

Judd Solutions

Development, Mentoring and Training



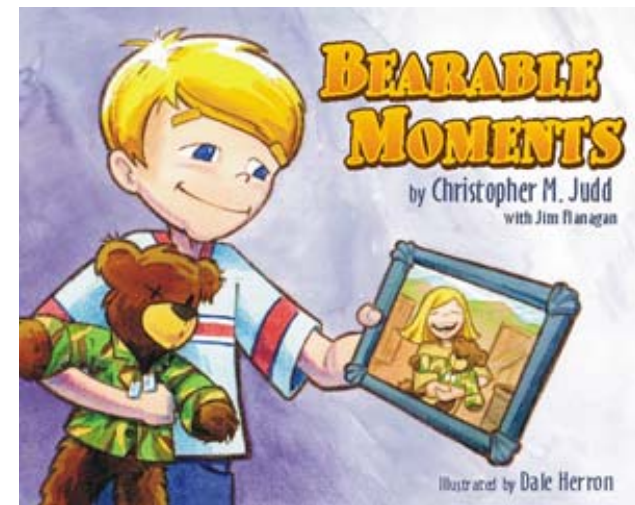
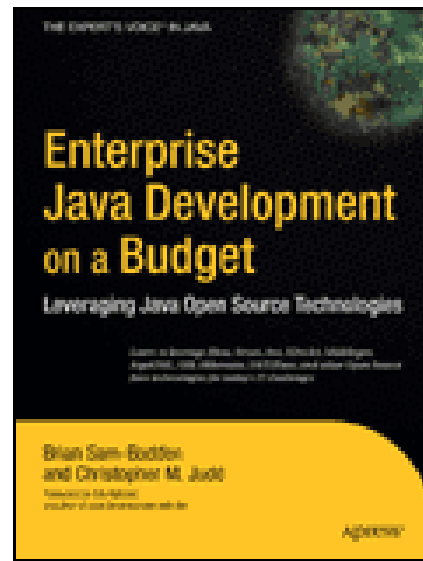
Scripting for the Java Platform

Christopher M. Judd
President/Consultant
Judd Solutions, LLC



Christopher M. Judd

- President/Consultant of Judd Solutions
- Central Ohio Java User Group (COJUG) coordinator





Agenda

- Java Scripting Overview
- Examples
 - API
 - Script Shell
 - Java EE Debugging
- Alternative Java Scripting Engines
 - Configuring
 - Creating
- Closing Thoughts
- Resources
- Q&A



**Java is the greatest
language ever invented**





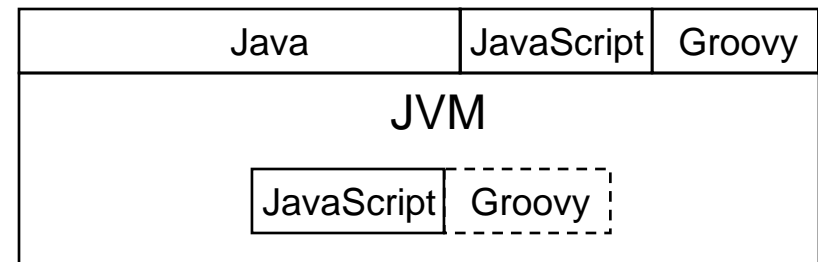
Developer's tools

Every developer's toolbox should contain a static typed language like Java or C# and a dynamically typed scripting language like JavaScript, Ruby or Groovy.



Java Scripting

- Java Scripting support added in Java SE 6
- JSR 223: Scripting for the Java Platform
- Java Virtual Machine
 - Executes “language-neutral” bytecode
 - Rich class library
 - Multi-platform
- Features
 - API to evaluate scripts
 - Embedded JavaScript engine (Rhino 1.6R2)
 - Scripting engine discovery mechanism
 - Java Scripting command-line interpreter (jrunscript)





Reasons for Scripting

- Flexibility
- Simplicity (Domain Specific Language)
- Interpreted
- Development productivity
- Dynamic typing
- Expressive syntax
- FUN



