Web 2.0 JavaScript Frameworks & Toolkits

Sang Shin
Michèle Garoche
http://www.javapassion.com
"Learn with Passion!"

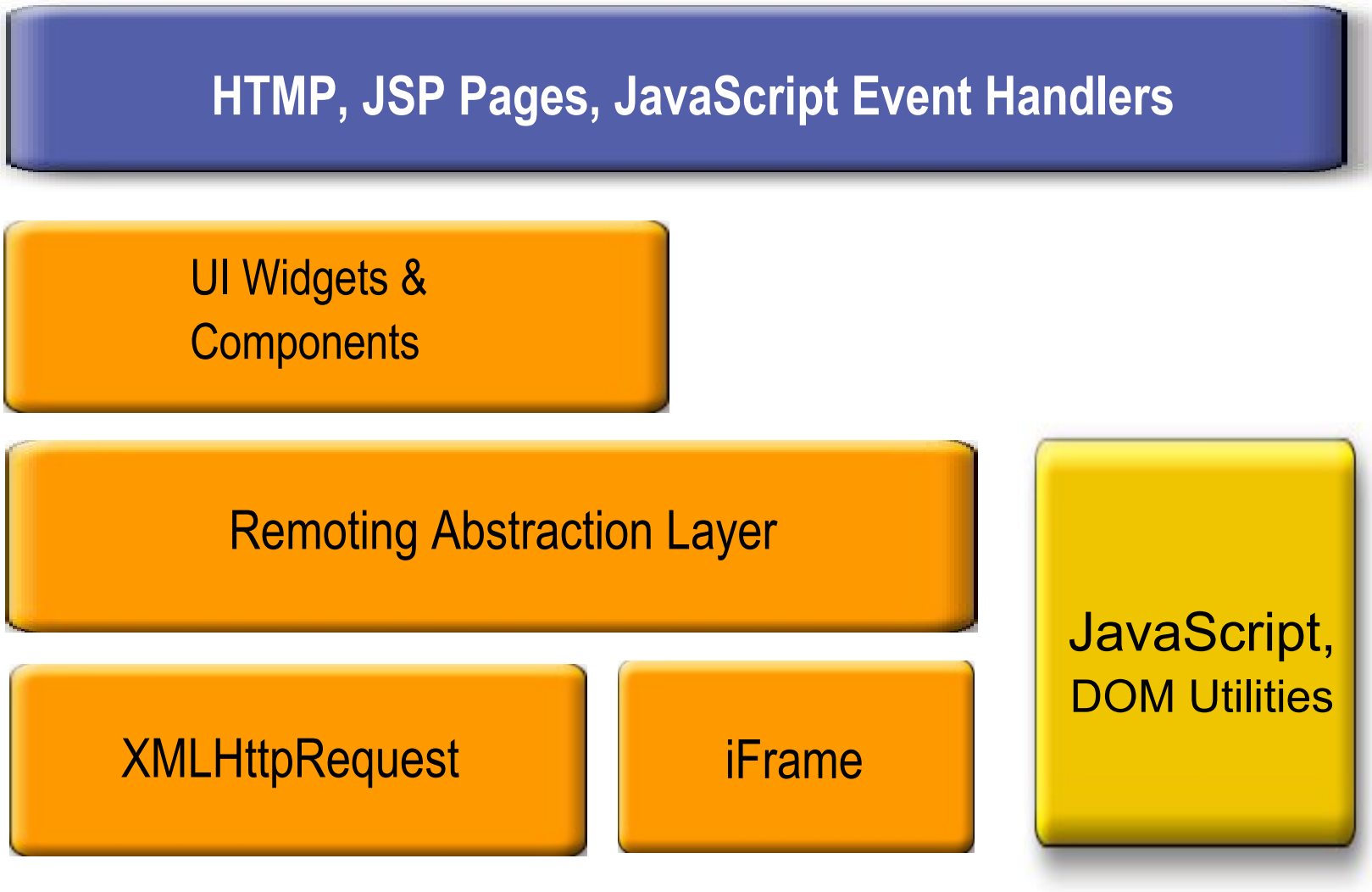
Types of Web 2.0 Toolkit and Framework Solutions of Today

- Clients-side JavaScript Libraries (ex: Dojo toolkit, jQuery, Ext JS)
- RMI-like remoting via proxy (ex: DWR)
- Java to JavaScript/HTML translator (ex: GWT)
- AJAX-enabled JSF components (ex: ICEFaces, DynaFaces)
  - JSF 2.0 of Java EE 6 supports Ajax as built-in feature
- Web Application Frameworks with AJAX extension (ex: Echo2 or Wicket)
- Integrator (ex: jMaki)

The goal is this presentation is to give you a high-level exposure of possible technologies and NOT to teach each of these technologies in detail. In the rest of this course, you will learn each of these technologies in detail.

