AJAX: LEVERAGING THE LANDSCAPES OF WEB SERVICES

PREMIER 2006 VOLUME 1 ISSUE 1 • WWW.AJAX.SYS-CON.COM

AjaxWorld™ MAGAZINE

The OpenAjax Technology Vision

OPEN & INTEROPERABLE

PLUS

Drag-and-Drop Shopping Carts

From 'View Source' to Open Source

Intelligent Web Applications with AJAX

AJAX + SOA: The Next Killer App

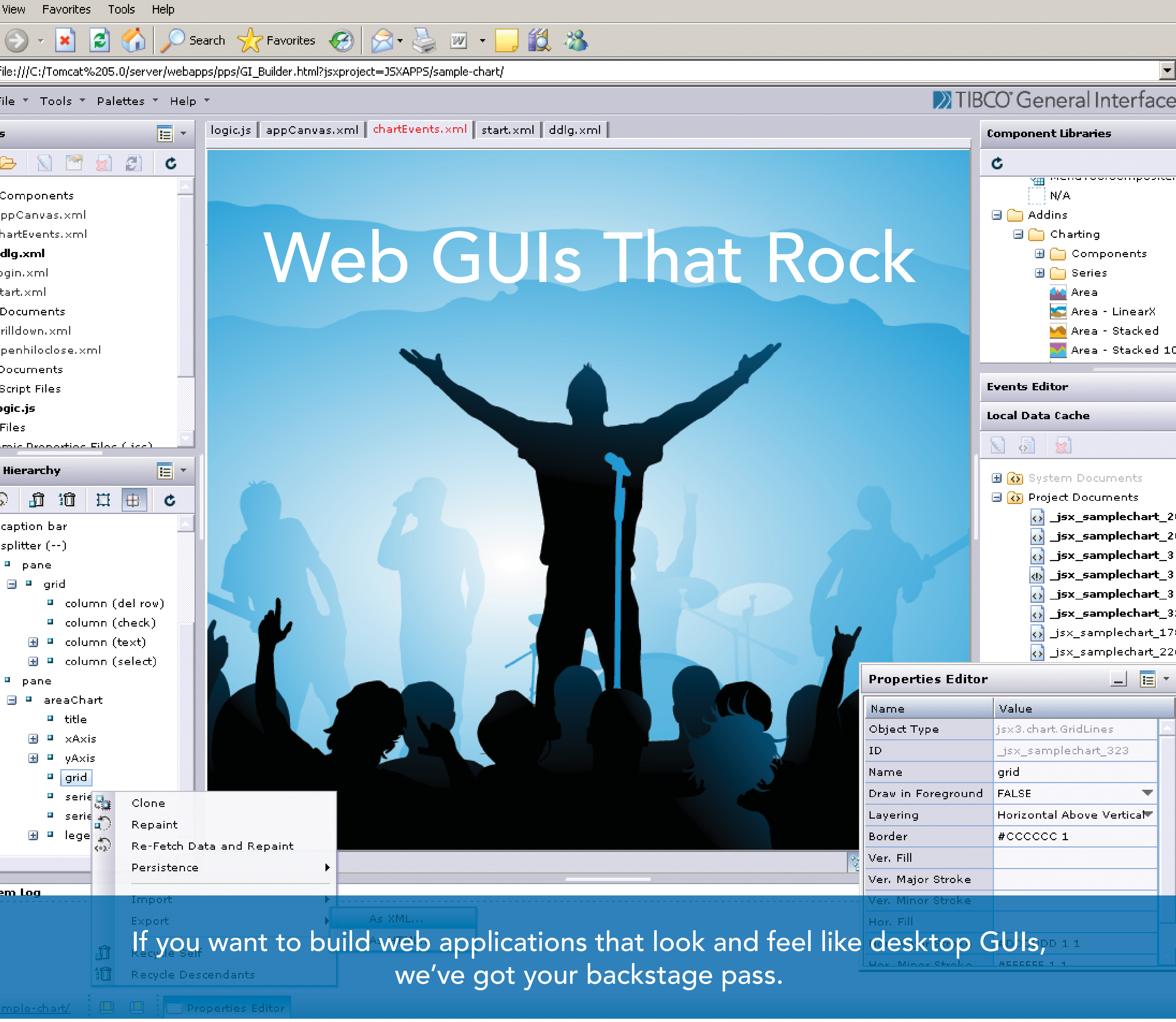
Integrating AJAX with JMX

JavaServer Faces & AJAX for Google Fans

Custom Error Handling Using AJAX

RETAILERS PLEASE DISPLAY UNTIL DECEMBER 31, 2006

$5.99US $6.99CAN



Name	Value
Object Type	jsx3.chart.GridLines
ID	_jsx_samplechart_323
Name	grid
Draw in Foreground	FALSE
Layering	Horizontal Above Vertical
Border	#CCCCCC 1
Ver. Fill	
Ver. Major Stroke	
Ver. Minor Stroke	
Hor. Fill	

Why are professional developers choosing TIBCO General Interface?
With more components and tools, TIBCO General Interface™ enables you to create AJAX applications that look and feel like desktop GUIs with astounding speed. No wonder it's the #1 independently rated AJAX Rich Internet Applications toolkit.

SOA and AJAX Rich Internet Applications are changing the way software is developed and deployed. TIBCO helps you make the shift from 3-tier to SOA-based computing to deliver astoundingly rich and agile solutions fast.

TIBCO®
The Power of Now®

Open Ajax alliance MEMBER

Are *you* ready to rock? Learn more at http://developer.tibco.com/.

You Hold the Future of the Web in Your Hands

Once upon a time – in a world before MashupCamps and online widget platforms like live.com, before Google's personalized homepage and pageflakes, and before JSON, Comet, Dojo, and Apache Derby – there was a term "DHTML" (for dynamic HTML). It was used, as Wikipedia reminds us, for "a collection of technologies, used together to create interactive and animated web sites by using a combination of static markup language (such as HTML), a client-side scripting language (such as JavaScript), the presentation definition language (e.g. Cascading Style Sheets [CSS]), and the Document Object Model."

Recent history has proved that "DHTML" was one letter too many, that the world prefers snappy acronyms to klunky initialisms. Plus there was the small matter of how DHTML scripts often tended to not work well cross-platform.

Now that very same approach – JavaScript, CSS, and the DOM, plus the secret sauce of the XMLHttpRequest object to exchange data asynchronously with the web server – is, as we all know, named "AJAX." And this four-letter word, and the approach it crystallizes, has catalyzed a profound transformation in the way that users and businesses alike are going to be using the Web.

Without getting into the debate of whether "Web 2.0" is a useful term for that transformation or not, the undeniable fact is that something big is happening, just fifteen years after Tim (now Sir Tim) Berners-Lee made public a little project he called the World Wide Web. And that it involves, if not AJAX, then some kind of similar approach: which is why we at SYS-CON Media are confident that what you are holding in your hands is the first issue of a new magazine destined to grow, widen and deepen in the very best tradition of SYS-CON's market-leading titles like *Java Developer's Journal*, *SOA Web Services Journal*, *Web Developer's & Designer's Journal*, and *Enterprise Open Source Magazine*.

It is also why we at SYS-CON Events are proud to be bringing you the first AJAXWorld Conference & Expo – at which you may well be reading this copy of *AJAXWorld Magazine*. Few technologies this young achieve the two milestones of a totally dedicated magazine and a totally dedicated conference and trade show, all within less than 18 months of their creation. But then AJAX, as the contributors to this premier issue remind us time and time again, is an approach not a technology.

It is an approach whose time has come. It now has even reached a third milestone, that of having a major cross-industry body formed because of it, and to help guide and inform its penetration and adoption – in this case, the OpenAjax Alliance.

You can find out more about that from our cover story, "The OpenAjax Technology Vision." Just as you can find out more about myriad other aspects of AJAX and Rich Internet Applications in the one hundred pages which follow – our largest-ever premier issue in the thirteen-year history of the company.

I look forward to seeing the magazine flourish, and am honored that right from the get-go we secured Dion Hinchcliffe to be its Editor-in-Chief. Many of you will know Dion from his significant footprint in the blogosphere, and you will be able to infer – rightly – that the articles and features published in *AJAXWorld Magazine* under his custodianship will never be anything less than technically astute and/or strategically prescient, while all the time being anchored by two things: a preference for actionable software over whiteboard-ware…and the all-important centrality of the user.

Take it away, Dion!

Jeremy Geelan
Sr. Vice-President, Editorial & Events
SYS-CON Media

Jeremy Geelan

Facing a few barriers on the road to SOA?

JackBe's Rich Enterprise Application (REA) platform clears the road to SOA business benefits.

There's an abundance of products and vendors to help you *create* your SOA. Now, *consume* those SOA services with JackBe REAs to achieve the business productivity and results that led you to SOA in the first place. Our new REA platform combines the power of Ajax and SOA with reliable, secure communications to extend your SOA directly into powerful business applications.

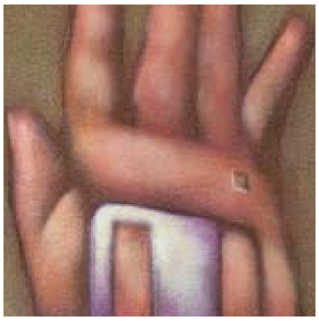
A fully visual IDE reduces manual coding and accelerates the development process. And our lightweight, vendor-neutral platform easily integrates with existing middleware and services while maintaining server-side governance and control--unlike products that leave governance to the browser or create middleware platform dependencies.

Join over 35 industry leaders like Citigroup, the U.S. Defense Intelligence Agency, Sears, Tupperware, and Forbes who are already optimizing their business activity with JackBe solutions. Call or visit our website—let us help you remove the barriers on the road to achieving real business value from your SOA investment.

JACKBE™
Optimizing Business Activity

Web: www.jackbe.com
Phone: (240) 744-7620
Email: info@jackbe.com

AjaxWorld Magazine
Premier 2006



AJAXWORLD MAGAZINE

Group Publisher Jeremy Geelan
Art Director Louis F. Cuffari

Editorial
Editor-in-chief
Dion Hinchcliffe
dion@sys-con.com

Editor
Nancy Valentine, 201 802-3044
nancy@sys-con.com

To submit a proposal for an article, go to http://grids.sys-con.com/proposal.

Subscriptions
E-mail: subscribe@sys-con.com
U.S. Toll Free: 888 303-5282
International: 201 802-3012
Fax: 201 782-9600
Cover Price U.S. $5.99
U.S. $19.99 (12 issues/1 year)
Canada/Mexico: $29.99/year
International: $39.99/year
Credit Card, U.S. Banks
or Money Orders
Back Issues: $12/each

Editorial and Advertising Offices
Postmaster: Send all address changes to:
SYS-CON Media
135 Chestnut Ridge Rd.
Montvale, NJ 07645

Worldwide Newsstand Distribution
Curtis Circulation Company, New Milford, NJ

List Rental Information
Kevin Collopy: 845 731-2684,
kevin.collopy@edithroman.com,
Frank Cipolla: 845 731-3832,
frank.cipolla@epostdirect.com

Promotional Reprints
Megan Mussa
megan@sys-con.com

AJAXWorld Magazine
is published monthly (4 times a year) by SYS-CON Publications, Inc., 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677.
SYS-CON Media and SYS-CON Publications, Inc., reserve the right to revise, republish, and authorize its readers to use the articles submitted for publication.

AJAX Comes of Age

DION HINCHCLIFFE

It's just been about a year and half now since Jesse James Garrett of Adaptive Path conceptually linked together the elements of the browser into that now-famous term, AJAX, which has gone on capture the imagination of the entire software industry. Since then, the result has been nothing short of amazing. The term has formed a rallying cry to a software industry much in need of a fresh view of the things we'd too often taken for granted. The result has been access to the full power of the Web browser, which has become a computing platform in its own right. This became fully realized on that fateful day in February, 2005 when Jesse published his seminal essay. AJAX was the right idea, at the right time.

At roughly the same time last year, the software world was beginning to undergo a full post-dot-com recovery with Internet startups appearing again in large numbers for the first time since 2000. These new software startups, many of which appeared under another new rubric, "Web 2.0", were predominantly Web-based and consisted of innovative new companies that were developing consumer and corporate Web applications that rivaled, and even exceeded, what was possible with native software. The very connectedness of the Web offered interesting capabilities not possible with disconnected software.

Many of these new Web startups, such as 37signals or Writely (who was subsequently purchased with much fanfare by Google), used AJAX to deliver rich visual experiences every bit as compelling as native software. Google in particular has famously been an early adopter of AJAX, with GMail and Google Maps being preeminent examples that have set the bar for the rest of the industry. Even Microsoft, never content to be an also-ran, has been embracing AJAX with a zeal that hasn't been seen in a large company for a while. And it's not stopped there.

There is the OpenAJAX initiative, a high-profile industry coalition formed in January, 2006 by IBM, Oracle, Mozilla, Yahoo, Red Hat, and other industry leaders, which intends to keep AJAX as independent as possible from any individual browser, client operating system, server language, and server operating system. True platform freedom is one of the major benefits of AJAX, and keeping it that way will keep its potential intact.

Then there is the mashup phenomenon of the last year, which shows that combining code from multiple sources together in the browser to create new hybrid applications was surprisingly easy, with the right approach. A brief tour of MashupFeed.com is all that one needs to get a sense of the innovation and bustling endeavor that the global community of mashup creators has been inspired by. A major source? You guessed it, AJAX, along with underlying ingredients, DHTML, JavaScript, and XmlHttpRequest(), has been a key enabler of this approach.

The wave of new and existing developer tools, libraries, and frameworks that have come onto the market, or have been reinvigorated, by the rise of AJAX also points to a growing issue: AJAX development isn't easy. For now, AJAX is not primarily for use by Web designers, and requires capable developers to create. Writing good AJAX software can be a significant engineering challenge when used for more than simple, visual eye-candy. AJAX also depends on Web services to provide information to the user, and this aspect alone requires experience with distributed computing technologies and server-side development. Then there are performance and throttling issues that come into play with AJAX applications that support more than a few users. AJAX is sophisticated, and though that means it's able to deliver robust software, it also means there's a major learning curve.

Finally, organizations are finding that AJAX is a terrific way to leverage the landscapes of Web services that many organizations are delivering internally for their developers to reuse. The client/SOA model actively encourages the use of AJAX since it requires nothing more than what you find in your local browser: no administrator rights, no installation, and no permanent client footprint at all. And AJAX is also fairly friendly with the burgeoning world of service-oriented architectures, even though the browser does not speak SOAP, the most popular official Web services standard. Cross-browser SOAP stacks are now quite easy to get if you want them.

All of these issues are far from show stoppers, but it's going to be a while until the AJAX development techniques, design patterns, and best practices become widespread and common knowledge. It's been said that civilization advances when things that were formerly hard to do become easy. And along that line of thinking, a lot of smart folks are putting their efforts into doing precisely this with AJAX.

As part of this industry evolution, AJAXWorld will be there in the forefront, bringing you the very latest in news, insight, technical information, product reviews, and everything else that you will need in order to develop the best AJAX-powered software. I hope you'll be reading these pages as well as participating in our terrific, world-class educational events on AJAX being held around the world. ■

Dion Hinchcliffe, president and CTO of Hinchcliffe & Company, is editor-in-chief of AJAXWorld Magazine.

Adobe Solution PARTNER

CYNERGY

THE LEADERS
IN RIA DEVELOPMENT SERVICES

INCREDIBLE APPLICATIONS
PASSIONATE USERS
PROVEN SUCCESS

Unlock your potential with the help of industry leaders in Rich Internet Application development. 10 Years. 1400+ Customers. **Your app starts here.**

cynergysystems.com

SYS-CON MEDIA

SYS-CON Media
President & CEO
Fuat Kircaali, 201 802-3001
fuat@sys-con.com
Group Publisher
Jeremy Geelan, 201 802-3040
jeremy@sys-con.com

Advertising
Senior Vice President, Sales & Marketing
Carmen Gonzalez, 201 802-3021
carmen@sys-con.com
Vice President, Sales & Marketing
Miles Silverman , 201 802-3029
miles@sys-con.com
Advertising Sales Director
Robyn Forma, 201 802-3022
robyn@sys-con.com
Advertising Sales Manager
Megan Mussa, 201 802-3023
megan@sys-con.com
Associate Sales Managers
Kerry Mealia, 201 802-3026
kerry@sys-con.com
Lauren Orsi, 201 802-3024
lauren@sys-con.com

Production
Lead Designer
Louis F. Cuffari, 201 802-3035
louis@sys-con.com
Art Director
Alex Botero, 201 802-3031
alex@sys-con.com
Associate Art Directors
Tami Beatty, 201 802-3038
tami@sys-con.com
Abraham Addo, 201 802-3037
abraham@sys-con.com
Assistant Art Director
Mandy Ekman, 201 802-3032
mandy@sys-con.com

SYS-CON.COM
Consultant, Information Systems
Robert Diamond, 201 802-3051
robert@sys-con.com
Web Designers
Stephen Kilmurray, 201 802-3053
stephen@sys-con.com
Paula Zagar, 201 802-3057
paula@sys-con.com

Accounting
Financial Analyst
Joan LaRose, 201 802-3081
joan@sys-con.com
Accounts Payable
Betty White, 201 802-3002
betty@sys-con.com
Accounts Receivable
Gail Naples, 201 802-3062
gailn@sys-con.com

Customer Relations
Circulation Service Coordinators
Edna Earle Russell, 201 802-3081
edna@sys-con.com

Achieving Business and IT Agility with Enterprise Web 2.0

Coach Wei

Web 2.0 technologies promise to turn the Internet into a true application platform, featuring robust client-side logic and rich interfaces that put users back in control of application flow. For the enterprise IT community, achieving the aims of Web 2.0 requires looking beyond the adoption of popular Rich Internet Applications (RIAs) development languages like AJAX, Flash, Java, and .NET.

Companies looking to implement an Enterprise Web 2.0 (EW2.0) strategy require a platform that provides standardization and simplification across different business applications and development technologies, while enabling the flexibility required for innovation within business units, otherwise called "common flexibility." The platform must provide Web 2.0 applications with a reliable and secure communication between client and server - whether online, offline or mobile - across any network. Finally, it needs to support Service-Oriented Architecture (SOA) initiatives by enabling the consumption of loosely coupled services that provide access to business functionality and data in real time, while leveraging existing code, development standards, tools, skills and infrastructure.

Over the last 20 years, mainstream enterprise applications have swung back and forth between server- and client-centric architectures. Originally, mainframe architectures were server-based and sent the user interface (UI) to display terminals. In the 1990s, attracted by the power of graphical desktop environments like Microsoft Windows, the pendulum swung to the other end of the spectrum - client/server, which was entirely client-based except for server-side databases. In the late 1990s, the next pendulum swing was caused by the low-cost, global deployment model of the Web, leading to the development of browser-based HTML and J2EE applications, which were again, entirely server-based with the UI running on the client-side browsers.

Today, the architectural flexibility of Web 2.0 offers developers the best of both worlds. By delivering the high performance and robust functionality of desktop or client/server - combined with the universal reinstall deployment and centralized management of browser-based applications - EW2.0 applications deliver a quantum operational efficiency and end-user productivity, while decreasing IT costs. For the first time in 30 years of application development, EW2.0 enables application developers to partition client- and server-side layers, using appropriate technologies to meet different business objectives and specific end-user requirements.

Although EW2.0 architectures combine the best qualities of their historic predecessors, there are some lingering challenges surrounding integration, security, UI development and performance. To overcome these challenges, enterprise IT organizations should consider implementing a "reference architecture" that provides these services consistently to all EW2.0 developers, regardless of their development technology and target deployment platforms.

The EW2.0 application landscape currently includes a mix of Java, .NET, AJAX, and Flash on the client side; different ways of communicating between client and server; different server technologies ranging from HTTP servers to J2EE; different ways to integrate data sources; as well as many different business processes and services, development tools, and methodologies. Simply put, there is a lot of complexity and heterogeneity to manage.

An EW2.0 reference architecture provides "common flexibility." In other words, it gives EW2.0 application developers common services and architecture without limiting the capability to leverage different landscape components and approaches to meet specific business needs. It provides a higher level of abstraction to enable developers to focus on application requirements rather than being bogged down with platform or technology specifics. This way, developers can integrate various existing systems without rewriting them, creating a composite application that can be universally deployed.

By implementing an EW2.0 reference architecture that provides "common flexibility," companies become more agile and significantly decrease the cost and risk of application development and deployment. Companies will be able to reduce development, deployment, and maintenance costs; improve responsiveness to business drivers; and reduce technology investment risk. As such, developers can build applications - using AJAX, Flash, Java, or .NET - that best suit the needs of the enterprise. ■

Coach Wei combines in-depth IT industry expertise with extensive education and research experience at MIT to drive technology innovation and business direction for Nexaweb. He founded Nexaweb in 2000 and served as CEO until summer 2003.

Build Richer, Thinner, Faster
Enterprise Applications with Ajax and Java

- Want to deploy enterprise-class Ajax applications that leverage your existing Java code, skills, and infrastructure?
- Want to build composite applications in a declarative XML environment without rewriting code?
- Want to be the hero that unlocks the ROI potential of your company's Web services and SOA investments?
- Want to do it using the latest in open source Ajax technology?

Do it all with Nexaweb!

nexaweb

Visit us at AJAXWorld in Santa Clara to register to win a Panasonic 37" Plasma HDTV and get an evaluation copy of Nexaweb's latest products.

The OpenAjax Technology Vision

Open and Interoperable

JON FERRAIOLO

OpenAjax – Fulfilling AJAX's Promise

One would think that an industry would slow down as it matures, but the Web has proven to be just the opposite. Innovations are happening at breakneck speed. Companies have to move faster than ever to keep up and survive.

AJAX is clearly a case in point. The term "AJAX" was first mentioned publicly in February 2005 by Jesse James Garrett. But roughly 18 months later, we have hundreds of companies delivering AJAX products, dozens of AJAX open source projects, and nearly everyone in the industry planning to adopt AJAX techniques as part of their future Web development strategies.

OpenAjax – How It Was Born

In late 2005, thanks largely to the globetrotting of David Boloker, IBM's CTO of Emerging Internet Technologies, a small number of leading companies brainstormed about how to ensure that AJAX fulfills its potential as the industry standard rich application platform based on open technologies. These early discussions came to a climax on Feb. 1, 2006, with the announcement of the "OpenAjax Initiative,"whose 15 original members included BEA, Borland, the Dojo Foundation, Eclipse Foundation, Google, IBM, Laszlo Systems, Mozilla Corporation, Novell, Openwave Systems, Oracle, Red Hat, Yahoo, Zend and Zimbra.

Between February 1 and May 15, another 15 organizations joined "OpenAjax,"and the (then) 30 members held a two-day kickoff meeting in San Francisco to lay out the blue-print for the initiative moving forward. At the meeting, the group decided to call itself the OpenAjax Alliance, defined its mission, agreed on an interim organizational process, and established its initial activities. Today, the organization has more than 50 members and is growing rapidly.

OpenAjax Alliance – Mission and Objectives

The OpenAjax Alliance is an organization of leading vendors, open source projects, and companies using AJAX that are dedicated to the successful adoption of open and interoperable

AJAX-based Web technologies. The prime objective is to accelerate customer success with AJAX by promoting a customer's ability to mix and match solutions from AJAX technology providers and by helping to drive the future of the AJAX ecosystem.

The set of technologies under the OpenAjax banner will provide the following benefits to Web developers:

- Lower development costs and faster delivery of Web 2.0 innovations
- Vendor choice and interoperability
- Richer web experience and greater collaboration that can be added incrementally to existing HTML websites or used for creating new applications

Alliance Activities

The OpenAjax Alliance will provide value to the software community through both marketing/communications and technical channels.

Marketing and communications

The OpenAjax Alliance will engage in various educational and communications activities. Its Web site will provide a standard vocabulary for industry terms such as "AJAX" and "OpenAjax," and will include whitepapers and block diagrams on AJAX technologies and associated best practices, with a focus on cross-vendor interoperability. Representatives will speak about OpenAjax at conferences and other industry events.

The OpenAjax Alliance Web site will provide a central point of information about the OpenAjax vision, explaining how to adopt AJAX successfully so that IT developers will feel confident about their technology and vendor choices.

Technical Committee Work

Most of the technical work is accomplished in committees. The following committees were established at the initial kickoff meeting in May:

- ***Marketing / Architecture Committee*** – This committee produces the Alliance's architecture vision documents, which help the IT manager and developer understand the AJAX landscape, the various AJAX technology alternatives (e.g., client-side AJAX toolkits vs. server-side AJAX toolkits), and criteria by which a product or technology is OpenAjax-compliant.
- ***Interoperability Committee*** - This group focuses on the ability to mix JavaScript components from different AJAX toolkits within the same Web application. Among the first topics is JavaScript name collision prevention, toolkit loading and event management in the presence of multiple AJAX runtimes used in the same Web application.
- ***Declarative Markup Committee*** - This group focuses on HTML/XML markup interoperability issues. Among the topics so far is mixing markup (HTML and/or XML markup) from different AJAX toolkits within the same Web application.

It is likely that other committees, such as a Mobile Committee, will be formed and begin work soon.

The technical committees will produce documents and (open source) source code. Documents will include roadmaps, whitepapers, and technical specifications. Given the focus on JavaScript issues, it is likely that most of the open source code produced by the Alliance will be JavaScript.

The Alliance is committed to royalty-free, unencumbered technologies in a manner that is friendly toward inclusion within commercial software products.

How OpenAjax Alliance Differs from Established Standards Bodies and Open Source Initiatives

The Alliance will purposely avoid competition with existing open standards and open source initiatives and instead will collaborate with and support any relevant open technology initiative.

The OpenAjax Alliance fills the AJAX interoperability gap in the industry. Other standards organizations such as W3C develop standards focused on what building-block features browsers must support, such as HTML, CSS, DOM, SVG, and JavaScript/ECMAScript. The OpenAjax Alliance addresses a technology layer above these browser formats, where the alliance defines "OpenAjax" specifications and best practices such that multiple AJAX toolkits will coexist and interoperate with the same AJAX-powered application.

Membership

To date, more than 50 organizations have joined the OpenAjax Alliance, including:

Adobe
Ajaxian
American Greetings (AG/Interactive)
Backbase
BEA
Bling Software
Borland
Curl
Dojo Foundation
Eclipse Foundation
edge IPK
eLink Business Innovations
ENOVIA MatrixOne
Fair Isaac
Finetooth
The Front Side
Google
IBM
ICEsoft
Ikivo
ILOG
IN2
Innoopract
Intel
IT MILL
JackBe
Javeline
JWAX
Laszlo Systems
Merced Systems
Mozilla Corporation
Nexaweb
Nitobi
Novell
OpenLink Software
Openwave Systems
Opera
Oracle
Red Hat
SAP
Scalix
Seagull Software
Sitepen
Software AG
Sun Microsystems
Tibco
Vertex Logic
Vircom
Webtide
XML11
Zend Zimbra
Zoho

OpenAjax alliance

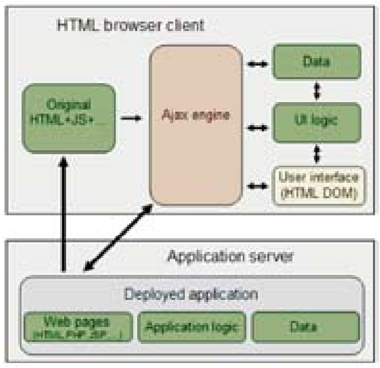


Figure 1

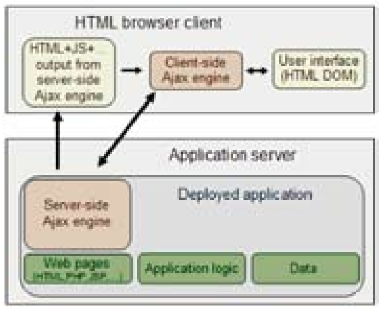


Figure 2

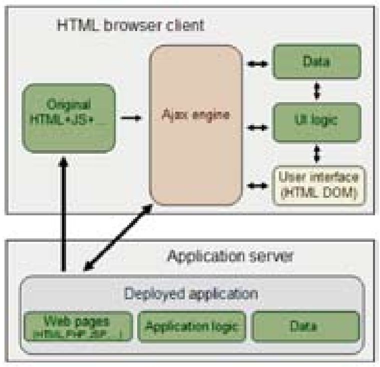


Figure 3

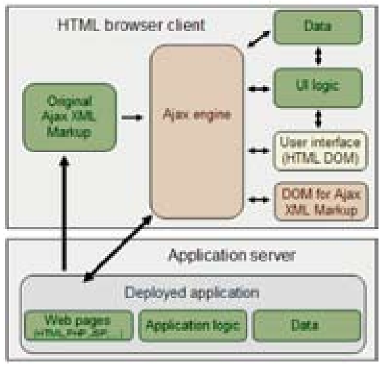


Figure 4

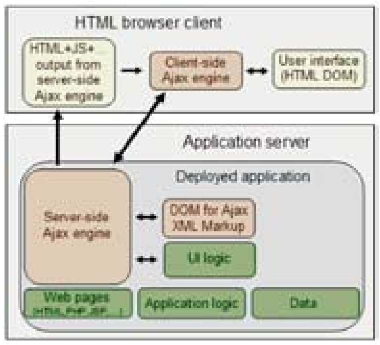


Figure 5

Open and Interoperable: The OpenAjax Technology Vision

The Alliance's mission is to accelerate customer success with AJAX by promoting a customer's ability to mix and match solutions from AJAX technology providers and by helping to drive the future of the AJAX ecosystem.

The members of the Alliance have worked together to produce a shared AJAX technology vision that is described in this article. The most fundamental aspects of the technology vision were established during the two-day kickoff meeting for the alliance (May 15-16, 2006). The rest of the vision is the result of the Alliance's committee work since the kickoff meeting.

Getting the Best of Web 2.0 and 1.0

First and foremost, the OpenAjax Alliance's technology vision represents a commitment towards open technologies. The Alliance believes that a large part of industry excitement over AJAX is due to its promise of rich user experiences through open technologies; therefore, open technologies are central to the OpenAjax technology vision.

Another reason for industry excitement with AJAX is that it promises the benefits of Web 2.0 through incremental adoption of new application development techniques while retaining the existing back-end infrastructure that has been set up around HTML. But AJAX frameworks deliver AJAX benefits in many different ways. Some AJAX frameworks do most of the AJAX work on the client, while others do most of the work on the server. Some AJAX frameworks integrate tightly with server application frameworks such as JSF; others move large parts of the application logic to the client. The OpenAjax Alliance embraces this diversity because it promotes innovation and provides customers with a rich set of choices, thereby enabling them to choose the AJAX technology approach that best matches their needs and preferences.

Open Technologies

Open standards

OpenAjax builds on the open technologies that are native to browsers. Most of the base technologies are official Web standards, while many of the rest have been implemented widely in browsers but have not been formally recognized by a standards body. Standards include JavaScript, HTML, CSS, DOM, DOM Events, XMLHttpRequest, XML and SVG.

Open source and commercial products

While open source software is not mandatory for AJAX projects, a large part of AJAX's momentum is due to the open source community's commitment to AJAX. OpenAjax embraces open source alternatives and seeks partnership with both AJAX open source initiatives and commercial products.

Platform independence (OS, server, browser, IDE)

One of the main attractions of AJAX is that it does not lock developers to a particular hardware platform, operating system, application server, web browser or IDE. Developers are free to choose among many technology providers, commercial and open source, to find the products or open source technologies that best match their unique requirements and preferences, while achieving the key benefit of write-once, run-everywhere, across multiple computing devices, operating systems, and web browsers.

Typically, AJAX toolkits deliver cross-platform and cross-browser functionality by providing a platform-neutral and browser-neutral abstraction layer to the developer. This layer is sometimes delivered as a set of client-side JavaScript libraries, and other times in the form of server-side software (e.g., Java).

Open Architectures

Multiple architecture options

AJAX developers have the ability to choose among multiple technical approaches to find the ones whose programming model best fits their needs. The following are some of the ways to categorize AJAX toolkits and frameworks.

Client-side vs. server-side

Most AJAX technologies transform a platform-independent definition of the application (usually consisting of a combination of markup and logic) into the appropriate HTML and JavaScript content that is then processed by the browser to deliver a rich user experience. Some AJAX designs perform most of their transformations on the client. Others perform transformations on the server.

Client-side AJAX transformations

Figure 1 shows a common approach to client-side AJAX.

The developer provides the green-color components, the AJAX toolkit provides the orange-colored components, and the yellow-colored components represent the final HTML DOM after the AJAX runtime library has performed its client-side transformations.

For client-side AJAX, the application server usually requires no additional AJAX-specific server software. All AJAX transformations come from the client-side AJAX runtime library.

One advantage of this option is the independence from the server side technology. The server code creates and serves the page and responds to the client's asynchronous requests. This way either side can be swapped with another implementation approach.

Server-side transformations

Server-side AJAX sometimes follows the model shown in Figure 2.

For server-side AJAX, an AJAX server component performs most or all of the AJAX transformations into appropriate client-side HTML+JavaScript. Often, the server-side AJAX toolkit downloads its own client-side AJAX library which communicates directly with the toolkit's server-side AJAX component.