Scripting Uses

- Configuration
- Customization
- Automation
- Debugging
- Templating
- Unit Testing
- Prototyping
- Web Scripting
- Data transport



Scripting Options

- JavaScript
 - Rhino – www.mozilla.org/rhino/
- Groovy (JSR-241)
 - groovy.codehaus.org
- Python
 - Jython – www.jython.org
- Ruby
 - JRuby – jruby.codehaus.org
- TCL
 - Jacl – tcljava.sourceforge.net
- Java (JSR-274)
 - BeanShell www.beanshell.org
- 189 Others
 - <http://flp.cs.tu-berlin.de/~tolk/vmlanguages.html>



Scripting Options

- Java 6
- JSR 223 Reference Implementation
- Apache Bean Scripting Framework (BSF)
- Native Scripting Library



JSR 223 Reference Implementation

- Same scripting API as Java 6
- Works in Java 5
- Download from <http://jcp.org/aboutJava/communityprocess/final/jsr223/>
- Install
 - Unzip sjp-1_0-fr-ri.zip
 - Add js.jar, script-api.jar and script-js.jar to classpath



Apache Bean Scripting Framework (BSF)

- Framework for incorporating scripting into Java applications
- JDK 1.5 and prior
- Current version is 2.4.0
- Originally developed by IBM
- jakarta.apache.org/bsf/
- Supported Languages
 - JavaScript (Rino)
 - NetRexx
 - Python (Jython)
 - Tcl (Jacl)
 - XSLT Stylesheets
- Available Languages
 - Groovy
 - Java (BeanShell)
 - JLog (PROLOG)
 - Ruby (JRuby)



Native Scripting Library

- Every scripting language framework has its own API
- See desired scripting language for details



Scripting Engines features beyond Java 6

- Compile to byte code
- Running Scripts
 - Shell
 - Command-line
 - Console



Agenda

- Java Scripting Overview
- Examples
 - API
 - Script Shell
 - Java EE Debugging
- Alternative Java Scripting Engines
 - Configuring
 - Creating
- Closing Thoughts
- Resources
- Q&A



Scripting API

- Simple API
- javax.script package

Interfaces	Description
ScriptEngine	Scripting framework wrapper.
ScriptEngineFactory	Factory for describing and instantiating ScriptEngine instances.
Bindings	A mapping of key/value pairs for passing references to the script.
Compilable	ScriptEngine who has methods to compile scripts for repeated execution.
Invocable	ScriptEngine who allows script procedures to be invoked.
ScriptContext	Script execution context.

Classes	Description
ScriptEngineManager	Discovery and instantiation mechanism for ScriptEngines.
AbstractScriptEngine	Standard Super class for ScriptEngine implementations.
CompiledScript	Result of a compiled script.
SimpleBindings	Simple implementation of the Bindings interface.
SimpleScriptContext	Simple implementation of the ScriptContext interface.
ScriptException	Generic scripting exception.



Evaluating a Script

- Simple Hello World example

```
public static void main(String[] args) {  
    ScriptEngineManager mgr = new ScriptEngineManager();  
    ScriptEngine engine = mgr.getEngineByName("JavaScript");  
    try {  
        engine.eval("print('Hello, world!')");  
    } catch (ScriptException ex) {  
        ex.printStackTrace();  
    }  
}
```

Console Output

```
Hello, world!
```



ScriptEngineManager

- Uses the service provider mechanism to discover scripting engines
 - <http://java.sun.com/javase/6/docs/technotes/guides/jar/jar.html#Service%20Provider>
- Sets Globally scoped values
- Registers engines programmatically



Returning Values

- Return value is the last value of the script

```
ScriptEngineManager mgr = new ScriptEngineManager();  
ScriptEngine engine = mgr.getEngineByName("JavaScript");  
Object result = engine.eval("5+4");  
System.out.println("Result = " + result);  
System.out.println("Result class = " + result.getClass().getName());
```