Client Side JavaScript Libraries (Examples: Dojo Toolkit, jQuery, Ext JS)

Client Side JavaScript Libraries



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

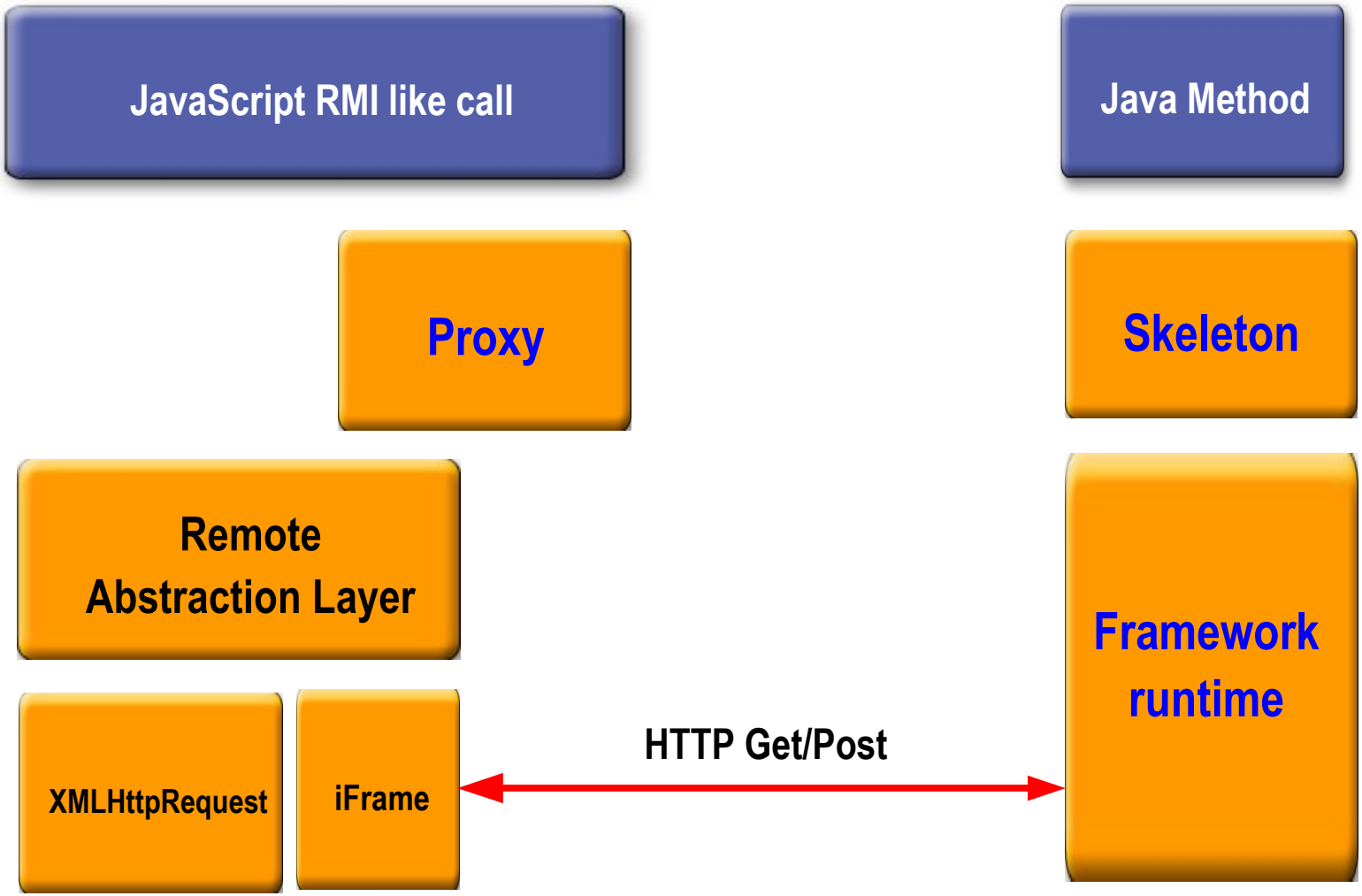
- DOJO Toolkit
  - Most comprehensive
  - http://dojotoolkit.com/
- jQuery
  - Gaining popularity
  - http://jquery.com/
- Ext JS
  - Gaining popularity
  - http://extjs.com/

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/

RMI-like Remoting via Proxy (Example: DWR)

RMI-like Remoting via Proxy



Why DWR?

