



JAVAPOLIS

11 - 15 DECEMBER ■ ANTWERP ■ BELGIUM





Server Side Scripting: Project Phobos

Ludovic Champenois
Senior Staff Engineer
Sun Microsystems



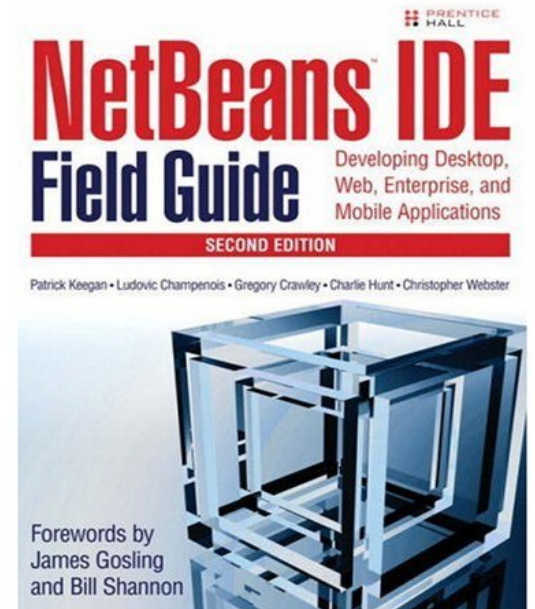
Overall Presentation Goal

Learn why JavaScript is relevant on the server side...as well as the client side

Learn how to use JavaScript on the server side

Speaker's Qualifications

- ☕ Senior architect at Sun Microsystems and...
 - ➔ Java EE 5, GlassFish, NetBeans, jMaki, Phobos
 - ➔ Tools developer
- ☕ Ludo wrote a Java EE book
 - ➔ NetBeans IDE field guide
- ☕ Ludo speaks French
- ☕ Ludo speaks Tools
 - ➔ GlassFish Eclipse:-) plugin



What Gartner thinks of JavaScript

”JavaScript on servers will emerge as one of several programming models popularized by Web platforms by 2009 (0.7 probability).”

(November, 21, 2006)

JavaScript

- ☕ Most visible technology for Web 2.0
- ☕ Around for almost 10 years
- ☕ In all relevant browsers
- ☕ Now in JDK 1.6...
- ☕ Server side JS in Netscape Enterprise Server (1998-)
- ☕ Emerging **affinity** between Client and Server
- ☕ Not Evil, especially on the server side



JavaScript Tools

Browsers centric

- ⇒ Venkman
- ⇒ Firebug (Written in JS, XUL UI)

IDEs

- ⇒ IntelliJ, Eclipse, NetBeans
- ⇒ Scripting languages support: Key feature for the brand new NetBeans editor framework
- ⇒ Inline errors, code completion, debugger
- ⇒ Embedded editors

Project Phobos

- ☕ Lightweight web application framework
- ☕ Runs on the Java platform
- ☕ Runs embedded (With Grizzly)
- ☕ Complementary to existing technologies
- ☕ Supports multiple scripting languages
- ☕ Current focus is on **JavaScript**
- ☕ Bottom-up framework development

