



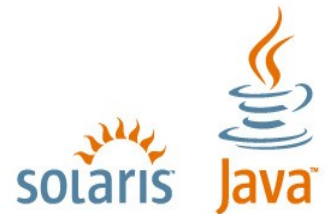
Ajax Frameworks and Toolkits

Sang Shin

sang.shin@sun.com

www.javapassion.com

Java Technology Architect



**UNLOCK
OPPORTUNITY**

What will you open?

SUN TECH DAYS 2006-2007
A Worldwide Developer Conference

**This Presentation is
part of 18-Week
(18 Topics)
Free AJAX Programming
online course.**

**[www.javapassion.com/
ajaxcodecamp](http://www.javapassion.com/ajaxcodecamp)**

Types of AJAX Toolkit and Framework Solutions of Today

- Clients-side JavaScript Libraries (ex: Dojo toolkit)
- RMI-like remoting via proxy (ex: DWR)
- AJAX-enabled JSF components (ex: NetBeans VWP)
- Wrapper (ex: jMaki)
- Java to JavaScript/HTML translator (ex: GWT)
- MVC server-side scripting (ex: Phobos)
- Web Application Frameworks with AJAX extension (ex: Shale or Echo2)

Client Side JavaScript Libraries

Client Side JavaScript Libraries

HTMP, JSP Pages, JavaScript Event Handlers

UI Widgets &
Components

Remoting Abstraction Layer

XMLHttpRequest

iFrame

JavaScript,
DOM Utilities

Characteristics of Client Side JavaScript Libraries

- Server side technology agnostic
 - > The server side technology can be Java EE, .Net, PHP, Ruby on Rails, etc.
- You can use them in combination in a single app
 - > You might want to use widgets and JavaScript utilities from multiple sources

Technical Reasons for using Client-side JavaScript Libraries

- Handles remote asynch. communication (remoting)
 - > Hides low-level `XMLHttpRequest` operation
- Handles browser incompatibilities
 - > No need to clutter your code with if/else's
- Handles graceful degradation
 - > Uses `IFrame` if the browser does not support `XMLHttpRequest`
- Provides page navigation hooks over Ajax
 - > Back and forward buttons
 - > Bookmarking

Technical Reasons for using Client-side JavaScript Libraries

- Provides ready-to-use widgets
 - > Tree, Calendar, Textfield, Button, Split panes, Fisheye, etc.
- Provides easy-to-use DOM utility
 - > Easier to use than original DOM APIs
- Provides useful JavaScript utilities
 - > Example: Table management, Timer, etc
- Provides error handling hook
 - > Easier to add error handler
- Provides more flexible event handling
 - > DOM node based, Function call based, AOP style

Technical Reasons for using Client-side JavaScript Libraries

- Provides advanced UI features
 - > Animation
 - > Drag and drop
 - > Fade out and Fade in
- Generally encourages OO programming style
 - > Helps you write better JavaScript code

Business Reasons for using Client-side JavaScript Libraries

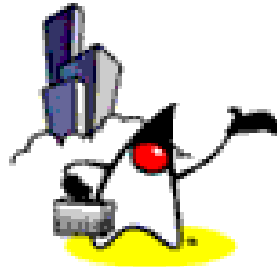
- Proven in the market
 - > Generally higher quality than your own
- Established developer/user communities
 - > Community keeps improving/adding features
 - > Easy to get help from community forums
- Easy to use
 - > It is just a matter of having them in the root directory of your Web application or providing URL location
- Tool support
 - > IDE's will support them in time

Client-side JavaScript Libraries

- DOJO Toolkit
 - > Most prominent and comprehensive
 - > Gaining a leadership in this space
 - > Major industry support (Sun, IBM)
 - > <http://dojotoolkit.com/>
- Prototype
 - > Used by other toolkit libraries
 - > <http://prototype.conio.net/>

Client-side JavaScript Libraries

- Script.aculo.us
 - > Built on Prototype
 - > Nice set of visual effects and controls
 - > <http://script.aculo.us/>
- Rico
 - > Built on Prototype
 - > Rich AJAX components and effects
 - > <http://openrico.org/>
- DHTML Goodies
 - > Various DHTML and AJAX scripts
 - > <http://www.dhtmlgoodies.com/>



