Java Scripting: One VM, Many Languages

Sang Shin
sang.shin@sun.com
javapassion.com
Sun Microsystems, Inc.

UNLOCK
OPPORTUNITY
What will you open?

SUN TECH DAYS 2006-2007
A Worldwide Developer Conference

Agenda

- Quick overview
- Scripting API
- Java SE 6 Scripting Support
- Demo
- Future Directions
- Resources

Quick Overview

Scripting Languages

- Typically dynamically typed languages
  - No need to define variables before you use them
  - Many type conversions happen automagically
  - Can be good...
  - Can be bad...
- Most scripting languages are interpreted
  - Perform the script compilation and execution within the same process
- Very good for fast results for small jobs
  - Write application faster, execute commands repeatedly

Different Languages, different jobs

- Perl
  - Text processing, report generation
- Bash, sh, ksh
  - job control
- Ruby
  - Web based applications

Java Programming Language and Ruby Compared

```
public class Filter {
  public static void main(String[] args) {
    List list = new java.util.ArrayList();
    list.add("Tim"); list.add("Ike"); list.add("Tina");
    Filter filter = new Filter();
    for (String item : filter.filterLongerThan(list, 3)) {
      System.out.println( item );
    }
  }
  public List filterLongerThan(List list, int length) {
    List result = new ArrayList();
    for (String item : list) {
      if (item.length() >= length) { result.add( item ); }
    }
    return result;
  }
}
```

Java Programming Language and Ruby Compared

Ruby!

```
list = [‘Tim’, ‘Ike’, ‘Tina’]
list.select {|n| n.length > 3}.each {|n| puts n}

=> ‘Tina’
```

Scripting Over Java Platform

Why Scripting Languages & Java together?

- Combining scripting languages with the Java platform provides developers and end-users an opportunity to leverage the abilities of both environments
- Use scripting languages for quick and easy development & testing for certain parts of your applications
- Use Java programming language and platform for what it is known for
  - Scalable and highly performing business logics

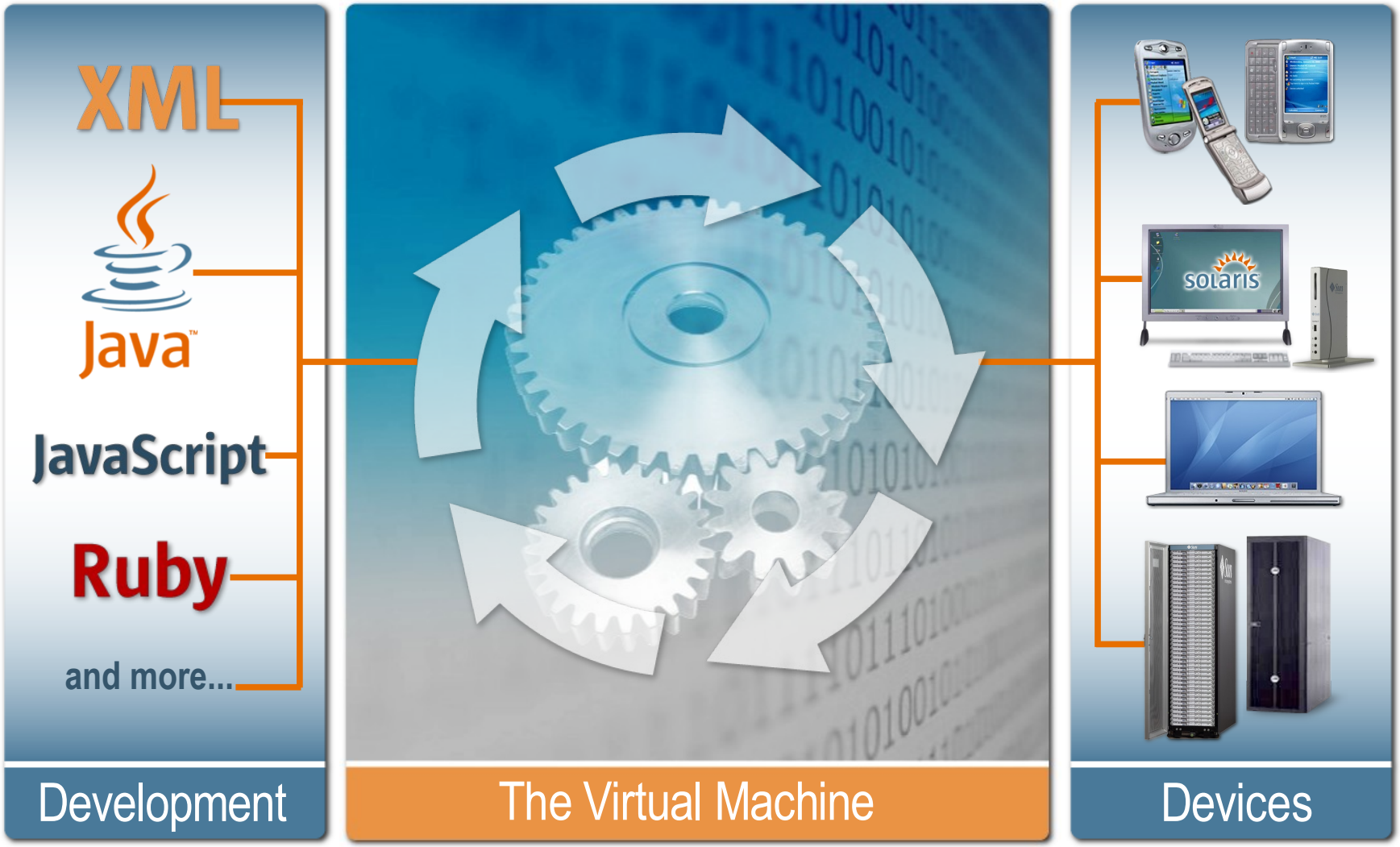
Why Scripting Languages & Java together?