The implementation strategies of server-based AJAX frameworks vary. At one side of the spectrum, the server handles all the application events. On the other side of the spectrum, many events are handled on the client. For some frameworks, the development and debugging phase handles all events on the server, but in production mode many events are handled on the client.

The main benefits of this approach are that it preserves back-end server-resident infrastructure, eliminates the need for extensive JavaScript programming, and preserves developer core competencies, development practices, and tools because it allows the use of server-side languages for debugging, editing, and refactoring tools with which developers are already familiar. This can be at the price of dependence on a particular server-side technology. As a general rule, server-side AJAX frameworks expect application code to be written in the server-side language (e.g. Java or RoR). These frameworks typically hide all the JavaScript that runs in the browser inside widgets, including their events. If pre-built capabilities don't suffice, however, new component development exposes the programmer to JavaScript.

Single-DOM vs. Dual-DOM (client-side and server-side)

Single-DOM

Some AJAX runtime toolkits use a Single-DOM approach where the toolkit uses the browser's DOM for any original source markup and any HTML+JavaScript that results from the toolkit's AJAX-to-HTML transformation logic. Figure 3 illustrates the Single-DOM Approach.

The Single-DOM approach is particularly well-suited for situations where the developer is adding islands of AJAX capability within an otherwise non-AJAX DHTML application, as the programming model matches the traditional approach used in DHTML applications.

Dual-DOM (client-side)

Other AJAX runtime toolkits adopt a Dual-DOM approach that separates the data for the AJAX-related markup into an "AJAX DOM" tree that is kept separate from the original "Browser DOM" tree. Figure 4 illustrates the Dual-DOM (client-side) approach:

With this approach, in model view controller (MVC) terms, the AJAX DOM can be thought of as the model, the Browser DOM as the generated view, and the AJAX runtime toolkit as the controller.

It is usually necessary to establish bidirectional event listeners between the AJAX DOM and the Browser DOM in order to maintain synchronization. Sometimes having a separate AJAX DOM enables a more complete set of XML and DOM support, such as full support for XML Namespaces, than is possible in the Browser DOM.

The Dual-DOM approach has two flavors: Client-side Dual-DOM and Server-side Dual-DOM. With Client-side Dual-DOM, the second DOM tree typically consists of a separate tree of JavaScript objects. With Server-side Dual-DOM, the second DOM tree is stored on the server. The following section describes Server-side Dual-DOM.

Dual-DOM (server-side)

Some AJAX technologies combine server-side AJAX transformations with a Dual-DOM approach (see Figure 5).

In this scenario, the primary job of the client AJAX engine is to reflect back to the server any interaction state changes. Most of the UI logic is handled on the server.

Server-side Dual-DOM enables tight application integration with server-side development technologies such as Java Server Faces (JSF).

Conclusion

The OpenAjax Alliance has a commitment to open technologies and architectural diversity.

A key part of the Alliance technical work are its technology initiatives focused on AJAX interoperability. The OpenAjax Hub, one of the Alliance's first interoperability initiatives, prevents collisions and enables linking together of multiple AJAX technologies within the same Web application (see sidebar).

OpenAjax Hub: Enabling AJAX Togetherness

The OpenAjax Hub is a set of standard JavaScript functionality defined by the OpenAjax Alliance that addresses key interoperability issues that arise when multiple AJAX libraries are used within the same web page. The OpenAjax Hub represents a key part of the initial technical work of the OpenAjax Alliance.

The following features have been identified for the first version of the OpenAjax Hub:

1. *AJAX library loading and unloading control - The Hub will have APIs that AJAX libraries must invoke when those libraries are initialized, identifying load and unload callback functions which are invoked at document load and unload time. The Hub manages and coordinates the invocation of these callback functions. The callback functions must be non-destructive outside of the global objects that the library has registered with the Hub (see next item). Markup handling should not occur as part of the load event callback function; instead, markup handling should occur via callback functions that are invoked by the Hub's markup scanner (see below).*
2. *Techniques and tools to minimize JavaScript object collision - The AJAX libraries must register with the Hub at initialization time a full and accurate list of all global objects and functions that they define when included in a page. The Hub will include runtime error check logic that can be turned on to verify that AJAX libraries are not colliding due to attempting to manage the same global objects.*
3. *Event hub - The event hub allows libraries to publish and subscribe to events in a reliable manner in order to coordinate actions across components from different AJAX libraries.*
4. *Markup scanner - The OpenAjax Hub controls markup scanning at the highest level. At initialization time, AJAX libraries identify to the Hub the pattern matching criteria for the markup that the given library controls. When the Hub scans the document, whenever it encounters markup that matches the criteria, it invokes the library's appropriate callback function.*

Unlocking Productivity and Flexibility Gains with OpenAjax

AJAX is an industry phenomenon. Hundreds of companies, including industry heavyweights, have announced AJAX products, and dozens of AJAX open source projects have been launched. Most Web developers say they are already using AJAX or plan to use AJAX in upcoming projects in order to achieve next-generation user experiences, improve developer productivity, and lower bandwidth requirements.

However, various challenges remain for AJAX to achieve its potential and deliver its full set of benefits. Some of these challenges are informational - providing the industry with easy access to information on how to be successful with AJAX. Other challenges are technical - today's AJAX products work well, but do not interoperate with other AJAX products. In recognition of these various challenges, leading companies and open source projects have joined forces to create the OpenAjax Alliance, whose mission is to take on these challenges to accelerate industry adoption and success with AJAX.

In this article, I provide an overview of what the OpenAjax Alliance envisions as the applications that AJAX and OpenAjax will enable and the value proposition that we want to see fulfilled as quickly and completely as possible.

More Productive End-Users, Lower BandWidth, and Strong ROI

In most businesses, decision makers are interested mainly in how information technology can increase revenue, reduce costs or make better use of information assets. Factors that are typically considered include:

1. Time spent waiting for data to be transmitted. Over many repetitions, the time employees spend waiting for pages to load can add up to significant costs.
2. Time spent completing a particular task. Inefficient user interfaces can translate into long periods of lowered end-user productivity.
3. Bandwidth consumed for the entire task. If repetitious tasks consume considerable bandwidth, IT costs can grow dramatically.

AJAX technologies help with all of the above. Because AJAX supports dynamic and continuous user experiences, user productivity increases significantly and measurably (e.g., seconds-saved-per-transaction x number-of-transactions-per-year). Because AJAX provides similar advanced user interface features to those in desktop applications (e.g., advanced UI controls, animated effects and adjustable layout controls) – thereby providing the visual and interaction tools needed to make the application self-explanatory – users spend less time learning and operating the application. AJAX's partial page update feature offers reduced communications overhead versus HTML's traditional approach of click, wait, and then full-page refresh.

Next-Generation Applications (Web 2.0 and RIAs)

Replacement for desktop applications

AJAX offers a desktop-like user experience while retaining the benefits of server-based application deployment, centralized infrastructure administration, easier scalability and immediate availability of application updates to all users. As a result, AJAX is accelerating the movement away from installable, two-tier, client-server applications to multi-tier web applications.

The runtime companion for SOA

As the natural evolution of HTML, AJAX's platform-independent runtime technology is well-suited for next-generation service-oriented architecture (SOA) applications. AJAX offers standards compliance and platform independence on the client, while SOA offers similar benefits on the server.

Mashups, dashboards and other composite applications

AJAX enables mixing and matching of multiple component technologies within the same application. This enables AJAX developers to build composite applications that leverage best-of-breed AJAX technologies from multiple suppliers, achieving many different types of composite applications, such as:

- ***Mashups*** - A mashup is a Web site or Web application that uses content from more than one source to create a completely new service. One concrete example is a custom mapping application that talks to a company's own server to pull down XML address data and then leverages mapping services (e.g., Google or Yahoo) to show a map view in combination in an application-specific custom user interface. Mashups often enable rapid application development by integrating ready-made third-party components.
- ***Portals and dashboards*** - Portals and dashboard applications consist of multiple panes that often can be configured by the user. Often, each pane is a separate application. AJAX technologies can be

used to manage the entire composite application and its integrated applications.

- ***Service compositions*** - In an SOA environment, composite business applications assemble multiple application services to create new functionality that supports innovative business processes. In some cases, the services from which the new application is composed have no user interface, so the composite application is completely responsible for the user interface. In other cases the services do have user interfaces, so that the composite application knits together not only back-end behavior components, but user interface components as well.

Collaboration

AJAX also enables collaborative applications, such as:

- Next-generation chat and instant messaging (IM) - AJAX enables Web-based messaging applications that run in the browser.
- Web meetings and whiteboarding - AJAX's rich user interface, network programming, and vector graphics features provide the infrastructure for browser-based Web meetings and shared whiteboards.
- Collaborative content creation - Web-based collaborative applications such as wikis can transition from users having to learn arcane wiki markup languages to a WYSIWYG user experiences, comparable to what users experience today with desktop content creation tools.
- Trip planning - A social group can use an AJAX-powered application to work together to plan and schedule their activities.
- Photo sharing - AJAX enables richer sharing of experiences, such as attaching annotations, tagging or marking up photos.

Collaboration capabilities are based on XMLHttpRequest and other network communications technologies.

AJAX-powered wikis

An important new application area is the AJAX-powered wiki, where wikis go beyond text-based collaborative documents into the following scenarios:

- ***Rich, lightweight portals*** – AJAX-powered wikis enables fast deployment of Enterprise rich dashboards by allowing a wiki page to contain both text content and AJAX-powered user interface components, such as data grids, forms fillout, mapping, and charting.
- Personal mashups and dashboards – Many leading Internet companies provide the option today for customized home pages. When personalization is paired with AJAX-powered components and wiki back-end services, IT gains easy deployment and end-users gain the ability to manage their view on information.

Cross-device applications (desktop and mobile)

AJAX offers multiple approaches to achieving cross-device applications. This includes:

- ***Using AJAX desktop technologies on high-end mobile devices*** - Today, some mobile devices offer mobile Web browsers that support the same feature set (HTML, JavaScript, etc.) as desktop mobile browsers. Examples include the Opera browser and Nokia's mobile browser, which is built from the same KHTML/WebKit code base as the Apple Safari browser. Web developers can reach this subset of mobile devices using the same AJAX source code as they use for desktop AJAX. Over time, as mobile devices become more powerful, increasingly larger percentages of mobile devices will ship with Web browsers that offer full desktop AJAX support.
- ***Using a mobile subset of AJAX*** - The W3C and the Open Mobile Alliance (OMA) are working together to standardize appropriate mobile subsets of XHTML, SVG, CSS and ECMAScript to take into account the constraints of today's mobile devices, and combine them together to form a mobile standard for rich content using AJAX.
- ***Using a server-side, multi-target AJAX toolkit*** - To reach a large number of mobile devices, some of which ship with more limited features sets, Web developers can take advantage of AJAX toolkits that provide a cross-platform abstraction layer. These toolkits adapt AJAX source into appropriate client-side instructions, such as mobile subsets of HTML+JavaScript, mobile SVG, or J2ME.

Conclusion

AJAX offers great promise, but challenges remain before AJAX can realize its full potential and deliver the benefits of Web 2.0. The OpenAjax Alliance has been established to address those challenges and accelerate industry success with AJAX-powered, next-generation rich applications. Target applications include replacements for desktop applications, SOA applications, composite applications, collaboration tools, next-generation wikis and cross-device applications. ■

Jon Ferraiolo joined IBM's Emerging Technologies group in May 2006 to play a leadership role in the OpenAjax initiative. Before IBM, Jon worked at Adobe for 13 years where he was an architect, engineering manager and product manager on multiple products and participated in various document format standards activities, including editorship of the SVG 1.0 specification. jferrai@us.ibm.com

Building a Drag-and-Drop Shopping Cart with AJAX

Creating an interactive shopping experience

Joe Danziger

Keeping up with the latest Web technologies is tough nowadays. Every week it seems new sites are launched that push the envelope further and further in terms of what can be accomplished using just a Web browser.

The rise of AJAX over the past several months has taken over the development world and breathed new life into the Web. Although these techniques have been possible for many years now, the maturity of Web standards like XHTML and CSS now make it a viable alternative that will be viewable by all but the oldest browsers.

It's also been possible to accomplish many of the same things using Flex or Flash, but the development cycle with those applications is typically more involved and the overhead often not justified.

We're going to harness the power of the Scipt.aculo.us JavaScript library to provide our interaction. As their Web site states, this library "provides you with easy-to-use, compatible and, ultimately, totally cool JavaScript libraries to make your web sites and web applications fly, Web 2.0 style." We're also going to utilize the <CF_SRS> library to handle the actual AJAX data piping to our application. Both of these libraries are free for all to use, and they're easier to integrate than you would think.

For this article, we'll create an interactive shopping experience allowing us to add items to our shopping basket by dragging and dropping them onto an icon of a shopping cart. We'll add AJAX functionality, allowing us to update our shopping cart without redrawing the entire screen. To save the trouble of setting up a product database, we'll use Amazon Web Services to search for DVDs and use those to shop from.

Start with a blank index.cfm in your root directory. You'll need to visit http://script.aculo.us/downloads to download the latest distribution (they're nearing a final release for version 1.5 as of this writing). Copy the "lib" and "src" directories into your empty directory. You'll need all of the .js files so just copy over the entire directory in each case. Next, type the following lines into the <head></head> section of your page:

```
<script src="./lib/prototype.js" type="text/javas-
cript"></script>
<script src="./src/scriptaculous.js" type="text/
javascript"></script>
```

We'll need a search box to submit our query to Amazon:

```
<form action="index.cfm" method="post">
Search: <input type="text" name="keywords" size="20"
/>
<input type="submit" name="search" value="Go" />
</form>
```

The page will look for a form.search variable and run an Amazon search when it is defined. Each item returned will be placed in its own styled div that will be able to be picked up and dragged.

The Scriptaculous library makes it easy to create "draggables" (the only required argument is the ID of the object that you want draggable). Listing 1 contains the code to search Amazon and return the results as draggable divs.

At this point, all of the items returned from the search will be in their own box and should be draggable around the screen. When we created each draggable, we set "revert=true", which will snap each object back to its original location if not placed directly on a drop zone.

Next, we'll add a graphic of a shopping cart to our page, which will become a drop target on which to drag items. The Scriptaculous library also makes it easy to create these "droppables". The syntax is simply:

```
Droppables.add('id_of_element',[options]);
```

The code below creates a droppable zone of id "cart1" and also runs a function onDrop() that pops up an alert box letting the user know an item has being added. We then hide the element from view, which allows the other divs to slide over and adjust accordingly.

```
<img src="shopcart.jpg" id="cart1" style="float:
left;">
<script language="javascript" type="text/javas-
cript">
Droppables.add('cart1', { onDrop:function(element) {
alert('Added UPC ' + element.id + ' to your shop-
ping cart.');
Element.hide(element.id);}})
</script>
```

The items should now be disappearing when dropped onto the shopping cart, but there's nothing going on behind the scenes yet. Now it's time to add some AJAX to process our shopping cart.

Although there are several AJAX libraries to choose from, we're going to use the ColdFusion Simple Remote Scripting <CF_SRS> library made available free of charge by Matthew Walker of ESWsoftware in New Zealand. <CF_SRS> uses an IFRAME for communication and encapsulates all of the dirty work for you. This library was chosen for its ability to handle HTML tables well and for its ability to interact directly with the browser's Document Object Model (DOM) to output our shopping cart rows.

We'll start with an empty cart by including the following code:

```
<fieldset style="width:400px;">
<legend>Your shopping cart</legend>
<table border="0" cellspacing="0" cellpadding="5"
id="tableCart">
<thead></thead><tbody></tbody></table>
<button onclick="emptyCartButton_onClick()" id="empt
yCartButton">Clear Shopping Cart</button>
</fieldset>
```

(Don't worry about the fact that our table body (<tbody></tbody>) is empty right now - we'll be populating it in just a second through AJAX.)

Next, you'll need to download the <CF_SRS> package from http://www.eswsoftware.com/products/download/. Copy the srs.cfm file into your Webroot (or you can add it to your CustomTags directory if you plan to do more AJAXing). You'll also need to create a subdirectory to hold the gateway pages that handle our AJAX data passing. Name the directory "SRS" and copy the Application.cfm and OnRequestEnd.cfm files into there from the "serverpages" directory in the zip file. You can use either regular CFM files or CFCs for these gateway pages (the download provides examples of each). The main thing to remember is that these pages should always return their results to "request.response".

Simply adding a <cf_srs> call to your page will handle the creation of the hidden IFRAME for you. Another great feature of the CF_SRS library is the ability to view an inline debugging window right inside the page you are working on. This allows you to see all of the data being passed back and forth through the gateway. You can enable this debugging by calling the tag as <cf_srs trace>. This line can be placed anywhere but we'll add it at the very end of the file.

Next, we'll need to create some JavaScript functions to handle the AJAX interactions. Add an onLoad function to your body tag as such: <body onload="body_onLoad()">. This will execute body_onLoad() when the page loads and we'll use this

Listing 1: Run Amazon search and return results:

```
<cfif isDefined("form.search")>
<!--- Submit REST query --->
<cfhttp url="http://webservices.amazon.com/onca/xml?Service=AWSECommerceService&AW
SAccessKeyId=#REPLACE_WITH_YOUR_KEY#&Operation=ItemSearch&Keywords=#form.keywords#
&SearchIndex=DVD&Sort=relevancerank&ResponseGroup=Small,ItemAttributes,Images,Offe
rs" throwOnError="yes" charset="UTF-8"></cfhttp>
<!--- parse XML document --->
<cfset myXMLdoc = XmlParse(cfhttp.filecontent)>
<CFSET xnSearch = myXMLdoc.xmlRoot>

<cfoutput>
<!--- if results LT 10, loop through those, otherwise show first 10 --->
<cfloop index="i" from="1" to="#IIF(xnSearch.Items.TotalResults.XMLText gt
10,'10',xnSearch.Items.TotalResults.XMLText)#">

<!--- add cftry/cfcatch in case no image or other error // we'll use UPC as div
id --->
<cfif xnSearch.Items.Item[i].Offers.TotalOffers.XmlText gt 0>
<div style="font-size:10px;font-family:Arial;text-align:center; background-
color:##eee;float:left;margin:10px; border:1px solid ##666; padding:5px;"
id="#xnSearch.Items.Item[i].ItemAttributes.UPC.XmlText#">
<img src="#xnSearch.Items.Item[i].MediumImage.URL.XmlText#"><br />
#xnSearch.Items.Item[i].ItemAttributes.Title.XmlText#<br />
#xnSearch.Items.Item[i].ItemAttributes.UPC.XmlText# &nbsp;&nbsp;&nbsp;
#xnSearch.Items.Item[i].ItemAttributes.ListPrice.FormattedPrice.XmlText#
</div>

 <!--- make this item draggable // reference by UPC as div id --->
<script type="text/javascript">
new Draggable('#xnSearch.Items.Item[i].ItemAttributes.UPC.XmlText#', {revert:
true});
</script>
</cfif>
</cfloop>
</cfoutput>
<br clear="all" />
</cfif>
```

Listing 2

```
function cartPacket_onReceive(packet) { // generates an HTML table of cart items

 var theTable = document.getElementById('tableCart');
 var tbody = document.createElement("tbody");
 var tr, td;

 if ( packet.upc.length == 0 ) { // if no results just send back empty cart
  tr = document.createElement("tr");
  td = document.createElement("td");
  td.colSpan = 4;
  td.innerHTML = "Your cart is empty.";
  tr.appendChild(td);
  tbody.appendChild(tr);
 }
 else
  for ( var i = 0 ; i < packet.upc.length; i++ ) { // loop and create new row
   tr = document.createElement("tr");
   td = document.createElement("td");
   td.innerHTML = packet.qty[i];
   tr.appendChild(td);
   td = document.createElement("td");
   td.innerHTML = '<img src="' + packet.image[i] + '">';
   tr.appendChild(td);
   td = document.createElement("td");
   td.innerHTML = packet.title[i];
   tr.appendChild(td);
   td = document.createElement("td");
```

function to set up our gateway. The function should read as follows:

```
function body_onLoad() {
// create an SRS gateway to the cart.cfm page
objGateway = new gateway("srs/cart.cfm?");
// update cart in case of return visit
// code for this function is below
updateCart();
}
```

Once you have created your gateway, you can invoke the methods below to send requests to the server:

- ***objGateway.setListener(str):*** Use this method to specify the name of the function in your Web page that will handle the server's response. str is a string representing the function's name. The listener defaults to "alert", which will pop up a JavaScript alert() box containing the server's response. Note that while ColdFusion is a case-insensitive language, JavaScript is case-sensitive. If you return a structure to your listener function, all the structure keys will be rendered in JavaScript as lowercase.
- ***objGateway.setArguments(obj):*** Set the arguments and values to pass to the server. obj is an object literal, which is basically just a set of one or more attribute/value pairs wrapped in curly braces. Here's an example: { name:'Joe', age:30, country:'US' }. You can see that string values need to be wrapped in quotation marks, and colons (:) are used in place of equals signs (=).
- ***objGateway.resetArguments():*** Remove all the arguments previously set.
- ***objGateway.request():*** Send the request to the server.

Note that you can chain these methods together. For example, it is perfectly acceptable to write:

```
 objGateway.resetArguments().setArguments( {
state:'NY' } ).request()
```

Using what we know now, let's take another look at our updateCart() function that we're calling onLoad.

```
function updateCart() {      objGateway.
setListener('cartPacket_onReceive').setArguments(
{action:'getCart'} ).request(); }
```

The function chains together several commands. It sets the listener to "cartPacket_onReceive". That means that we'll execute this JavaScript function whenever data is returned from our gateway. This function handles the generation of our table body that contains our cart rows (see Listing 2).

In our updateCart() function, we're also passing in an argument: action=getCart. This is going to be passed through to our cart.cfm gateway page. The full text of the gateway page is displayed in Listing 3.

We're passing in the action variable with a value of "getCart". This gets passed to our cart.cfm gateway page and causes the user's session cart to be returned as a query object. Whenever we need to update our cart to add or delete rows, we'll set our listener to 'cartPacket_onReceive' and then redraw the table body.

When we created our shopping cart on screen, we added the following button to clear our cart:

```
<button onclick="emptyCartButton_onClick()" id="empt
yCartButton">Clear Shopping Cart</button>
```

We'll add two JavaScript functions to go along with that button. The first will confirm the delete and the second will issue a call to remove the items and redraw the cart:

```
function emptyCartButton_onClick() {
 if ( confirm('Are you sure you want to empty your
cart?') ) clearCart();
}

function clearCart() {
 objGateway.setListener('cartPacket_onReceive').
setArguments( {action:'clearCart'} ).request();
}
```

Finally one more JavaScript function to be called when adding items to our cart:

```
function addToCart(upc) {
 objGateway.setListener('cartPacket_onReceive').
setArguments( { action:'addToCart',upc:upc }
).request();
}
```

Now that we have our addToCart() function coded, add the line "addToCart(element.id);" right before the Element.hide call in the shopping cart droppable. This will execute our addToCart() function and redraw the shopping cart when an item is dropped onto it.

And that's all there is to it! With just 150 lines of code, we were able to create an interactive, drag-and-drop shopping experience that many did not think was possible using just the browser. ■

Joe Danziger is the founder and president of DJCentral.com, an online promotional tool for disc jockeys and other members of the electronic dance music industry. He has been developing professional ColdFusion solutions for over six years since version 1.5.
danziger@yahoo.com

```
  td.innerHTML = packet.price[i];
  tr.appendChild(td);
  tbody.appendChild(tr);
 }

 theTable.replaceChild(tbody, theTable.childNodes[1]);
}
```

Listing 3: cart.cfm gateway page:

```
<cfswitch expression="#action#">

<cfcase value="getCart">
 <cfparam name="session.cart" default="#ArrayNew(1)#">
 <cfset cartQuery = QueryNew("qty,upc,title,image,price")>
 <cfloop from="1" to="#ArrayLen(session.cart)#" index="i">
  <cfset queryAddRow(cartQuery)>
  <cfset querySetCell(cartQuery, "qty", session.cart[i].qty, i)>
  <cfset querySetCell(cartQuery, "upc", session.cart[i].upc, i)>
  <cfset querySetCell(cartQuery, "title", session.cart[i].title, i)>
  <cfset querySetCell(cartQuery, "image", session.cart[i].image, i)>
  <cfset querySetCell(cartQuery, "price", session.cart[i].price, i)>
 </cfloop>
<cfset request.response = cartQuery>
</cfcase>

<cfcase value="addToCart">
<cfparam name="session.cart" default="#ArrayNew(1)#">

<!--- get items from Amazon Web Services -‡
<cfhttp url="http://webservices.amazon.com/onca/xml?Service=AWSECommerceService&A
WSAccessKeyId=1MEQ9VMKAJS5A8DSHER2&Operation=ItemLookup&IdType=UPC&SearchIndex=DV
D&ItemId=#url.upc#&ResponseGroup=Small,ItemAttributes,Images" throwOnError="yes"
charset="UTF-8"></cfhttp>
<cfset myXMLdoc = XmlParse(cfhttp.filecontent)>
<CFSET xnProduct = myXMLdoc.xmlRoot>
<cfscript>
addItem = StructNew();
addItem.qty = 1;
addItem.upc = xnProduct.Items.Item.ItemAttributes.UPC.XmlText;
addItem.title = xnProduct.Items.Item.ItemAttributes.Title.XmlText;
 addItem.image = xnProduct.Items.Item.SmallImage.URL.XmlText;
 addItem.price = xnProduct.Items.Item.ItemAttributes.ListPrice.FormattedPrice.
XmlText;
</cfscript>
<cfset ArrayAppend(session.cart,addItem)>

<!--- return cart --->
<cfset cartQuery = QueryNew("qty,upc,title,image,price")>
<cfloop from="1" to="#ArrayLen(session.cart)#" index="i">
  <cfset queryAddRow(cartQuery)>
  <cfset querySetCell(cartQuery, "qty", session.cart[i].qty, i)>
  <cfset querySetCell(cartQuery, "upc", session.cart[i].upc, i)>
  <cfset querySetCell(cartQuery, "title", session.cart[i].title, i)>
  <cfset querySetCell(cartQuery, "image", session.cart[i].image, i)>
  <cfset querySetCell(cartQuery, "price", session.cart[i].price, i)>
</cfloop>
<cfset request.response = cartQuery>
</cfcase>

<cfcase value="clearCart">
 <cfset ArrayClear(session.cart)>
 <cfset cartQuery = QueryNew("qty,upc,title,image,price")>
 <cfset request.response = cartQuery>
</cfcase>

</cfswitch>
```

From 'View Source' to Open Source

AJAX and the maturation of Web development

John Eckman

From the beginning, the World Wide Web that Tim Berners-Lee imagined was a place where the architecture of participation ruled. Berners-Lee's first application for accessing the information Web was both a browser and an editor, and throughout the early 1990s he worked diligently to encourage Web browser development groups to develop editors and servers as well as browsers. As early as the spring of 1992, the challenge was clear: "Although browsers were starting to spread, no one working on them tried to include writing and editing functions….As soon as developers got their client working as a browser and released it to the world, very few bothered to continue to develop it as an editor" (*Weaving the World Wide Web* by Tim Berners-Lee).

Developers tended to defer the editing functionality for a number of reasons, mostly to compress development schedules and to get the browser out the door - because they felt many people, if they didn't need it, would at least use it - without the editor, which was more complex and useful to a smaller audience. Netscape Communicator 4.0, released in 1997, did finally include Netscape Composer, although its licensing terms only allowed free use for non-commercial purposes. Internet Explorer never contained an editor directly, though Microsoft acquired FrontPage from Vermeer in 1996; FrontPage 1.0 had been released in 1995 (http://www.seoconsultants.com/frontpage/history/).

It wasn't just the complexity of the editing functions themselves, of course, but also the fact that reading pages required a much simpler authorization model, in which the user either has access to the document or does not. In fact, the cluster of issues at hand - from version control of Web pages to multiple authors editing the same page, sometimes at the same time, to control over who should have access to change what pages - would busy the content management industry for the better part of the next decade.

While the early Web browser teams deferred creation of an HTML editor, they retained a key element of Sir Berners-Lee's original Web browser/editor:

The 'View Source' menu item migrated from Tim Berners-Lee's original browser, to Mosaic, and then on to Netscape Navigator and even Microsoft's Internet Explorer. Though no one thinks of HTML as an open source technology, its openness was absolutely key to the explosive spread of the Web. Barriers to entry for "amateurs" were low, because anyone could look "over the shoulder" of anyone else producing a Web page ("The Architecture of Participation" by Tim O'Reilly).

This "View Source" menu item, which was not buried in developer editions or professional versions but was part of the core browser, created a culture of easy access to knowledge.

As the complexity of presentation-tier Web development grew, with Cascading Style Sheets, JavaScript, and DHTML in the mix, the View Source culture of Web development evolved into an open source culture of frameworks and libraries. It is this culture that enables the viability of current AJAX-based approaches to Web development.

The View Source Culture

From a narrow perspective, the decision to include a View Source option in the Web browser was an insignificant choice, perhaps useful for troubleshooting formatting issues, but of interest to a very small community. As Berners-Lee puts it, "I never intended HTML source code (the stuff with the angle brackets) to be seen by users. A browser/editor would let a user simply view or edit the language of a page of hypertext, as if he were using a word processor. The idea of asking people to write the angle brackets by hand was to me, and I assume to many, as unacceptable as asking one to prepare a Microsoft Word document by writing out its binary coded format" (Weaving the World Wide Web by Tim Berners-Lee). What View Source did (and still does!) was let users who were interested in learning to create Web pages see what HTML source was delivered to the browser to produce the page currently being rendered. Perhaps because many of the early Web users were developers of one kind or another, it became an expectation that any reasonable browser would include the ability to View Source.

Viewed more broadly, however, the View Source command was nothing short of revolutionary. It set the expectation that users should be able to not only view the "rendered" document, but also the "code" that created it. Because early browsers often differed in their interpretation of HTML, this was critical. Significantly, though, the View Source option was not buried in a developer's edition but was part of the edition everyone used, which encouraged even neophyte users to view the source of pages, whereupon they would see the relative simplicity of (especially early) HTML. (An interesting discussion about the need for the View Source option can be found in this bug report: https://bugzilla.mozilla.org/show_bug.cgi?id=256213 – which was a request to move View Source into a developer build of Firefox, and was ultimately rejected.)

This same expectation - that users should be able to view the raw source of files served by Web servers in addition to the rendered effect - was later extended to Cascading Style Sheets (.css files) and JavaScript (.js). In order to be rendered and displayed, browsers would need to download HTML, CSS, and JavaScript files, along with images and other binary files referenced in pages. However, the fact that the major browser developers chose to expose access to raw source as a first-level menu item was extraordinary. (In some cases accessing .css and .js files required a bit more ingenuity on the user's part, but nothing like the difficulty of accessing the source files in any other format such as Microsoft's Word or Adobe's PDF.)

The View Source option was, however, perfectly consistent with the culture of the Web as a whole. Certainly, as I've said above, this was consistent with Tim Berners-Lee's vision of the information Web in which all could participate, and in the choice of HTML as an immensely simplified version of SGML. One could argue that the View Source option simply reinforced the already "open" nature of the Web as a phenomenon, which was firmly in place before the first browser was ever made widely available.

Regardless of whether the View Source option set the tone for the culture or merely reflected it, it is fair to characterize the culture of Web development that came into being as a View Source culture. In the early years of the Web, people learned HTML by example. When you saw a site that was doing something interesting, you would view the source of that site, and (often quite directly) copy their code into your own pages and edit from there. It became quite customary for such "imitation" (which was in some cases arguably copyright infringement by modern definitions) to occur even without the polite inquiry that originally preceded it. At first people asked, "Can I borrow your Web page template? I love what you've done with tables," but over time it became so customary people didn't even ask. People did continue to ask for access to server-side features, like Perl scripts for the common gateway interface, but perhaps due to Perl's heritage (version 3.0, in 1989 had been under GPL, even before Linux's public release), such scripts were commonly shared as well.

As the Web bubble grew, people started to offer well-designed HTML/CSS templates; useful JavaScript snippets; cascading drop-down, slide-out menu frameworks; scrolling marquees; and the like. Some of these libraries and templates were offered as freeware or shareware; some were sold as commercial software. Of course, developers couldn't easily prevent end users with the necessary JavaScript, CSS, and HTML in their browser cache from "borrowing" their scripts – it was very common for such developers to find unlicensed versions of their code on Web pages from around the globe - but people had begun to realize that there was some value in the effort involved in making a useful function work as a cross-browser. In fact, one of the purported advantages of Java applets and Flash movies, when they first arrived, was the fact that they closed this loophole and were emphatically not View Source enabled.

Early efforts, however, to leverage JavaScript and CSS to provide a more compelling browser experience – what we then called DHTML – were hampered by the complexity of developing for multiple browsers, each of which had their own interpretation of the Document Object Model and implementation

John Eckman leads the Next Generation Internet Practice at Optaros (www.optaros.com), and has over a decade of experience designing and building Web applications for organizations ranging from small non-profit organizations to Fortune 500 enterprises. John´s technical background includes J2EE and .NET frameworks as well as scripting languages and presentation-tier development approaches, in addition to information architecture, usability testing, and project management. He received a BA from Boston University and a PhD from the University of Washington at Seattle; he expects to complete an MS in Information Systems from Northeastern University in 2006. He also achieved PMP certification in 2003.

of Cascading Style Sheets. Two major events provided a way out of this complexity: the recognition of a community of practices now referred to as AJAX (itself enabled by the release of compelling competitors to Microsoft's Internet Explorer) and the appearance of a number of open source frameworks for developing AJAX applications.