Demo #1: Running Dojo Toolkit

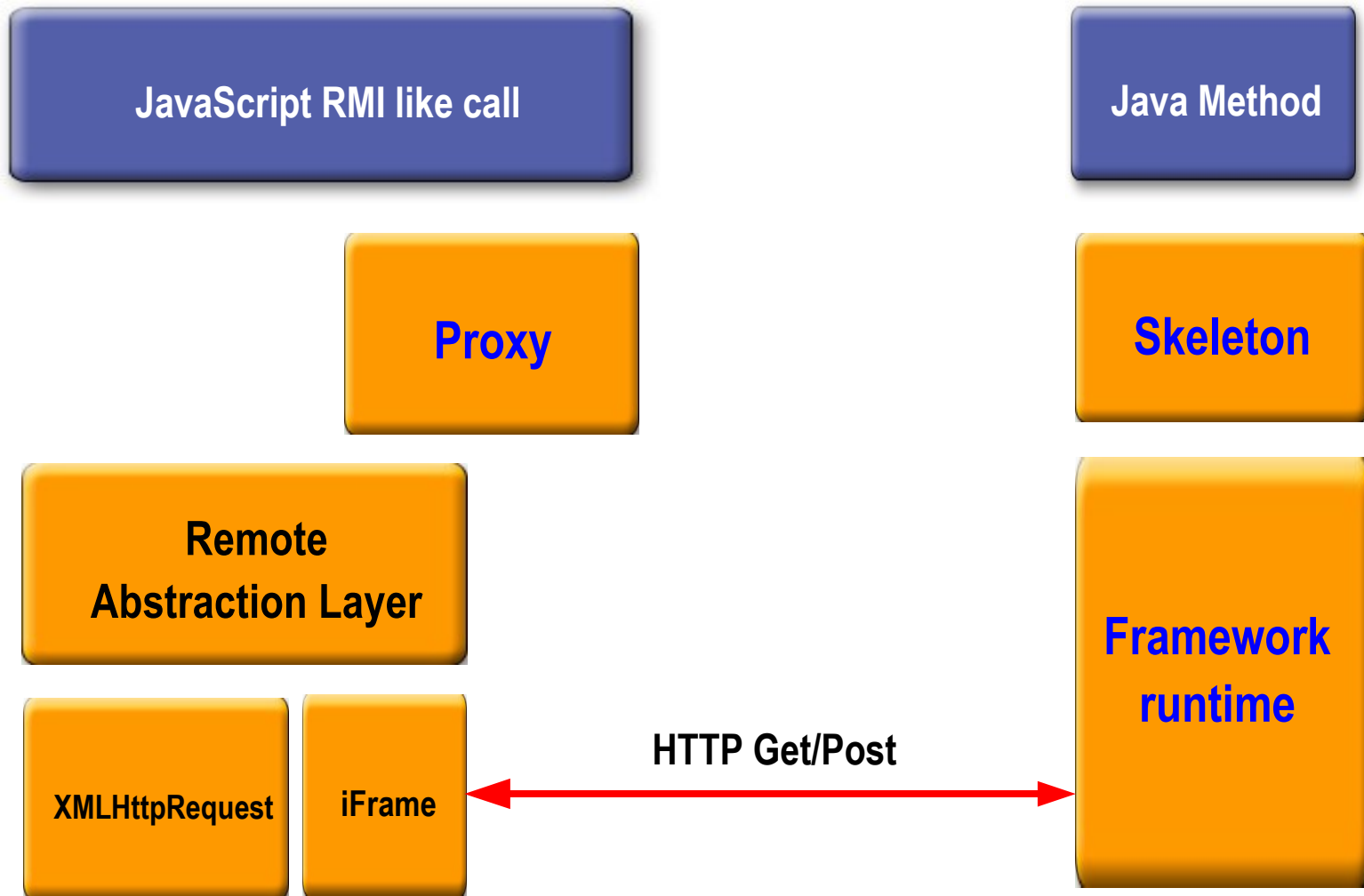
http://www.javapassion.com/handsonlabs/4260_ajaxdojointro.zip