Console Output

```
Result = 9.0  
Result class = java.lang.Double
```



Passing Values

- Bindings or ScriptEngineManager can be used to pass variables to the script
- Keys become the variable name

```
ScriptEngineManager mgr = new ScriptEngineManager();  
mgr.put("op1", 4);  
  
ScriptEngine engine = mgr.getEngineByName("JavaScript");  
  
Bindings bindings = engine.createBindings();  
bindings.put("op2", 5);  
  
Object result = engine.eval("op1 + op2", bindings);  
System.out.println("Result = " + result);
```

Console Output

```
Result = 9.0
```



Evaluating a File

- ScriptEngine contains an overloaded eval methods that takes a java.io.Reader

```
String script = args[0];
String extension = script.substring(script.lastIndexOf('.') + 1, script.length());

InputStream is = ScriptEngineScripts.class.getResourceAsStream("/" + script);
Reader reader = new InputStreamReader(is);

ScriptEngineManager mgr = new ScriptEngineManager();
ScriptEngine engine = mgr.getEngineByExtension(extension);

engine.eval(reader);
```

Demo

Judd Solutions



Using Java Classes in Scripting Language

- Import the package in language specific way
 - JavaScript – `importPackage(java.sql)`
 - Groovy – `import java.sql`

```
importPackage(java.sql);

var conn = DriverManager.getConnection("jdbc:derby:sample;create=true");
var stmt = conn.createStatement();
var rs = stmt.executeQuery("select * from SYS.SYSTABLES");

println("System Tables:\n");

while(rs.next()) {
    println(rs.getString("tablename"));
}
```



Compiling Scripts

- Compiling scripts can increase performance
- Script Engine must support it by supporting the Compilable interface

```
ScriptEngineManager manager = new ScriptEngineManager();
ScriptEngine engine = manager.getEngineByName("js");
Compilable compilable = (Compilable) engine;

CompiledScript script = compilable.compile("op1 + op2");

Bindings bindings = engine.createBindings();
bindings.put("op1", 26);
bindings.put("op2", 100);
Object result = script.eval(bindings);
```



Invoking Script Methods

- Individual methods in a scripting language can be invoked besides an entire script.
- Script Engine must support it by supporting the Invocable interface

```
ScriptEngineManager manager = new ScriptEngineManager();  
ScriptEngine engine = manager.getEngineByName("js");  
engine.eval("function add(opt1, opt2) {return opt1 + opt2;}");  
Invocable invocable = (Invocable) engine;  
Object result = invocable.invokeFunction("add", new Object[] { 10, 15 });  
System.out.println("Result = " + result);
```



Script Shell

- Java 6 includes an experimental script shell
- `jrscript`
(<http://java.sun.com/javase/6/docs/technotes/tools/share/jrscript.html>)
 - Interactive
 - Batch
- JavaScript includes built-in functions
(<http://java.sun.com/javase/6/docs/technotes/tools/share/jsdocs/index.html>)
 - `which(cmd)`
 - `date()`
 - `cat(object, pattern)`
 - `grep(pattern,files)`
 - Etc.
- Executes scripting languages other than JavaScript with the `-l` argument



Executing jrunscript

Evaluate

```
C:\windows\System32\cmd.exe
F:\java\jdk1.6.0\bin>jrunscript -e "which('mvn')"
F:\java\maven-2.0.3\bin\mvn
```

Interactive

```
C:\windows\System32\cmd.exe
F:\java\jdk1.6.0\bin>jrunscript
js> date()
January 14, 2007 3:02:12 PM GMT-05:00
js> importPackage(java.util)
js> var now = new Date()
js> echo(now)
Sun Jan 14 2007 15:02:38 GMT-0500 (GMT-05:00)
js> echo(date)

function date() {
    println(new Date().toLocaleString());
}

js> exit()
F:\java\jdk1.6.0\bin>
```