Target Audience

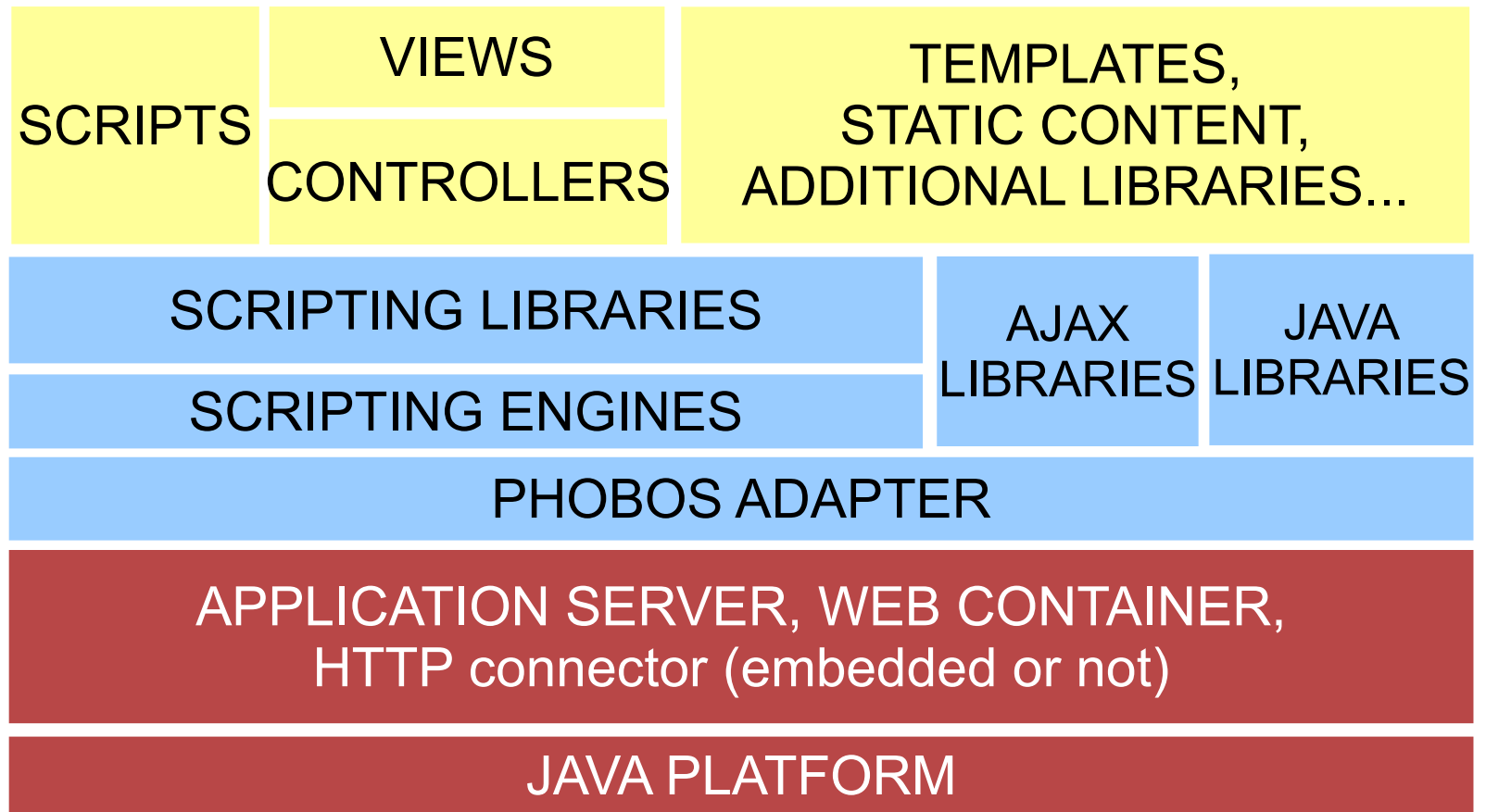
- ☕ Web application developers
- ☕ With a need for agility of scripting
- ☕ Feeling pain from compile/deploy cycle (especially true with Ajax)
- ☕ Glueing things together
- ☕ With investment in Java
- ☕ Libraries, EE servers/resources, Interop
- ☕ Seeking proven, high-performance deployment platform



Phobos Goal

- ☕ Deliver a great developer experience
- ☕ Tune your server logic while it is running:
 - ➔ Save the script
 - ➔ Reload the page that interacts with it

Architecture



Performance: grizzly+Javascript

jRuby: Apache bench:(<http://recompile.net/2006/11/>)

Server Software: WEBrick/1.3.1
Requests per second: **7.20** [#/sec] (mean)

Server Software: Grizzly/1.0.4
Requests per second: **13.45** [#/sec] (mean)