Dojo Demo Scenario

- Build and run “input validation” Ajax application using Dojo toolkit's `dojo.io.bind()`
 - > Forward, backward capabilities
- Build and run “Fisheye application” using `dojo.event.connect()`
 - > Event model

RMI-like Remoting via Proxy

RMI-like Remoting via Proxy

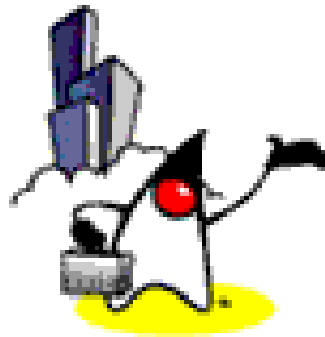


Characteristics of “RMI-like Remoting via Proxy” Framework

- Similar to general RPC communication schemes
 - > Stub and Skeleton based architecture
- Allows RMI like syntax in the client side JavaScript code
 - > ex) `Chat.addMessage(text, gotMessages);`
- Framework
 - > Generates client stub (Proxy), which is a JavaScript code
 - > Provides server side runtime as well
- Client stub (Proxy) handles marshalling of parameters and return value

Remoting via Proxy Implementations

- Direct Web Remoting (DWR)
 - > Designed specifically for Java application at the backend
 - > <http://getahead.ltd.uk/dwr>
- JSON-RPC
 - > Lightweight remote procedure call protocol similar to XML-RPC
 - > <http://json-rpc.org/>
 - > There are language-specific implementations
 - > JSON-RPC-Java
 - > <http://oss.metaparadigm.com/jsonrpc/>



Demo: Building and running DWR Application

http://www.javapassion.com/handsonlabs/4265_ajaxdwrintro.zip

DWR Demo Scenario

- Build and run Chat application
 - > Test with 2 browser clients
- Show test feature of DWR
- Show Java class and its methods that are exposed
 - > Chat has two methods – `getMessages()` and `addMessage()`
- Show [dwr.xml](#) configuration file
 - > Specifies Chat class for remoting
- Show client-side JavaScript code
 - > RMI-like syntax (application-level)
 - > Asynchronous callback

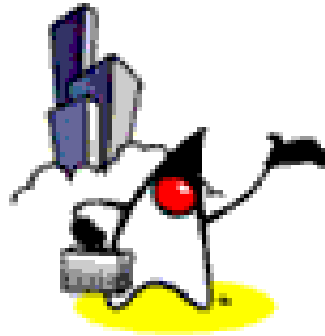
AJAX-Enabled JSF Components

AJAX-enabled JSF Components

- AJAX-enabled JSF components hides all the complexity of AJAX programming
 - > Page author does not need to know JavaScript
 - > The burden is shifted to component developers
- Leverages drag-and-drop Web application development model of JSF through an IDE
 - > You can drag and drop AJAX-enabled JSF components within Sun Java Studio Creator 2 or NetBeans Visual Web Pack (and other JSF-aware IDE's) to build AJAX applications
- JSF components are reusable
 - > More AJAX-enabled JSF components are being built by the community

Implementations

- Blueprint AJAX-enabled JSF components (open-source)
 - > <http://developers.sun.com/ajax/componentscatalog.jsp>
 - > <https://bpcatalog.dev.java.net/ajax/jsf-ajax/>
- ajax4jsf (open-source)
 - > Can add AJAX capability to existing applications
 - > <https://ajax4jsf.dev.java.net/>
- ICEfaces (ICESoft) - commercial
 - > <http://www.icesoft.com/products/icefaces.html>
- DynaFaces (development on-going)
 - > <https://jsf-extensions.dev.java.net/nonav/mvn/slides.html>



Demo: Building Ajax Application using Ajax- enabled JSF Components

<http://www.netbeans.org/kb/55/v/wp-ajaxprogressbar.html>

NetBeans VWP Demo Scenario

- Build a Web application by drag-and-dropping Ajax enabled JSF component
 - > Auto-complete
- Modify the server side logic to enable the Ajax behavior of the component