- Allows end-users to customize the applications further

Java Platform Supports Scripting Languages Well!

- Java Language != Java Platform
  - Java VM runs “language-neutral” bytecode
  - Rich set of Class libraries are “language-neutral”
  - “Write once run anywhere” applies to Platform
  - Leverage programmer skills and advantages of particular languages
- Time-tested technologies
  - Open-source projects for various languages
  - Jakarta BSF

The Virtual Machine



And Announced Recently

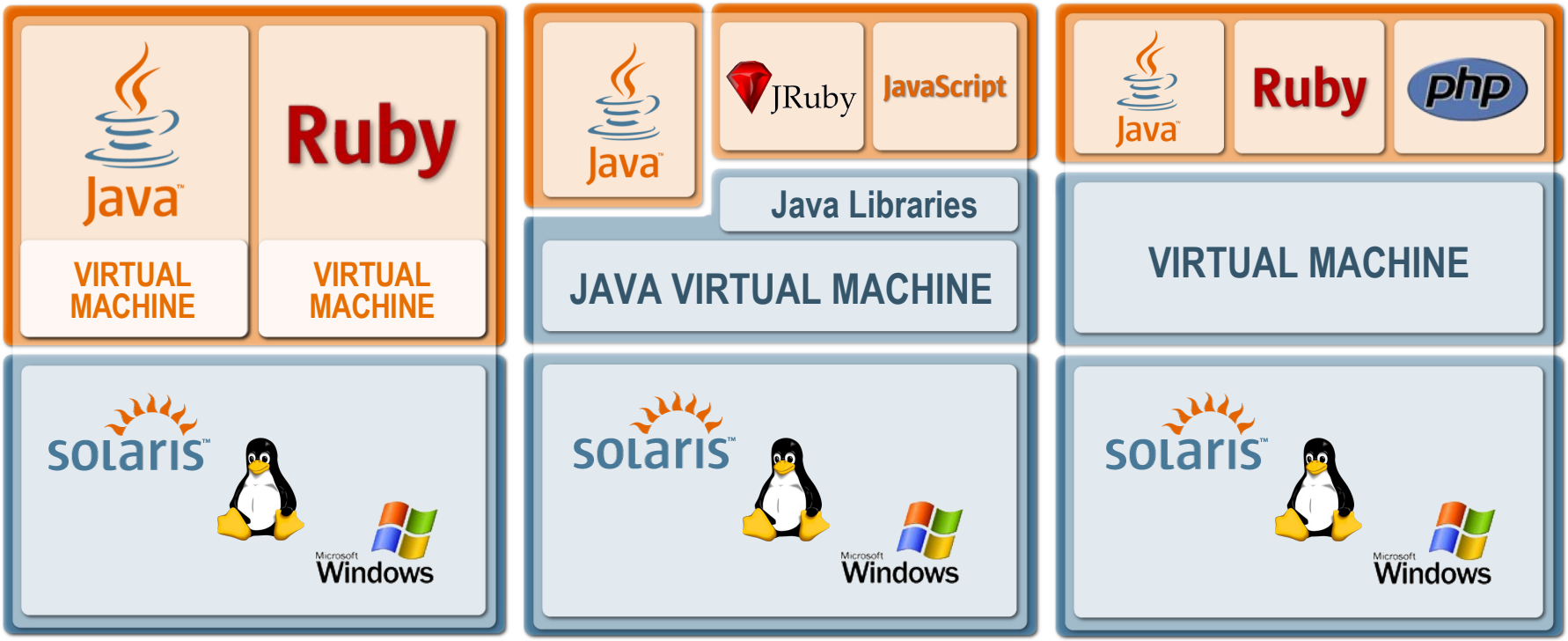
NetBeans

- Ruby Support from Sun
  - JRuby @ Sun
  - Building full Ruby and Rails Support right in the Virtual Machine
  - A new team
- NetBeans Tools
  - Ruby and Rails
  - JavaScript Support

Client Scripting Scenarios

- Class files written in other languages
  - Groovy
  - Jython Compiler
  - Kawa Scheme
  - JRuby
- Java applications execute script programs
  - Stand-alone interpreter
  - Macro interpreters
  - Web Scripting
- In both cases, programs use Java Objects/Libraries

Scripting Scenarios



Native Scripting

The Community does Both...
(port and run)

Java Virtual Machine

Living the Java Lifestyle...

Web

Leverage the VM
(multiple languages)

= You do = We do

Scripting Framework & API over Java Platform

Scripting framework

- JSR 223 defines the scripting framework
- It supports pluggable framework for third-party script engines
  - Resembles BSF ActiveX Scripting
  - "Java application runs script programs" scenario
- *javax.script* package
- Optional *javax.script.http* package for Web scripting
- Part of Java SE 6
- Available for Java 5.0

Scripting API

- *ScriptEngine*
- *ScriptContext, Bindings*
- *ScriptEngineFactory*
- *ScriptEngineManager*

Interfaces

- *ScriptEngine* interface—required
  - Execute scripts—“eval” methods
  - Map Java objects to script variables (“put” method)
- *Invocable* interface—optional
  - Invoke script functions/methods
  - Implement Java interface using script functions/methods
- *Compilable* interface—optional
  - Compile Script to intermediate form
  - Execute multiple times without recompilation

ScriptEngine API

- **ScriptEngine** (Interface)
  - **eval()**
  - **put()**
  - **get()**
  - **getBindings()/setBindings()**
  - **createBindings()**
  - **getContext()/setContext()**
  - **getFactory()**
- **AbstractScriptEngine**
  - Standard implementation of several **eval()** methods

ScriptEngineManager

- Provides the ScriptEngine discovery mechanism
  - `getEngineByName()`
  - `getEngineByExtension()`
  - `getEngineByMimeType()`
  - `getEngineFactories()`
- Developers can add script engines to a JRE
  - with the JAR Service Provider specification