AJAX

In what is now a seminal article, Jesse James Garrett noted, in February of 2005, a growing phenomenon of using standards-based presentation (XHTML and CSS), the Document Object Model (DOM), the XMLHttpRequest, and JavaScript to generate rich interface Web-based applications. He called this approach AJAX, and issued a call to designers and developers "to forget what we think we know about the limitations of the Web, and begin to imagine a wider, richer range of possibilities" ("AJAX: A New Approach to Web Applications" by Jesse James Garrett).

While many of the raw materials of the AJAX approach had been available for many years – Microsoft first introduced the XMLHttpRequest object in Internet Explorer 5, and it was leveraged in production by Outlook Web Access – it only took off when the approach was validated by inclusion in the Mozilla framework (and thus Firefox) and Apple's Safari browser (itself based on KHTML, the framework used for Konqueror in the K Desktop Environment for Linux). So long as what Microsoft called "remote scripting" was only available in a single browser (Internet Explorer), many developers felt it was unusable in any public application. (It might be acceptable, for some, to dictate browser usage on a corporate intranet, but most were unwilling to try to do so on the public Internet.)

As the user base of the old Netscape 4.x applications waned, and the availability of new platforms rose, developers were emboldened by the relative ease of developing cross-browser standard applications with rich interaction in JavaScript and CSS, and more and more developers started to create AJAX applications. Garrett's essay crystallized this movement and gave it a name. The AJAX meme spread like wildfire, and the essay became so influential, exactly because so much work of this kind was already being done.

Open Source Frameworks

In addition to the AJAX phenomenon, the other major change that supported the new explosion of activity around Web development was the emergence of open source frameworks for AJAX-style development. While the diminishing use of Netscape 4.x (specifically, the acceptance among developers and their clients of the decision to not include complete functionality in Netscape 4.x as a primary requirement) and the relative stability of standards for HTML, CSS, and JavaScript all combined to make developing cross-browser Web applications considerably easier, writing such applications from scratch still required too much repetitive effort. Though the Mozilla-based platforms had adopted an approach similar to Microsoft's remote scripting, there were enough differences to create a barrier to cost-effective and reproducible development.

The frameworks that evolved abstracted away the messy details (the difference between Internet Explorer's ActiveX-based XMLHttpRequest object and Mozilla's true JavaScript version, for example) and enabled developers to add snazzy AJAX functionality to their Web applications simply by leveraging APIs provided by the framework.

In addition to ease of use, broad adoption of these frameworks has brought along with it an open source culture, characterized by an emphasis on community, de jure licensing, and more sophisticated planning and architecture for the presentation-tier of Web applications. (While the definition of open source typically focuses specifically on what licenses qualify, these licenses carry with them a set of cultural impacts, just as the View Source command brought with it a set of cultural impacts.)

First, and most obviously, an open source culture is characterized by explicit license rather than borderline theft or gift of use. While the View Source culture sometimes included explicit permission ("Can I copy what you've done here?"), it more often relied on a kind of borrowing that ranged from imitation to outright copyright infringement or theft. An open source culture relies not only on a kind of de facto or assumption-based sharing of information, but a formally stated, de jure grant of specific rights to all users. This explicit license also facilitates innovation, as improvements made by users of the code can be contributed back into the community and the benefits of those innovations shared with other users directly.

Additionally, an open source culture is characterized by communities that develop around the frameworks. These communities have differing levels of formality, professionalism, and "helpfulness," but all represent a great leap forward compared to the random deciphering of other people's code, which characterized the View Source culture. To revisit Tim O'Reilly's metaphor about "View Source," in an open source culture, a would-be developer cannot only look over the shoulder of other developers, but he or she can join them in a conference room and discuss the

OpenLaszlo
Advancing the Web Experience™

Hit the road to freedom with OpenLaszlo, the only open source advanced Ajax development platform that lets you choose between Flash® and DHTML for running your applications.

Open Source:
Create and control cost-effective, mission-critical web applications

Open Standards:
Build web applications with the most mature and advanced AJAX (JavaScript and XML) development platform

Open Architecture:
Designed to support multiple runtimes, including Flash, DHTML and embedded devices (mobile and TVs)

Laszlo®

Download today at www.openlaszlo.org

Framework	License	Notes
Prototype.js	*MIT Style (slightly modified)*	*Server-Side Platform independent Fully integrated into the Ruby on Rails framework as well as Symfony framework for PHP*
Script.aculo.us	*MIT*	*Builds on Prototype.js*
RicoApache 2.0	*Builds on Prototype.js*	
Dojo	*Dual: Academic Free 2.1 and BSD*	*Server-Side Platform independent Dojo foundation is supported by IBM, OpenLaszlo, AOL, JotSpot and others*
YUI (Yahoo User Interface Elements)	*BSD*	*Utilities and Controls (Widgets) in JavaScript, css*
ZimbraTk	*Dual: Mozilla Public or Apache*	*Client Developer Library – Widgets, Controls in JavaScript*

"I have always imagined the information space as something to which everyone has immediate and intuitive access, and not just to browse but to create."

–(*Weaving the World Wide Web* by Tim Berners-Lee)

code directly. Open source culture also helps to ensure repeatability and maintenance of the leveraged code base because when newer versions or patches are released, there is a well-organized mechanism for announcing and providing access to the code, something that the View Source culture generally lacked.

Finally, an open source culture is characterized by a greater degree of attention to standards and interoperability. Because open source projects gain their strength from the breadth of their use and the size of their community, they tend to focus much more clearly on methods of sharing. Even in cases where an individual's needs are contrary to the direction of the overall project, open source licensing allows for and encourages the development of extensions and alternate versions to meet specific problem sets.

Community and Professionalism

In some ways, one could point out the essential difference between the Web development culture, which relies heavily on open source frameworks and libraries, and the first generation of Web development, which relied on View Source and imitation, in the same way that free / open source software advocates have long distinguished between them: by drawing the "free as in beer" versus "free as in speech" (or "free as in freedom") comparison.

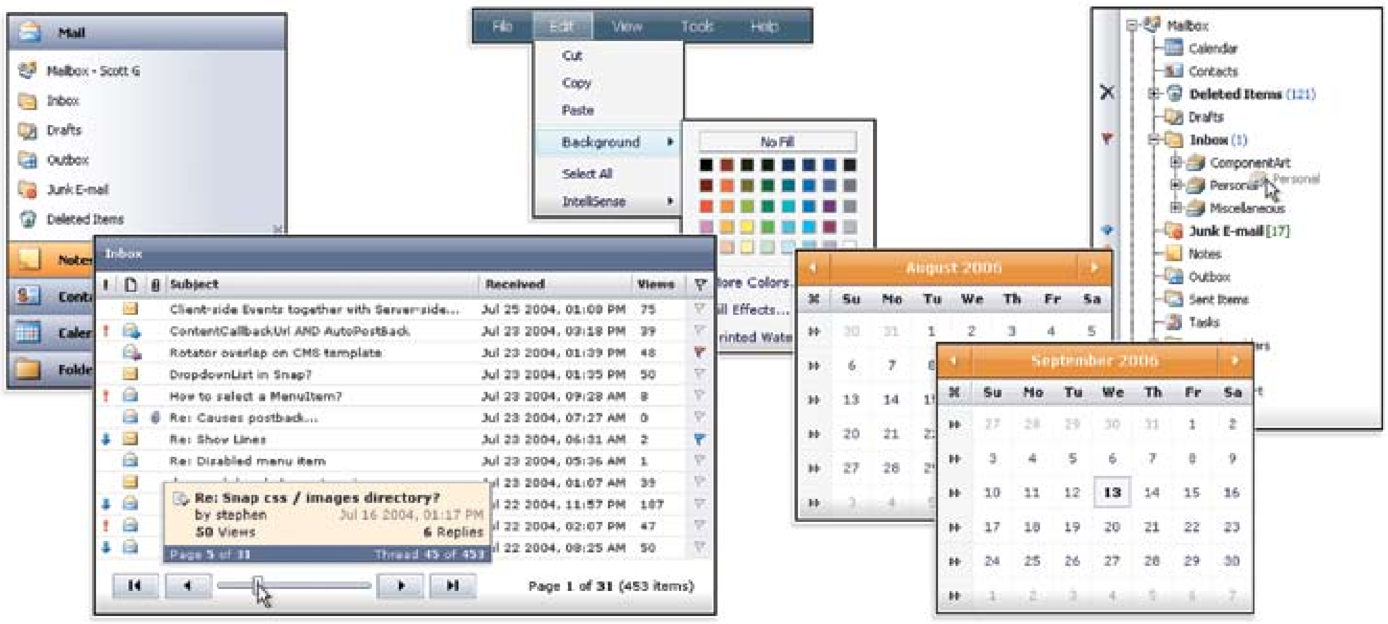
Viewing the source of a Web page developed by someone else, including digging into the CSS and JavaScript that accompanies the HTML, has always been free, as in beer. If you have access to a Web browser and can display the page, you can get to the source. Open source frameworks like Prototype.js, Dojo, DWR, the YUI library, and the Zimbra toolkit, however, are free as in speech. Not only can you access the source code without cost, you are encouraged and explicitly granted the right to use, modify, and redistribute your modifications to others.

The early days of the World Wide Web were characterized by open experimentation and the de facto sharing of source, whereas the trend today is toward maturity, professionalism, and the de jure sharing of open source frameworks. This evolution is a sign of the growing professionalism of the Web development community.

While the current AJAX-style development community faces many ongoing challenges - working out JavaScript namespace issues, encouraging better adherence to security best practices, and dealing with accessibility issues are at the top of a substantial list – the broad, active, and transparent communities behind open source AJAX frameworks bode well for the possibilities of solving such challenges and continuing the evolution of Web development. ■

AJAX Web User Interfaces, evolved.

The most advanced set of AJAX-enabled UI controls for ASP.NET and "Atlas"



Subject	Received	Views
Client-side Events together with Server-side...	Jul 25 2004, 01:09 PM	75
ContentCallbackUrl AND AutoPostBack	Jul 23 2004, 03:18 PM	39
Rotator overlap on CMS template	Jul 23 2004, 01:39 PM	48
DropdownList in Snap?	Jul 23 2004, 01:35 PM	50
How to select a MenuItem?	Jul 23 2004, 09:28 AM	8
Re: Causes postback...	Jul 23 2004, 07:27 AM	0
Re: Show Lines	Jul 23 2004, 06:31 AM	2
Re: Disabled menu item	Jul 23 2004, 05:36 AM	1
	l 23 2004, 01:07 AM	39
	l 22 2004, 11:57 PM	187
	l 22 2004, 02:07 PM	47
	l 22 2004, 08:25 AM	50

ComponentArt

Web.UI™ for ASP.NET

TRY WEB.UI DEMOS TODAY AT WWW.COMPONENTART.COM

Web.UI combines ComponentArt's patent-pending rendering technology with the power of AJAX to deliver the richest set of user interface controls available for ASP.NET. The latest version features an unsurpassed level of client-side programmatic control, along with extensive support for ASP.NET 2.0 and Microsoft's forthcoming "Atlas" framework.

ComponentArt
www.componentart.com

Intelligent Web Applications with AJAX

A peek into modern technologies for browser-based applications

Victor Rasputnis Igor Nys

Anatole Tartakovsky

Browser-based applications are widely used and we like the fact that we can access them from anywhere. But from the users' perspective, the productivity level of Web applications still doesn't approximate the productivity of desktop programs. The good news is the gap is closing: the accumulated potential of multiple technologies has boosted a whole new breed of HTML-based apps that are as powerful as the desktop ones. Meet AJAX.

What Is AJAX?

The name stands for Asynchronous JavaScript + XMLHTTPRequest and means you can establish socket communication between browser-based JavaScript and the server. AJAX isn't a new technology. It's a successful branding of possibilities implanted in several technologies available in modern browsers. All AJAX applications deliver a rich UI via extensive JavaScript manipulation of the HTML Document Object Model based on the precision-pointed data retrieval via XMLHttpRequest. Typical examples of AJAX applications are Google Maps and Google Suggest from Google Labs (http://labs.google.com). These applications actively monitor user input and provide real-time page updates. Most importantly, this happens without a page refresh while the user navigates through the map or types a search string.

In fact, the technologies behind these wonders have been around for a while, although they require sophisticated skills in using browser-specific tricks. Proprietary offerings with similar capabilities - Macromedia Flash plug-in, Java Applets or .NET runtime - have been available for quite some time too. The idea of integrating a scriptable transport component talking to the server into the browser was pioneered by IE 5.0. Then Firefox and other popular browsers joined the club of browsers in supporting XMLHTTPRequest as a built-in object. With cross-browser availability, these technologies gained visibility and in March of 2004 a company called Adaptive Path introduced AJAX.

In short, backing from Google and having the right browser technologies available out-of-the-box tipped the scale: these days everyone is adding client-side technologies to Web applications for a "better user experience."

AJAX vs. Classical Applications

A classic Web application model is literally a triumph of form over substance: users are forced to submit forms in exchange for pages. That said, the form submission and page delivery aren't guaranteed: worse case the user clicks again though some pages specifically warn against that. It's quite different with AJAX, where the data travels across the wire instead of entire HTML pages. This data exchange is scripted via a specific browser object - XMLHttpRequest; the appropriate logic handles the outcome of each data request, the specifc region of the page is updated instead of the entire page. The results are more speed, less traffic, and better control of information delivery.

Traditional “click-refresh” Web applications force users to interrupt the work process while waiting for the page to reload. With AJAX, a client-side script can asynchronously talk to the server while the user keeps entering data. Besides being transparent to the user, such asynchrony means more time for the server to process the request.

Classic Web applications delegate all processing to the server and force the server to manage the state. AJAX allows flexible partitioning of the application logic and state management between the client and the server. This eliminates a “click-refresh” dependancy and provides better server scalability. When the state is stored on the client-side you don’t have to maintain sessions across the servers or save/expire state: the lifespan is defined by client.

AJAX: Distributed MVC

Although AJAX applications rely on JavaScript for the presentation layer, the processing power and knowledge base remain on the server. For that matter, AJAX applications talk heavily to J2EE servers, feeding data to and from Web Services and servlets. The difference between J2EE applications with an AJAX-based presentation tier and standard J2EE application is that in the first case MVC is distributed over the wire. With AJAX, View is local, while Model and Controller are distributed giving the developer the flexibility to decide which components will be client-based. Specifically, a local View renders graphics by manipulating with HTML DOM; the controller handles user input locally and at the developer’s discretion extends the processing to the server via HTTP requests (Web Services, XML/RPC or others); the remote part of the Model is downloaded as needed to the client achieving in-place real-time updates of the client page; and state is collected on the client.

In future AJAX articles we’ll talk about each of these components in depth and provide examples of how they came to play together. Now, without further ado, let’s dive into a simple AJAX example.

Zip Codes Validation and Lookup

We’ll create an HTML page containing three INPUT fields: Zip, City, and State. We’ll make sure that as soon as the user enters the first three digits of the zip code, the state will get populated with the first matching state value. Once the user types in all five zip digits, we’ll instantly determine and populate the appropriate city. If the zip code isn’t valid (not found in the server’s database), we’ll turn the zip’s border color to red. Such visual clues are helpful to users and have become standard in modern browsers (as an example, Firefox finds matching words in an HTML page by highlighting them in the browser search field while you type).

Let’s start with a simple HTML containing three input fields: zip, city, and state. Please note that the method zipChanged() is called as soon as a character is entered in the zip field. In turn, the JavaScript function zipChanged() (see below) calls the function updateState() when the zip length is three and updateCity() when the length of the zip is five. Both updateCity() and updateState() delegate most of the work to another function - ask().

```
Zip:<input  id=”zipcode” type=”text” maxlength=”5”
onKeyUp=”zipChanged()” style=”width:60”/>
City: <input id=”city” disabled maxlength=”32”
style=”width:160”/>
State:<input id=”state” disabled maxlength=”2”
style=”width:30”/>

<script src=”xmlhttp.js”></script>
<script>
var zipField = null;
function zipChanged(){
 zipField = document.getElementById(“zipcode”)
 var zip = zipField.value;
 zip.length == 3?updateState(zip):zip.length ==
5?updateCity(zip):””;

}
function updateState(zip) {
 var stateField = document.getElementById(“state”);
 ask(“resolveZip.jsp?lookupType=state&zip=”+zip,
stateField, zipField);
}
function updateCity(zip) {
 var cityField = document.getElementById(“city”);
 ask(“resolveZip.jsp? lookupType=city&zip=”+zip,
cityField, zipField);
}
</script>
```

The function ask() communicates with the server and assigns a callback to process the server’s response (see the following code). Later, we’ll look at the content of the dual-natured resolveZip.jsp that looks up the city or state information depending on the number of characters in the zip field. Importantly, ask() uses the asynchronous flavor of the XmlHttpRequest so that populating the state and city fields or coloring the zip border is done without slowing data entry down. First, we call request.open(), which opens the socket channel with the server using one of the HTTP verbs (GET or POST) as the first argument and the URL of the data provider as a second one. The last argument of the request.open() is set to true, which indicates the asynchronous nature of the request. Note that the request hasn’t been submitted yet. That happens with the request.send() call, which can provide any necessary payload for POST. With asynchronous requests we have to assign the request's callback using the request.onreadystatechanged attribute. (If the request had been synchronous, we could have processed the results immediately after request.send, but we would have blocked the user until the request was completed.)

```
HTTPRequest = function () {
   var xmlhttp=null;
   try {
      xmlhttp = new ActiveXObject("Msxml2.
XMLHTTP");
   } catch (_e) {
      try {
         xmlhttp = new ActiveXObject("Microsoft.
XMLHTTP");
       } catch (_E) {    }
   }
   if (!xmlhttp && typeof XMLHttpRequest != 'unde-
fined') {
     try {
        xmlhttp = new XMLHttpRequest();
     } catch (e) {
        xmlhttp = false;
   }  }
   return xmlhttp;
}

function ask(url, fieldToFill, lookupField) {
   var http = new HTTPRequest();
     http.open("GET", url, true);
   http.onreadystatechange = function (){
handleHttpResponse(http, fieldToFill, lookupField)};
   http.send(null);
}

function handleHttpResponse(http, fieldToFill, look-
upField) {
  if (http.readyState == 4) {
    result = http.responseText;
   if ( -1 != result.search("null") ) {
 lookupField.style.borderColor = "red";
 fieldToFill.value = "";
} else {
 lookupField.style.borderColor = "";
 fieldToFill.value = result;
} } }
```

The HttpRequest() function (see above) used by ask() is a cross-browser constructor of an instance of the XMLHTTPRequest; we'll look at it a bit later. For now, note how the invocation of handleResponse() is wrapped by an anonymous function (a so-called closure) function (){ handleHttpResponse(http, fieldToFill, lookupField)}.

The code for that function is dynamically created and compiled every time we do an assignment to the http.onreadstatechange property. As a result, JavaScript creates a pointer to the context with all variables that the enclosing method - ask() - has access to. It's done so the anonymous function and handleResponse() are guaranteed full access to all context-hosted variables until the reference to the anonymous function is garbage-collected. In other words, whenever our anonymous function gets invoked, it can refer to the request, fieldToFill, and lookupField variables as seamlessly as if they were global. It's also true that every invocation of ask() will create a separate copy of the environment with the variables holding the values of the moment the closure was formed.

Let's look at the function handleResponse(). Since it can be invoked at different states of the request processing, the function ignores all cases except the one when the request processing is complete. This corresponds to the request.readyState property equal to 4 ("Completed"). At this point the function reads the server's response text. Contrary to what its name may suggest, neither the input nor the output of XmlHttpRequest has to be restrained to XML. In particular, our resolveZip.jsp (see Listing 1) returns plain text. If the return value is "unknown" the function assumes that the zip code was invalid and changes the border color of the lookup field (zip) to red. Otherwise, the return value is used to populate the fill field (state or city), and zip's border is assigned a default color.

XMLHttpRequest - the Transport Object

Let's return to our cross-browser implementation of XMLHTTPRequest. The last listing contains an HttpRequest() function that's upward-compatible with IE5.0 and Mozilla 1.8/FireFox. For simplicity's sake, we just try to create a Microsoft XMLHTTPRequest object - and if that fails we assume it's Firefox/Mozilla.

At the heart of this function is the XMLHTTPRequest - a native browser object, which facilitates anything that involves HTTP protocol in commu-nicating with the server. It allows specifying any HTTP verbs, headers, and payload and works in either asynchronous or synchronous mode. No downloads or plugins are required, although in the case of IE, XMLHTTPRequest is an ActiveX integrated inside the browser. Accordingly, the "Run ActiveX Control and Plugins" default IE permission should be in place to use it.

Most important, XMLHTTPRequest allows an RPC-style programmatic query to the server without any page refresh.

It does it in a predictable, controlled way, offering complete access to all details of the HTTP protocol, including the headers and any custom formatting of the data. In future articles, we'll show you industrial protocols that you can run on top of this transport including Web Services and XML-RPC that greatly simplify developing and maintaining large-scale applications.

The Server-Side Logic

Finally, the server-side resolveZip.jsp is invoked from the function ask() as shown in Listing 1. The resolveZip.jsp is called in two separate scenarios differentiated by the current length of the zip code (see the zipChanged() function.) The value of the request parameter lookupType is either state or city. For simplicity's sake, we'll assume that two files, state. properties and city.properties, are located in the root directory of the c: drive of the server. The resolveZip. jsp logic is confined to returning the lookup value with the appropriate pre-loaded
file - once in each case of course.

Our AJAX-enabled page is ready. The complete working example is available at: http://www.ajax-maker.com:8080/blog/zipsearch.htm.

Remote Scripting - An Alternative Approach

Some older AJAX implementations are based on so-called remote scripting. The idea is that the user's actions result in querying the server via IFRAME, and the server responds with the JavaScript, which is immediately executed as soon as it reaches the client. This is a big difference compared to XMLHttpRequest approach, where the server responds with the data and the client interprets the data. The advantage is that this solution supports older browsers.

The HTML portion of the IFRAME-based example (see Listing 2) is similar to the one we've used in the XMLHTTPRequest scenario, but this time we'll introduce an extra IFRAME element - controller:

```
Zip:<input  id="zipcode" type="text" maxlength="5"
onKeyUp="zipChanged()" style="width:60" size="20"/>
City: <input id="city" disabled maxlength="32"
style="width:160" size="20"/>
State:<input id="state" disabled maxlength="2"
style="width:30" size="20"/>
<iframe id="controller" style="visibility:hidden;wid
th:0;height:0"></iframe>
```

We keep calling zipChanged() per every key stroke, but this time the function ask(), called from zipChanged() (see Listing 3), sets the IFRAME's src property, instead of invoking an XMLHTTPRequest:

```
function ask(url, fieldToFill, lookupField)
{
 var controller = document.getElementById("controlle
r");
 controller.src = url+"&field="+fieldToFill.
```

Listing 1: resolveZip.jsp

```
<%@ page info="resolveZip" %>
<%@page import="java.util.Properties"%>

<%
 String lookupType = request.getParameter("lookupType");

 Properties properties = (Properties)getServletContext().getAttribute(lookupTyp
e);

 if (properties == null)
 {
  properties = new Properties();
  try {
                    properties.load(new java.io.FileInputStream("c:\\" +
lookupType + ".property"));
   getServletContext().setAttribute(key, properties);
  } catch (java.io.IOException e) {
   out.println("Property File not found");
  }
 }
 out.println(properties.getProperty(request.getParameter("zip")));
%>
```

Listing 2: zipsearch.html (IFRAME)

```
Zip:<input  id="zipcode" type="text" maxlength="5"  onKeyUp="zipchanged()"
style="width:60" size="20"/>
City: <input id="city" disabled maxlength="32" style="width:160" size="20"/>
State:<input id="state" disabled maxlength="2" style="width:30" size="20"/>
<iframe id="controller" style="visibility:hidden;width:0;height:0"></iframe>

<script src="iframe.js"></script>
<script>
var zipField = null;
function zipchanged(){
 zipField = document.getElementById("zipcode")
 var zip = zipField.value;
 zip.length == 3?updateState(zip):zip.length == 5?updateCity(zip):"";

}
function updateState(zip) {
 var stateField = document.getElementById("state");
 ask("resolveZip.jsp?lookupType=state&zip="+zip, stateField, zipField);
}
function updateCity(zip) {
 var cityField = document.getElementById("city");
 ask("resolveZip.jsp?lookupType=city&zip="+zip, cityField, zipField);
}
</script>
```

Listing 3: iframe.js

```
function response(result, fieldToFill, lookupField)
{
 var lookup = document.getElementById(lookupField);
 var fill = document.getElementById(fieldToFill);

 if ( result == "null" ) {
  lookup.style.borderColor = "red";
  fill.value = "";
 } else {
  lookup.style.borderColor = "";
  fill.value = result;
 }
}

function ask(url, fieldToFill, lookupField)
{
 var controller = document.getElementById("controller");
 controller.src = url+"&field="+fieldToFill.id+"&lookup="+lookupField.id;
}
```

Listing 4: resolveZip.jsp (IFRAME)

```
<%@ page info="resolveZip" %>
<%@page import="java.util.Properties"%>

<script>
<%
 String lookupType = request.getParameter("lookupType");

 Properties properties = (Properties)getServletContext().getAttribute(lookupType);

 if (properties == null)
 {
  properties = new Properties();
  try {
                    properties.load(new java.io.FileInputStream("c:\\" + lookup-
Type + ".property"));
   getServletContext().setAttribute(key, properties);
  } catch (java.io.IOException e) {
   out.println("Property File not found");
  }
 }

 out.println("var field= '" + request.getParameter("field") +"'");
 out.println("var lookup= '" + request.getParameter("zip") +"'");
 out.println("var result= '" + properties.getProperty(request.getParameter("zip"))
+"'");
%>
 window.top.window.response(result, field, lookup);
</script>
```

Listing 5

```
id+"&zip="+lookupField.id;
}
```

The server-side logic is presented by a sketchy resolveZip.jsp (see Listing 4). It's different from its XMLHTTPRequest counterpart in that it returns JavaScript statements, which set the global values of the variables field lookup and city and the call function response() from the global window's execution context as soon as it gets to the browser. (Listings 5-7 can be downloaded from www.jdj.sys-con.com.)

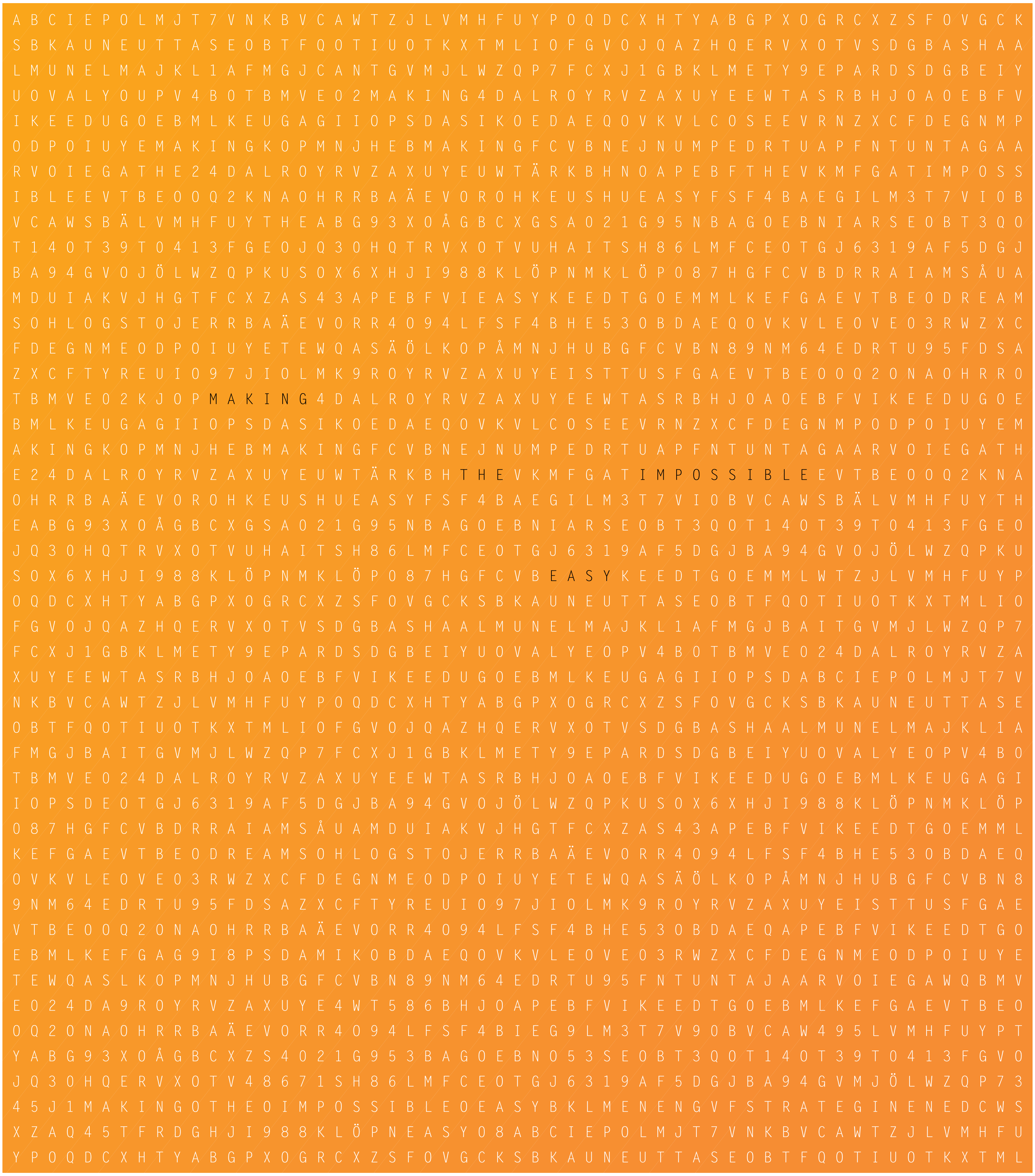
The function response() is a modified version of the handleResponse() which is absolved from dealing with uncompleted requests (see Listing 2.)

The Fine Print

For simplicity's sake, we've "overlooked" some important issues in our sample code:

1. The fact that the instances of the XMLHTTPRequest object and callback invocations haven't been destroyed after being used, which causes memory leaks after every call. Properly written code should destroy or reuse such instances in the object pool. Object management techniques common to the server software have to be used for the client
2. In quite a few places the errors weren't handled properly. For example, the call to request.open() in the method ask() can throw an exception that has to be caught and processed even though JavaScript exceptions don't have to be checked. The handleResponse() function is another example. It has to check headers and responseText for possible server-side and communication errors. In case of an error, it has to try to recover and/or report an error. Properly developed AJAX applications eliminate loosing data on "submissions" due to disconnects and other low-level communication problems via a robust, self-recovering framework.
3. Current server-side frameworks provide quite a few functions that have to be reconciled with a refresh-free approach. For example, let's consider a custom server-side authentication with a timeout. In that case we'd have to intercept security system response to the XMLHTTPRequest calls, bring up the login screen, and then re-issue the request after the user was authenticated.

All these problems are typical of any application code working with low-level APIs and all of them can be resolved. The good news is that the technologies needed to resolve these issues are quite familiar to most Java developers like Web Services, custom tags, and XML/XSLT. The only



AJAX developers, do you want to develop RIAs more quickly and easily?

Are you under the gun to get high quality applications out the door? Do you need better solutions? Helmi's open source, AJAX based RIA platform helps you build rich, highly interactive web applications and speed them to market fast. You might say we make the impossible easy. For more information, visit www.helmitechnologies.com

helmi technologies

difference is that nowadays these technologies come to the rescue on the client in the form of:

- Web Services using SOAP/REST/RPC for a simple communication standard
- Client-side custom tags for packaging rich client-side controls with integrated AJAX functionality
- XML- and XSLT-based data manipulation

Summary

The AJAX approach offers a rich Internet experience on a par with that of desktop applications. AJAX features have to be applied selectively: you definitely don't want your credit card charged by the background process while you're still shopping. Is AJAX momentum sustainable? We certainly hope so. We've been developing AJAX applications for the last sive years and can attest that it's sound and very effective. However, it requires that a developer be exposed to a much wider set of technologies than the ones used in the traditional "click-refresh" Web applications.

Part 2

The publicity that AJAX grabbed over the last half a year is based on closing the gap between the Web applications and the desktop applications, combining the "reach" and "rich." At the same time, the gap between the technological level of AJAX and what corporate developers expect in their modern arsenal is really astonishing. After all, AJAX is neither a tool nor a platform. There is no AJAX standards committee or community process in place. While software vendors are crafting proprietary development platforms on top of AJAX (which pretty much means "from scratch"), early adopters of AJAX are left on their own.

