      textField(text: entry.value)
    }
    button(text: 'About', actionPerformed: {
      def pane = swing.optionPane(message: 'SwingBuilder')
      def dialog = pane.createDialog(null, 'About')
      dialog.show()
    })
    button(text:'Quit', actionPerformed:{ System.exit(0) })
  }
}
frame.pack()
frame.show()
```

AntBuilder

- Reuse hundreds of Ant tasks as simple method calls
- Even create your own build system!

```
def ant = new AntBuilder()
def patterns = ["**/*.groovy", "**/*.java"]
ant.sequential {
    mkdir(dir: "dir1")
    copy(todir: "dir1") {
        fileset(dir: "dir2") {
            for(pattern in patterns)
                include(name: pattern)
        }
    }
    echo("done")
}
```

DEMO

Administer an Application Server by Manipulating JMX-Based Beans with Groovy

Status

- Groovy RC-1 released recently
- Next month: release of **Groovy 1.0 final**!
- **JSR 241** will standardize Groovy 1.0
- Groovy already JSR 223 compliant
- First version of **Grails**
- Two books soon on the shelves, including
  - "**Groovy in Action**", Manning

Agenda

Scripting back on rails

What is Groovy, and why it matters?

Groovy syntax, APIs and beyond

SOA scripting

JSR 223, an API to rule them all

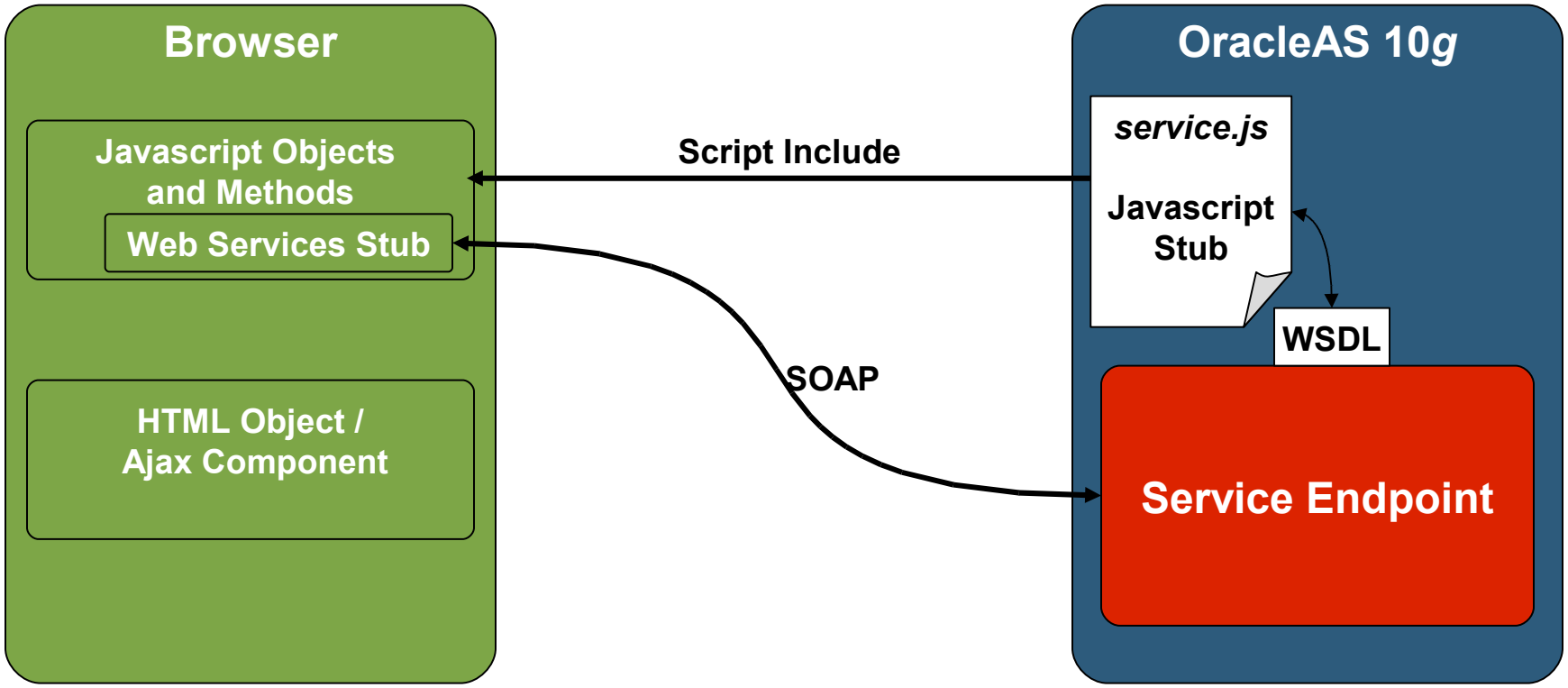
Conclusion

Simplify SOA With JavaScript Technology

- Generating JavaScript-based stub using Web Services metadata (WSDL)
  - Leverage the simplicity of JavaScript technology
  - Leverage Objects programming
- Use JavaScript
  - In the browser for AJAX applications
  - In the Java VM (Rhino/Java SE 6)

AJAX Web Services

Consume Web Services Directly From the Client



DEMO

Generate and Use Web Services Stubs for AJAX

Agenda

Scripting back on rails

What is Groovy, and why it matters?

Groovy syntax, APIs and beyond

SOA scripting

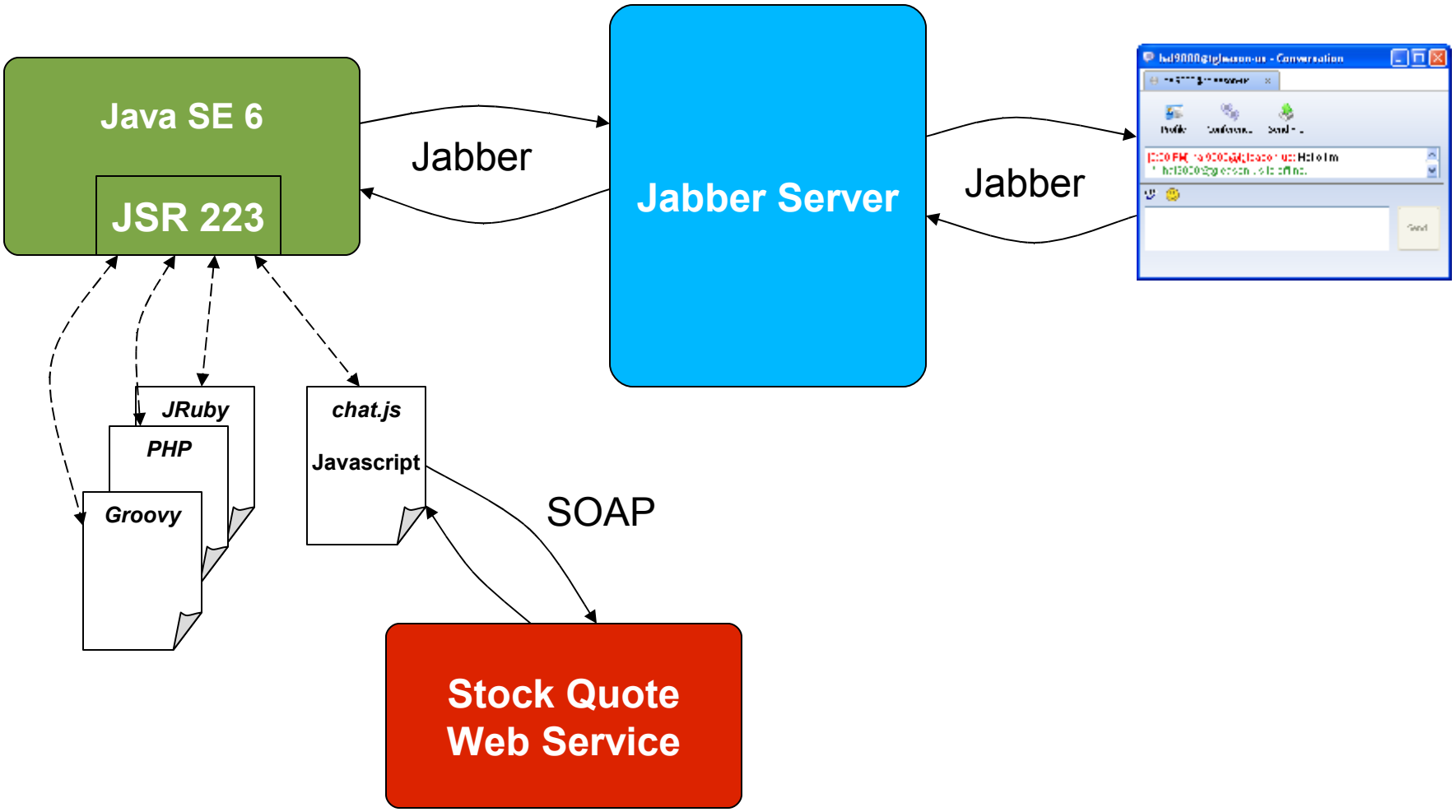
JSR 223, an API to rule them all

Conclusion

JSR 223—Scripting for the Java™ Platform

- **One API to rule them all!**
- JSE™ 6 includes JSR 223 and embeds JavaScript™ (Rhino)
- RI already usable starting from JDK 1.4
  - With Groovy, PHP, and Rhino
- ScriptEngines provide a common way to integrate stateful interpreters into Java
- Invocable and Compilable interfaces extend ScriptEngines to support generic function invocations and compilation of scripts

Chatting With Scripts



DEMO

Java SE 6 and Scripting

Chatting With Scripts