Example – Hello world

```
import javax.script.*;
public class Main {
    public static void main(String[] args) throws ScriptException {
        // Create a script engine manager
        ScriptEngineManager factory = new ScriptEngineManager();

        // Create JavaScript engine
        ScriptEngine engine = factory.getEngineByName("JavaScript");

        // Add a script variable whose value is a Java Object
        engine.put(“greeting”, new Exception(“Hello World!”));

        // Evaluate JavaScript code from String
        engine.eval("print(greeting.toString())");
    }
}
```

Example - “eval” script file

```
// Create script engine manager
ScriptEngineManager manager = new ScriptEngineManager();

// Create JavaScript engine
ScriptEngine engine = manager.getEngineByExtension(“js”);

// Evaluate a file (or any java.io.Reader)
engine.eval(new FileReader(“test.js”));
```

Example – Invoking functions

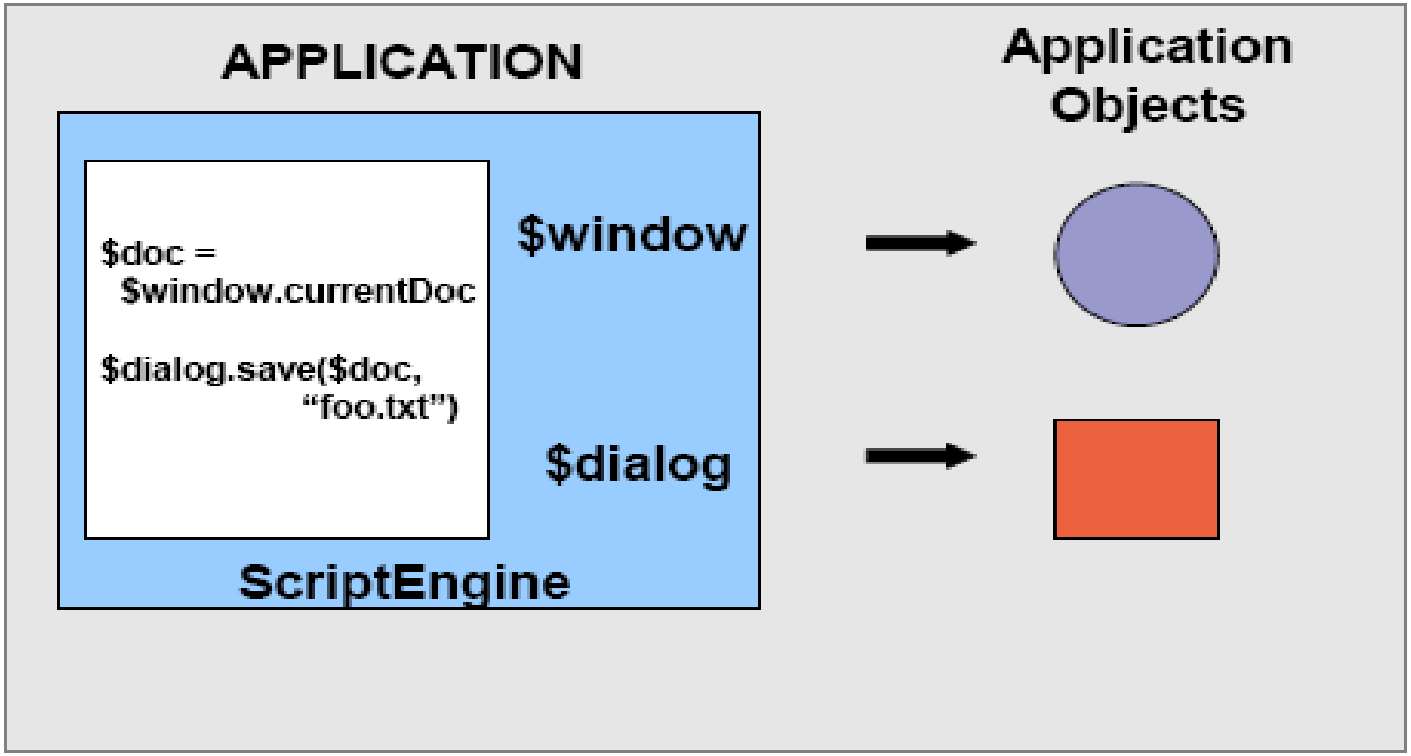
```
// JavaScript code in a String
String script = "function hello(name) { print('Hello, ' + name); }";

// Evaluate script
engine.eval(script);

// JavaScript engine implements Invocable interface
Invocable inv = (Invocable) engine;

// Invoke a global function called “hello”
inv.invoke("hello", new Object[] {"Scripting!!"} );
```