In the previous section, we touched on the foundation of AJAX - the ability to establish script-to-server communication. This is what makes HTML pages dynamic and responsive. Does it mean we are ready to kick-off our own version of Yahoo mail? No, we are not. Here is why: AJAX is a mixed blessing. On one hand it enables us to create rich, desktop-class applications on the Web. On the other, if we compare "page-flipping" Web applications with the client/server or Swing ones, the development practices are not quite the same. What about management expectations? We'll need to get used to the fact that it takes time to build a rich UI. The more flexibility with more permutations the user is allowed - the more time it takes.

The answer, of course, is component libraries, frameworks, and industrial-strength tools. Leaving tools aside, this article concentrates on what is available for AJAX enthusiasts today. Addressing a need to build reusable business components, it focuses on the "hidden" object-oriented power of JavaScript. Also, by addressing a need to build custom-rich UI components, it illustrates a convenient way to encapsulate presentation logic in custom client-side HTML tags.

AJAX Language: Object-Oriented JavaScript

By definition, JavaScript is the language of classic AJAX. Unlike Java, JavaScript does not enforce the OO style of coding. That said, it's surprising how often it's overlooked that Java-Script fully supports all the major attributes of an OO language: inheritance, polymorphism, and encapsulation. Douglas Crockford even named Java Script "The World's Most Misunderstood Programming Language." Let's review the object-oriented side of JavaScript.

Data Types

In Java, a class defines a combination of data and its associated behaviors. While JavaScript reserves the class keyword, it does not support the same semantic as in conventional OOP languages.

It may sound strange but in JavaScript, functions are used as object definitions. By defining a function in the example below you, in fact, define a simple empty class - Calculator:

```
function Calculator() {}
```

A new instance is created the same way as in Java - by using the new operator:

```
var myCalculator = new Calculator();
```

The function not only defines a class, but also acts as a constructor. The operator new does the magic, instantiating an object of class Calculator and returning an object reference in contrast to merely calling the function.

Creating an empty class is nice but not really useful in real life. We are going to fill-in the class definition using a Java-Script prototype construct. JavaScript uses prototype to serve as a template for object creation. All prototype properties and methods are copied by reference into each object of a class, so they all have the same values. You can change the value of a prototype property in one object, and the new value overrides the default, copied from the prototype, but only in that one instance. The following statement will add a new property to the prototype of the Calculator object:

```
Calculator.prototype._prop = 0;
```

Since JavaScript does not provide a way to syntactically denote a class definition, we'll use the with statement to mark the class definition boundaries. This will also make the example code smaller as the with statement is allowed to perform a series of statements on a specified object without qualifying the attributes.

```
function Calculator()  {};
with (Calculator) {
 prototype._prop = 0;
 prototype.setProp = function(p) {_prop = p};
 prototype.getProp = function() {return  _prop};
}
```

So far we have defined and initialized the public _prop variable as well as provided getter and setter methods for it.

Need to define a static variable? Just think of the static variable as being a variable owned by the class. Because classes in JavaScript are represented by function objects, we just need to add a new property to the function:

```
Calculator.iCount = 0;
```

Now that the iCount variable is a property of the Calculator object, it will be shared between all instances of the class calculator.

```
function Calculator()  {
 Calculator.iCount++;
};
```

The above code will count all created instances of the class Calculator.

Encapsulation

Using “Calculator”, as defined above, permits access to all the “class” data, increasing the risk of name collisions in inherited classes. We clearly need encapsulation to view objects as self-contained entities.

A standard language mechanism of data encapsulation is private variables. And a common JavaScript technique for emulating a private variable is to define a local variable in the constructor, so that this local variable is accessible exclusively via getter and setter - inner functions of the very same constructor. In the following example, the _prop variable is defined within the Calculator function and is not visible outside of the function scope. Two anonymous inner functions, assigned to setProp and getProp attributes, provide access to our “private” variable. Also, please note the use of this, quite similar to how it is used in Java:

```
function Calculator()  {
 var _prop = 0;
 this.setProp = function (p){_prop = p};
 this.getProp = function() {return  _prop};
};
```

eliminate postbacks **forever.**

NetAdvantage® 2006 Vol. 2

The toolset for user interface design and development

Save Time – Enable AJAX support without hand-coded JavaScript.

Increase Performance – Eliminate postbacks with AJAX-enabled Grid, Tree, Tab, and Schedule.

Empower Your Users – Simplify your interfaces with user-definable Grid row filtering.

Leverage a Rich Toolset – Deliver robust scheduling applications with recurring appointments and disconnected user support.

No -Touch AJAX

Visit Booth #208 - Enter To Win a FREE Sirius Radio & 1-Year Subscription

SIRIUS SATELLITE RADIO THE BEST RADIO ON RADIO™

Infragistics® Powering The Presentation Layer

learn more: infragistics.com/ajax

Infragistics Sales - 800 231 8588
Infragistics Europe Sales - +44 (0) 800 298 9055

```
// Constructor
function Calculator(ops) {
     this.ops = ops;
}
with (Calculator) {
 // Make constructor available as a member function
 prototype.Calculator = Calculator;
 // Member functions
 prototype.evaluate = function () {
  this.opsStack = new Array();
  for (var i = 0; i < this.ops.length; i++) {
   var op = this.ops[i];
   if (typeof op == "number")
    this.push(op);
   else if (typeof this[op] == "function")
    this[op]();
  }
  return this.pop();
 };
 prototype.pop = function () {return this.opsStack.pop();}
 prototype.push = function (val) {this.opsStack.push(val);}
}

//Constructor
function ArithmeticCalcuator(ops) {
 this.Calculator(ops);         // super
};
with (ArithmeticCalcuator) {
 // Establish inheritance
 ArithmeticCalcuator.prototype = new Calculator();
 prototype.constructor = ArithmeticCalcuator;
 // Make constructor available as a member function
 prototype.ArithmeticCalcuator = ArithmeticCalcuator;
 // Member functions
 prototype.add = function () {this.push(this.pop() + this.pop()); }
 prototype.sub = function () {this.push(this.pop() - this.pop());}
 prototype.mul = function () {this.push(this.pop() * this.pop());}
 prototype.div = function () {this.push(this.pop() / this.pop());}
}
```

Listing 6

```
<PUBLIC:COMPONENT NAME="cbx" tagName="CHECKBOX">
<PROPERTY NAME="label" />
<PROPERTY NAME="checked" GET="getChecked" PUT="putChecked" />
<PROPERTY NAME="labelonleft" VALUE="true"/>
<PROPERTY NAME="value" GET="getValue" PUT="putValue" />
<PROPERTY NAME="onValue" VALUE="true"/>
<PROPERTY NAME="offValue" VALUE="false"/>
<METHOD NAME="show"/>
<EVENT NAME="onItemChanging" ID="onItemChanging"/>
<EVENT NAME="onItemChanged" ID="onItemChanged" />
```

What is often overlooked is the cost of such encapsulation in JavaScript. It can be tremendous, because inner function objects get repeatedly created for each instance of the "class".

Accordingly, since constructing objects based on the prototype is faster and consumes less memory, we cast our vote in favor of public variables wherever performance is critical. You can use naming conventions to avoid name collisions, for example, by prefixing public variables with the class name.

Inheritance

At first glance, JavaScript lacks support for the class hierarchy similar to what programmers of conventional object-oriented languages expect from the modern language. However, although JavaScript syntax does not support class inheritance as in Java, inheritance can still be implemented by copying an instance of a previously defined class into the prototype of the derived one.

Before we provide an illustration, we need to introduce a constructor property. JavaScript makes sure that every prototype contains constructor, which holds a reference to the constructor function. In other words, Calculator.prototype.constructor contains a reference to Calculator().

Now, the code below shows how to derive the class ArithmeticCalculator from the base class Calculator. "Line 1" results in borrowing all properties of the Calculator, while "Line 2" restores the value of the prototype, constructor back to ArithmeticCalculator:

```
function ArithmeticCalculator() { };
with (ArithmeticCalculator) {
    ArithmeticCalculator .prototype = new
Calculator();        //Line 1
   prototype.constructor = ArithmeticCalculator;
//Line 2
}
```

Even if the example above looks like a composition rather than inheritance, the JavaScript engine knows about the prototype chain. In particular, the instanceof operator will work correctly with both the base and derived classes. Assuming you create a new instance of a class ArithmeticCalculator:

```
var  c = new ArithmeticCalculator;
```

expressions c instanceof Calculator and c instanceof ArithmeticCalculator will both evaluate to true.

Notice, that the constructor of the base class in the example above is called at the point when the ArithmeticCalculator prototype is initialized and not when an instance of the derived class is created. This could have unwanted side effects and you should consider creating a separate function for initialization purposes. As the constructor is not a member function, it can't be called through this reference directly. We will need to create a "Calculator" member function to be able to call super:

```
function Calculator(ops) { ...};
with (Calculator) {
 prototype.Calculator =  Calculator;
}

 Now we can write an inherited class that explic-
itly calls the constructor in the base class:

function ArithmeticCalculator(ops) {
 this.Calculator(ops);
};
with (ArithmeticCalculator) {
 ArithmeticCalculator .prototype = new Calculator;
 prototype.constructor = ArithmeticCalculator;

 prototype.ArithmeticCalculator =
ArithmeticCalculator;
}
```

Polymorphism

JavaScript is a non-typed language where everything is an object. Accordingly, if there are two classes A and B, both defining method foo(), JavaScript will allow polymorphic invocation of foo() across instances of A and B even if there is no hierarchical relation (albeit implementational) whatsoever. From that perspective, JavaScript provides a wider polymorphism then Java. The flexibility, as usual, comes at a price. In this case, it is a price of delegating the type checking job to application code. Specifically, if there is a need to check that a reference indeed points to a desired base class, it can be done with the instanceof operator.

On the other hand, JavaScript doesn't check parameters in the function calls, which prevents from defining polymorphic functions with the same name and different parameters (and let the compiler choose the right signature). Instead, JavaScript provides an argument object - Java 5 style - within a function scope that allows you to implement a different behavior depending on the parameter's type and quantity.

Example

Listing 5 implements a calculator that calculates expressions in a reverse Polish notation. It illustrates the main techniques described in the articles and also shows the usage of the unique JavaScript features, such as accessing object properties as an array element for a dynamic function call.

To make Listing 5 work we also need to provide a piece of code that instantiates the calculator objects and calls the evaluate method:

```
var e = new ArithmeticCalcuator([2,2,5,"add","mul
"]);
alert(e.evaluate());
```

AJAX Component Authoring

All AJAX component authoring solutions known today can be logically divided into two groups. The first group specifically targets the seamless integration with the HTML-based UI definition. The second group drops HTML as a UI definition language in favor of certain XML. In this article we illustrate one approach from the first group, an analog to JSP tags, albeit in the browser. These browser-specific component authoring extensions are called element behaviors in the IE case or extensible bindings in the case of the latest versions of Firefox, Mozilla, and Netscape 8.

Custom Tag Dialects

Internet Explorer, starting with version 5.5, enables the JavaScript authoring of custom, client-side HTML elements. Unlike JSP tags, these objects are not preprocessed into HTML on the server side. Rather, they're legitimate extensions of a standard HTML object model and everything, including control construction, happens dynamically on the client. Similarly, Gecko-engine based browsers can dynamically decorate any existing HTML element with a reusable functionality.

It's possible, therefore, to build a library of rich UI components with methods, events, and attributes that will have HTML syntax. Such components can be freely mixed with standard HTML. Internally, these components will communicate with application servers, AJAX style. In other words, it's possible (and relatively simple) to build your own AJAX object model.

The IE flavor of this approach is called HTC or HTML components; the Gecko version is called XBL - eXtensible Bindings Language. For the purposes of this article, we'll focus on IE.

Enter the HTML Components - HTC

HTC or HTML components are also called behaviors. In turn they are divided into attached behaviors that decorate any existing HTML element with a set of properties, events, and methods, and element behaviors that look like an extended set of custom HTML tags to the hosting page. Together, element and attached behaviors provide a simple solution for authoring both components and applications. Here we'll illustrate the most comprehensive case, element behaviors.

Data-Bound Checkbox Control

As an illustration of element behavior, we'll construct a custom data-bound checkbox. The rationale behind building such a control is that a standard HTML checkbox has several noticeable shortcomings:

- It requires application code to map the value of the "checked" attribute into business domain values, such as "Y[es]"/"N[o]", "M[ale]"/"F[emale]", etc. The HTML checkbox uses "checked" contrary to many other HTML controls using "value".
- It requires application code to maintain the state of the control (modified versus not modified). This is actually a common problem with all HTML controls.
- It requires application code to create an associated label that should accept click and change the state of the checkbox accordingly.
- The standard HTML checkbox doesn't support "validation" events to allow the canceling of a GUI action under certain application conditions.

To settle on a syntax, let's say that a sample usage of the control we are building could look the following way:

```
<checkbox id="cbx_1" value="N" labelonleft="true"
label="Show Details:" onValue="Y" offValue="N"/>
```

In addition, our control will support the cancelable event onItemChanging and the notification event onItemChanged.

Custom Tag Definition

Structurally, a custom tag is a file with an HTC extension that describes its properties, methods, and events between <PUBLIC:COMPONENT> and </PUBLIC:COMPONENT>.

To define a custom CHECKBOX tag, we create a file checkbox.htc as in the following snippet where the first line sets the tag name of the component:

```
<PUBLIC:COMPONENT NAME="cbx" tagName="CHECKBOX">
<PROPERTY NAME="value" GET="getValue" PUT="putValue"
/>
// Here we place all other properties of the com-
ponent
<METHOD NAME="show" />
// Here we place all other methods of the compo-
nents
<EVENT NAME="onItemChanging" ID="onItemChanging"/>
// Here we place all other events that component
will fire to
application
<ATTACH EVENT="oncontentready" HANDLER="constructor"
/>
// Here we place all other events that component
handles itself
<SCRIPT >
// Here we place all methods, properties getters
and setters and
event handlers
</SCRIPT>
</PUBLIC:COMPONENT>
```

Custom Tag Use Case

While the contents of the HTC file matter a lot, the name of the file is irrelevant, although, ultimately, the URL to the HTC file needs to be specified using the IMPORT instruction. It has to be done before the corresponding custom tag is mentioned for the first time (on the page). Here is how the simplest possible page utilizing a custom checkbox might look, assuming the page and the HTC file are located in one folder:

```
<HTML xmlns:myns>
<?IMPORT namespace="myns" implementation="checkbox.
htc" >
<BODY>
<myns:checkbox id='cbx_1' label='Hello'/>
</BODY>
</HTML>
```

Please notice how custom CHECKBOX has been mapped to a nondefault namespace "myns" in the opening HTML tag. The IMPORT instruction performs a synchronous load of the HTC into the browser's memory and also instructs the browser how to perform name resolution for the appropriate namespace (HTC to namespace association can be many-to-one).

Constructor of the Custom Tag

The best way to initialize HTC, once it's loaded, is to process the "oncontentready" event. Accordingly, we define a handler function, which for sheer clarity is called constructor:

```
<ATTACH EVENT="oncontentready" HANDLER="constructor"
/>
```

The logic of constructor() is simple: concatenate a regular HTML checkbox and HTML label in the order dependent on the value of property labelonleft (see property definition below):

```
function constructor()       {
  // We will add an HTML checkbox and label to the
element body
  // See Listing 6 for details
}
```

Defining Custom Tag Properties

To define property labelonleft, we add one more line to the <PUBLIC:COMPONENT> section:

```
<PROPERTY NAME="labelonleft" VALUE="true"/>
```

_INFRASTRUCTURE LOG

_DAY 15: This project is out of control. The development team's trying to write apps supporting a service oriented architecture...but it's taking FOREVER!

_DAY 16: Gil has resorted to giving the team coffee IVs. Now they're on java while using JAVA. Oh, the irony.

_DAY 18: I've found a better way: IBM Rational. It's a modular software development platform based on Eclipse that helps the team model, assemble, deploy and manage SOA projects. The whole process is simpler, faster and all our apps are flexible and reusable. :)

_The team says it's nice to taste coffee again, but drinking it is sooo inefficient!

IBM

Rational

Download the IBM Software Architect Kit at:
IBM.COM/**TAKEBACKCONTROL**/FLEXIBLE

```
<ATTACH EVENT="oncontentready" HANDLER="constructor" />
<SCRIPT >

 var _value;

 function constructor()       {
  var s;
  s = '<INPUT name="cb_'+ uniqueID + '" id="cb_' + uniqueID + '" type="checkbox" ' 
+
   'onclick="' + uniqueID + '.checked= +cb_' + uniqueID+ '.checked" ';
  if( _value == onValue )
   s +=' checked="true" ';
  s+= "/>";
  element.insertAdjacentHTML('afterBegin', s);
  element.insertAdjacentHTML((labelonleft == "true")?'afterBegin':'beforeEnd',
   '<LABEL for="cb_' + uniqueID+ '">' + label + '</LABEL>');
 }

 function putChecked( newValue ) {
  value = (newValue?onValue:offValue);
 }
 function getChecked(){
  return ( _value == onValue);
 }

 function getValue(){
  return _value ;
 }
 function putValue( newValue ) {
  if (window.event == null)
   _value = newValue; /* initial*/
  else if (_value != newValue) {
   var oEvent = createEventObject();
   oEvent.newValue = newValue;
   oEvent.returnValue = true;
   onItemChanging.fire(oEvent);
   if (oEvent.returnValue)
    _value = newValue;
   eval('cb_'+uniqueID).checked= ( _value == onValue );
   if (_value == newValue)
    onItemChanged.fire(oEvent);
  }
 }

 function show(bCmdShow){
  style.visibility= (bCmdShow?'visible':'hidden') ;
 }

</SCRIPT>
</PUBLIC:COMPONENT>
```

Please note that this property does not contain getter and/or setter methods. Properties onValue and offValue that provide the mapping of the checkbox status into a business value domain also don't need getters and setters:

```
<PROPERTY NAME="onValue" VALUE="true"/>
<PROPERTY NAME="offValue" VALUE="false" />
```

However, property checked is defined with both getter and setter:

```
<PROPERTY NAME="checked" GET="getChecked"
PUT="putChecked" />
```

Accordingly, we have definitions for both methods in the <SCRIPT> section. As you can see, setter putChecked(), which is triggered every time checked is modified, sets the value property to one of two variants: onValue or OffValue. Please note that putChecked() will get triggered not only by the script on the checkbox-hosting page, but also by any assignment to checked done inside this very checkbox.htc.

```
var  _value;
function putChecked( newValue ) {
 value = (newValue?onValue:offValue);
}
function getChecked(){
 return ( _value == onValue);
}
```

Defining Events for the Custom Tag

Let's look at the definition of onItemChanging and onItemChanged events and how these events are being fired and processed inside the setter for value property (see Listing 6).

Method putValue() has a couple of points of interest. First, it can be called during the parsing on the CHECKBOX tag, as long as the HTML value attribute is specified. That's why we have a separate logic branch for not constructed objects, to differentiate the process of construction from a reaction to a user click. Second, we illustrate here the creation and processing of the custom event onItemChanging, which allows the application to cancel (block) the action. Please note that both clicking as well as programmatic assignment to the value can get cancelled this way.

Event Canceling

To cancel the event, an application should intercept the event and set event.returnValue to false. The schematic snippet of code illustrating how the application would cancel the processing is presented below:=

```
cbx_1::onItemChanging() {
. . . . .
if (canNotBeAllowed) {
 event.returnValue=false;
.  .  .  .  .
}
```

If the event is not cancelled, putValue() sets the internal, plain HTML checkbox's checked property as per the current value: if that is equal to onValue, the internal checkbox will get checked; if it is equal to offValue (there is no third option), unchecked (the full listing is presented in Listing 6).

HTML Internals of the CHECKBOX

The painting of our control is done by the helper functions addLabel() and addCheckBox() and is called from within a constructor(). These functions inject HTML inside the element's innerHTML (element is the analog for this in HTC parlor). The injected HTML, in the simplified form, looks like the following:

```
<LABEL for=cb_{uniqueID}>Show Details:</LABEL>
<INPUT id=cb_{uniqueID} type=checkbox />
```

where uniqueID is a unique (within a page) string literal generated by IE, which identifies the instance of the HTC.

Encapsulation ... Again

There is one shortcoming in our CHECKBOX. The way we built it, the HTML injected during the constructor() contributes to the DOM of the page that hosts the HTC. Also global JavaScript variables like_value fall into the global scope of the document they're included in. This is dangerous since we run into the possibility of name clashes: the most obvious case is more than one instance of the same control. In addition, it presents a possibility that our control may accidentally reference other objects with the same names and vice versa.

To put it simply, a special mechanism is required to enable a truly modular approach for object authoring. Fortunately, HTC technology supports an intelligent answer - viewLink.

The easiest way to declare a control as encapsulated is to put one extra declaration between opening and closing PUBLIC:COMPONENT tags:

```
<PUBLIC:DEFAULTS viewLinkContent/>
```

Instantly, the control becomes encapsulated; it has its own HTML document tree, atomic to the master document. Every instance of the object has its own set of instantiated values and only public methods and properties can be accessed by the outside code. In other words, the viewLink mechanism fully enables the design and implementation of sophisticated Web applications using a true OO component-based approach.

In particular, we can simplify our code by removing uniqueID suffixes from the definition of internal checkboxes and HTML labels, since we are not afraid of name clashes anymore. Accordingly, we may replace the line:

```
  eval( 'cb_'+uniqueID).checked = ( _value ==
onValue );

with the

  cb.checked = ( _value == onValue );
```

as well as change addCheckbox() and addLabel() accordingly.

Conclusion

Since the AJAX race has just started, there are no AJAX standards and no commonly accepted RAD tools you can rely on to build your applications. While software vendors have a long way to go to create the robust development platforms, AJAX enthusiasts can prepare by encapsulating reusable blocks of code as business components with a well-defined API.

Navigating in this direction, this article outlined the OO "powers" of the AJAX language - JavaScript. It also illustrated one of the available component-authoring strategies - client-side custom tags technology. While we presented only IE-specific custom tags, we also provide a downloadable example of extensible bindings example for the Mozilla browser. All article-related examples can be downloaded from http://www.ajaxmaker.com/JDJ/AJAX/partII.html. ■

Victor Rasputnis is an IT consultant who has been working in Java, PowerBuilder, C, Assembler - whatever language has appeared since 1976. Victor is one of the creators of XMLSP, the product that pioneered AJAX in 1999. victorrasputnis@teamcti.com

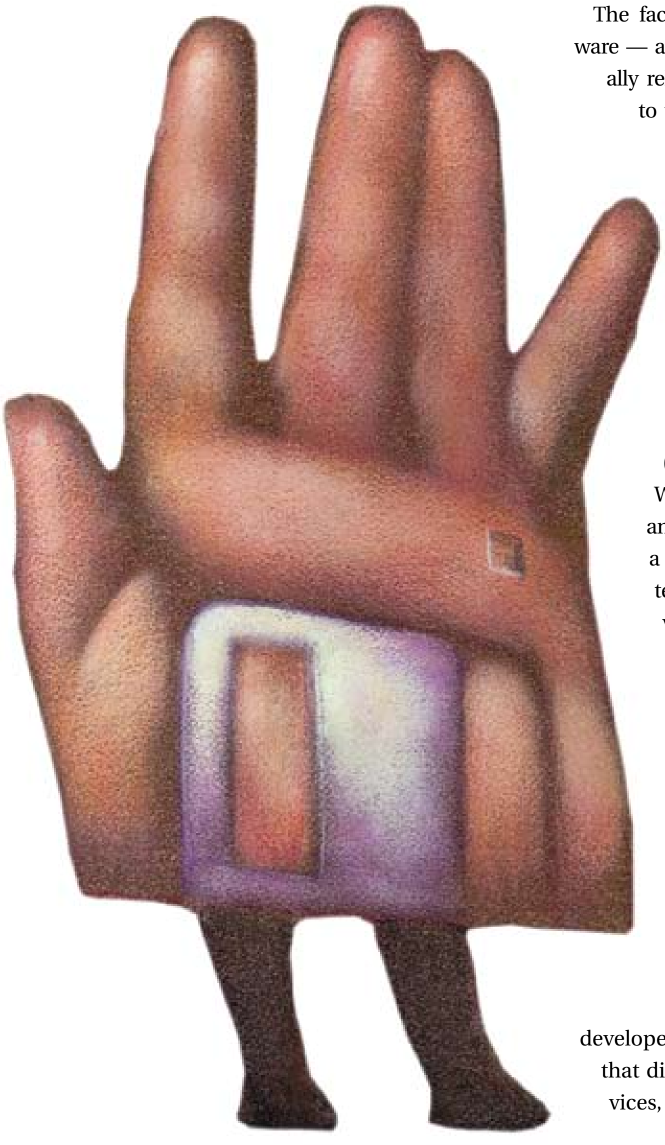
Anatole Tartakovsky is a New York-based software developer, lecturer, consultant, and author. He is currently working as the CTO at Computer Technology, Inc., focused on developing AJAX/FLEX solutions for the financial and retail industries. anatolet@teamcti.com

Igor Nys is a director of technology solutions at EPAM Systems. He was closely involved in the software development based on XMLSP technology - one of the AJAX pioneers. igordnys@gmail.com

AJAX + SOA: The Next Killer App

SOA lacks a face; that's where AJAX comes in — it puts a face on SOA

John Crupi



Enterprises trying to improve business unit productivity and the reuse of IT assets continue to struggle. IT organizations have achieved some success by attacking these challenges with Service Oriented Architecture (SOA), but in most cases have still only exposed small portions of the overall IT service portfolio. Much of this struggle has been to deliver a "just enough" SOA to the business unit to improve its ability to build applications and features to get to market faster, better, and cheaper. And as we've learned, accomplishing this is easier said than done.

The fact is that SOA is middleware — and middleware traditionally relies on more middleware to translate data into a consumer-friendly state. It's certainly a major disappointment when you finally get your SOA right only to find that building a composite application requires using a portal (middleware) and/or orchestrating it with a BPEL engine (even more middleware). Worse yet, you may be in an organization that deploys a UDDI registry and registers a bunch of Web services. Unfortunately, in most cases there are very few applications built to actually consume these services. How can this be?

Should we conclude that something is wrong if applications aren't being built to consume these SOA services? Is it too difficult for business unit developers to build applications that directly consume SOA services, forcing us to rely on IT to create these applications? Is the absence of a SOA governance architecture holding us back? I think the answer is "yes" to all of these questions. And there is one standout reason: it has simply been much too difficult for the business unit developer to consume and leverage the SOA services exposed by IT. What's been missing is an easy way to put a "face" on SOA — and that's precisely the benefit of using AJAX in combination with SOA.

SOA services are typically implemented as loosely coupled Web services that encapsulate and expose business functionality. This sounds relatively straightforward, but is quite complex and difficult to achieve in practice. Developers frequently argue about the granularity of SOA services, but most now agree that "business-grained" granularity is the most appropriate. However, it still takes a great deal of domain expertise and collaboration with the business unit to size services properly.

Fortunately, there's been a recent surge in SOA interest. Perhaps enterprises are finally coming to recognize that SOA can really help the bottom line. Maybe it's being driven by better tooling and Web services evangelizing by Amazon, Yahoo, and eBay. Or could it be AJAX? Of course, it's AJAX — why else would I be writing this article? Seriously, I do think that AJAX is driving a renewed interest in SOA, especially in the mashup space. But how can two very different technologies combine and connect to provide something far greater than the parts? Take a look at Wikipedia's current definition of AJAX. It talks about Web pages, but there's no mention of SOA. It says:

"AJAX, shorthand for Asynchronous JavaScript and XML, is a Web development technique for creating interactive Web applications. The intent is to make Web pages feel more responsive by exchang-ing small amounts of data with the server behind the scenes, so that the entire Web page does not have to be reloaded each time the user makes a change. This is meant to increase the Web page's interactivity, speed, and usability."

The absence of SOA in this definition comes as no big surprise since the early buzz around AJAX has centered on enhancing Web page functionality and usability. This has been showcased in many applications such as Google Maps, Flickr, and Yahoo Mail. It's not these consumer-facing applications that personally get me excited about the potential of AJAX though. Instead, it's the *business* applications that run *behind* the corporate firewalls that can really exploit and benefit from AJAX, because it gives us two key features: a client-side programming model and the ease of making asynchronous calls to the server. These two key capabilities — the ability to apply logic in the client (browser) and to access server data without disrupting the Web page — are what al-low the new Web 2.0 paradigm to open up so many enticing possibilities for rich enterprise applications.

Earlier I said that SOA was lacking a face. That's where AJAX comes in — it puts a face on SOA. Let me explain this a bit further. Think about what happens when SOA services come online. They usually get registered in a registry/repository (if we're lucky) and become available for consumption. For instance, take a look at what StrikeIron (www.StrikeIron.com) provides. StrikeIron has created a "Web services marketplace" for the general population. At first glance, StrikeIron's catalog looks like a list of mini-business applications. But then you realize that these aren't applications — they're actually Web services. The concept of a company providing WSDL/REST Web services for general consumption makes a lot of sense. But before we get too excited, let's take a look at what's selling. According to StrikeIron, which licenses access to these services, its most popular Web services include:

* U.S. Address Verification
* Global SMS Pro
* Sales and Use Tax
* E-mail Verification
* Reverse Phone Lookup

Without a doubt, all of these Web services are useful and can be applied to many different domains. But at the same time, they're also pretty commoditized. In other words, I may not care about who provides these services, only that I get the desired information. On the other hand, would I use just any Web service to transfer money from my checking account to my savings account? I don't think so. I need to trust the service, so I'd have to have some sort of relationship with the vendor providing that service. This "circle of trust" between me, the consumer, and the service provider is exactly the kind of relationship enterprises have internally and with partners.

The same approach could be taken by enterprises to begin offering their Web services to a wider audience. Through a Web services marketplace, enterprises could register various Web services that would normally only be available internally and/or to their partners. The marketplace vendors would obviously love to see this happen, but more importantly, I see this as an opportunity to begin to apply AJAX + SOA to drive a whole new class of Web 2.0 business applications.

For the first time, it's starting to feel like our application develop-ment and SOA efforts are coming together. We have business functionality represented in a reusable form – SOA services. We have ubiquitous connectivity – the Web. We have what's turning out to be the new application container – the browser. We have a programming model in the application container/browser – JavaScript. And they are all using open standards! What more could we ask for? Actually, quite a bit more.

In particular, I'd like to see a more rapid solution for developing applications based on all of this — a way to build applications without having to rely on more middleware to integrate with the SOA services. This is what I'll call the ability for Web applications to perform "direct connect SOA." Direct connect SOA conveys the ability to punch through the traditional curtain of portals and heavyweight process engines and directly (at least conceptually; more about this later) access SOA services. I don't just mean Web services either. It could be BPEL orchestration services, coarse-grained POJO services, RSS feeds, or anything else that can be exposed as a "service," albeit at the right level of business granularity. And of course the interfaces should be exposed using open standards.

This novel development and runtime model creates a new way to build application-driven composite applications. It has the appeal of client/server, without all the traditional heavyweight client/server baggage. It runs in the browser and is delivered on-demand.

We've all heard a lot about "composite applications" over the past few years. But most vendors have been talking about composing services as a way to re-factor their hosted services into more palatable services or portal applications. Let me clarify by using an analogy.

AcmeGrid, a fictitious grid vendor, provides a service grid that lets you run your applications as services. Its customers tell it they want a way to "compose" a combination of services into coarser-grained services. So, naturally, AcmeGrid announces an Eclipse-based AcmeGrid Composite Application Builder (CAB). Interestingly, CAB looks a lot like a BPEL designer, but has tighter integration with the AcmeGrid deployed services. Pretty slick, but it's not really an application as much as it is a service. In essence, CAB is more like a service builder. But who wants a service builder when we're trying to build applications? Soon, another fictitious vendor, we'll call them AcmePortal, announces its Por-tal Composite Application Builder (PCAB). It too releases an Eclipse-based designer that looks and feels like a BPEL designer, but this one knows how to build portals. In many cases, a portal is fine as an application. But if you're forcing a portal to be an application, it's just adding unnecessary weight.

What I really want is a user-based composite application, not a middleware-based composite application. To make this happen, I need a development and runtime platform that not only speaks AJAX and SOA, but governs the two as well. Some vendors promote the concept of AJAX applications calling WSDL-based Web services directly from the browser, essentially making SOAP calls. This approach even has a name – "client/SOA." This may be fine for simple non-enterprise or pure consumer applications, but it's a no-go for the enterprise. Why? Because when you call Web services directly from the browser, governance is left to the browser — which is another way of saying, there's no governance. Figure 1 shows ungoverned Web service consumption. I've never run into an enterprise that doesn't govern its services and strongly doubt that enterprises would allow this to happen merely because we have the technology to do so efficiently. If you doubt me, just remember that enterprises never opened up the firewall (for applications) to anything but HTTP and SSL. No matter what we told system administrators, no other ports were being opened.

So what we're talking about isn't just AJAX + SOA. It's really a platform that provides the governance necessary for AJAX and SOA to co-exist in the enterprise. This is a platform that provides the ability to consume SOA services on behalf of the client, but also governs the service consumption. Figure 2 shows how Web services can be governed by a service gateway. A service gateway is a server-side abstraction that governs and mediates service access on behalf of the client, which in this case is the AJAX application in the browser. The beauty of using a serv-ice gateway is that you're not restricted to accessing only services run-ning in the enterprise. The service gateway governs any service that's registered in the enterprise. In WSDL-based Web services, an enterprise would register the WSDL, and WSDL provides the bind to the service at runtime. This might be a service running in the enterprise's data center, but it could just as easily be a service running in a partner's data center. If the enterprise allows (governs) applications to access services, it doesn't matter where they're running.