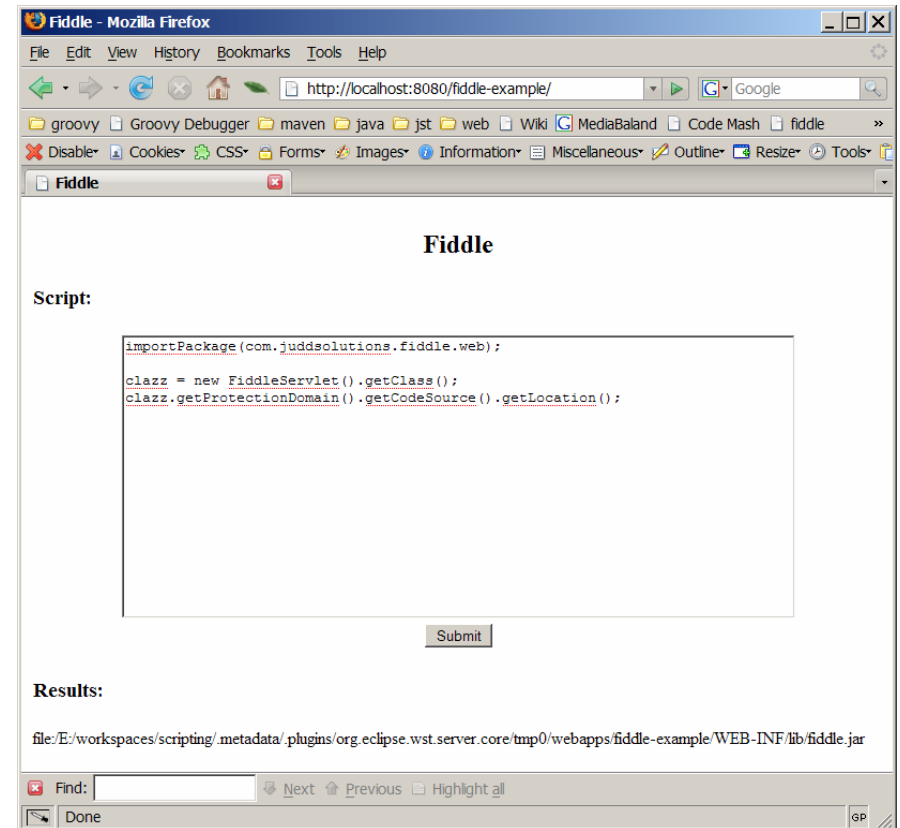
Batch

```
C:\windows\System32\cmd.exe
F:\java\jdk1.6.0\bin>jrunscript -f simple.js
F:\java\maven-2.0.3\bin\mvn
January 14, 2007 2:42:58 PM GMT-05:00
```



Java EE Debugging

- Fiddle Project
 - Servlet used for executing Java Scripts using the Java Scripting API for the purpose of debugging and fiddling around with web and Java EE based applications in-container.
 - Open Source
 - <http://fiddle.dev.java.net/>
- Uses
 - Display session data
 - Validate JDBC connections
 - Access EJBs in-container
 - Validate JNDI settings
 - Verify Classpath issues



Demo

Judd Solutions



Agenda

- Java Scripting Overview
- Examples
 - API
 - Script Shell
 - Java EE Debugging
- Alternative Java Scripting Engines
 - Configuring
 - Creating
- Closing Thoughts
- Resources
- Q&A



Alternative Java Scripting Engines

- JDK 6 Scripting Engine Implementations can be found at <http://scripting.dev.java.net/> for many common scripting languages
 - BeanShell
 - FreeMarker
 - Groovy
 - Java
 - JavaScript (newer versions of Rhino)
 - Python
 - Ruby
 - Scheme
 - Tcl
 - Velocity
- Also has links to other Scripting Engines
 - AppleScript
 - PHP



Configuring Alternative Java Scripting Engines

1. Download and uncompress jsr223-engines.tar.gz or jsr223-engines.zip
2. Add appropriate engine jar in classpath
 - `<jsr223-engines>/<script-framework>/build/<script-framework>-engine.jar`
 - `<jsr223-engines>/groovy/build/groovy-engine.jar`
3. Add appropriate scripting framework jars and dependant jars to classpath