GlassFish V3 early prototype:
Starts Phobos engine in **0.8** seconds



Programming Model

- ☕ Less prescriptive than ROR
- ☕ Meaningful directory structure
- ☕ HTTP-centric
- ☕ Map out the URLs
- ☕ Attach logic to them
- ☕ Test out interactively
- ☕ REST modeled directly
- ☕ Background tasks, AJAX support, etc.
- ☕ Freely use Java APIs



Application Layout

`/application`

`/controller`

`mycontroller.js`

`/module`

`application.js`

`/script`

`index.js`

`hello.rb`

`/template`

`/view`

`layout.ejs`

`myview.ejs`

`/static`

`/dojo`

`dojo.js`

`/css`

`main.css`

`faq.html`

`release_notes.html`

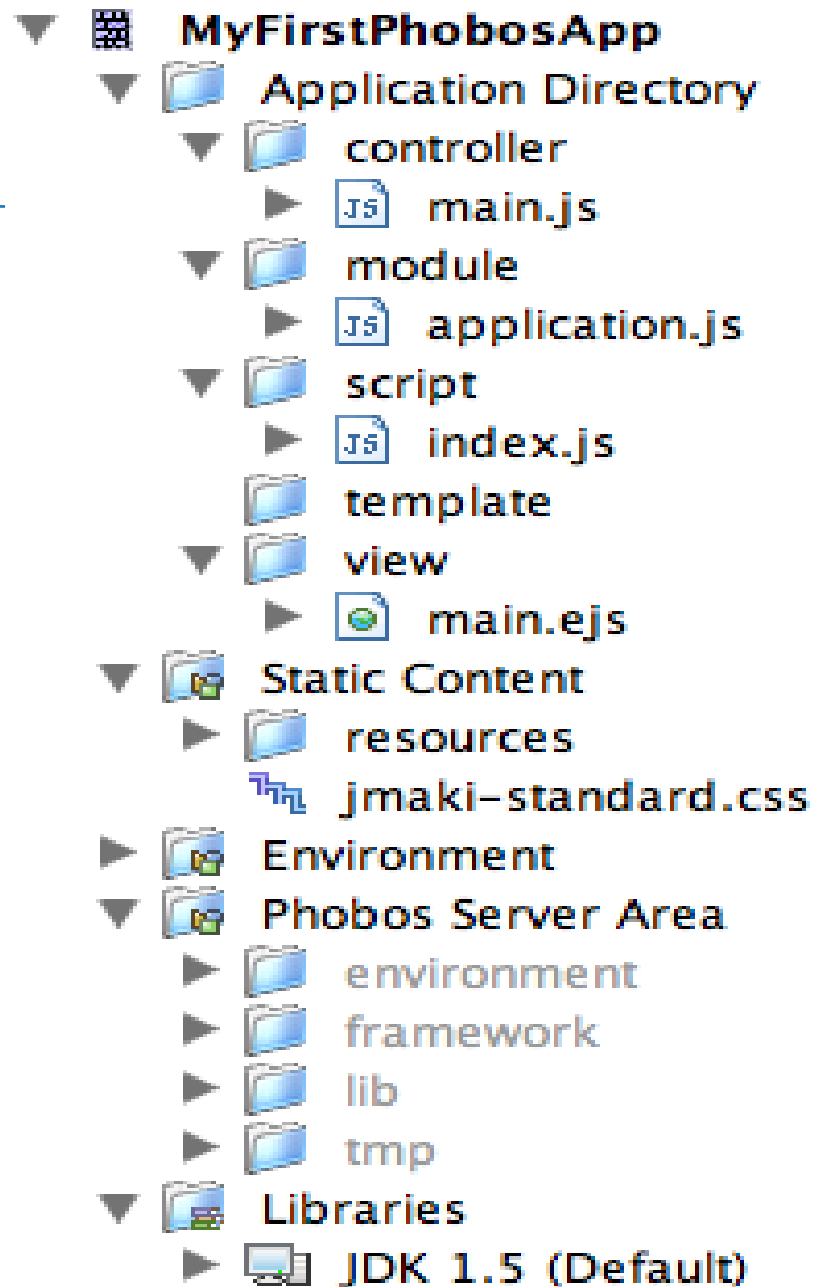
`/environment`

`development.js`

`startup-glassfish.js`



Application



Where does the logic go?