I hope you're starting to appreciate the power of combining AJAX and SOA – specifically, how the two can co-exist and deliver new Web services–based applications with the governance that enterprises require. I truly believe we're entering a new era with amazing opportunity. Web 2.0 social networks, photo-sharing and tagging are great, but the real corporate impact comes in a form of Web 2.0 for the enterprise. ■

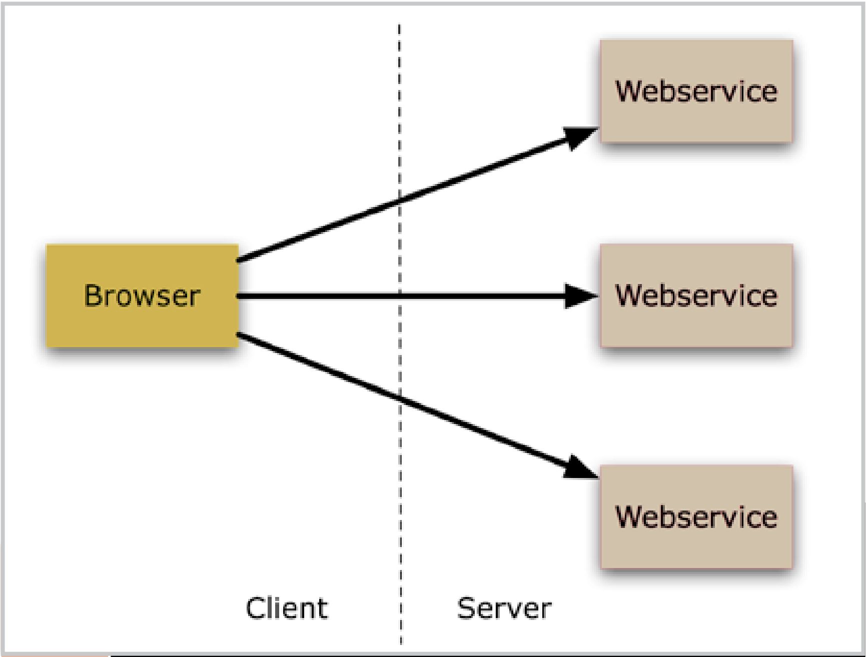


Figure 1: Non-governed Web service consumption

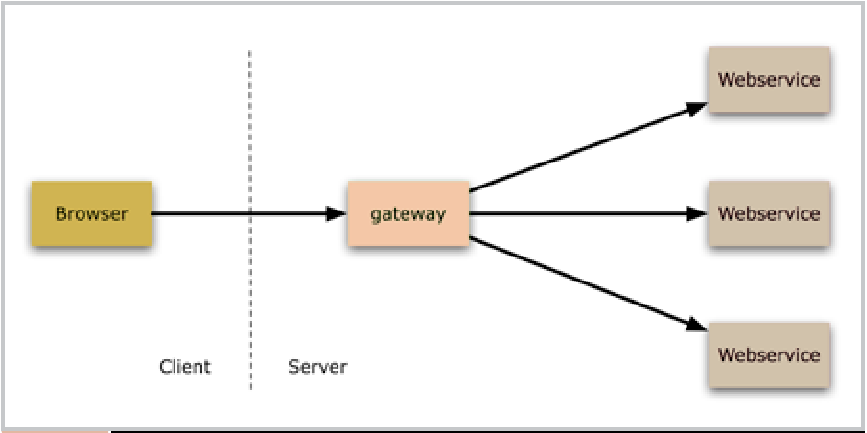


Figure 1: Governed Web service consumption

John Crupi is the CTO at JackBe. See www.jackbe.com.

Enterprise AJAX is ICEfaces

- Create secure Rich Internet Applications (RIA)
- Develop in Java, not JavaScript
- Transform the User Experience

Visit **www.ICEfaces.org** to try it out!

RIA Solution for SOA

Rich User Experience

Create a new class of collaborative and dynamic enterprise applications. Unleash the unique power of Ajax Push Technology to deliver server-initiated, instantaneous presentation updates. Easily migrate existing JSP-based and traditional client-server applications to RIAs.

Standards-Based

Develop and deploy scalable RIAs in pure Java using an integrated Ajax framework complete with rich JSF-based components. Harness the power of Ajax and leverage the entire standards-based Java EE ecosystem of familiar tools and runtime environments.

Performance, Scale and Security

Deploy rich enterprise applications with advanced Ajax connection management for maximum application reliability. Seamlessly scale applications across clustered Java EE servers. Extend the existing web security model and avoid transferring sensitive business logic and data to the client browser.

ICESOFT
THE RICH WEB COMPANY

ICESOFT TECHNOLOGIES
Toll Free 1.877.263.3822
www.icesoft.com

Integrating AJAX with JMX

Opposite ends of the Systems Management stack

Graham Paul Harrison

AJAX and JMX are at opposite ends of the Systems Management stack. However, the emerging ubiquity of the AJAX model for rich browser clients has obscured the benefits the model provides in the architectural space for enhancing support patterns within the problem resolution pipeline.

This article elaborates on an architectural benefit of AJAX that lets the management state be 'broadcast' to a browser-enabled user base without waiting for a page refresh.

This architecture is an extension of a general pattern for logging JMX events and properties to a server-side log file; this variation logs or 'broadcasts' management information to the (AJAX-enabled) user base.

Emphasis is placed on the AJAX request/response

model and the painting of management data to the page, together with the beautiful JMX notification framework, all neatly integrated in the middle with an instrumented servlet.

Also included is a cursory view of issues not usually covered in the standard AJAX discussion; namely, security and capacity models.

BEA WebLogic 8.1 was used as the deployment platform for the software, although the architecture and method applies to other J2EE application servers. (The souce code for this article can be downloaded from http://jdj.sys-con.com.)

Key Requirements

The Systems Management stack for Enterprise Java and J2EE applications forms part of the problem resolution pipeline in which the Java/J2EE application interacts with a management tier to monitor potential problems such as application server thread starvation, heap overflow or stale connections to a database.

The management tier usually consists of JMX MBeans, which the application is instrumented to use, together with products such as Wily Introscope to read these JMX properties, and HP OpenView, which can be alerted from Wily if a configured threshold is violated.

One problem with this model is that server-side patterns for JMX management don't help the client at the browser if the system is going down behind him; it's all server-centric. For example, if a new J2EE Web application is to be deployed, or the application is to shut down in a few minutes because Wily has detected a problem, the user at the browser is unaware of management events that are imminent.

By gracefully allowing the user into the problem resolution pipeline, the systems administrator can manage the end-user experience to his advantage by broadcasting management information to the user base and to some extent control the user's behavior.

To communicate management information to clients over HTTP, a problem exists: how can the management aspect send management information to an HTTP client if the user at the client doesn't overtly refresh the page using GET or POST, and if the client doesn't covertly refresh a hidden frame?

Solution Description

Implementing a simple AJAX script that will take management information in the form of an XML message from the MBean server via a servlet solves the problem. This management information must be administered and fed to all participating application servers that can receive AJAX requests.

On the server side:

- A cluster of J2EE application servers is used to service requests from browser-based users, the on-line transaction processing (OLTP) user-base (two or four servers, for example). User requests are load-balanced across this cluster (OLTP cluster) using a third-party Web server.
- A standard MBean (UserWeb) is used to hold management information, e.g., administration message plus metadata properties. The MBean is hosted on both the J2EE 'administration' server and the remaining J2EE servers in the OLTP cluster.
- On the administration server, a system administrator uses the HTMLAdaptor for JMX to set the alert status, retry interval (the interval between XMLHttp-Requests), and alert message; for example, 'System Down in 10 Minutes.' The administrator then broad-casts this state to the MBeans on the managed servers, which reset their state to the master administration state.

On the client side:

- AJAX-enabled clients retrieve the status, retry interval, and message using an XMLHttpRequest, which invokes a servlet to return the relevant MBean values as an XML message.
- JavaScript on the client then parses this XML message, resets the retry interval, and repaints a part of the screen with the management message.
- After retry-interval seconds, the client does another XMLHttpRequest and the client cycle begins again.

Essential Architecture

Figure 1 shows the overall solution architecture. The essential architecture elements are described in Table 1- Architecture Elements.

JMX Notification Model

This model involves two components:

- MBean to raise events for registered listeners, both local and remote
- Listeners, which register themselves with the MBean to listen for events generated by that MBean

The first is implemented by the UserWeb MBean, the second by the ManagementListener.

JMX MBean for Managing User Information

The UserWeb standard MBean is a simple class that contains key properties and methods as shown in Table 2 - UserWeb MBean Properties and Methods.

Event Listener

The Singleton ManagementListener class implements Weblogic.management.RemoteNotificationListener, which extends javax.management.NotificationListener and java.rmi.Remote to allow events in a remote WebLogic JVM to be notified of remote listeners using RMI.

At application server start-up, a listener on each JVM registers itself with the UserWeb MBean on the administration server.

MBean Helper

It's best practice to use a helper class as a façade for Mbeans.This helper is invoked from instrumented code to call MBean methods.

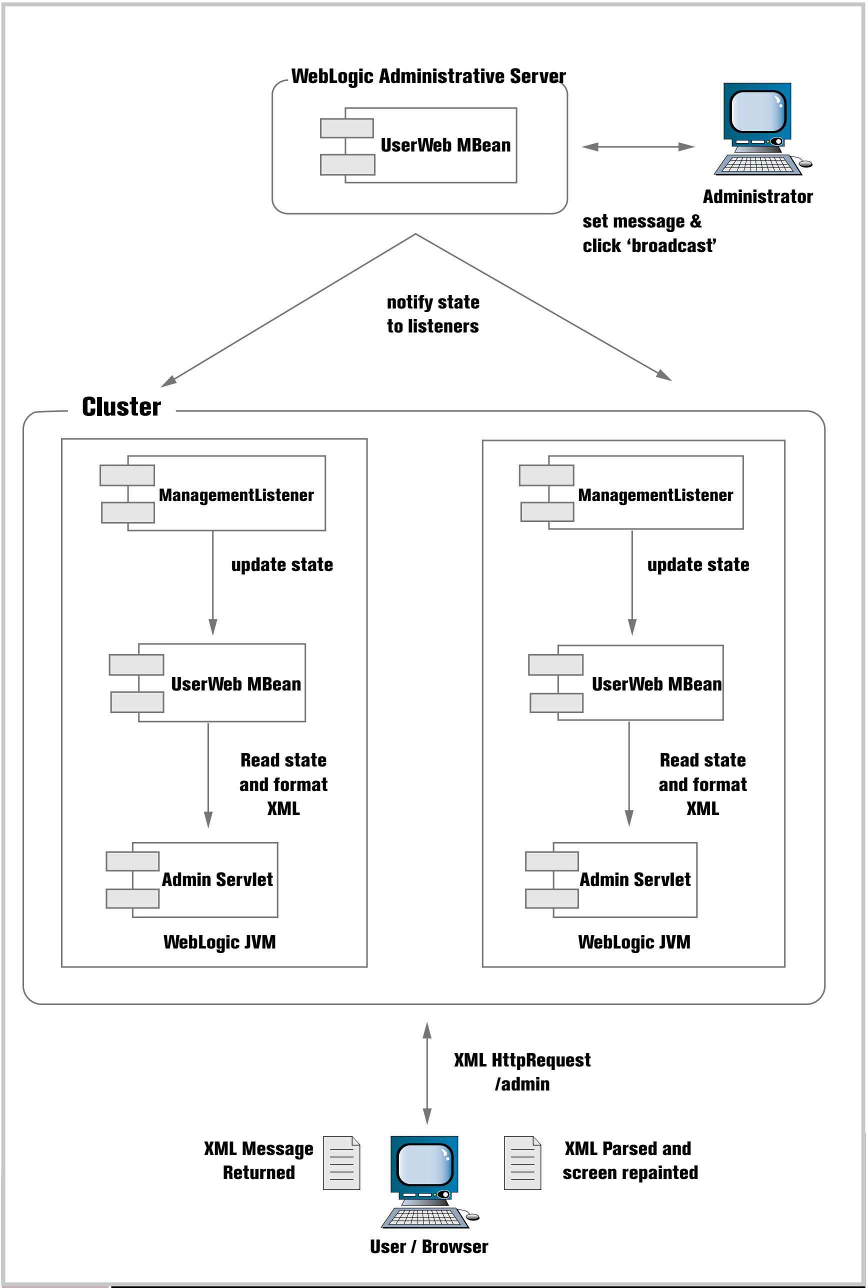


Figure 1: Solution Architecture

Element	Description
AlertEnabled	True if AlertStatus > -1
AlertMessage	The message the user will see repainted on the screen
AlertReady	True if AlertStatus > 0
CallBack	Number of milliseconds between each XMLHttpRequest
BroadcastState	Method that noties all registered listeners on local/remote JVM's with an event (alert.broadcast) and passes the MBean state as event data

Table 1: UserWeb MBean Properties & Methods

The UserWebMBeanHelper class is used as the façade for the UserWeb MBean. The ancestor of all helpers is ApplicationMBeanHelper, which:

- Looks up the local and remote MBean servers
- Invokes those servers to get/set MBean properties and invoke MBean methods

To ensure parity, both the MBean and MBean helper implement the interface UserWebMBean.

Instrumented Servlet

An application is instrumented to use JMX. In AOP terms, the management aspect is woven into the application code. The first JMX instrumentation point for this article is an HTTPServlet. This servlet is the target of the AJAX request and implements a controller pattern that can be elaborated down to handle other AJAX requests using simple request parameters.

From an MVC perspective, the model is the UserWeb Mbean, the view is the AJAX-enabled (JSP) page, and the controller is the instrumented servlet.

Client AJAX Engine

This is a set of JavaScript functions that:

- Manage the XMLHttpRequest and response processing iterations
- Parse the XML message returned by the XMLHttpRequest
- Repaint the screen with the XML message contents

Client Presentation

This is the main.jsp page that contains the client AJAX engine and repaintable section.

Sequence in Action

Essentially, the server sequence is concerned with managing the setting of the management properties and broadcasting these properties to all interested (listening) JVMs. The client sequence is concerned with retrieving these properties and repainting the HTML page with important management information at management-specified intervals.

JMX Notification (Server Sequence)

- UserWeb MBeans and MBean event listeners are created and registered at application server start-up using start-up classes
- The administrator sets the 'master' UserWeb MBean properties (alert message and retry interval), then broadcasts, or notifies, this state to the listeners hosted on the remote managed servers
- The remote listeners handle the notification by copying the master (notification) data to the local UserWeb MBean

XMLHttpRequest Poll (Client Sequence)

- AJAX-enabled clients invoke a servlet at intervals to query management state
- The servlet reads the local UserWeb MBean properties, inserts them into an XML message, and returns the XML message as an XML response to the browser client (alternative message formats are discussed later)
- The AJAX client then parses the XML document, extracts the alert message and retry interval, repaints the screen, and then uses the retry interval to set the delay for the next XMLHttpRequest

Each step is described in detail below.

Register MBeans and MBean Listeners

On each J2EE server instance, two start-up classes are run at server start-up:

- ***ManagementStartup:*** Registers the UserWeb MBean in the local MBean server. Startup class parameters include default settings of the alert status, as well as the MBean name and MBean class. For example:

```
<StartupClass
    Arguments="ServerName=admin,
           MBeanName=ExampleApp:Name=UserWeb,
           MBeanClass=com.grahamh.management.user-
Web.UserWeb"
           ClassName="com.grahamh.management.start-
up.ManagementStartup"
    FailureIsFatal="true" Name="UserWEB" Notes=""
    Targets="admin,OLTPCluster"/>
```

- ***MbeanRegistrations:*** Used to register a Singleton POJO, ManagementListener, with the UserWeb MBean on the administration server.

A javax.management.NotificationFilterSupport object is used to list the notification types that the UserWeb MBean will generate and the listener will receive:

```
// MbeanRegistrations.java
MBeanHelperFactory.getWebHelper().registerListen-
er();

// UserWebMbeanListener.java
```

Infragistics® NetAdvantage® family of products brings you:

No-Touch AJAX

for ASP.NET and JavaServer™ Faces

VISIT US AT AJAXWORLD BOOTH #208

Enter to Win a FREE Sirius Radio & 1-Year Subscription

SIRIUS SATELLITE RADIO

THE BEST RADIO ON RADIO™

Though it's simply a combination of relatively mature technologies, AJAX, or Asynchronous JavaScript and XML, is now commonly used to create highly interactive and responsive web interfaces. The basic idea of AJAX is to open up a new line of communication between the client and server, where data can be transferred asynchronously, usually as XML. The end user no longer waits for the page transfer to complete; bytes are being transferred in the background, eliminating page postbacks. A common usage flow is to use client-side JavaScript to combine the XML results returned by the web server with DHTML to alter what the end user sees in the browser.

AJAX and Infragistics NetAdvantage components

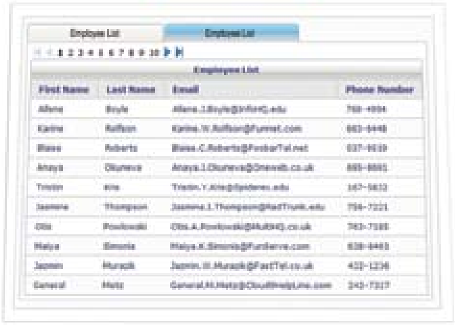
As the leader in rich presentation layer technologies, Infragistics delivered AJAX features in October 2004, well before the term 'AJAX' was coined. In fact, our NetAdvantage customers have been enjoying the benefits of AJAX for almost 2 years.

NetAdvantage for ASP.NET WebGrid™

Infragistics NetAdvantage for ASP.NET and NetAdvantage for JSF components take full advantage of AJAX technology, yielding robust, high performance web applications. Boosting performance, the WebGrid™ has built-in AJAX features including XML LoadOnDemand, sorting, updating, and more. This same incredible performance can be realized with our WebTree™, which also has the AJAX-enabled LoadOnDemand feature. In fact, all of our JSF components and all major ASP.NET components are AJAX-enabled already, with more on the way.

A Word about "Atlas"

The Microsoft® client-side framework, code named "Atlas", provides a solid foundation for creating rich client-side behaviors through JavaScript and simplifies AJAX programming in general. NetAdvantage for ASP.NET includes more than 20,000 lines of JavaScript, as does "Atlas". Due to the large nature of these two frameworks and the fact that they are likely to be used together, compatibility is a top priority. Infragistics is working directly with Microsoft to ensure compatibility between NetAdvantage and "Atlas". We will continue to work through any issues found to ensure that NetAdvantage web components will maintain full compatibility for "Atlas" going forward.



Employee List

First Name	Last Name	Email	Phone Number
Allene	Boyle	Allene.J.Boyle@InfoHQ.edu	768-4994
Karine	Rolfson	Karine.W.Rolfson@Funnet.com	663-5448
Blaise	Roberts	Blaise.C.Roberts@FoobarTel.net	037-9939
Anaya	Olunuva	Anaya.J.Olunuva@Oneweb.co.uk	895-8881
Tristin	Kris	Tristin.Y.Kris@Spiderex.edu	167-5832
Jasmine	Thompson	Jasmine.J.Thompson@RedTrunk.edu	756-7221
Otis	Powlowski	Otis.A.Powlowski@MultiHQ.co.uk	763-7185
Maiya	Simonis	Maiya.K.Simonis@Fundierre.com	638-6463
Jazmin	Murazik	Jazmin.W.Murazik@FastTel.co.uk	432-1236
General	Metz	General.M.Metz@CloudHelpLine.com	243-7317

NetAdvantage for JSF WebGrid™

For more information about Infragistics and AJAX, please visit **www.infragistics.com/ajax**

NetAdvantage® for ASP.NET
www.infragistics.com/asp.net

NetAdvantage® for JSF
www.infragistics.com/jsf

Infragistics®
Powering The Presentation Layer

WINDOWS® FORMS ASP.NET WPF JSF

grids scheduling charting toolbars navigation menus listbars trees tabs explorer bars editors

Listing 1: admin.js

```
var url = './admin?reqid=0';
var baseCtx;
var callBack=null;
var callbackTimeout=null;

function trapAlert() {
  if (window.XMLHttpRequest) {
      req = new XMLHttpRequest();
  } else if (window.ActiveXObject) {
      req = new ActiveXObject("Microsoft.XMLHTTP");
  }

  req.onreadystatechange = processRequest;
  req.open("GET", url, true);
  req.send(null);
}

//
// readyState Status Codes:
//0 = uninitialized
//1 = loading
//2 = loaded
//3 = interactive
//4 = complete

function processRequest() {
    if (req.readyState == 4) {
        if (req.status == 200) {
          parseMessages();
        } else {
          updateStatus(null)
        }
        setCallback(); // only do this when complete
    }
}

function parseMessages() {
 response  = req.responseXML;
 itemStatus = response.getElementsByTagName('status')[0].firstChild.nodeValue;
        itemText = response.getElementsByTagName('textBody')[0].firstChild.
nodeValue;
        paintAlertText(itemText);
        callbackTimeout = parseInt(response.getElementsByTagName('callBack')[0
].firstChild.nodeValue);
        updateStatus(itemText);

}

function paintAlertText(itemText) {
    document.getElementById("adminBanner").innerHTML = itemText;

}
function setCallback() {
  callBack = setTimeout('trapAlert()',callbackTimeout);
}

function initAdmin(uri) {
  setCallback();
}

function updateStatus(msg) {
  if (msg == null ) {
    window.status = "No Alerts";
    document.getElementById("adminBanner").innerHTML = " ";
  } else {
    window.status = "Example Alert: " + msg
  }
}
```

Listing 2: admin.java (servlet)

```
package com.grahamh.management.servlet;
/**
```

```
public void registerListener() throws
UserWebException{

    try {
      // get listener and filter
      ManagementListener listener =
MBeanHelperFactory.getListener();
      NotificationFilterSupport filter = listener.
getSupportedEvents();

      // get admin mbean server;
      //register the listener and filter with the
UserWeb MBean
      RemoteMBeanServer rmbs = getAdminMbeanServ-
er();
      rmbs.addNotificationListener("ExampleApp:
Name=UserWeb",
                                    listener,
filter, null);
    }
    catch (Exception e) {
      throw new UserWebException("Unable to regis-
terListener: "+
                                  e.getMessage(),
e);
    }
}
```

```
 The listener.getSupportedEvents() method returns
the following filter:

    NotificationFilterSupport filter = new Notifica
tionFilterSupport();
    filter.enableType("alert.broadcast");
```

When ManagementListener is run at server start-up, a connection is made to the MBean server on the (remote) administration server and the (local) ManagementListener is registered as a listener on events generated by the UserWeb MBean, with a filter set to "alert.broadcast" event types.

Because the ManagementListener implements Weblogic.management.RemoteNotificationListener , it can get JMX notifications that are generated in either the local JVM or a remote JVM; in this case, generated in the remote administration server JVM.

Broadcast Admin MBean properties

The administration and managed UserWeb MBeans can be set independently, giving any one J2EE server a localized AJAX response. However, a general Operations, Administration & Support (OA&M) support pattern would set the admin MBean properties and then broadcast these properties to the MBeans on the remote application servers, using the Notification model, for subsequent AJAX retrieval.

Because the UserWeb MBean is based on ApplicationMBean, which extends javax.management.NotificationBroadcasterSupport, the infrastructure is in place for the UserWeb MBean to notify all listeners. Hence, the administrator sets the relevant MBean properties (using the HTMLAdaptor) and clicks BroadcastState (see Figure 2).

Consequently, the UserWeb.broadcastState() method is executed, which notifies all listeners synchronously with the state of the admin MBean:

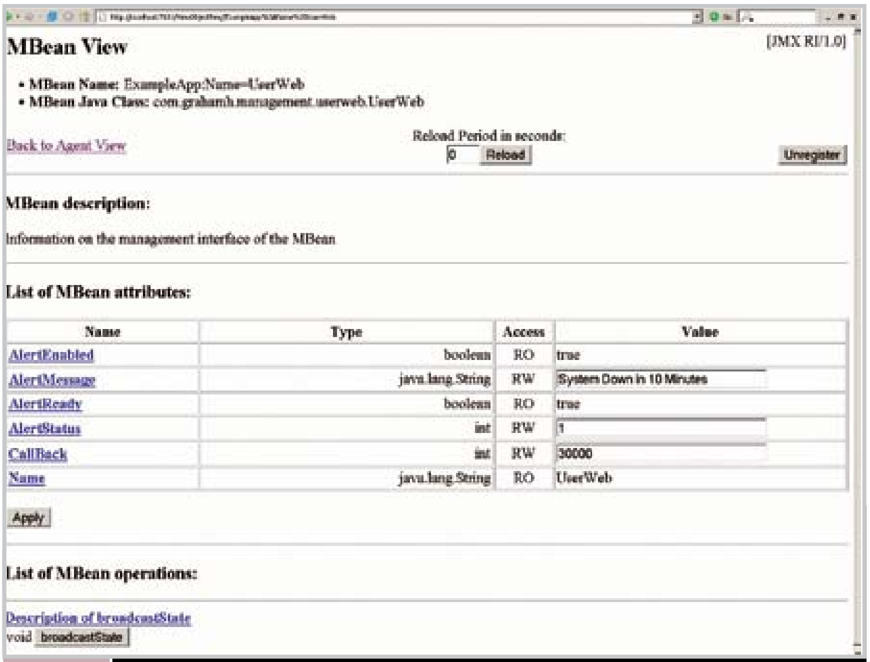
```
public void broadcastState() throws Exception {
   try {
     Notification n = new Notification(“alert.
broadcast”,

 “ExampleApp:Name=UserWeb”, 0);
     n.setUserData(new UserWeb(this));
     this.sendNotification(n);
   }
   catch (Exception e) {
     throw e;
   }
 }
```

Because the data is serialized over the network, the non-transient object graph must be serializable.

Receive Notification of MBean Props via Listener

The event listener is the ManagementListener Singleton. The JMX Notification framework on the administration server makes a remote call to the ManagementListener handleNotification() method in each listener on each of the OLTP cluster JVMs, which registered on server

```
start-up:
  public void handleNotification(Notification noti-
fication, Object              handback) {
    System.out.println(“Received alert: “ + notifi-
cation.get-
   Type());

    // get event userdata from notification
    Object userData = notification.getUserData();

    if (userData instanceof UserWeb) {
      // comes from destin8 Web
      UserWeb WebVo = (UserWeb)userData;
      UserWebMBeanHelper helper =
MBeanHelperFactory.getWeb-
    Helper();

      // get from value object and set into local
MBean using
```



MBean View

[JMX RI/1.0]

- **MBean Name:** ExampleApp:Name=UserWeb
- **MBean Java Class:** com.grahamh.management.userweb.UserWeb

Back to Agent View

Reload Period in seconds:
0 Reload

Unregister

MBean description:

Information on the management interface of the MBean

List of MBean attributes:

Name	Type	Access	Value
AlertEnabled	boolean	RO	true
AlertMessage	java.lang.String	RW	System Down in 10 Minutes
AlertReady	boolean	RO	true
AlertStatus	int	RW	1
CallBack	int	RW	30000
Name	java.lang.String	RO	UserWeb

Apply

List of MBean operations:

Description of broadcastState
void broadcastState

Figure 2 MBean view using HTMLAdaptor

```
    MBeanHelper
      helper.setAlertMessage(WebVo.getAlertMes-
sage());
      helper.setAlertStatus(WebVo.getAlertSta-
tus());
      helper.setCallBack(WebVo.getCallBack());
      helper.setRefreshAlertStatus(WebVo.getRefre-
shAlertStatus());
    }
  }
```

The ‘master’ UserWeb data is set into the local User-Web MBean via its MBean helper. Consequently each managed server is updated with the master UserWeb
state.

That’s as far as the JMX elements need to go.

AJAX Request of Management State

The browser client is AJAX-enabled as follows:

- ***main.jsp*** - instrumented JSP page that checks the (JMX) alert status and polls the server for alerts. It includes admin.js
- ***admin.js*** - JavaScript utilities that use XMLHttpRequest to poll the server for the management state, parse the XML response, and repaint the ‘status’ area of the screen

JavaScript (described below) is included in main.jsp as follows:

```
<script type=”text/javascript” src=”./js/admin.js”
></script>
```

```
 * Admin
 * Processes a request to get various admin states for the OLTP user
 * @author GH
 */

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

import com.grahamh.management.UserWeb.UserWebMBeanHelper;
import com.grahamh.management.utils.MBeanHelperFactory;


public class Admin extends HttpServlet
{
  static final private String CONTENT_TYPE = "text/xml";

  public void init() throws ServletException
  {
  }
  //Process the HTTP Get request
  public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
  {
    /**
     * use requestId in future to switch on admin events
     */
    String requestId = request.getParameter("reqid");
    if ((requestId == null) || (!MBeanHelperFactory.getWebHelper().isAler-
tReady()) )
    {
        //nothing to show
        // there re two models - if nothing to show :
        // 1. dont send anything, or
        // 2. send a message and let the client interpret the
   alertStatus.
        // (1) used, so client just displays an alert if one
   exists.
       response.setStatus(HttpServletResponse.SC_NO_CONTENT);


    } else {
      // only send a message if there is an alert notification to
   send
      StringBuffer sb = new StringBuffer();
      sb.append("<message><status>");
      sb.append(MBeanHelperFactory.getWebHelper().getAlertStatus());
      sb.append("</status><textBody><![CDATA[");
      sb.append(MBeanHelperFactory.getWebHelper().getAlertMessage());
      sb.append("]]></textBody>");
      sb.append("<callBack>");
      sb.append(MBeanHelperFactory.getWebHelper().getCallBack());
      sb.append("</callBack></message>");

      response.setContentType( "text/xml" );
      response.setHeader( "Cache-Control", "no-cache" );
      PrintWriter out = response.getWriter();
      out.write(sb.toString());
      System.out.println(sb.toString());
    }
  }
  //Process the HTTP Post request
  public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
  {
    doGet(request, response);
  }
  //Clean up resources
  public void destroy()
  {
  }
}
```

Rather than poll continuously, we will only poll if alerting is enabled. We use the UserWebMBeanHelper to check this status. If enabled, the JavaScript function initAdmin() is invoked when the page loads:

```
<%
if (MBeanHelperFactory.getWebHelper().isAlertEn-
abled()) {
%>
        <body bgcolor="#F4FFE4"
onload="initAdmin();">
<%
} else {
%>
        <body bgcolor="#F4FFE4">
<%
}
%>
```

The repaintable 'status' area of the screen is defined as follows:

```
<span id="adminBanner" class="style1"></span>
```

'adminBanner' will be used to identify the repaintable area when the XML response is parsed and the message extracted.

The initAdmin() method schedules a JavaScript method, trapAlert(), to be executed after callbackTimeout milliseconds:

```
function setCallback() {
  callBack = setTimeout('trapAlert()',callbackTime
out);
}

function initAdmin() {
  setCallback();
}
```

It's the trapAlert() method that initiates the XMLHttpRequest with the now familiar ring:

```
function trapAlert() {
  if (window.XMLHttpRequest) {
      req = new XMLHttpRequest();
  } else if (window.ActiveXObject) {
      req = new ActiveXObject("Microsoft.XMLHTTP");
  }

  req.onreadystatechange = processRequest;
  req.open("GET", './admin?reqid=0', true);
  req.send(null);
}
```

HTTP GET is used to read data (only a small request parameter is used), and the admin servlet is

BACKBASE

AJAX for Java

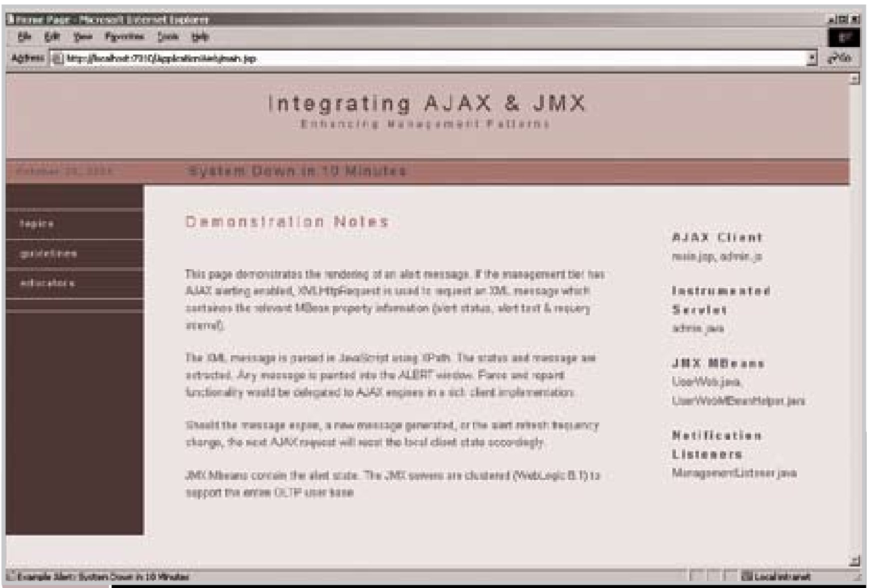
Backbase offers a comprehensive AJAX Development Framework for building Rich Internet Applications that have the same richness and productivity as desktop applications.