```
ScriptEngineManager manager =
        new ScriptEngineManager();

ScriptEngine engine =
        manager.getEngineByExtension(
            getExtension(args[0]));
```

Chatting With Scripts

Once:

```
engine.eval(new InputStreamReader(
    new FileInputStream(args[0])));
```

For Each Message:

```
invocableEngine.invoke("onMessage",
    new Object[]{chat, msgBody});
```

Chatting With Scripts: JavaScript

```
var SOAPScript = new Packages.soapscript.SOAPGen();
var namespace = SOAPScript.useWSDL("http://localhost:8888/…");
var service = SOAPScript.makeStub(namespace, "StockQuote");

var count = 0;

function onMessage (chat, message) {
    var params = message.split(" ");
    if (params[0] == "echo") {
        chat.sendMessage(params[1]);
    } else if (params[0] == "count") {
        count = count + 1;
        chat.sendMessage ("Count: " + count);
    } else if (params[0] == "quote") {
        var quote = service.StockQuotePort.getQuote(params[1]);
        chat.sendMessage(quote.name + ": $" + quote.price);
    }
}
```

Dynamic Java Objects in Scripts

- When you want to code in Java without a fixed interface
- Different for every language:
  - Groovy:
    - GroovyObject interface
    - `Object invokeMethod (String name, Object args)`
  - JavaScript (Rhino):
    - Scriptable and Function interfaces
    - `Object call(Context cx, Scriptable scope, Scriptable thisObj, Object[] args)`

Agenda

Scripting back on rails

Integration patterns and use cases

What is Groovy, and why it matters?

Groovy syntax, APIs and beyond

SOA scripting

JSR-223, an API to rule them all

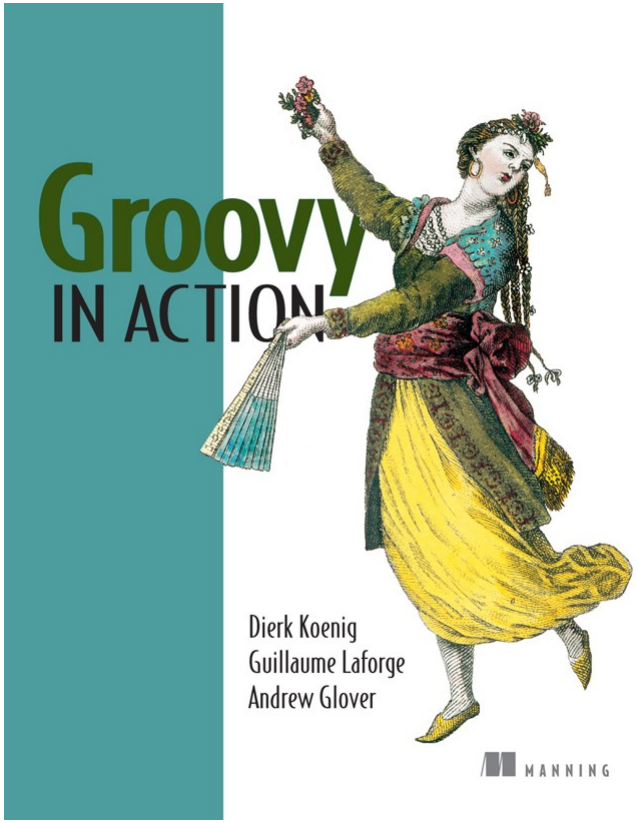
Conclusion

Summary

- Scripting simplifies development in the enterprise
- Scripting languages, such as **Groovy**, leverage the richness of Java technology
- Scripts can be used in various contexts
  - From the client using AJAX to the server
- The Java platform is embracing scripting with various standards

For More Information to Get the Groove…

- Attend Graeme Rocher's **Grails session**
- Web resources:
  - http://groovy.codehaus.org
  - http://grails.codehaus.org
- Read the upcoming **"Groovy in Action"** book from **Manning**, written by Dierk Koenig, Andrew Glover, and Guillaume Laforge



Q&A

With Guillaume, Tugdual, and Tim

Simplify Enterprise Development With Scripting

Guillaume Laforge

Software Architect
OCTO Technology
http://www.octo.com

Tim Gleason/Tugdual Grall

OracleAS Development