Mapping script variables to application objects



ScriptContext and Bindings (interface)

- ScriptContext—Script's view of host application
- ScriptContext contains one or more Bindings
- Bindings is subtype of Map<String, Object>
- Scope is a set of named attributes
- Engine Scope Bindings
  - Script variables → application objects
- Global Scope Bindings
  - Variables shared across engines
- Writers for stdout, stderr
- Reader for stdin

ScriptContext and Bindings (cont.)

- Exposes readers/writers for script engines to use for input and output
  - `setBindings()/getBindings()`
  - `setAttributes()/getAttribute()`
  - `setWriter()/getWriter()`
  - `setReader()/getReader()`
- `SimpleScriptContext`

Example – Script variables

```
// Create script engine manager
ScriptEngineManager manager = new ScriptEngineManager();

// Create JavaScript engine
ScriptEngine engine = manager.getEngineByName("JavaScript");
File f = new File("test.txt");

// Expose File object as variable to script
engine.put("file", f);

// Evaluate a script string wherein the "file" variable is accessed, and a
// method is called upon it
engine.eval("print(file.getAbsolutePath())");
```

ScriptEngineFactory (interface)

- Describe and instantiate script engines
  - 1-1 with ScriptEngines
- Factory method—getScriptEngine
- Metadata methods
  - Script file extensions, mimetypes
  - Implementation-specific behavior (threading)
- Script generation methods
  - Generate method call
  - Generate “print” call

ScriptEngineFactory (cont.)

- Each script engine has a ScriptEngineFactory
  - `getEngineName()`
  - `getEngineVersion()`
  - `getExtensions()`
  - `getMimeTypes()`
  - `getLanguageName()`
  - `getProgram()`
  - `getScriptEngine()`

Other Scripting Classes

- **`CompiledScript`**
  - Compiled version of script
  - No requirement for reparsing
  - Associated with a script engine
- **`ScriptException`**
  - All checked exceptions must be wrapped in this type
  - Records line number, column number, filename
- **`Bindings`/`SimpleBindings`**
  - Mapping of key/value pairs, all strings

Java SE 6 Scripting Support

Javascript Engine

- Based on Mozilla Rhino 1.6v2
- Features omitted for security/footprint reasons
  - Optimizer (script-to-bytecode compiler – only interpreter support)
  - E4X (XML language support) – depends on xmlbeans.jar
  - Rhino command line tools (shell, debugger etc.)
- Security Tweaks

Scripting Tools / Samples

- Tools
  - <JDK>/bin directory
  - jrunscript
    - Interactive command-line interpreter.
  - jhat
    - Processes heap analysis tool output
  - jconsole scripting plugin
- Samples
  - Script notepad
    - Swing application mostly implemented in Javascript
    - Fancy Javascript programming.

Demo

Programmable Calculator

- From “Scripting for the Java Platform” by John O'Connor
  http://java.sun.com/developer/technicalArticles/J2SE/Desktop/scripting/
- 100% Java Swing application
- Customizable using end-users' scripts
- Uses Java SE Javascript engine
- Enhanced to use any JSR 223 Engine

Demo: Scripting over Java SE

- Build and run ScriptPad sample app from JDK 6 samples
  - You can build and run as NetBeans project
- Executing JavaScript code
- Invoking Java methods from JavaScript code

Scripting on the Server side

Scripting in Java EE

- Web-tier is a natural place for scripting
  - tends to have high rate of change
- JSP is already very script-like
  - allow mixing of Java language and tags on HTML page
- Project Phobos supports JavaScript
  - as server-side web page scripting language
  - as lightweight way of implementing servlets
  - see phobos.dev.java.net