The Backbase AJAX JSF & Struts Editions:

- are based on JavaServer Faces (JSF) or Struts/JSP
- run in all major Application Server
- support development, debugging and deployment in Eclipse
- embrace web standards (HTML, CSS, XML, XSLT)

Download a 30-day Trial at www.backbase.com/jsf

i: www.backbase.com **t:** (866) 800-8996 **e:** sales-us@backbase.com


Integrating AJAX & JMX

Enhancing Management Patterns

System Down in 10 Minutes

Demonstration Notes

This page demonstrates the rendering of an alert message. If the management tier has AJAX alerting enabled, XMLHttpRequest is used to request an XML message which contains the relevant MBean property information (alert status, alert text & requery interval).

The XML message is parsed in JavaScript using XPath. The status and message are extracted. Any message is painted into the ALERT window. Parse and repaint functionality would be delegated to AJAX engines in a rich client implementation.

Should the message expire, a new message generated, or the alert refresh frequency change, the next AJAX request will reset the local client state accordingly.

JMX MBeans contain the alert state. The JMX servers are clustered (WebLogic 8.1) to support the entire OLTP user base.

AJAX Client
main.jsp, admin.js

Instrumented Servlet
admin.java

JMX MBeans
UserWeb.java
UserWebMBeanHelper.java

Notification Listeners
ManagementListener.java

Example Alert: System Down in 10 Minutes

Figure 3: Repainted screen

targeted. The request is asynchronous, and when the request state changes, the processRequest JavaScript function is invoked:

```
  req.onreadystatechange = processRequest;
```

It might seem reasonable to wait for a response before continuing processing. However, you run the risk of having your script hang if a network or server problem prevents completion of the transaction. An asynchronous call with the onreadystatechange event is more resilient.

As the request cycles through to completion, the processRequest event handler is invoked:

```
function processRequest() {
     if (req.readyState == 4) {
          if (req.status == 200) {
            parseMessages();
          }
....
          setCallback(); // only do this when com-
plete
     }
}
```

Listing 1 shows the available status codes. When the request is complete and HTTP status code 200 (OK) is returned, the parseMessages() method is called to extract the data from the XML message. Then the trapAlert() method is rescheduled again. If the XML response has a different retry interval, this will have been set by the parseMessages() function.

Parse XML Response and Repaint Screen

The parseMessages() function first extracts the XML response

and extracts the elements for alert status, alert text, and retry interval:

```
 itemStatus = response.getElementsByTagName('status'
)[0].firstChild.nodeValue;
 itemText = response.getElementsByTagName('textBody'
)[0].firstChild.nodeValue;
 callbackTimeout = parseInt(response.getElementsByTa
gName('callBack')[0].firstChild.nodeValue);
```

The alert text is then repainted on to the adminBanner document element (see above):

```
        document.getElementById("adminBanner").
innerHTML = itemText;
```

The alert message appears on the screen as shown in Figure 3.

Servlet Formats the XML Response

For the browser to display management alerts to the user, XMLHttpRequest is used to request the management state. When the browser sends the request, the servlet uses the MBean helper to check the alert state and, if an alert is available, constructs an XML document as a response.

If there's no state to return, the response status is set as follows:

```
response.setStatus(HttpServletResponse.SC_NO_
CONTENT);
```

Otherwise the text/XML response type is set:

```
response.setContentType("text/xml");
```

Listing 2 shows the servlet method in full.

When the servlet is invoked and the XML content returned, the console should print:

```
Received alert: alert.broadcast
<message><status>1</status><textBody><![CDATA[System
Down in 10 Minutes]]></textBody><callBack>10000</
callBack></message>
```

Capacity Modelling and Security

Because AJAX opens up the architecture in interesting ways, two key areas require elaboration:

- Capacity Modelling
- Security

Caching and response message type (XML or text) can also be important.

Capacity Modelling

AJAX-enabled rich clients won't necessarily submit requests any more frequently than before. But with XMLHttpRequest executed asynchronously in the browser, the number of HTTP requests to the server increases in line with the retry interval.

- Retry interval (Think Time) = 20 seconds
- Number of connected users = 5000
- Transactions per second (TPS) = 5000/20 = 250

We expect an extra 250 requests (transactions) per second generated from the HTTP user base.

Of course, it depends what these requests do at the server to increase latency in response time. In our example, each request must look up MBean properties and format an XML response, but the response is very small and the MBean is in local memory. With each Web server thread able to handle approximately 200 GET requests a second, and the user requests load balanced across a J2EE server cluster of perhaps 200 threads, the increased loading isn't significant.

When modelling AJAX architectures, increased load injection can be offset by reduced bandwidth, since the response contains data only in contrast to data plus mark-up.

Security

Suppose you only wanted users in the WebUser group to access the Admin servlet?

If only authenticated users can access the admin servlet, then the XMLHttpRequest will run as that user if that user has authenticated.

For example, once user Joe logs into the application, and Joe is a member of group WebUser, the XMLHttpRequest will be able to invoke the Admin servlet.

Adding the following code to the admin servlet would confirm the authenticated subject, returning true and Joe respectively:

```
request.isUserInRole("WebUser");
request.getRemoteUser();
```

Caching

Some users have found that IE will cache the response from the AJAX request. This could be due to browser/page settings, but a forced workaround is to timestamp the URL:

```
var urlstr = "./admin?reqId=0&ts=" + new Date().
getTimeStamp();
```

To XML or Not to XML

Not to XML. Some AJAX designers happily dispense with XML and send the response as plain text:

```
response.setContentType("text/plain");
```

"AJAX presents architectural opportunities that shouldn't be overshadowed by the rich-client frenzy"

This clearly depends on your client-side requirement and how loosely coupled the client is from the data required. A simple text response would suffice for a text alert. However, the advantage of the XML model for this article is that the response data can be elaborated further to refine both the status and status-related data. This article demonstrates how you'd parse a more complicated response that the client may have to code to accept.

Conclusion

AJAX presents architectural opportunities that shouldn't be overshadowed by the rich-client frenzy. This article has leveraged the architectural advantage AJAX can provide while addressing the technical requirements for both capacity and security that more exotic uses of this technology will demand.

References

- http://java.sun.com/j2ee/1.4/docs/api/javax/management/NotificationFilterSupport.html - Details the alert types functionality
- http://e-docs.bea.com/wls/docs81/jmx/notifications.html - Details the use of WebLogic MBean Notifications for remote listeners
- http://e-docs.bea.com/wls/docs81/ConsoleHelp/startup_shutdown.html - Configuring start-up and shutdown classes in WebLogic Server 8.1 ■

Graham Paul Harrison is the author of Dynamic Web Programming and a former senior consultant with BEA Systems in Europe. He now works as a freelance J2EE architect, specializing in performance.
gpharrison@btinternet.com

Security and AJAX

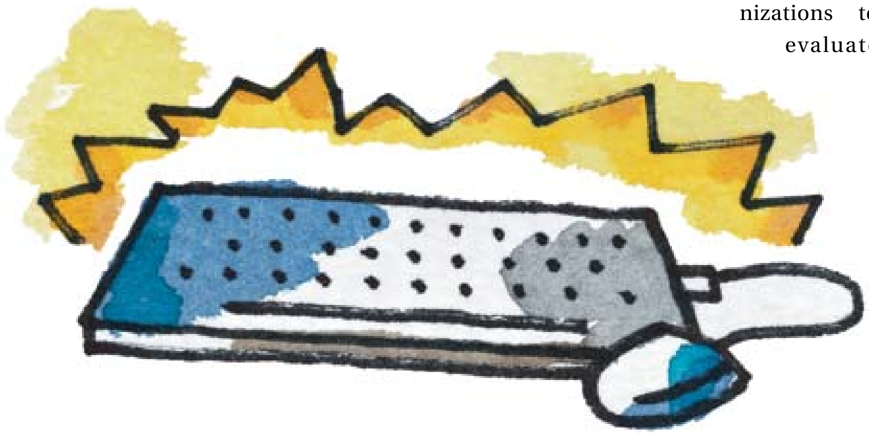
Best practices

Samir Kapuria

Flexible software development approaches such as AJAX are making it easier for developers to deliver fast and responsive interactive Web applications. With AJAX, users no longer have to wait while an entire Web page reloads after they make a change. This performance shift allows new data to be called up almost as soon as it is input.

From a response perspective, it's almost like working with a desktop application.

However, the use of AJAX also opens up a number of security challenges for organizations to evaluate



as they consider leveraging this robust architecture to enhance their Web services offerings.

The potential vulnerabilities associated with AJAX are especially important to consider in today's Internet-based business world. Information technology serves as the foundational vehicle for business communications, operations, transactions, and more. Consumers, too, are taking advantage of Internet connections to bank, shop, manage accounts, and interact with their institutions of choice from the comfort of their own homes.

At the same time, threats to these highly interactive and responsive Web environments are appearing on the Internet landscape. Vulnerabilities in Web applications and browsers and risks associated with AJAX-style application environments may present organizations with serious challenges to security. To protect against such risks, organizations must understand the issues surrounding AJAX security and follow best practices for more secure Web application development.

AJAX Pressure Points

Moving business logic from the server to the client presents a number of potential security risks with AJAX. These risks range from performance problems to exposing applications to Web services vulnerabilities, the implications of which can be serious.

For example, system-wide performance degradation can occur as the amount of XML network traffic increases. The steady parsing and exception handling resulting from malformed messages can also lead to server performance disruptions. The asynchronous nature of AJAX makes denial of service (DoS) attacks a possibility. And Web browsers can be used by attackers to send corrupted data.

In addition, improper input validation can cause Web application components to crash. The complexities of cryptographic functions and coding may make it difficult for programmers to produce strong protection. Attackers can exploit flaws in access controls to get at confidential data, use functions they are not authorized to use, and more. Improper error handling can be exploited by an attacker to access system information, bring servers down, or make security mechanisms ineffective.

In October 2005, such concerns about AJAX vulnerabilities were highlighted when an AJAX worm was able to infect the MySpace network by bypassing security checks on the server. But AJAX vulnerabilities are not an organization's only concern. Software vulnerabilities in general are steadily increasing—and the volume has never been higher.

Vulnerabilities in Web Applications

First, the good news. During the second half of 2005, there was only a slight increase in the total number of vulnerabilities disclosed over the previous six months—from 1,871 to 1,896. That's according to the latest Internet Security Threat Report from Symantec Corp. The semiannual report provides a six-month update of Internet threat activity and includes an analysis of network-based attacks, a review of known vulnerabilities, and highlights of malicious code and additional security risks. The current report covers the period from July 1 to Dec. 31, 2005.

Now, the bad news. During all of last year, the total volume of vulnerabilities reached 3,767—the highest yearly total volume recorded since 1998.

And to what is much of this growth attributed? Web application vulnerabilities. Of the vulnerabilities disclosed during the last half of 2005, 69 percent were associated with Web applications. This number reflects the shift toward the Web as a platform for applications that were previously stand-alone software suites or client-server solutions. And because traditional security infrastructures allow Web traffic onto a network by default, organizations that host Web applications are often left exposed to attacks that are both difficult to detect and to prevent.

Web application vulnerabilities are a serious security concern because they are typically exposed to the Internet through Web servers, which are often the external face of an organization on the Internet. Web servers make a popular target because attackers can exploit them to steal information that passes through them, such as credit card and bank information. Web servers can also serve as potential jump-off points into databases that hold sensitive client or user information. Compromised Web servers can also be used to host phishing sites and to launch attacks against Web browsers that access them. According to the Symantec report, such attacks are becoming more prominent.

The rise in Web application vulnerabilities is likely due to the ease with which they can be introduced into source code. Also, small Web applications are often developed on an ad hoc basis. And while AJAX-type development methods provide a rich set of application capabilities, allowing more people to develop such programs in a shorter period of time, not all developers may be trained to incorporate security in the programs they develop.

The first examples of malicious code that propagates by exploiting vulnerabilities in Web-based applications or services were seen during the second half of 2005. These included the first known Web application worm, Perl.Santy, which affected the widely deployed phpBB forum. The next was a segment of JavaScipt code that quickly spread through Web sites of MySpace users by taking advantage of a vulnerability in the social networking Web site, as previously mentioned. And the most recent example of propagating code exploited a vulnerability in the Mambo content management system.

Browser Blemishes

The Web browser is a crucial application that has security implications for AJAX and other Web application environments. Vulnerabilities in Web browsers can allow attackers to circumvent traditional perimeter security devices. Their ubiquity in homes and businesses across the country makes the exploitation of Web browser vulnerabilities one of the most effective ways to attack users.

During the last half of 2005, 24 new vendor-confirmed and non-vendor-confirmed vulnerabilities were disclosed that affected at least one version of Microsoft Internet Explorer. During this same period, the Mozilla Firefox browser was affected by 17 new vendor-confirmed and non-vendor-confirmed vulnerabilities.

Attacks that target Web browser and Web application vulnerabilities are often conducted by HTTP and, therefore, may bypass filtering mechanisms in place on the network perimeter. And the widespread deployment of Web applications and Web browsers gives attackers a large number of easily exploitable targets. For example, Web browser vulnerabilities can lead to the exploitation of vulnerabilities in operating system components and individual applications, which can lead to the installation of malicious code, including bots.

In fact, both Web browser and Web application vulnerabilities may create the potential for large increases in bots and bot networks. Bot networks are groups of compromised computers on which attackers have installed software that listens for and responds to commands. Bots can have a number of effects on all Internet users, including home users, small businesses, and large organizations. A single infected host within a network can allow a bot to propagate to other computers that are normally protected against external attacks by corporate firewalls. Bots can be used by external attackers to perform denial of service (DoS) attacks against the enterprise's Web site. And bots within an organization's network can be used to attack other organizations' Web sites, which can have serious legal consequences.

Keeping It Clean

As the number of Web application and browser vulnerabilities grows, they will likely serve as a more attractive target for potential attackers to exploit. And, because the Web is a popular medium for the

Samir Kapuria is the director of strategy practice at Symantec Global Security Consulting.

delivery of products and services, understanding and securing against Web-based attacks is an important security objective.

As a result, organizations must manage their Web-based assets carefully. If they are developing Web applications in-house, developers should be educated about secure development and the use of secure shared computers. If possible, all Web applications should be audited for security prior to deployment.

In working toward more secure AJAX and similar Web applications, developers must recognize that vulnerabilities can occur anywhere in the application lifecycle as well as throughout all of the components of an application. A Web server, database, business logic, application server, and operating system are all required to make up a Web application. Each component can be coded or configured differently, but all of the components come together to form a Web application. The potential attack vectors for such applications are tied to each technology component and type.

What's more, the information associated with any given Web application is vulnerable as it passes through the stages of the information lifecycle. The first phase is creation, wherein the user enters data into a Web application, which then puts the information in a temporary format while processing it into the other application components. The second phase is information transfer, wherein data is moved to a backend database. The third phase, storage, represents another potential vulnerability as the application running on the database writes data to a hard drive. Information is again vulnerable during the final storage--the retrieval phase--when the user subsequently views his or her data via the browser.

Developers of AJAX-based applications can help mitigate risk to their technology infrastructure by implementing server-side validation, keeping business logic on the server, being aware of and checking for known attacks, like SQL injections, crossing site scripting vulnerabilities, and by ensuring that each request is authentic and authorized.

In addition, while many security risks are associated with application vulnerabilities, others are the result of poor operational practices such as improper data storage, unpatched software, lost backup tapes or laptops, mail theft, social engineering, and more.

Needless to say, to provide a more protected Web services environment, businesses must not only apply best practices for developing secure AJAX applications but they must also put in place the policies and procedures for mitigating risk throughout their organizations. This includes understanding their risk posture regarding applications—including regulatory compliance issues, liabilities, and operational dependencies—and then defining their desired risk posture, developing a plan to achieve that posture, and then following it.

Guarding the Gateway

Digital interactions have become ubiquitous. Consumers entrust organizations with their financial data, email messages, and family photos. And organizations house more and more sensitive customer and corporate data in Web applications. And customers demand faster service from their Web services, opting for rapid interactions that respond almost as soon as they click a key.

At the same time, vulnerabilities in Web application technologies are causing organizations to evaluate the risks associated with new, more flexible development techniques such as AJAX. Attackers have quickly recognized that targeting Web vulnerabilities allows them to access data—including everything from financial data to internal employee data—much more quickly than was possible through network-level attacks.

Compounding these risks are vulnerabilities in Web browsers, which may enable attackers to bypass security mechanisms to exploit operating systems and applications, and even install bot software.

To continue to leverage the robust and flexible AJAX architecture to provide the world-class services that customers require, organizations must demonstrate that they are trusted partners by taking steps to improve customer confidence and incorporating security throughout the lifecycle of the application. Security must also be evaluated in every component of a Web application, from the Web server to the database, business logic, application server, and operating system.

Secure application development is complemented by best practices for mitigating risk across the enterprise. Organizations must put in place and follow a risk management policy that addresses security as it applies not only to technology but to people and processes as well.

In today's technology-based culture, applications will continue to serve as the gateway to information—a company's greatest asset and a hacker's favored target. Identifying application vulnerabilities and determining how to protect against their exploit, in turn, will remain a priority as enterprises move critical applications and services to the Web.

By following best practices for more secure code and a more protected Web services infrastructure, organizations can leverage innovative and rich technology platforms such as AJAX to provide the highly responsive interaction customers demand—in a safer, more secure environment. ■

Complex and evolving systems are hard to test...

Parasoft helps you code smarter and test faster.

Start improving quality and accelerating delivery with these products:

Awarded **"Best SOA Testing Tool"** by Sys-Con Media Readers	**InfoWorld's 2006 Technology of the Year pick** for automated Java unit testing.	Automated unit testing and code analysis for C/C++ quality.	Memory errors? Corruptions? Leaks? Buffer overflows? Turn to…	Easier Microsoft .NET testing by auto-generating test cases, harnesses & stubs	Automate Web testing and analysis.
SOAtest™	Jtest™	C++test™	Insure++™	.TEST™	WebKing™

PARASOFT®

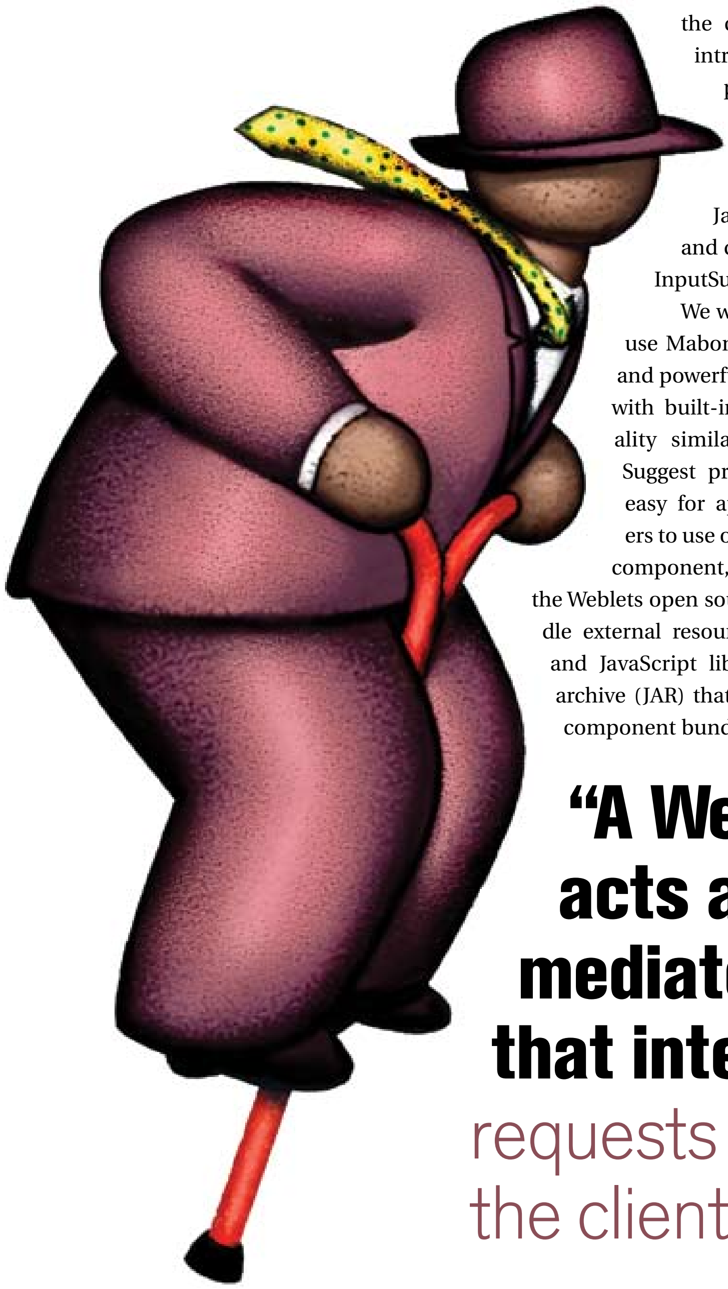
We make software work.™

Go to www.parasoft.com/WSJmagazine • Or call (888) 305-0041, x3501

JavaServer Faces and AJAX for Google Fans

Create your own custom components and build RIAs

Jonas Jacobi John Fallows



This is our last article in a series of four that have been introducing the concepts of creating AJAX-enabled JavaServer Faces (JSF) components. In this article we are going to summarize and encapsulate the concepts that were introduced in the three previous articles starting with the "Rich Internet Components with JavaServer Faces", and design a Google-like InputSuggest component.

We will show you how to use Mabon to create a simple and powerful input component with built-in suggest functionality similar to what Google Suggest provides. To make it easy for application developers to use our JDJ InputSuggest component, we are going to use the Weblets open source project to bundle external resources, such as icons and JavaScript libraries, into a Java archive (JAR) that represents our JSF component bundle.

"A Weblet acts as a mediator that intercepts requests from the client"

Creating an AJAX-Enabled JSF Input Suggest Component

The JSF AJAX input suggest solution consists of four classes as shown in Figure 1.

These classes are as follows:

- The HtmlInputSuggest is the renderer-specific subclass.
- The HtmlRenderer superclass provides some convenient methods for encoding resources.
- The HtmlInputSuggestRenderer is your new custom Renderer, which is in charge of the markup rendered to the client, including resources needed such as JavaScript libraries and style sheets.
- The HtmlInputSuggestTag is the tag handler.

The part of our input suggest solution that will provide the richness is a JavaScript library - inputSuggest.js - that contains functions needed to leverage Mabon to retrieve data from the application developer's backing bean. We'll focus on the following artifacts - the inputSuggest.js file and the HtmlInputSuggestRenderer - both impacted by Mabon and provide the input field with type-ahead and suggest list functionality.

The Input Suggest JavaScript Library

Since we use Mabon, there is no need to worry about fetching data from the backing bean. We can leave this to Mabon. What we do need to be concerned about, though, is how to handle the returned data on the XMLHttpRequest object, how to populate the actual suggest list, and how to handle user interactions. The inputSuggest.js library contains a number of functions that are used to handle keyboard navigation and mouse interactions, and for space limitations we'll be focusing on the functions that have the most impact on the JSF HtmlInputSuggest component.

The doKeyPress Function

The doKeyPress function, as shown in Listing 1, handles keypress events and checks whether the user pressed the TAB key or not. The TAB key would, under normal circumstances, navigate out of the input field and raising the blur event. For an input suggest solution, a TAB key can also be used to select an active row in the list of suggestions and as such we need to trap the TAB key and select a row in the suggest list and add the value to the input field, or, if no list is available, navigate out of the input field. If navigation occurs, the doBlur() function will be invoked and close the list of suggestions.

The doKeyUp Function

This function is invoked on any keyup event, and, depending on which key was activated, it will perform certain actions. Even though the TAB key has been managed by the doKeyPress function, we still have to make sure to catch it and terminate the process.

Another aspect of key strokes is the UP and DOWN arrow keys. Normally these keys will cause the cursor to navigate left and right in a regular input field. With an input suggest component a user is expecting these keys to navigate the list of suggestions.

The doKeyUp function's most important part, for this article, is to catch all key strokes that end up with a new character entered in the input suggest field, and is not a Backspace (see Listing 2). In this case the doKeyUp function will invoke the blur() function, which evaluates the value entered by the user and if changed raise a change event. We also have to reset the focus to the input suggest field, input.focus(), so the user can continue to enter more text.

If a user enters a new value, the doChange() function is invoked (see Listing 3). The doChange() function will call the mabon.send() function, passing a Map containing information about the value entered by the user, the JSF managed bean to invoke, and the callback function for this solution - _callback().

The _callback function, as shown in Listing 4, is responsible for handling the returned result from the response object and creating a list with suggestions according to the value entered by the user. Since we asynchronously communicate with the server, there is a chance that the user has entered a new character before the response comes back. To ensure that we can handle this, we have added a timer that delays the type-ahead feature until a certain time has passed.

If the value has changed since the request was made or the value is the same as the suggested value, we do nothing (see Listing 5). If the value is the same as the initial value entered, we add the first suggested value in the list to the input field and highlight the part that is appended to the value entered by the user.

The HtmlInputSuggestRenderer

It's time to have a look at how we can leverage the client-side functionality provided by the inputSuggest.js library and encapsulate it into one single JSF component. In this sample the main player is the JSF Renderer - HtmlInputSuggestRenderer. The Renderer is responsible for making sure that the correct markup is written to the client including any references to additional resources such as JavaScript libraries, CSS, icons, etc.... As we described in our previous JDJ article - "JSF and AJAX: Introducing a new open source project" (Vol. 11, issue 1) - you can use Weblets to package these resources into the same library as your JSF components.

Using Weblets

The open source Weblets project (http://weblets.dev.java.net) aims to solve the resource-packaging problem in a generic and extensible way so that all JSF component writers can leverage it, and it places no additional installation burden on the application developer.

A Weblet acts as a mediator that intercepts requests from the client and uses short URLs to serve resources from a JAR file. Unlike the servlet or filter approach, a Weblet can be registered and configured inside a JAR file, so the component library renderers, their resource files, and the Weblet con-

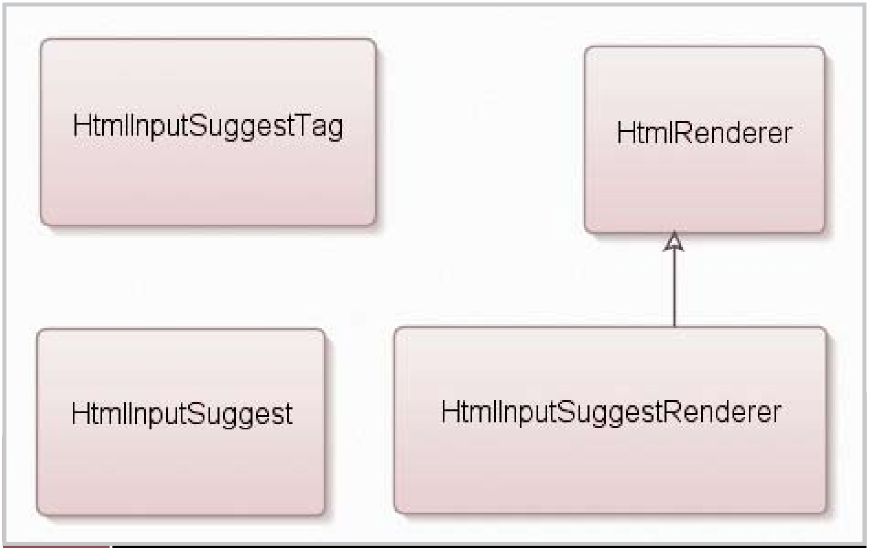


Figure 1: Class diagram showing classes needed for the input suggest component

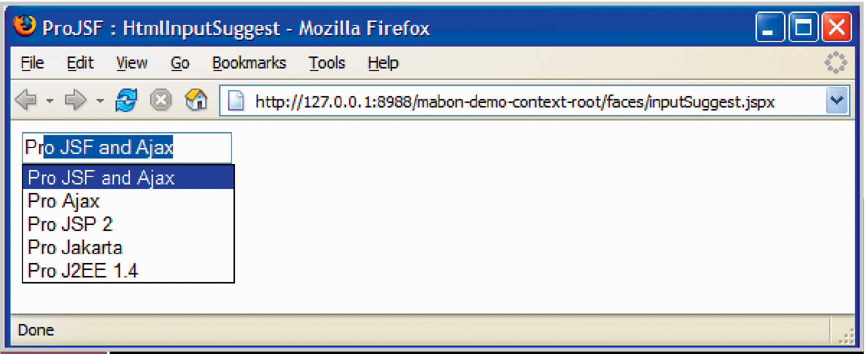

ProJSF : HtmlInputSuggest - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://127.0.0.1:8988/mabon-demo-context-root/faces/inputSuggest.jspx
Pro JSF and Ajax
Pro JSF and Ajax
Pro Ajax
Pro JSP 2
Pro Jakarta
Pro J2EE 1.4
Done

Figure 2: The HtmlInputSuggest component rendered in a browser

Listing 1: the doKeyPress function

```
projsf.jdj.doKeyPress = function(
  event)
{
  var input = (event.srcElement || event.target);
  var inputId = input.id;
  var div = document.getElementById(inputId + “$suggest”);
  var divStyle = (div.currentStyle || div.style);
  if (event.keyCode == 9 && divStyle.display == “block”)
  {
    div.style.display = ”none”;
    var activeRow = projsf.jdj._findActiveRow(div);
    input.value = activeRow.innerHTML;
    return false; //Cancel Tab out
  }

  return true; //Proceed with Tab out, which will invoke the doBlur()
}
```

Listing 2: The doKeyUp function

```
projsf.jdj.doKeyUp = function(
  event)
{
  var input = (event.srcElement || event.target);
  var inputId = input.id;

  var div = document.getElementById(inputId + ”$suggest”);
  if (event.keyCode == 9)//Tab key
  {
    return false;
  }
  else if ((div.style.display == ”block” || div.childNodes.length > 0) &&
           (event.keyCode == 40 || event.keyCode == 38))
  {
    if (div.style.display == ”none”)
    {
      div.style.display = ”block”;
    }
    else
    {
      var activeRow = projsf.jdj._findActiveRow(div);

      switch (event.keyCode)
      {
        case 40: // Arrow Down
          if (activeRow.nextSibling)
          {
            activeRow.className = ”HtmlInputSuggestRow”;
            activeRow = activeRow.nextSibling;
            activeRow.className = ”HtmlInputSuggestActiveRow”;
          }
          break;

        case 38: // Arrow Up
          if (activeRow.previousSibling)
          {
            activeRow.className = ”HtmlInputSuggestRow”;
            activeRow = activeRow.previousSibling;
            activeRow.className = ”HtmlInputSuggestActiveRow”;
          }
          break;
      }

      input.value = activeRow.innerHTML;
    }

    return false;
  }

  if (event.keyCode != 8)//Not a Backspace
```

figuration file (weblets-config.xml) can all be packaged together in the same JAR file. You don't need to separately deploy additional installables when the component libraries are upgraded to new versions. For the application developer, no configuration steps are needed.

It's important to note that all resources served up by Weblets are internal resources, used only by the Renderer. Any resources, such as images, that are provided by the application are supplied as component attribute values and loaded from the context root as external resources.

The HtmlInputSuggestRenderer Class

The two most important methods in this HtmlInputSuggestRenderer are the encodeBegin() and encodeEnd() methods. The encodeBegin() method, as shown in Listing 6, is responsible for writing out the needed resources for this component. The writeScriptResource() method and writeStyleResource() method are convenience methods provided by the HtmlRenderer and provide "write-only-once" semantics, preventing the same library from being written multiple times in case the application developer adds more than one input suggest component to the page.

Using Mabon

Mabon is an open source project hosted on the http://mabon.dev.java.net Web site. Mabon offers a convenient way to hook in a specially designed life cycle that is ideal for AJAX-enabled components that need to fetch data directly from a backing bean, without the overhead of a full JSF life cycle. It also provides a Mabon protocol - mabon:/ - that is used to reference the backing bean and a JavaScript convenience function that is used to send the target URL and any arguments needed and then asynchronously receive data from the managed bean.

Mabon and JSON

As you know, the XMLHttpRequest provides two response types - responseText and responseXML - that can be used to fetch data. The question to ask is, when should I use each? Answers to this question can differ depending on whom you ask, but we can recommend one rule. Ask yourself whether you control the syntax of the response.

The responseXML type returns a complete DOM object (which gives you ample ways of walking the DOM tree), allowing you to find the information needed and apply changes to the current document. This is useful when your component will impact surrounding elements and you don't control the response (for example, when you're communicating with a Web service).

For the input suggest component, you do control the response and you are looking at only fetching data for your component, not modifying the whole page's DOM structure. The responseText type returns plain text, which allows you to leverage JSON syntax for the response. For components leveraging AJAX, JSON is an extremely useful data-interchange format, since it can be easily parsed with the eval() function.

The eval() function takes one argument, a string of JavaScript code, and parses and executes this string in one go rather than trying to process each part separately. This is significantly faster than any other type of parsing, such as XML DOM parsing.

This is the reason why Mabon implements JSON - you control the response, and JSON syntax is easy and fast to parse.