 Three choices:

- ➔ Plain scripts
- ➔ Controllers
- ➔ Resources

 Different URL “shapes”:

- ➔ `/doSomething.js`
- ➔ `/store/display_cart`
- ➔ `/catalog/isbn/1234-5678-90`

 These reflect how people think of URLs

 Custom mappings (choice #4)



#1 - Plain Scripts

- ☕ Servlet-like programming
- ☕ No need to extend servlet base class
- ☕ Request/response bound to variables

```
response.setStatus(200);  
response.setContentType("text/html");  
writer = response.getWriter();  
writer.println(<html><head><title>Hello from  
    Javascript</title></head><body>Hello from  
    Javascript!</body></html>);  
writer.flush();
```



#2 - Controllers

- ☕ MVC framework
- ☕ Controllers are JavaScript classes
- ☕ Action handlers are methods
- ☕ Views are scripts
- ☕ URL mapping: `/@controller/@action`
- ☕ User-specified mappings possible, e.g.
- ☕ `/photo/@id/@action`

Sample Controller

```
Var mycontroller = function() {  
  this.Main = function() {  
    this.frontpage = function () {  
      library.view.render("frontpage.ejs");  
    }  
  }  
};
```

```
library.common.define(controller, "main", mycontroller);
```

/@controller/@action → /main/frontpage



Embedded Javascript (.ejs)

- ☕ Templating system for views
- ☕ Embed Javascript statements:
 - `<% library.jmaki.insert({component: "sudoku"}) %>`
- ☕ Or expressions:
 - `<%= java.lang.System.currentTimeMillis() %>`
- ☕ No mandated way to share state between controller and views
 - ➔ Conventionally, use *model* global variable


#3 - Resources

- ☕ REST framework
- ☕ Resources are classes
- ☕ Methods are HTTP methods
- ☕ Code deals with HTTP entities
- ☕ Many HTTP aspects offloaded to framework
- ☕ Atom Protocol server as testbed
- ☕ Plan to investigate WADL

Ajax on the client



All jMaki components built in

```
 <% library.jmaki.insert({component: "dojo.richtext",  
id: "text1",  
args: { ... }}); %>
```

Jmaki CSS Layouts

Dojo Toolkit bundled

Multiple versions of a library OK

JSON natural data format

jMaki XmlHttpRequest support



Other Libraries

- ☕ In general: all that Java EE supports
- ☕ Templating using FreeMarker
- ☕ Persistence using JPA
- ☕ REST clients
- ☕ JSF/facelets as views
- ☕ Logging, Tango, etc.
- ☕ JPA CRUD generator
- ☕ Intriguing:-) : Dojo on the Server side

JPA sample

```
var em = library.persistence.  
    getEntityManager("jpaExample1-pu");  
  
// Insert some Authors if none are defined  
var authors = em.createQuery("select a from Author a").  
    getResultList().toArray();  
if (authors.length==0){  
    var tx = em.getTransaction();  
    tx.begin();  
  
    var author = new Packages.jpalexample.Author();  
    //equivalent to author.setName("Danny Goodman");  
    author.name = "Danny Goodman";  
    author.organisation= "O'Reilly";  
    em.persist(author);  
}
```

Script engine: Rhino (Mozilla)

- ☕ Robust, fast JavaScript implementation
- ☕ Bytecode compilation
- ☕ Built-in debugging support
- ☕ Great interface to Java code
- ☕ Many language extensions
- ☕ Interface via JSR-223