Wrapper Technology: jMaki

Motivations for jMaki

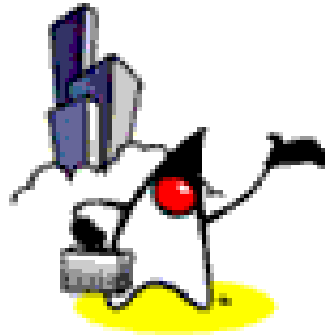
- You want to leverage widgets from existing and future AJAX toolkits and frameworks in reusable fashion
 - > Dojo, Scriptaculus, Yahoo UI Widgets and DHTML Goodies
- Today, there are multiple AJAX frameworks with their own widgets and with different syntax
 - > There is a need for a common programming model for using widgets from multiple AJAX toolkits and frameworks
- JavaScript coding is too alien to many Java EE developers
 - > There is a need for using JavaScript widgets using Java EE syntax and programming model (JSP custom tags)

What is jMaki?

- JavaScript Wrapper framework for the Java platform
 - > The name, jMaki, was derived from "j," for JavaScript, and "Maki," a Japanese word for wrap
- Allows developers to take widgets from many popular AJAX toolkits and frameworks, and wrap them into a JSP or JSF tag
 - > Provides a common programming model to developers
 - > Leverages the widgets from popular frameworks
 - > JSP and JSF tags are familiar to Java EE application developers
- Publish and subscribe event model

Wrapper Technology Implementations

- jMaki
 - > <https://ajax.dev.java.net/>



Demo: Building and running jMaki Application

http://www.javapassion.com/handsonlabs/4270_ajaxjmakiintro.zip

jMaki Demo Scenario

- Build a simple jMaki application using widgets from various sources
 - > Using a pre-defined layout

Java Code To JavaScript/HTML Translator: GWT

What is GWT?

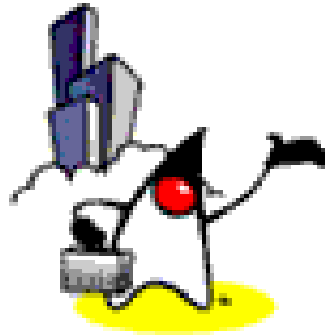
- Java software development framework that makes writing AJAX applications easy
- Let you **develop and debug AJAX applications in the Java language** using the Java development tools of your choice
 - > NetBeans or Eclipse
- Provides **Java-to-JavaScript compiler** and a special web browser that helps you debug your GWT applications
 - > When you deploy your application to production, the compiler translates your Java application to browser-compliant JavaScript and HTML

Why GWT?

- No need to learn/use JavaScript language
 - > Leverage Java programming knowledge you already have
 - > You can call JavaScript code if you want
- No need to handle browser incompatibilities and quirks
 - > GWT handles them for you
- No need to learn/use DOM APIs
 - > Use Java APIs
- No need to handle forward/backward buttons browser-history
 - > GWT handles it for you
- No need to build commonly used Widgets
 - > Most of them come with GWT

Why GWT?

- Leverage various tools of Java programming language for writing/debugging/testing
 - > For example, NetBeans or Eclipse
- JUnit integration
 - > GWT's direct integration with JUnit lets you unit test both in a debugger and in a browser and you can even unit test asynchronous RPCs
- Internationalization
 - > GWT includes a flexible set of tools to help you internationalize your applications and libraries
 - > GWT internationalization support provides a variety of techniques to internationalize strings, typed values, and classes



Demo: Building and running GWT Application

http://www.javapassion.com/handsonlabs/4275_ajaxgwtintro.zip

GWT Demo Scenario

- Build and run a simple HelloWorld GWT application
 - > Write the code in Java programming language
 - > Run it in both hosted and web mode
- Open “KitchenSink” NetBeans GWT project and run
 - > Play around various widgets provided by the GWT
 - > Backward and forward button behavior
- Show how to invoke JavaScript code from Java code and vice versa

Phobos (MVC Server Side Scripting for the Java Platform)

What is Phobos?