The encodeEnd() Method

The real "work" is done in the encodeEnd() method, as shown in Listing 7. In the encodeEnd() method we get the Map of attributes from the HtmlInputSuggest component. One of the attributes on this component is the doSuggest attribute. From this attribute we can get hold of the MethodBinding, if any, and from the MethodBinding object we can get hold of the actual MethodBinding expression defined by the application developer, for example, #{backingBean.doSuggest}. We then trim the #{} from the

```
  {
    input.blur();
    input.focus();
  }

  if (input.value.length <= 2)
    div.style.display = "none";
}
```

Listing 3: The doChange function

```
projsf.jdj.doChange = function(
  event,
  doSuggestURL)
{
  var input = (event.srcElement || event.target);
  var inputId = input.id;

  var context = { _inputId: inputId };
  net.java.dev.mabon.send({ url: doSuggestURL,
                            args: [input.value],
                            callback: function(result) {
                              projsf.jdj._callback.call(context,
                                                           result);} });
  return true;
}
```

Listing 4: The _callback function

```
projsf.jdj._callback = function(
  results)
{
  var inputId = this._inputId;
  var input = document.getElementById(inputId);
```

ComponentArt

Web.UI™ for ASP.NET

Next-Generation Web Application Development, Today.

See the new breed of AJAX-enabled controls for ASP.NET and Microsoft's forthcoming "ATLAS" framework at AjaxWorld **Booth #207**

and

Attend the Technical Session "Client-centric Web Programming with ComponentArt Web.UI" on Wednesday, October 4th at 10:35 am.

ComponentArt
www.componentart.com

```
  var div = document.getElementById(inputId + "$suggest");

  if (results.length <= 1)
  {
    div.style.display = "none";
    return;
  }

  // get the input from the context
  var input = document.getElementById(inputId);
  div.style.width = input.offsetWidth;
  while (div.firstChild)
  {
    div.removeChild(div.firstChild);
  }

  for (var i=0; i < results.length; i++)
  {
    var row = document.createElement("div");
    var span = document.createElement("span");
    var text = document.createTextNode(results[i]);
    row.className = "HtmlInputSuggestRow";
    row.appendChild(text);
    row.onmouseover = new Function("event",
                                   "projsf.jdj._doMouseOver(event ||
                                                      window.event)");
    row.onclick = new Function("event",
                               "projsf.jdj._doMouseClick(event ||
                                                      window.event)");
    div.appendChild(row);
  }

  div.firstChild.className = "HtmlInputSuggestActiveRow";

  div.style.display = "block";

  window.setTimeout("projsf.jdj._selectText('" + inputId + "', " +
                                           "'" + input.value + "', " +
                                           "'" + results[0] + "')",
                    200);
}
```

Listing 5: The _selectText Function

```
projsf.jdj._selectText = function(
  inputId,
  initialValue,
  suggestion)
{
  var input = document.getElementById(inputId);
  if (input.value != initialValue)
    return;

  if (input.value == suggestion)
    return;

  if (input.createTextRange)//IE Specific
  {
    var selectionStart = input.value.length;
    input.value = suggestion;
    var range = input.createTextRange();
    range.moveStart("character", selectionStart);
    range.moveEnd("character", input.value.length);
    range.select();
  }
  else //DOM Compliant
  {
    var selectionStart = input.value.length;
    input.value = suggestion;
    input.selectionStart = selectionStart;
    input.selectionEnd = input.value.length;
  }
}
```

expression and concatenate the remainder of the string with the mabon:/ protocol-like syntax. The MabonViewHandler will recognize the string and return a resource URL that will be written to the client (for example, /context-root/mabon-servlet-mapping/backingBean.doSuggest).

Using the Input Suggest Component

Creating an AJAX solution is not a simple task, although there are several AJAX toolkits available, such as the Dojo Toolkit (http://www.dojotoolkit.org), that make it a lot easier. What JSF can offer is an even simpler programming model and one that is known by millions of developers: JSP and Java. To finish this article off in a fashion that suits a proper Ajax solution, let's look at how you can use this input suggest component in a JSF application, as shown in Listing 8.

This page contains one HtmlInputSuggest component, <jdj:inputSuggest>, that has the value attribute set to a value binding expression. This expression is pointing to a value property on the backing bean. The doSuggest attribute contains a method binding expression pointing to a doSuggest() method on the same backing bean. Let's have a look at the backing bean, as shown in Listing 9.

The value property is just a plain old JavaBean property. The doSuggest() method, as shown in Listing 10, is a bit more interesting. This method takes the initial value entered by the user and passed to it via Mabon from the doChange() function (see Listing 3). The doSuggest() method then returns an Array filtered based on the users initial value to the client. It is important that the returned value conforms to the supported JSON syntax.

The end result of this HtmlInputSuggest component looks like Figure 2.

Conclusion

From this article, we hope you have gained an understanding of how to Ajax-enable data fetch for your JSF components using Mabon, and how you can package external resources needed for your JSF component into the same archive as your Java classes leveraging the Weblets project.

Finally, now that you know how to create reusable rich Internet components with JSF and AJAX, we hope you will apply the techniques you have learned in this article series to create your own custom components and build Rich Internet Applications (RIAs) with JSF.

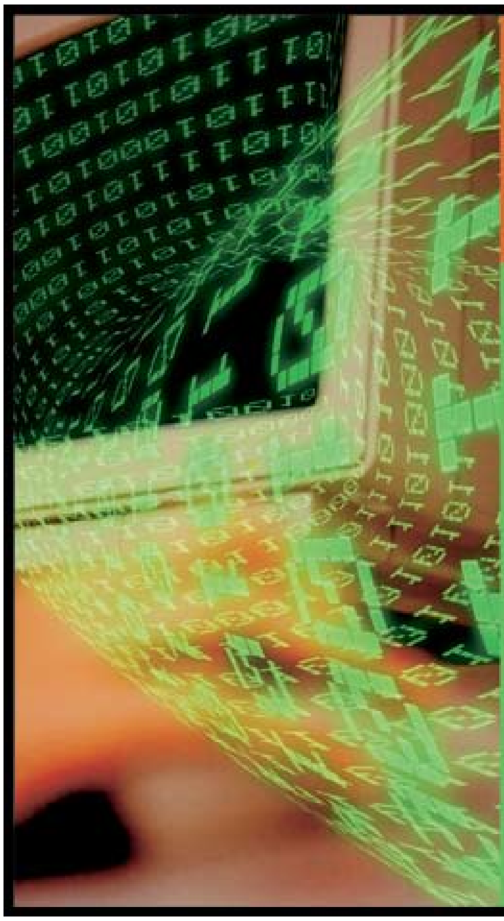
This article is based on, and contains excerpts from, The Book Pro JSF and AJAX: Building Rich Internet Components *by Jonas Jacobi and John Fallows, published by Apress.* ■

Listing 6: The HtmlInputSuggestRenderer encodeBegin() Method

```
package com.apress.projsf.jdj.render.html;

Import ...

/**
 * HtmlInputSuggestRenderer renders a traditional HtmlInputText field
 * with auto-suggest behavior.
 */
public class HtmlInputSuggestRenderer extends HtmlRenderer
{
...
  public static String TITLE_ATTR = “title”;
  public static String DO_SUGGEST_ATTR = “doSuggest”;

  public void encodeBegin(
    FacesContext context,
    UIComponent  component) throws IOException
  {
    writeScriptResource(context,
                       “weblet://org.dojotoolkit.browserio/dojo.js”);
    writeScriptResource(context,
                       “weblet://net.java.dev.mabon/mabon.js”);
    writeScriptResource(context,
                       “weblet://com.apress.projsf.jdj/inputSuggest.js”);
    writeStyleResource(context,
                      “weblet://com.apress.projsf.jdj/inputSuggest.css”);
}
```

Jonas Jacobi is a technology evangelist for Oracle's Java/J2EE tool offering, JDeveloper, and over the past three years has been responsible for JavaServer Faces, Oracle ADF Faces, and Oracle ADF Faces Rich Client development features within Oracle JDeveloper. Jonas has been in the software business for 15 years. Prior to joining Oracle, he worked at several software companies in Europe, covering many roles including support, consulting, development, and project team leadership. jonas.jacobi@oracle.com

John Fallows is a consulting member of technical staff for server technologies at Oracle Corporation, and has been working in distributed systems for over a decade. During the past five years, he has focused on designing, developing, and evolving Oracle ADF Faces, and is now lead developer for Oracle ADF Faces Rich Client. john.fallows@oracle.com



Listing 7: The HtmlInputSuggestRenderer encodeEnd() Method

```
  public void encodeEnd(
    FacesContext context,
    UIComponent  component) throws IOException
  {
    String valueString = _getValueAsString(context, component);
    String clientId = component.getClientId(context);

    Map attrs = component.getAttributes();
    String title = (String)attrs.get(TITLE_ATTR);
    String onchange = (String)attrs.get(ONCHANGE_ATTR);
    MethodBinding doSuggest = (MethodBinding)attrs.get(DO_SUGGEST_
ATTR);

    ResponseWriter out = context.getResponseWriter();
    out.startElement("div", component);

    if (title != null)
      out.writeAttribute("title", title, TITLE_ATTR);

    // <input id="[clientId]" name="[clientId]"
    //         value="[converted-value]" onchange="[onchange]" />
    out.startElement("input", component);
    out.writeAttribute("id", clientId, null);
    out.writeAttribute("name", clientId, null);
    if (valueString != null)
      out.writeAttribute("value", valueString, null);
    if (doSuggest != null)
    {
      // disable browser auto-complete when using server-side
suggest
      out.writeAttribute("autocomplete", "off", null);

      String expression = doSuggest.getExpressionString();
      // trim #{} from expression
      String bindingRef =
              expression.substring(2, expression.length() - 1);
      ViewHandler handler =
              context.getApplication().getViewHandler();
      String doSuggestURL =
              handler.getResourceURL(context, "mabon:/" + bindin-
gRef);
      out.writeAttribute("onkeypress",
                          "return projsf.jdj.doKeyPress(event);",
null);
      out.writeAttribute("onkeyup",
                          "return projsf.jdj.doKeyUp(event);",
null);
      out.writeAttribute("onchange",
                          "projsf.jdj.doChange(event, '" + doSug-
gestURL
                                                  + "');", null);
      out.writeAttribute("onblur",
                          "return projsf.jdj.doBlur(event);",
null);
    }
    out.endElement("input");
    out.startElement("br", null);
    out.endElement("br");
    out.startElement("div", null);
    out.writeAttribute("id", clientId + "$suggest", null);
    out.writeAttribute("class", "HtmlInputSuggest", null);
    out.endElement("div");

  }
```

Listing 8: A JSP page using the JSF input suggest component

```
<?xml version="1.0" encoding="UTF-8" ?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="1.2"
           xmlns:jdj="http://projsf.apress.com/jdj"
           xmlns:f="http://java.sun.com/jsf/core"
           xmlns:h="http://java.sun.com/jsf/html" >
  <jsp:directive.page contentType="text/html" />
  <f:view>
    <html>
      <head>
        <title>ProJSF : HtmlInputSuggest</title>
      </head>
      <body>
        <h:form id="form" >
          <jdj:inputSuggest id="inputSuggest"
                             title="Input Suggest Component"
                             value="#{backingBean.value}"
                             doSuggest="#{backingBean.doSuggest}"
/>
        </h:form>
      </body>
    </html>
  </f:view>
</jsp:root>
```

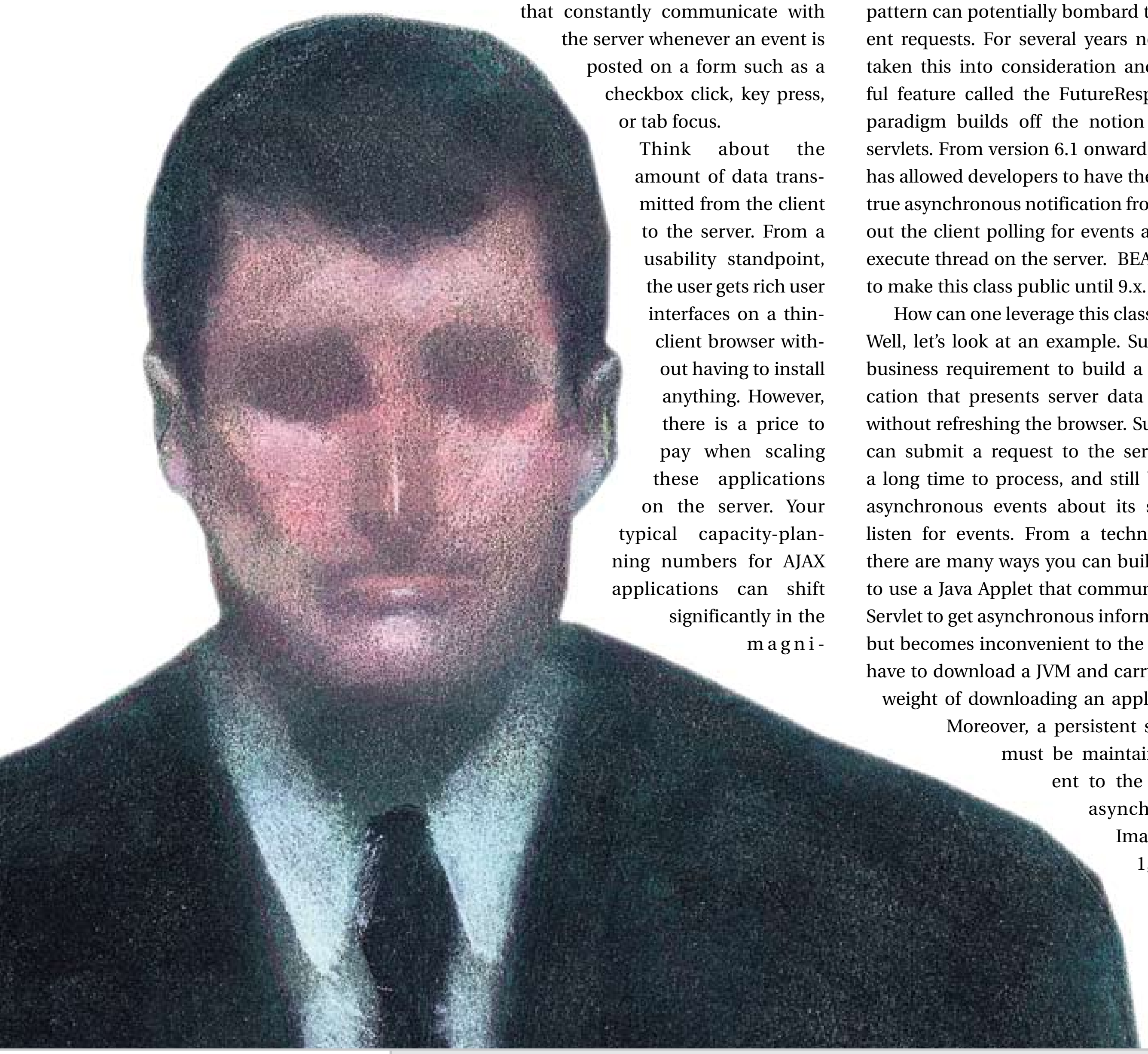
Listing 9: The backing bean value property

```
package com.apress.projsf.jdj.application;

import java.util.ArrayList;
import java.util.List;

/**
 * BackingBean is a backing bean for inputSuggest.jspx document.
 */
public class BackingBean
{
  public void setValue(
    Object value)
  {
    _value = value;
  }

  public Object getValue()
  {
    return _value;
  }

Listing 10: The backing bean doSuggest() method
  public String[] doSuggest(
    String initialValue)
  {
    List<String> suggestions = new ArrayList<String>();

    for (int i=0; i < _MASTER_LIST.length; i++)
    {
      if (_MASTER_LIST[i].startsWith(initialValue))
        suggestions.add(_MASTER_LIST[i]);
    }

    return suggestions.toArray(new String[0]);
  }

  private Object _value;

  static private final String[] _MASTER_LIST = new String[]
                                                {
                                                  "Pro JSF and
Ajax",

                                                  "Pro Ajax",
                                                  "Pro JSP 2",
                                                  "Pro
Jakarta",

                                                  "Pro J2EE
1.4"

                                                };
}
```

Scaling AJAX Applications Using Asynchronous Servlets

Bahar Limaye

Multiplexing client sockets



The advent of AJAX as a Web application model is significantly changing the traffic profile seen on the server side. The typical Web pattern usage of a user sitting idle on a Web page filling out fields and hitting the submit button to the next link is now transforming into sophisticated client-side JavaScript and rich user interfaces that constantly communicate with the server whenever an event is posted on a form such as a checkbox click, key press, or tab focus.

Think about the amount of data transmitted from the client to the server. From a usability standpoint, the user gets rich user interfaces on a thin-client browser without having to install anything. However, there is a price to pay when scaling these applications on the server. Your typical capacity-planning numbers for AJAX applications can shift significantly in the magnitude of three to four times than that of standard Web applications.

One might ask how would this impact the WebLogic Server. For every HTTP request to WebLogic, an execute thread is consumed. Given the nature of AJAX programming and many short-lived HTTP requests in the form of polling, this behavioral pattern can potentially bombard the server with client requests. For several years now, WebLogic has taken this into consideration and built a wonderful feature called the FutureResponseServlet. This paradigm builds off the notion of asynchronous servlets. From version 6.1 onward, this functionality has allowed developers to have the ability to provide true asynchronous notification from the server without the client polling for events and consuming an execute thread on the server. BEA was not too keen to make this class public until 9.x.

How can one leverage this class in the real world? Well, let's look at an example. Suppose you have a business requirement to build a Web-based application that presents server data in near-real time without refreshing the browser. Such an application can submit a request to the server that can take a long time to process, and still be able to receive asynchronous events about its status, as well as listen for events. From a technology standpoint, there are many ways you can build this. One way is to use a Java Applet that communicates with a Java Servlet to get asynchronous information. This is nice but becomes inconvenient to the user because they have to download a JVM and carry the baggage and weight of downloading an applet to the browser. Moreover, a persistent socket connection must be maintained from the client to the server to receive asynchronous events. Imagine: if there are 1,000 users using the applet, there are 1,000 execute threads that are mostly idle waiting to send event notifications back to the client. Yes, there are other approaches such as building polling mechanisms from an applet or AJAX application to check for new data every so often. If data is not received that often, it is useless to poll and waste server resources and consume execute threads. Instead, the server can poll periodically and relay events back to the client and maintain the socket threads without consuming a persistent execute thread. This is very similar to how Java NIO works. Ideally, you would want to build an application, whether it is an applet or a thin AJAX-based Web application, that "asynchronously" receives event notifications from the server without consuming a persistent execute thread on the server.

One solution to this problem is to create a servlet that extends the FutureResponseServlet class. The browser establishes a single connection to the FutureResponseServlet and registers itself as a listener in a different thread. Whenever an event is received on the server, the thread notifies the client with the event. The server maintains the client asynchronously without having to consume a persistent execute thread. This model can scale several concurrent users.

This article does not describe how to build an AJAX application. There are several articles available that discuss this topic. It discusses the importance of asynchronous processing for presentation layers, such as AJAX, applets, or any front-end application. Listing 1 shows an example.

As you can see, this example is extremely trivial. The AsynchronousServerResponseServlet class extends FutureResponseServlet and overrides the service method. A single thread, the Notifier class, processes all client connections response. For every HTTP request, the servlet registers the socket connection to the Notifier thread and returns. Asynchronous events get delivered to the client while the persistent socket connection is maintained.

A single thread can manage multiple client connections!

The run() method can be used to call back events to the client based on some message selection criteria. This example just performs a server-side push operation and is very simplistic in nature. Thread pools can be utilized for certain types of event processing.

In conclusion, when processing long running tasks, the FutureResponseServlet is a good feature that allows developers to increase performance and process responses in separate threads and minimize overhead. This approach allows scalability when building asynchronous applications. ■

Bahar Limaye is a system architect at The College Board. He has extensive experience building distributed object-oriented systems. blimaye@collegeboard.org

Listing 1

```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;
import java.util.Stack;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;

import weblogic.servlet.FutureResponseServlet;
import weblogic.servlet.FutureServletResponse;

// An AsynchronousServlet that handles HTTP requests from a "separate" thread and
// not the execute thread used to invoke this servlet.
public class AsynchronousServerResponseServlet extends FutureResponseServlet {
 private final Notifier notifier;

 public AsynchronousServerResponseServlet() {
  this.notifier = new Notifier();
  this.notifier.start();
 }

 public void service(HttpServletRequest request, FutureServletResponse response)
  throws IOException, ServletException {

  // push this client's request to a buffer and return immediately.
  // asynchronous processing occurs in the run method of the Notifier Thread
  notifier.poll(request, response);
 }

 class Notifier extends Thread {

  private static Stack clients = new Stack();

  void poll (HttpServletRequest request, FutureServletResponse response) {
   clients.push(new Client(request, response));
  }

  public void run() {
   while (!clients.empty()) {
    Client client = null;
    try
    {
     client = (Client) clients.pop();
     PrintWriter pw = client.response.getWriter();
     for(int j = 0; j < 10; j++) {
      pw.println("Time is:" + new Date() + "<BR>");
      pw.flush();

     }
     pw.close();
    } catch(Throwable t) {
     t.printStackTrace();
    } finally {
     try {
     client.response.send();

     } catch(IOException ioe) {
     ioe.printStackTrace();
     }
    }

   }
  }
 }

 // inner class that holds on to the clients http request and response
 class Client {
  private HttpServletRequest request;
  private FutureServletResponse response;

  private Client(HttpServletRequest request, FutureServletResponse response) {
   this.request = request;
   this.response = response;
  }
 }
```

ColdFusion and AJAX

A primer

Jeffry Houser

One of the most annoying things about the Web page paradigm is that you have to reload the page whenever you want to do something on the server side. If you want to do a site search, it's sprinkled across two pages.

On the first page, the user will enter a search term, then clicks a submit button to get to the second page where the search results are returned. If the user wants to filter or sort the search results it's usually refreshed again and each result is redisplayed in its filtered state.

Lately there's been a movement to help improve the Web by eliminating these constant page refreshes. Macromedia's Rich Internet Application (RIA) push with Flash and Flex has been at the forefront of this effort, however Flex' entry price has made it prohibitive for many ColdFusion Developers. That's where AJAX comes in. It's an alternate way to create a RIA and relies on technologies built into most browsers, making it completely free.

Dissecting AJAX

AJAX stands for Asynchronous JavaScript and XML. It's not a single technology, but rather a combination of different technologies that are used to create a Rich Internet Application feel. You've probably seen examples of this kind of application. Google uses AJAX a lot on such services as maps.google.com. After bringing up a location, you can click and drag the map, zoom in or out, or even bring up directions to and from the location. All this is done without any screen refreshes. You have to admit that the interface is pretty sweet. (Note: Yahoo has a similar mapping application, built entirely in Flex at http://maps.yahoo.com/beta.)

Since AJAX is a combination of technologies you're going to want to know what you're getting into before going any farther, right? I thought so. Here's the list:

- ***XHTML:*** To start with you need a display language. Most resources on AJAX say you have to use XHTML, but regular HTML will work too.
- ***XMLHttpRequest:*** The XMLHttpRequest object lets you retrieve data from the server without refreshing the page. It's a JavaScript object implemented in most major browsers. As the XMLHttpRequest object is being called, the user can do other things on the page. This is where the "asynchronous" part of the acronym comes in.
- ***iFrames:*** iFrames are in-line frames and are sometimes used as an alternative to the XMLHttpRequest object. A regular frame will divide the browser window into multiple separate pages, but an in-line frame embeds one page inside another. I won't get into any iFrames details in this article.
- ***XML:*** You all know what XML is, right? As a refresher, it's a way to describe data. If you need more details, read the article in this column from August 05 at http://coldfusion.sys-con.com/read/117667.htm.
- ***Document Object Model:*** The document object model defines your HTML page through a programmer API.
- ***JavaScript:*** I'm sure you all know what JavaScript is too. It's the most common language used for providing client-side functionality to HTML pages.

How do all these different technologies work together using AJAX? Well, it generally works something like this:

1. First, the user comes to your page and the page loads. Something (depending on what your page does) is loaded and displayed to the user.
2. The user does something such as clicking a link. Instead of reloading the page (as would happen in a normal Web app), that link fires off a JavaScript function.
3. The JavaScript function uses XMLHttpRequest object to retrieve XML data from the server.
4. Then the JavaScript parses the XML Data and through the Document Object Model will change the page without a refresh.

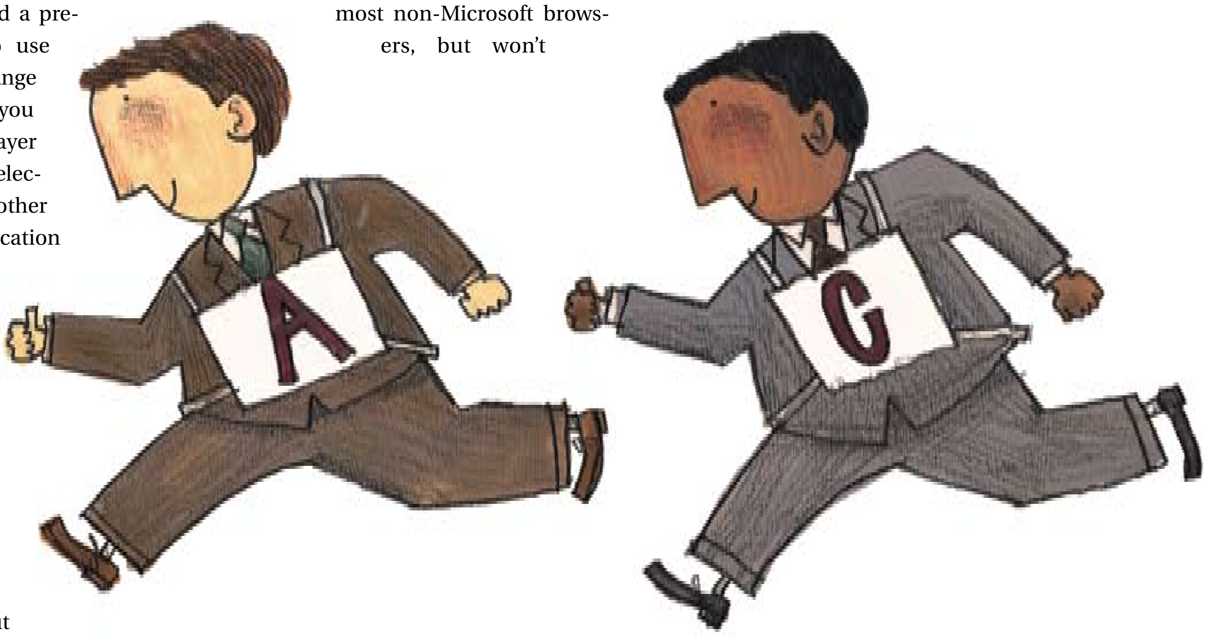
So the user gets to see more data quicker without the annoying page refreshes. That's fantastic, right? Why isn't everyone using this? Well, as with all things, AJAX isn't perfect. There are drawbacks:

- ***Development Time:*** Working with AJAX will bring you back to the old days of Web development, especially if you want any kind of cross-compatibility. There are different syntaxes for creating the XMLHttpRequest object in IE vs Netscape/Firefox, which can lead to extended development times.
- ***User Experience:*** When used properly, an AJAX can enhance the user experience. However, when

used improperly it can make it horrendous. Since the page isn't reloading, what happens when the user hits the back button? He'll leave the application altogether. I bet he expected to go to the app's "previous state." What happens when the user bookmarks a page? He'll go the "initial" state of the app. There are workarounds, of course, but they're not trivial to implement.

- ***Searches:*** The single-page nature of the app makes indexing by search engines a tetchy proposition at best.
- ***Response Times:*** The very thing that works for you (no page reloads) can also work against you. What happens if you're retrieving a lot of data from the server? And the user is given no indication that something is happening.
- ***JavaScript:*** JavaScript must be enabled for an AJAX app to work. If JavaScript is disabled, your user isn't going to have a lot of fun with an AJAX app.
- ***Accessibility:*** AJAX is, most likely, not going to meet accessibility guidelines. If this is a goal of your project you're going to have to provide a fallback option. That means developing two paths to the same goal.

There are a few real simple examples of AJAX that you've probably used or seen that existed long before the term AJAX was conceived. Have you ever rolled your mouse over a graphic navigation button only to have that graphic change? JavaScript rollovers can be considered a precursor to AJAX. They do use JavaScript and they do change the state of the page. Have you turned on or off a DHTML layer on a page based on some selection by the user? This is another example of AJAX. If an application has a select box with an "other" option in it I'll often use DHTML to display an input text box if "other" is selected. If other isn't selected the input box is hidden. Both of these are simple examples of changing the page without a screen refresh. They don't go out to the server to get data, but they still fit the RIA paradigm.



Putting This All Together

Let me show you how this all comes together, so you can actually do something useful. The first step in the process is to initialize the XMLHttpRequest object. Unfortunately, there's not consistent way to do this across browsers:

```
<script language="javascript" type="text/javas-
cript">
var request = false;
try {
 request = new XMLHttpRequest();
}
catch (trymicrosoft) {
 try {
  request = new ActiveXObject("Msxml2.XMLHTTP");
 }
 catch (othermicrosoft) {
  try {
   request = new ActiveXObject("Microsoft.
XMLHTTP");
  }
  catch (failed) {
   request = false;
  }
 }
}
</script>
```

The first line initializes the request variable, giving it a Boolean value of false. You can check against this later to make sure that your request object was properly initialized. The next code creates an instance of the XMLHttpRequest object using 'new XMLHttpRequest.' This works fine for most non-Microsoft browsers, but won't work in IE. If it doesn't initialize, then an error is thrown, which is where the first catch statement comes into play. It attempts to re-initialize using the method available in the current version of Internet Explorer: new ActiveXObject("Msxml2. XMLHTTP").

If that fails, it tries the version from older versions of Internet Explorer new ActiveXObject(“Microsoft. XMLHTTP”). That’s kind of complicated right? I borrowed this initialization code from the article http://www-128.ibm.com/developerworks/Web/library/wa-ajaxintro2/?ca=dgr-lnxw07AJAX-Request (which you should all read because it goes into a lot more details about error checking that are outside the scope of this article).

So, now the code has created the object, but what can you do with it? As you can probably see, it doesn’t do much yet. You want to assign it a URL, load the data, and then process the data in some way. This function will do that:

```
function getRSS() {
 var url = “/htdocs/jeffhousercom/wwwroot/rss.cfm”;
 request.open(“GET”, url, true);
 request.onreadystatechange = updatePage;
 request.send(null);
}
```

This is function to grab an RSS feed. The first line defines the URL. This is standard JavaScript for defining a variable. You should bear in mind that you can only request URLs from the current Web site. In this case, I’m grabbing a RSS feed from a local copy of my blog. The next line calls a method on the request object entitled open. Open prepares the request to be sent, but doesn’t actual send it. There are five arguments to the open function (although we only use three in this example):

- ***Request-type:*** This specifies the request type, either get or post, for the URL that you want to send.
- ***url:*** The URL specifies the URL that you’re getting (or posting) to.
- ***Asynchronous:*** The async value is a Boolean value that specifies whether this request is asynchronous or not. For AJAX use, you’ll want to specify true. If set to false, the application will hold until this request is finished, which defeats the purpose of an RIA in the first place.
- ***username:*** The username specifies the username required for accessing the URL, if applicable.
- ***password:*** The username specifies the username required for accessing the URL, if applicable.

Okay, you’ve set up the request to be ready to send. The next line specifies the onreadystatechange property. This property specifies what function to call once the request is complete. If you don’t specify a method to be called then nothing will happen once the request is complete. The final line calls the send method. The send method is the one that will execute the request. Its parameter is the data that you need to send along with the request. In our example, we aren’t sending parameters, so we pass a null value.

Once the request complete, the updatePage function will execute, so lets look at that next:

```
function updatePage() {
 if (request.readyState == 4){
  Form1.XMLValues.value = request.responseText;
 }
}
```

This function first checks the readyState of the XMLHttpRequest object. Exploring that is beyond the scope of this article, but there are a lot more details in some of the references provided at the end. For now, all you need to know is that if the readyState is 4, then the request is complete. This takes the resulting value from the server response and assigns it to a form textbox. The form is below:

```
<form name=”Form1”>
 <textarea name=”XMLValues” rows=”20” cols=”50”></
textarea>
</form>
```

The text box will contain the XML from the RSS feed. There’s another property associated with the XMLHttpRequest object called responseXML. This will contain the results as an XML Document object, which JavaScript can access natively. Unfortunately specific details are beyond the scope of this article, but I found this good resource to get you started on that: http://www.peachpit.com/articles/article.asp?p=29307&seqNum=4&rl=1 .

Where to Go from Here

If you want to learn more about AJAX, there are some great resources you can read. The article by Jesse James Garrett that started it all is at http://www.adaptivepath.com/publications/essays/archives/000385.php. Garrett is credited with coining the term AJAX. You can follow an AJAX blog at http://www.ajax-powered.net, which talks about all things AJAX. Finally, there are a few people who are working on ways to integrate ColdFusion and AJAX together. One is the CFAJAX project http://www.indiankey.com/cfajax. Another is the AJAXCFC project at http://www.robgonda.com/blog/projects/ajaxcfc. ■

Jeff Houser has been working with computers for over 20 years. He owns a DotComIt, a web consulting company, manages the CT Macromedia User Group, and routinely speaks and writes about development issues. You can find out what he’s up to by checking his Blog at www.jeffryhouser.com. jeff@instantcoldfusion.com

Ready to take the AJAX plunge?