JavaScript Extensions in Rhino

- ☕ Much more powerful language
- ☕ JavaAdapter
- ☕ JSAdapter
 - ➔ on-the-fly properties
- ☕ Assignable `__proto__` property
 - ➔ dynamic inheritance
- ☕ Continuations
- ☕ E4X

E4X sample

```
var sales = <sales shoppingcartID="12345">
  <item type="LCD TV" price="3000" quantity="5"/>
  <item type="Sony PSP 3" price="3" quantity="0"/>
  <item type="E4X for dummies" price="35" quantity="3"/>
</sales>;

writer.println( "quantity: "+sales.item.(@type ==
                "E4X for dummies").@quantity );
writer.println( "shopping cart ID is "+sales.@shoppingcartID);

for each( var price in sales..@price ) {
  writer.println("<br>price is: "+ price );
}
```



JSON

- ☕ Native JSON support
- ☕ Easier to read write
- ☕ Easier to pass back and forth
 - ☞ Clients<-->Server
- ☕ The X in Ajax! (POX versus POJ)
- ☕ jMaki widgets friendly

jRuby

- ☕ Usable via JSR-223 engine
- ☕ Work in progress:
 - ➔ JRuby ↔ JavaScript bridge
 - ➔ Dispatching code to call into a JRuby class
- ☕ Start with simple types, enrich the mapping as we go
- ☕ Goal is to have freedom of choice for what language to use for the application and the libraries



Under the Hood: PhobosAdapter

- ☕ Core Phobos class
- ☕ Handles startup/shutdown, request processing, background tasks
- ☕ Container-independent
- ☕ Framework assumes request/response classes are like the servlet ones
- ☕ Adapter can plug into:
 - ➔ Web container
 - ➔ Grizzly
 - ➔ JDK 6 HTTP server



Supported Deployment Models

- ☕ As a web application
- ☕ War file size with all the libraries: ~8MB
- ☕ Can put all Phobos jars in a shared directory
- ☕ Framework code and resources like jMaki packaged with the application itself

NetBeans Support for Phobos

- ☕ Delivered as NetBeans 5.5, 6.0 plugins via the Update Center
- ☕ Phobos project type
 - ➔ Knows about the logical structure of a Phobos application
 - ➔ Wizards to create controllers, etc.
 - ➔ jMaki palette works for .ejs files too
 - ➔ JPA, Tango support via dependent projects/jars
- ☕ OS JavaScript editor plugin or NB 6.0

Testing and Debugging

- 🔗 In-browser testing very time consuming
 - ➔ Speed up development cycle as much as possible
- 🔗 Run the app in test mode using a mock container with Grizzly as the HTTP connector
- 🔗 Run in the same JVM as NetBeans
- 🔗 JavaScript debugger attached to all requests, w/ breakpoints, call stack, etc.



Building, Deploying, Running

- ☕ Build step produces a war file
- ☕ Deployment to the appserver using the existing interfaces
- ☕ Run the app as a regular web application
- ☕ Debugging on deployed app will come later (remote debugging)



Future / Roadmap

- ☕ Other languages, esp. JRuby
- ☕ Debugging deployed applications
- ☕ Use GlassFish v3 when available
- ☕ Simplified access to more EE APIs
- ☕ Seamless scripting/Java debugging
- ☕ Other containers (e.g. JBI)
- ☕ More libraries (e.g. Eventing)
- ☕ Eclipse support (with your help)





DEMO

www.javapolis.com



Summary

- ☕ Phobos: version 0.5
- ☕ JavaScript on server side: a reality
- ☕ Jmaki integrated
- ☕ Rich client tier and server tier in harmony
- ☕ Already performant
- ☕ Tools developed at the same time



To learn more

-  <https://ajax.dev.java.net/>
-  <https://phobos.dev.java.net/>
-  <https://glassfish.dev.java.net/>
-  <https://grizzly.dev.java.net/>
-  <http://www.mozilla.org/rhino>
-  <http://dojotoolkit.org/>





Q&A

www.javapolis.com