- What happens if you have several methods in a class on the server that you want to invoke from the browser?
  - Each of these methods need to be addressable via URI whether you are using *XMLHttpRequest* directory or client-side only toolkit such as Dojo or Prototype
  - You would have to map parameters and return values to HTML input form parameters and responses yourself
- DWR provides a client/server framework that addresses these problems
- DWR comes with some JavaScript utility functions

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
- No need to handle browser incompatibilities and quirks
  - GWT handles them for you
  - Forward/backward buttons
  - Browser history
- No need to learn/use DOM APIs
  - Use Java APIs
- No need to build commonly used Widgets
  - Widgets 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 internationalization support provides a variety of techniques to internationalize strings, typed values, and classes

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 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

Open-Source Implementations

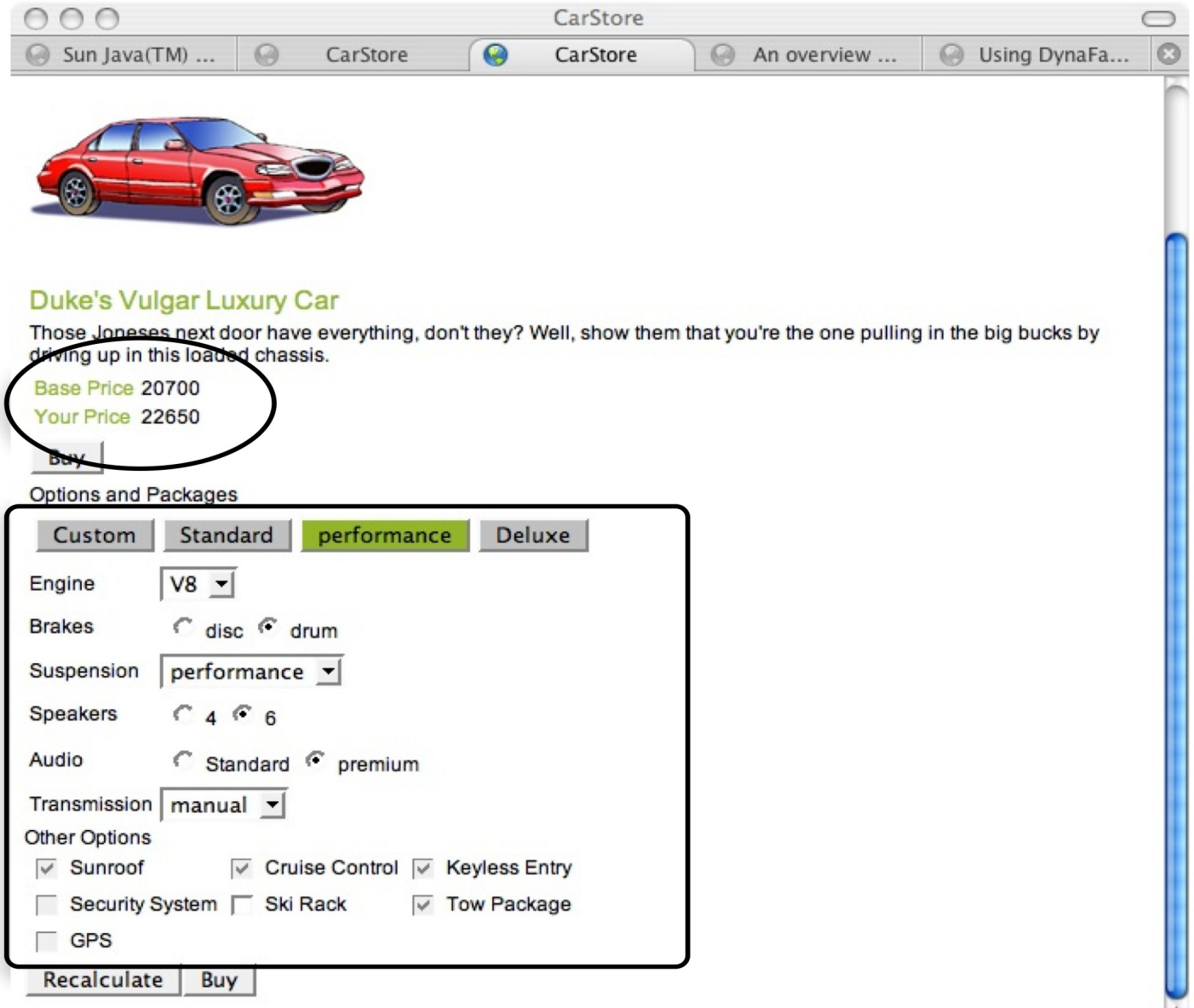
- ICEfaces
  - http://www.icefaces.org/main/home/index.jsp
- ajax4jsf
  - Can add AJAX capability to existing applications
  - https://ajax4jsf.dev.java.net/