Sample JRuby Script

```
$response.setStatus(200)

$response.setContentType("text/html")

writer = $response.getWriter()

writer.println("<html><head><title>Hello</title></hea
  d><body>Hello from JRuby!</body></html>")

writer.flush()
```

Application Layout

```
/application
    /controller
        test.js
    /module
        application.js
    /script
        index.js
        hello.rb
    /template
    /view
        layout.ejs
        test.ejs
```

```
/static
    /dojo
        dojo.js
    /css
        main.css
    faq.html
    release_notes.html
```

```
/environment
    development.js
    startup-glassfish.js
```

Future Direction

Language JSRs

- invokedynamic Bytecode – JSR 292
  - http://www.jcp.org/en/jsr/detail?id=292
  - Used for better compilation of dynamically-typed scripts
- Groovy – JSR 241
  - http://groovy.codehaus.org/
- BeanShell – JSR 272
  - http://www.beanshell.org

JSR 292 – invokedynamic bytecode

- To enable the compilation of dynamically typed languages such as Groovy, Jruby, Jython to JVM bytecodes, a new bytecode called **invokedynamic** is being proposed as part of JSR 292
- The invokedynamic will not require target class name, and the method signature.
- It will search the specified method on the target object based on the method name
  - JSR will specify how to handle method overloading in such scenario
  - JSR will specify how to handle failures

JSR 292 – invokedynamic bytecode

- There are 4 JVM bytecodes to call methods:
  - invokeinterface - used to call an interface method on an object
  - invokestatic - used to call a static method of a class
  - invokevirtual - used to call a overridable method
  - invokespecial - used to call
    - constructors
    - private instance methods
    - super class methods (super.foo() calls in the source)

JSR 292 – invokedynamic bytecode

- All these instructions require the specification of
  - target class (or interface for invokeinterface) name
  - the name of the method (or <init> for constructors)
  - the signature of the method.

JSR 292 – invokedynamic bytecode

Impact on Groovy

- Groovy today supports a flexible method dispatching mechanism

```
class Main {
  public static void main(String[] args) {
    // see Person class below..
    Person p = new Person();
    System.out.println("Starting...");

    // call methods that are defined in Person class
    p.work();
    p.greet();

    // call methods that are not defined in Person
    // or it's superclass
    p.surfTheNet();
    p.writeBlog();  }}
```

```
class Person {

  public void work() {
    System.out.println("Okay, I'll work tomorrow!");
  }

  public void greet() {
    System.out.println("Hello, World!");
  }

  public Object invokeMethod(String name,
  Object args) {
    System.out.println("Why are you calling " +
     name + "?");  }}
```

Server-side scripting – Phobos

- http://phobos.dev.java.net
- Borrows from Ruby on Rails
  - Speed of development
  - Well-organized application structure
- Access to enterprise Java
- Javascript libraries
- Support for other technologies
  - AJAX
  - RSS / Atom

Resources

Resources - scripting.dev.java.net

- BSD License
- Scripting Engines
  - jruby, groovy, beanshell, jacl, jaskell, java, jawk,jelly,jexl,jruby,javascript,jython,ognl,pnuts,scheme,sleep,xpath,xslt
- Applications
  - NetBeans Scripting module
- Also see coyote.dev.java.net
  - NetBeans Scripting IDE
  - Jython, groovy support

Resources - references

- JSR-223
  - http://jcp.org/en/jsr/detail?id=223
- A. Sundararajan's Blog
  - http://blogs.sun.com/sundararajan
- Roberto Chinnici's Blog (serverside scripting)
  - http://weblogs.java.net/blog/robc/
- JavaScript Developer Connection
  - http://java.sun.com/javascript

Java Scripting: One VM, Many Languages

Sang Shin
sang.shin@sun.com
javapassion.com
Sun Microsystems, Inc.

UNLOCK
OPPORTUNITY
What will you open?

SUN TECH DAYS 2006-2007
A Worldwide Developer Conference