Demo

Judd Solutions



Steps for Creating a Java Scripting Engine

1. Implement the ScriptEngine (extend AbstractScriptEngine) and ScriptEngineFactory interfaces
2. Optionally implement a Compilable and/or Invocable interface
3. Add a javax.script.ScriptEngineFactory file to the META-INF/services directory containing the name of your ScriptEngineFactory implementation



Agenda

- Java Scripting Overview
- Examples
 - API
 - Script Shell
 - Java EE Debugging
- Alternative Java Scripting Engines
 - Configuring
 - Creating
- Closing Thoughts
- Resources
- Q&A



Scripting Performance

- Simple performance benchmark
 - Interpreted JavaScript: 1,550ms
 - Compiled JavaScript: 579ms
 - Compiled Java: 0.0172ms
- Executed 10,000 times

```
public double calculateRiskFactor(int age, boolean noClaims) {  
    double riskFactor;  
    if (age < 25) {  
        riskFactor = 1.5;  
    } else if (noClaims) {  
        riskFactor = 0.75;  
    } else {  
        riskFactor = 1.0;  
    }  
    return riskFactor;  
}
```

Borrowed from John Ferguson Smart's The Mustang Meets the Rhino: Scripting in Java 6 article
<http://www.onjava.com/pub/a/onjava/2006/04/26/mustang-meets-rhino-java-se-6-scripting.html?page=3>



Evidence Sun is opening to other languages

- Adding Scripting to the API
- Providing JavaScript implementation
- Scripting JSRs
 - JSR 223 – Scripting for the Java Platform
 - JSR 241 – Groovy Programming Language
 - JSR 274 – BeanShell Scripting Language
 - JSR-292 - Dynamically Typed Language Support
- Hiring the primary JRuby developer



Future

- Additional scripting languages
- Dynamically Typed Language Support
 - JSR-292
 - <http://www.jcp.org/en/jsr/detail?id=292>
- Frameworks utilizing scripting
 - Phobos
 - lightweight, scripting-friendly, web application environment
 - <https://phobos.dev.java.net/>
 - Fiddle
 - In-container debugging
 - <https://fiddle.dev.java.net/>



Agenda

- Java Scripting Overview
- Examples
 - API
 - Script Shell
 - Java EE Debugging
- Alternative Java Scripting Engines
 - Configuring
 - Creating
- Closing Thoughts
- Resources
- Q&A



Resources

- Web sites
 - JSR-223 – Scripting for the Java Platform
 - <http://www.jcp.org/en/jsr/detail?id=223>
 - Rhino: JavaScript
 - <http://www.mozilla.org/rhino/>
 - Scripting java.net
 - <https://scripting.dev.java.net/>
 - JavaScript Developer Connection
 - <http://java.sun.com/javascript/>
- Java Scripting Documentation
 - JavaDoc
 - <http://java.sun.com/javase/6/docs/api/javax/script/package-summary.html>
 - Java Scripting Programmer's Guide
 - http://java.sun.com/javase/6/docs/technotes/guides/scripting/programmer_guide/



Resources

- Articles
 - Scripting for the Java Platform
 - <http://java.sun.com/developer/technicalArticles/J2SE/Desktop/scripting/>
 - Mustang Meets the Rhino: Scripting in Java 6
 - <http://www.onjava.com/pub/a/onjava/2006/04/26/mustang-meets-rhino-java-se-6-scripting.html>
- JavaScript Documentation
 - ECMAScript Language Specification
 - <http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf>
 - Mozilla JavaScript Documentation
 - <http://developer.mozilla.org/en/docs/JavaScript>
 - jrunscript JavaScript built-ins functions
 - <http://java.sun.com/javase/6/docs/technotes/tools/share/jsdocs/index.html>



Contact Information

- <http://www.juddsolutions.com>
- cjudd@juddsolutions.com
- Blog
 - <http://blogs.apress.com/authors.php?author=Christopher%20Judd>
- Pro Eclipse JST
 - <http://www.projst.com>
 - <http://www.apress.com/book/bookDisplay.html?bID=447>
- Enterprise Java Development on a Budget
 - <http://www.apress.com/book/bookDisplay.html?bID=197>





Thank you!

Questions ?