Dynamic Faces (DynaFaces) - This feature is included in JSF 2.0 of Java EE 6

DynaFaces — Usage Patterns

- Page Author
  - Use AJAX enabled components
  - Use *AjaxZone* tag to AJAXify regions of the page
  - Use provided JavaScript library to AJAXify page elements and components
- Component Author
  - Use provided JavaScript library in custom components
  - Write your own JavaScript that talks directly to the HTTP protocol and the XML application defined by DynaFaces

Views and Partial Views

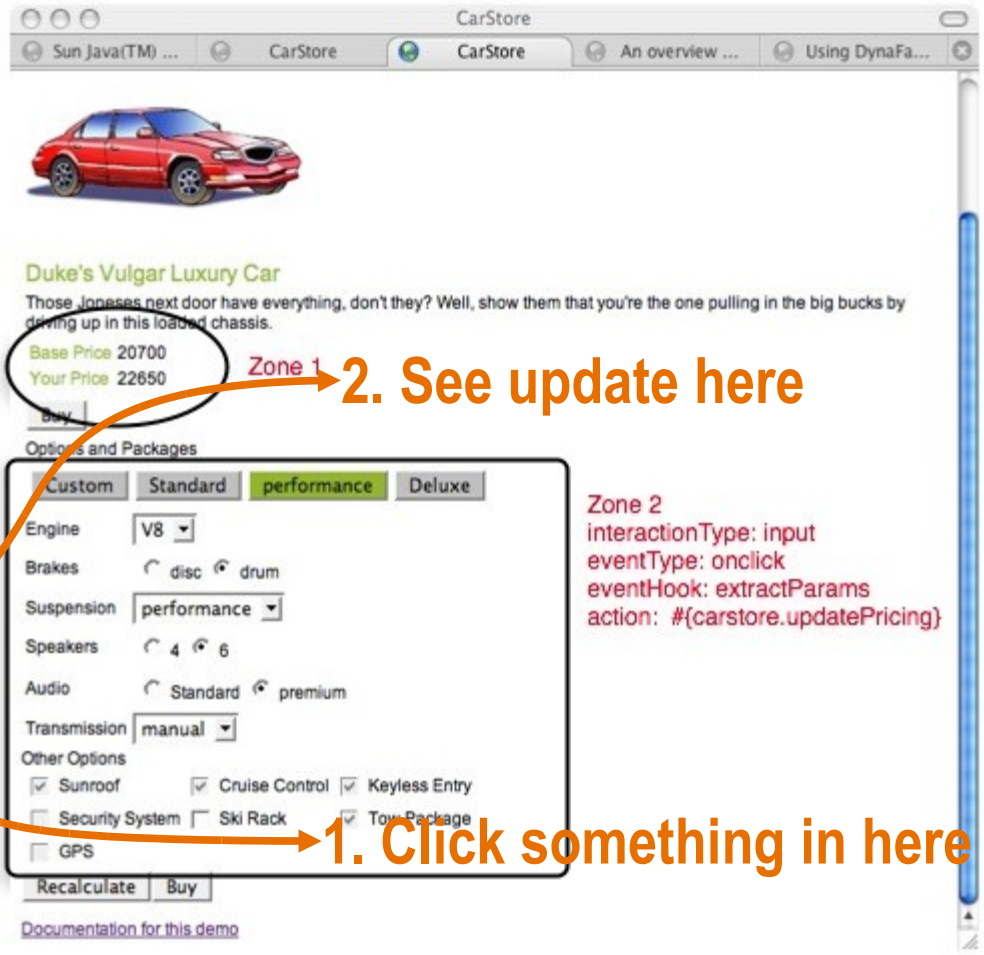


DynaFaces Usage Patterns: AjaxZones

- The easiest way to AJAXify an existing application
- Demarcate one or more AJAX zones within a page
- Zones will refresh via AJAX, without full page refresh.

causes reaction in another zone

Action in one zone



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/

So... What Should I Use?

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

- On the UI side
  - If your shop is already using JSF, use AJAX-enabled JSF components
  - If you want to have total control on the client side JavaScript coding with server side agnoscity, use Dojo toolkit, jQuery, or Ext JS
  - 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 Rails or Grails
  - Leverage agile development of scripting
  - Leverage the power of Java platform
  - MVC based
  - Multiple scripting languages

Thank you!

Sang Shin
Michèle Garoche
http://www.javapassion.com
“Learn with Passion!”