- Lightweight, scripting-friendly web application environment
 - > Develop Web applications using Scripting language at both client and server side
- Runs on the Java™ platform
 - > Can call Java APIs from Script
- Complementary to existing technologies
 - > Leverage all Java EE stack
- Supports multiple scripting languages
 - > Based on JSR 223
 - > Current focus is on Javascript

It is a Web Application Framework

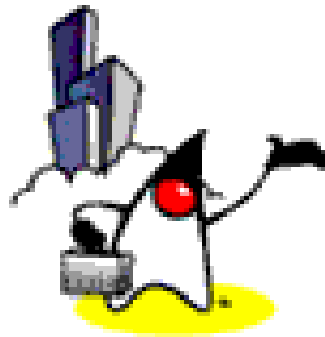
- Entirely written in Javascript
- MVC-based
 - > Model – provides data
 - > View - presentation
 - > Controller – handles HTTP request & dispatching
- Configurable, Rails-like URL mapping
 - > /@controller/@action
- Views using templates and layouts
 - > Embedded Javascript (.ejs)
 - > Freemarker template engine

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

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



Demo: Building and running Phobos Application

<https://phobos.dev.java.net/screenscasts/Phobos/Phobos.html>

Phobos Demo Scenario

- Build and run a simple Phobos application
- Show MVC architecture
- Show URL mapping
- Create and change model
- JavaScript source-code level debugging
- Access JPA from Scripting code

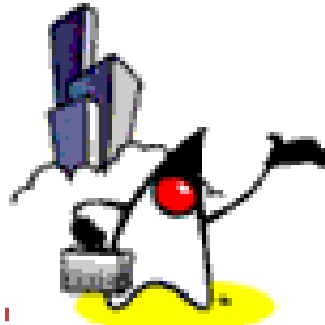
Web Application Frameworks with AJAX Extension

Web Application Framework with Ajax Extension

- Existing Web Application Frameworks add AJAX functionality
 - > Minimum or no requirement of JavaScript coding
- Uses JavaScript client library internally

“Web App Frameworks with AJAX Extension” Implementations

- Shale
 - > <http://struts.apache.org/struts-shale/>
- Echo2
 - > <http://www.nextapp.com/platform/echo2/echo/>
- Wicket
 - > <http://wicket.sourceforge.net/>
- Ruby on Rails
 - > <http://www.rubyonrails.org/>



Demo: Building and running Ajax enabled Shale Application

http://www.javapassion.com/handsonlabs/4281_ajaxshale.zip

So... What Should I Use?

So What Should I Use? Assuming You are using Java Tech.

- On the UI side
 - > Use AJAX-enabled JSF components whenever possible using an JSF-enabled IDE such as Visual Web Pack of NetBeans
 - > If you are not ready to commit yourself to JSF component solutions yet, use jMaki
 - > If you want to have total control on the client side JavaScript coding, use Dojo toolkit
 - > If you already have Swing apps that you want to expose as AJAX-fied Web apps or if you do not want to deal with JavaScript coding, use GWT

So What Should I Use? Assuming You are using Java Tech.

- On the business logic side
 - > If you already have Java EE business logic that you want to be exposed as RMI calls on the client with AJAX behavior, use DWR
 - > If you are already using a particular Web application framework for building majority of your web application and the framework has AJAX extension, use it

So What Should I Use?

If you want use Scripting Language as a Main enabler

- Use Phobos
 - > Leverage agile development of scripting
 - > Leverage the power of Java platform
 - > MVC based
 - > Multiple scripting languages



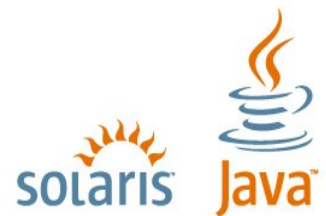
Ajax Frameworks and Toolkits

Sang Shin

sang.shin@sun.com

www.javapassion.com

Java Technology Architect



**UNLOCK
OPPORTUNITY**

What will you open?

SUN TECH DAYS 2006-2007
A Worldwide Developer Conference