Oracle Corporation
http://www.oracle.com

TS-1246

Grails—Groovy on Rails

Simplifying Web Application Development with Grails

Learn how **scripting** can allow you to dramatically speed up the **development of Web applications** and let you concentrate on the core parts of your project

Grails—Groovy on Rails

- New MVC Web Framework:
  - "**Convention over Configuration**"
- Reuses and innovates around standard bricks
  - **Spring**: Spring MVC, Spring WebFlow
  - **Hibernate**
  - **SiteMesh**
- Spiced up with some Groovy
  - GSP (Groovy Server Pages) and custom dynamic tags
  - Controllers and actions, Services
- But **no XML configuration**!

Grails Domain Classes

- Persistent domain beans

```
class Book {
    @Property Long id
    @Property Long version
    @Property String title
    @Property String author
}
```

No dependency

- Dynamic methods, thanks to the MOP

```
new Book(author: "x", title: "y").save()
Book.findByAuthor("Rowling").each{…}
```

- With Grails, it's possible to
  - Reuse existing Hibernate mappings
  - Specify validation constraints with a DSL
  - Define associations (1-1, 1-n, n-m)

Grails Services

- Create and injecting dependent services

```
class BookService {

    @Property MyService myService
    @Property boolean transactional = true

    def allBooks() {
        Book.findAll()
    }
}
```

No dependency

Automatic wiring and injection of services (No XML)

Transactional demarcation for atomic method calls

Manipulate your domain classes

- Possible to
  - Reuse existing Spring configuration
  - Services implemented in Java technology

Grails Controllers

- Create controllers and actions

```
class BookController {

    @Property BookService service

    @Property list = {
        ["books": service.allBooks()]
    }
}
```

Automatic wiring and injection of services

Pass the books to the view

- Injection of services
- Also integration of Spring WebFlow
- **Dynamic** CRUD **"scaffolding"** for views and controllers

Grails Views

- And now a GSP for the view

```
<html>
  <head><title>Book list</title></head>
  <body>
    <h1>Book list</h1>
    <table>
      <tr><th>Title</th><th>Author</th> </tr>
      <% books.each { %>
      <tr>
        <td>${it.title}</td>
        <td>${it.author}</td>
      </tr>
      <% } %>
    </table>
  </body>
</html>
```

- Possibility to use JavaServer Pages™ technology as well
- Grails dynamic tag library (AJAX, links, templates, etc…)