Use Zapatec's AJAX building blocks to build comprehensive applications quickly and easily

- Fast Deployment
- Modular Architecture
- Cross Browser Compliant
- Standards Driven

Jump start your AJAX deployment by using the leading Zapatec AJAX Suite. The suite provides a robust set of widgets and a rich library of effects and utilities. The Zapatec AJAX Suite's simple modular design means you can put selected components to use right away, adding depth and breadth to your web application as you go. Because the Suite is truly standards based, learning to integrate these components comes easily.

Zapatec

AJAX Suite 2.0

www.zapatec.com
866 522-7941
sales@zapatec.com

Custom Error Handling Using AJAX

RYAN ANKLAM

Enhancing the interactive experience

AJAX has become an increasingly popular tool to develop RIAs. With AJAX, as with many new technologies, developers often overlook core application issues such as error handling. While many current AJAX frameworks come with ways to handle errors, the built-in error-handling methods might not be quite what you need, and it's possible that you might not even want to adopt a specific AJAX framework at all. So how do you handle errors in AJAX?

This article will guide you through one possible way to implement custom error handling in AJAX using many of the same techniques that you'll likely read about in other parts of this magazine. Because AJAX represents a big paradigm shift in the way users interact with Web applications, it's easy to leave your users confused when things don't quite work as they'd expected. To enhance the user experience, it's equally important to alert them when something goes exactly as planned and enhances the interactive experience.

Consider for a moment a hypothetical AJAX application that updates employee information. Users will fill out fields and click on a "Save" button to process the update. In a traditional Web application the user expects to wait a moment while the server updates the record then anticipates another page that displays a confirmation message. This is the typical request/return scenario that our Internet conditioning forces us to accept.

Now let's look at the same example using AJAX techniques. The end user still fills out form fields and clicks on the "Save" button but instead of seeing the confirmation message, nothing seems to happen. This can leave the user confused and unsure that his information was saved, yet with AJAX, the database update occurred exactly as expected. Here's how you can implement a messaging and error system in a simple employee information maintenance application that will alert a user as records are updated.

Process at a Glance

The process isn't much different from any typical AJAX request/response. A request is created, sent to the server, checked for error conditions, XML is sent back to your request page, and checked in the browser for a status message, which is displayed, if it exists.

Create an Area to Display Status Messages

We'll begin to create our code by creating our CSS format classes for our status messages. Let's create three styles for our application's potential conditions: error, success, and hidden, which correspond to the two cardinal states (error and success) of our query (hidden being used when no update is currently active):

```
.error{
 font-family: Arial, Helvetica, sans-serif;
 font-size: 10px;
 font-weight: bold;
 color: #FFFFFF;
 background-color: #FF0000;
 display:block;
}
.success {
 font-family: Arial, Helvetica, sans-serif;
 font-size: 10px;
 font-weight: bold;
 color: #FFFFFF;
 background-color: #009900;
 display:block;
}
.hidden {
 display:none;
}
```

Once we have our styles set, we have to put them to use. We'll do this by creating an area in our application to display the status messages returned by the server. This display area can be any type of HTML container such as table, div, or span; we'll use a div. Once the container is created, we'll set the initial style to hidden, as follows:

```
<div id="message" name="message" class="hidden"></
div>
```

Creating the ColdFusion Page to Process the Request.

At this point, you'll find that the code you've created doesn't do much - you have three CSS styles, one of which is called by your container div, but its class is set to not display on the rendered page. To create conditions where an error or success message might actually exist, we'll look at a ColdFusion template that updates an employee's database record. Since the focus of this article is on error handling we'll skip the details of updating the record and get right to the process of building the XML that returns our status message.

First let's look at an example of processing an update of one of our employee records. Since this is a situation where no data is being returned (because we're not using a SELECT statement), we really only need to deliver either a success or error message to our user.

To do this, we'll have to create a variable to hold the XML string then add the XML declaration to it:

```
<cfset xml = "<?xml version=""1.0"" encoding=""UTF-
8"" standalone=""yes"" ?>">
```

Now that we've created our variable, "xml," we're going to want to do some simple data validation - in this case, to make sure that a valid department ID was passed into the template. If the department ID passed in is not valid, we're going to want to set the first node of the XML document to (<error>), add the error message, and close the error node (</error>). For our purposes, we'll assume that a "departmentID" value of 0 or of a non-numeric value constitutes an invalid condition. We're also going to include "try/catch" conditions to cover database errors and general failures:

```
<cfif departmentId eq 0 OR NOT
IsNumeric(departmentId)>
<cfset xml = xml & "<error>Invalid department spec-
ified.</error>">
<cfelse>
 <cftry>
<cfcatch type="database">
 <cfset xml = "<?xml version=""1.0"" encoding=""UTF-
8"" standalone=""yes"" ?><error>There was a error
communicating with the server, please call the help
desk at x555.</error>">
</cfcatch>
<cfcatch type="Any">
<cfset xml = "<?xml version=""1.0"" encoding=""UTF-
8"" standalone=""yes"" ?><error>There was a error
processing your request.  Please try again later or
call the help desk at x555.</error>">
</cfcatch>
</cftry>
</cfif>
```

"While many current AJAX frameworks come with ways to handle errors, the built-in error-handling methods might not be quite what you need"

Listing 1

```
<cffunction name="GetAllDepartments" access="public" returntype="array"
output="false">
 <cfargument name="dsn" required="true" type="string">
 <cftry>
  <cfquery name="GetAll" datasource="#arguments.dsn#">
   SELECT
    id
   FROM
    Department
  </cfquery>

  <cfscript>
   DepartmentArray = ArrayNew(1);

   //if no departments found, throw an error
   if(GetAll.RecordCount eq 0)
   {
    ThrowError('No departments found.');
   }

   for(i = 1; i lte GetAll.RecordCount; i = i + 1)
   {
    ArrayAppend(DepartmentArray,GetAll['id'][i]);
   }

   return DepartmentArray;
  </cfscript>
 <cfcatch type="database">
  <cfthrow message="There was an error communicating with the database server.
Please call the help desk at x555.">
 </cfcatch>
 <cfcatch type="any">
  <cfthrow message="#cfcatch.Message#">
 </cfcatch>
 </cftry>
 </cffunction>
```

One thing to note here is that the entire XML string is overwritten in the catch statement. This eliminates a situation caused by an error being thrown in the middle of building the XML string. Specifically this condition sends an incomplete and unpredictable XML document back to the requesting page.Let's walk through an example of a successful and unsuccessful update of an employee record. The main screen of the employee update application is shown in Figure 1. For this example let's say that a phone extension can only be used once per employee in a company. When a user clicks on the "Update Employee" link the main screen will prepare the AJAX request and send it to a ColdFusion template. The template will then process the update and send a XML message back to the main screen.

If everything in the update goes okay the ColdFusion template will send the page a success XML message that would look something like this:

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes" ?><data><Success>Successfully
updated employee record.</Success></data>
```

Now, let's say that the extension is in use. Our ColdFusion template will return a error XML message that would look something like:

```
<?xml version=""1.0"" encoding=""UTF-8""
standalone=""yes"" ?><data><error>There was a
error updating the employee, extension is in use
by Desmond Mason.  Please call the help desk at
x555.</error></data>
```

To complete the application we have to have a way to handle the XML sent back to the requesting page.

Creating JavaScript to Handle the Errors

Let's take a look at what we've done so far: we've created our CSS style classes, we've created our display container, we have our query, our template to update the employee record, and created our error-handling logic. The next step is to read the response from the server, parse through the XML, and set the appropriate display condition. We'll start this step by creating a function to show error messages. This function will need two parameters: the message text and an indication of whether or not this is an error message.

We'll start writing this function by setting a variable with a value that will represent the name of the message area - this, again, is our display container. Now that we have our display container we have to set the class of the display container conditionally based on the error parameter. Finally, we can set the innerHTML property of message node equal to the message's value:

```
function ShowMessage(message, isError)
{
 messageArea = document.getElementsByName('message')
[0];

 if(isError)
 {
  messageArea.className = 'error';
 }
 else
 {
  messageArea.className = 'message';
 }

 messageArea.style.display = 'block';
 messageArea.innerHTML = message;
}
if(response.childNodes[0].nodeName == ‘error’)
{
ShowMessage(response.childNodes[0].firstChild.
nodeValue,true);
}
```

After the ShowMessage function is written we have to write the logic to check the value of the first node returned. Since we called an update page we expect the first node to be named either “error” or “success” so we create code that will check to see if this is true:

```
if(response.childNodes[0].nodeName == ‘error’)
{
ShowMessage(response.childNodes[0].firstChild.node-
Value,
true);
}
else
{
ShowMessage(response.childNodes[0].firstChild.node-
Value,
 false);
}
```

Let’s look at our example of updating the employee record again. In the first case we got an XML message that looked like:

```
<?xml version=”1.0” encoding=”UTF-8”
standalone=”yes” ?><data><Success>Successfully
updated employee record.</Success></data>
```

In this case our script would call the ShowMessage function with the error Boolean value set to false. The resulting page would look something like Figure 2.

Now let’s look at the same page if the server returned an error because the employee’s extension is used by another employee. As mentioned above, the XML message returned from the ColdFusion template would look like:

```
<?xml version=””1.0”” encoding=””UTF-8””
standalone=””yes”” ?><data><error>There was a
error updating the employee, extension is in use
by Desmond Mason.   Please call the help desk at
x555.</error></data>
```

In this case our script would call the ShowMessage function with the error Boolean value set to true. The resulting page would look something like Figure 3.

Coupling with Client-Side and Template Error Checking

To make the messaging system complete we have to integrate traditional ColdFusion error checking as well as client-side error checking.

Let’s look at how to integrate traditional ColdFusion error handling into our messaging system first. We’ll start by creating two session variables: one to hold a Boolean value to determine if an error has occurred and a second one that will be a string to hold the error message that we’re going to return to the visitor. For this example I’ll name them session.error and session.errorMessage. The next step is to wrap all your ColdFusion logic in the <cftry> and <cfcatch> statements. In your <cfcatch> statement you’ll set the session.error to true and the session.errorMessage equal to the cfcatch.Message.

```
try
{
Department = CreateObject(“Component”,”CFDJ.
Components.Department”);
DepartmentArray = Department.GetAllDepartments(#ses
sion.dsn#);
}
catch(Any ex)
{
 session.error = true;
 session.errorMessage = ex.Message;
}
```

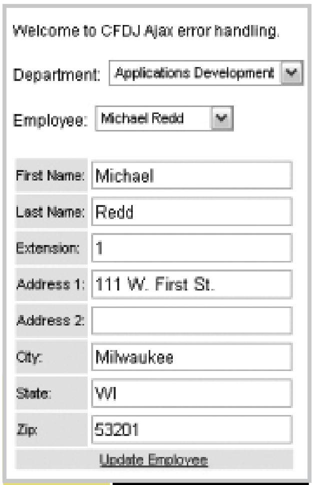
Finally we’ll have to add some logic to control the visibility of our message area. If the session.error message is false the display area is created exactly as before. However, if an error is found we’ll set the default style of the message area to “error” and display the error message:

```
<cfif NOT session.error>
 <span id=”message” name=”message” class=”hidden”></
span>
<cfelse>
<span id=”message” name=”message”
class=”error”>#session.errorMessage#</span>
 <cfset session.error = false>
</cfif>
```

Now let’s say that the connection to the database was down when the user first entered the application. The function in Listing 1 is used to get a list of all the departments in the company.

If the database server is down the cfcatch will catch a database error and throw a new error message that will tell the user that the database server is down, it’s up to the page that calls this function to trap the error and set the session.error and session.errorMessage variables:

```
<cfscript>
 try
 {
```

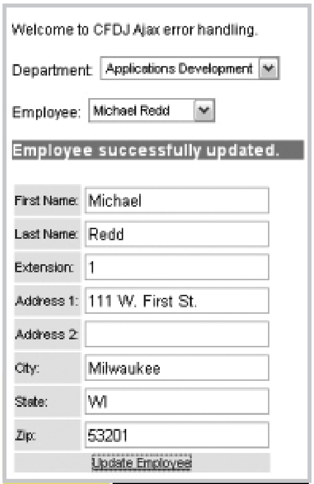

Welcome to CFDJ Ajax error handling.

Department: Applications Development

Employee: Michael Redd

First Name:	Michael
Last Name:	Redd
Extension:	1
Address 1:	111 W. First St.
Address 2:	
City:	Milwaukee
State:	WI
Zip:	53201

Figure 1: Main screen


Welcome to CFDJ Ajax error handling.

Department: Applications Development

Employee: Michael Redd

Employee successfully updated.

First Name:	Michael
Last Name:	Redd
Extension:	1
Address 1:	111 W. First St.
Address 2:	
City:	Milwaukee
State:	WI
Zip:	53201

Figure 2: Resulting page

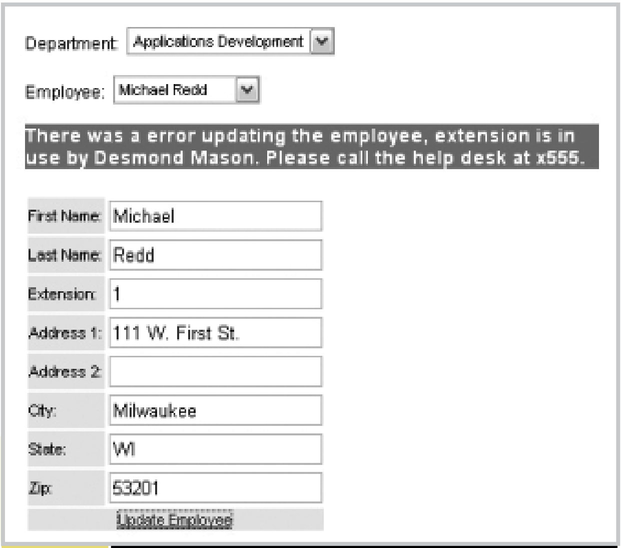
```
 Department = CreateObject("Component","CFDJ.
Components.Department");
 DepartmentArray = Department.GetAllDepartments(#se
ssion.dsn#);
 }
 catch(Any ex)
 {
 session.error = true;
 session.errorMessage = ex.Message;
 }
</cfscript>
```

In this case the employee update page would look like Figure 4 when the user enters the application.

Next let's look at using JavaScript to do some simple client-side error handling. The first step is to check for invalid data. I'll assume that you've had some experience doing simple JavaScript error checking and skip right to displaying the error. For our example, we're going to have the JavaScript check to make sure that the user has selected the name of a user to edit from the user drop-down. The first step would be to make sure that the selected index of a drop-down is not zero. Typically the next step would be to display some sort of message to the user with the Alert() function. In this case we'll simply call the ShowError() function that we wrote above:

```
if(departmentSelect.selectedIndex != 0)
{
 //do something
}
else
{
 ShowMessage('Please select an employee to
update',true);
}
```

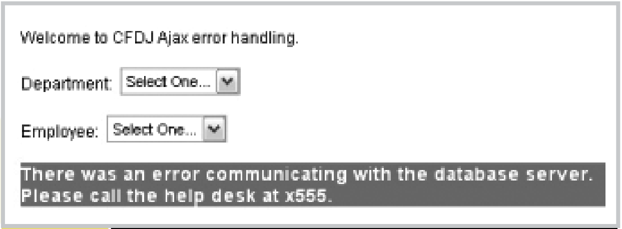
"Adding a messaging system to an AJAX application isn't very complicated"



There was a error updating the employee, extension is in use by Desmond Mason. Please call the help desk at x555.

First Name:	Michael
Last Name:	Redd
Extension:	1
Address 1:	111 W. First St.
Address 2:	
City:	Milwaukee
State:	WI
Zip:	53201

Figure 3: Server-returned error


Welcome to CFDJ Ajax error handling.

There was an error communicating with the database server. Please call the help desk at x555.

Figure 4: Employee update page

Conclusion

Adding a messaging system to an AJAX application isn't very complicated, but it can be an extremely useful technique for keeping your end users informed. This application can be viewed online at http://www.innovacreative.com/cfdj/ajax. The source code can be downloaded from http://www.innovacrative.com/cfdj/ajax/ajax.zip. ■

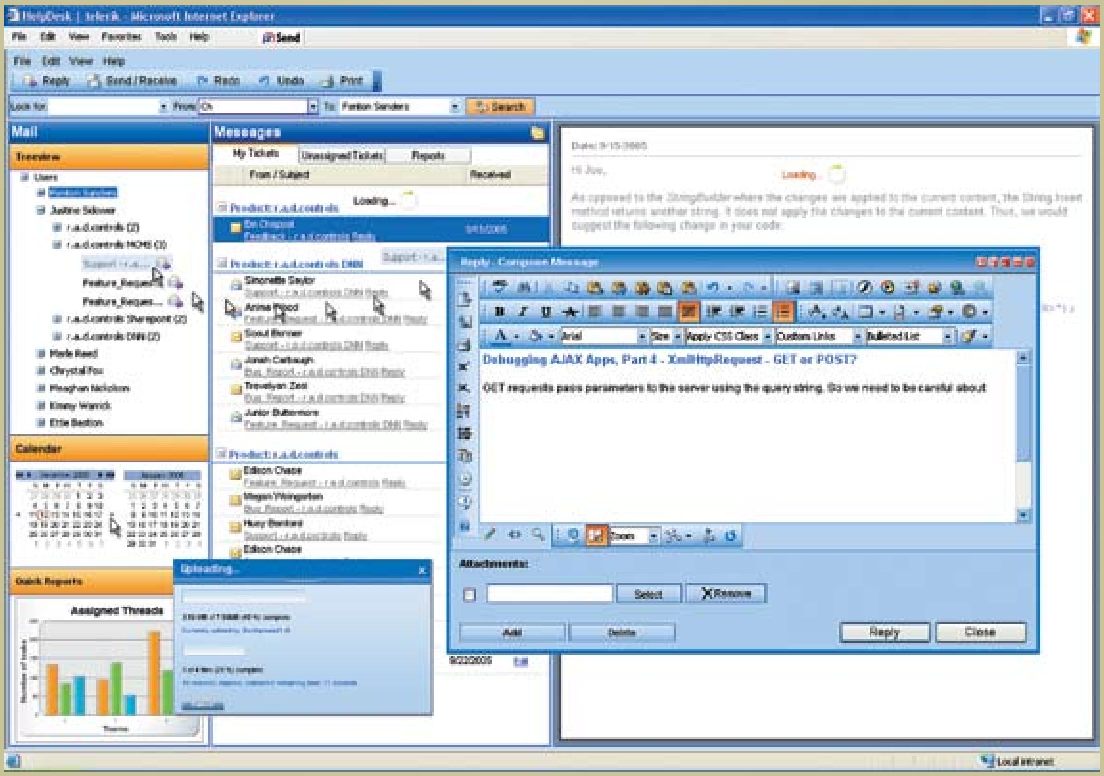
Ryan Anklam is the Chief Information Officer at Innova Creative Media, Inc. His current focus is on using ColdFusion to develop large scale hosted applications. Ryan has been developing ColdFusion applications since 1996. In addition, he is also a Microsoft Certified Professional with demonstrated skills in C# and SQL Server. ryan@innovacreative.com

telerik
deliver more than expected

The preferred UI suite for ASP.NET

Reporting Solution | ASP.NET UI Suite | AJAX framework | WinForms UI Suite

telerik r.a.d.controls for ASP.NET



Debugging AJAX Apps, Part 4 - XmlHttpRequest - GET or POST?

GET requests pass parameters to the server using the query string. So we need to be careful about

- 18 of the best ASP.NET UI and data components on the market
- Extensive AJAX capabilities (codeless AJAX-enabling), support for Microsoft Atlas
- Highly efficient semantic rendering of navigation controls – up to 70% reduced markup
- Widest cross-browser compatibility (IE, Netscape, Firefox, Mozilla, Opera, Safari)
- Standards compliant (XHTML 1.1, WCAG “Level A” or higher)
- Annual subscription entitles you to receive all updates and new ASP.NET components
- Quick-Start framework with 600+ examples, including VS projects

www.telerik.com

AJAX and the Evolution of the Web

Interview with Jesse James Garrett

Jesse James Garrett

At the Real-World AJAX seminar in New York City on March 13, SYS-CON Events had a chance to speak with Jesse James Garrett, the director of user experience strategy and founding partner of Adaptive Path as well as the "Father of AJAX."

Jeremy Geelan: *Hello, this is Jeremy Geelan for SYS-CON TV coming to you live from the Real-World AJAX seminar, the first of the series this year, and we have sitting next to me, Jesse James Garrett, Mr. Real-World AJAX himself. Jesse, thank you for talking with us.*
Jesse James Garrett: Thank you for having me.

Geelan: *I've got to ask you, and I know that everyone watching this, they want me to ask you, are you fed up with being the father of the term AJAX, because you don't look it?*
Garrett: No, no, not at all. It's been fun. It's been really great for me to have the opportunity to talk to people all over the world and it's so exciting to me what's happening with AJAX and with the evolution of Web technology. I'm excited to be a part of it and so many of the people that I talk to, people at events like this one, are excited to be a part of it, too. That energy is what keeps me going.

Geelan: *Let's just talk, because I know it's a lovely story and you told it in your keynote this morning. There you were coining this term in your little February essay last year, and you go on a trip, not knowing that anything would happen.*
Garrett: Right. I had no idea what was coming, obviously, because I put the essay up on our Web site and then I immediately left the country for two weeks…

Geelan: *This was quite a thoughtful essay. We're not talking the front of the New York Post. It was a thoughtful, scholarly essay that you posted and probably said, well, I got that out of my system. Now it's time to travel. What happens next?*
Garrett: I was out of the country for a couple of weeks with essentially no Internet access. I come back home and there's been this avalanche of e-mail waiting for me, e-mail from people all over the world,

people who want to know if they can buy some AJAX from me and people who have all kinds of questions about the concept. The site was on Slashdot one day and so there was a lot of feedback from that and it was really – I'm so glad in some ways that I wasn't there to have to deal with it in the moment.

Geelan: *You saw it maybe as a whole – oh my God, I've obviously touched some kind of nerve; I'd better back up and figure out what this is.*
Garrett: Exactly.

Geelan: *When you do back up, and you're such a good explicator, and figure out what it is, what's the easiest way to quickly, disabuse my question at once, turn it back to me and say, look, the thing is that someone else should have figured out this word AJAX but I did it, so it needed figuring out, right? I think that was always a business problem for you.*
Garrett: Absolutely. That was the business problem for me. My company, Adaptive Path, is a product strategy company. We do a lot of work with business people to help them figure out how to leverage technology to deliver compelling experiences to users.

Geelan: *For that they need to master the concepts.*
Garrett: Right, and so a lot of my job is as an interpreter between technology people and business people to help persuade the business people of the appropriate technological approach for their particular problem, and AJAX was one tool that I came up with in my work as a consultant to address that problem.

Geelan: *You're coming out with this sort of collection of technologies and thinking, how am I going to keep referring to that time and time again. You're going to have to just call it something. Maybe it is as simple as that. That's what's formed this term*
Garrett: I realized also that once you start talking about a collection of things, then you have to explain how those things fit together and that was where I felt that the conversation was going to really get away from the part that I wanted to talk about, the part that I thought was important, which was the impact on the user experience. That was really where the concept of AJAX came about.

Geelan: *And you have the A in place and you have the J in place. Most people would know about that, but perhaps you should just go over that.*
Garrett: For me, the really compelling thing about AJAX is this new asynchronous interaction model, because this is where we have the opportunity to change the way that people work with and think about the Web by making that interaction asynchronous, so what the user does and what the server does are no longer so tightly linked. This was really the main concept that I wanted to communicate to my clients. The addition of JavaScript and XML were just some choices to flesh it out, to help them understand that we were talking about client-side browser technologies that made this possible as opposed to technologies like Flash or Java.

Geelan: *Scroll forward then. It's February 2005, and, lo and behold, more or less a year and, wow, what a year.*
Garrett: Yeah, it's been crazy.

Geelan: *Who would have thunk it. Where will it go? Clearly this pace can't last, neither should it. It doesn't need to last, but that doesn't mean that the momentum can't increase; the speed may slow down, but the momentum is increasing, absolutely.*
Garrett: Oh, sure.

Geelan: *You're seeing a massive take-up. This sort of call to action at a seminar like this is clearly: go and start doing it, and find out about it, visit your site, nose around with it. What would you like to see happen in the course of the year? Were you hoping that the enterprise side of it would be sorted out by the community? What were your ambitions? There may be none.*
Garrett: There are certainly, at this point, things that I'd like to see happen with AJAX in the world. Obviously, the ongoing development of toolkits to make it easier for developers to put AJAX applications out there into the world, but it's going to be a process that's going to take some time for those to reach an appropriate level of maturity; I'm sure there's going to be a profusion of different approaches there. But what I think a lot of people miss in the discussion about AJAX is they get hung up on the technology, and they get hung up on code and things like that.

I think the reason AJAX is compelling to anyone at all is because of the impact that it has for the users, the way that it is able to create these applications that have these dynamic rich experiences to them that change the way we think about, the way we relate to the medium. My hope is that all of the people who right now are wrestling with the code, once they get to the point where they're more comfortable with the code, they can turn their attention to what it is about AJAX that makes it so compelling for people, and explore and push the boundaries of that.

Geelan: *By default now you're a bit of a wordsmith, so let me ask you this. In exactly this space we have that word "rich" that you just used, unpacked by Gartner quite usefully, and that's making some progress, but they're having to work hard because rich is such a vague word. Then we have, and you know what I'm going to say next, Web 2.0. Now somebody – it wasn't*

Advertising Index

Advertiser	URL	Phone	Page
Adobe	www.adobe.com	800-833-6687	99
Apress	www.apress.com		65
Backbase	www.backbase.com/jsf	866-800-8996	51
Component Art	www.componentart.com	416-622-2923	25, 61
Cynergy	www.cynergysystems.com	1.866.CYNERGY	4
helmi	www.helmitechnologies.com	+358 40 557 4004	31
IBM	www.ibm.com		37
Icesoft	www.icesoft.com	877-263-3822	43
Infragistics	www.infragistics.com	1-800-231-8588	33, 47
JackBe	www.jackbe.com	240-744-7620	7
Laszlo	www.openlaszlo.org		23
Nexaweb	www.nexaweb.com	781-345-5500	9
Parasoft	www.parasoft.com/WSJmagazine	888-305-0041 x3501	57
telerik	www.telerik.com	1-866-886-0452	77, C4
Tibco	http://developer.tibco.com/	1-650-846-1000	C2
Zapatec	www.zapatec.com	866-522-7941	71

Jesse this time – somebody comes up with it – you can argue whether it was Tim O'Reilly or someone else. Now comes another, even more strange than rich, what is your take on that? Is that going to help? It seems to me that AJAX comes to be the focal point precisely because it's tighter, better defined, and you came at it with a philosophy almost, which Web 2 is kind of searching for and rich just doesn't even get it.

Garrett: I think you're right, that Web 2.0 as a concept or as a label has been used and applied to a wide variety of concepts that seem to have very little in common.

Geelan: *That could change.*

Garrett: Yes, but I think that the essential thing that people are trying to express when they use the term Web 2.0 is that something new is happening, something exciting is happening, and that the medium itself…

Geelan: *It serves a purpose. It's like a rallying cry, something's going on.*

Garrett: What they're saying is that the medium is evolving, that for all of the people, the people who may not be as close to Web technology as we are, who maybe thought that the Web was something whose time had come and gone perhaps, or that the Web had reached this plateau state. This is a way of communicating to those people that the Web is still in motion and that there are still a lot of changes afoot, and that, I think, is the real value of Web 2.0.

Geelan: *So you're not against it. It's serving a purpose.*

Garrett: I'm not against it. I think it's something that the more you try to define it, the less useful it becomes because that's not really what it's about.

Geelan: *We have this richness, rich media, versus you've gone more with just user-centric, which seems to be much more friendly. You just said this, I like that. If the Web now is going to become user-centric, what centric was it before?*

Garrett: Well, I think we actually do see still, on a very regular basis, technology companies putting things out in the world because they were really interested in solving a particular technical problem.

Geelan: *Because it could be done.*

Garrett: Right, rather than doing something because there's a strong sense of a need out there in the marketplace or there's a way that we can deliver something compelling and innovative to users, they are saying, if we take this technology, this technology, and this technology and we put them all together, let's throw them out in the marketplace and see what people do with them. That ends up being a much more difficult kind of path to take.

Geelan: *Let's take another slice of it while we've got you sort of in a lexicological mode, the one-page Web; is that a useful metaphor? It's kind of floating about.*

Garrett: I think there's a limited sense in which the one-page Web is true. I still think that there are a lot of applications for which having multiple sections or pages still make a lot of sense, so I don't think we're going to be reducing the whole Web down to one page.

Geelan: *We seem to be on to something; let's now get what we want out of it, something useful, you seem to be very passionate about it.*

Garrett: Yes, and I think that we, as a community, we've learned a lot. We've learned a lot from the boom; we've learned a lot from the bust in the Web industry about how to pursue the evolution of this medium in a way that is a little bit smarter and a little bit more savvy and taking a course that will really deliver value for people.

Geelan: *Have you ever wavered in your love of the Web?*

Garrett: Not really.

Geelan: *You said yourself that some people are mainly thinking plateau; some of them think it's gone.*

Garrett: I haven't really. I had a lot of people around the time that the bubble burst in 2000, 2001, ask me if I was going to try to do something else, maybe get out of technology altogether. However, it was at that time that I kind of redoubled my commitment to the Web by starting Adaptive Path, which started in the spring of 2001. So I've always been a big believer in the Web and, if anything, my experience in the last year has just kind of confirmed that belief.

Geelan: *As the Chinese say of love, and we're talking about a love affair with the Web. The Chinese say that there are two types to love. It can be like a hot kettle that's put on a cold stove. After a while, of course, the steam goes out of it. It seems to me that AJAX and Jesse James Garrett are a cold kettle – no disrespect – but you're a cold kettle put on a hot stove; you're coming to the boil and it's going to – oh my god – just watch this space. Thank you so much, Jesse James Garrett.*

Garrett: Thank you for having me. ■

OFFICIAL SHOW DIRECTORY

PRODUCED BY
SYS-CON EVENTS
www.events.sys-con.com

AJAXWORLD™
CONFERENCE AND EXPO

Learn from the Best...

Experience AJAX, RIA, and Web 2.0 Knowledge and Best Practices

OCTOBER 2-4

THE SANTA CLARA HYATT REGENCY IN SANTA CLARA

AJAXWORLD CONFERENCE AND EXPO

WELCOME LETTER

Greetings, fellow i-Technology professionals, and welcome to the first-ever AJAXWorld Conference & Expo.

The ripple effect of Jesse James Garret's four-letter coinage from 2005 continues to rip through the entire software development world. The dynamic Web applications that characterize the "AJAX" approach have begun to alter forever the landscape of information architecture and user experience.

From AJAX to full-blown "Web 2.0" is but a hop and a skip, but just as there are those who contend that RSS, not AJAX, is the key enabler of Web 2.0 (just as http was of Web 1.0), so we are mindful of how many other technologies there are on the current scene, from CSS and JavaScript and the DOM to AJAX-Compatible RIA Technologies like Flex, Laszlo, Dojo, and the rest.

While this inaugural AJAXWorld Conference & Expo then is devoted to the rich-web world that's been catalyzed by the term given to us Garrett (who is keynoting on Tuesday, the opening day of the conference program) but it is inclusive in its approach, and we have sessions about complementary technologies of every stripe, as well as about the business aspects of AJAX-type applications and services.

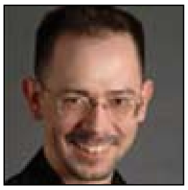
Please let me know personally if there is anything we have missed, and we will make sure it is addressed in a future AJAXWorld Conference & Expo. Meantime, hold onto your hat: AJAXWorld 2006 is going to be a blast!

Cordially,

Jeremy Geelan

Jeremy Geelan
Conference Chair

WHAT'S INSIDE



AJAXWORLD Keynote:
Jesse James Garrett

Jesse James Garrett is the Director of User Experience Strategy and a founding partner of Adaptive Path, the world's premier user experience consulting company. He is author of The Elements of User Experience (New Riders), and is recognized as a pioneer in the field of information architecture. Jesse's clients include AT&T, Intel, Crayola, Hewlett-Packard, Motorola, and National Public Radio. Since starting in the Internet industry in 1995, Jesse has had a hands-on role in almost every aspect of Web development, from interface design and programming to content development and high-level strategy. Today, information architects around the world depend on the tools and concepts he has developed, including the widely acclaimed "Elements of User Experience" model. He is co-founder of the Information Architecture Institute, the only professional organization dedicated to information architecture. He is also a frequent speaker and writer whose work has appeared in numerous publications, including New Architect, Digital Web, and Boxes and Arrows.

AJAXWORLD Keynote:
Patrick Grady

A recognized pioneer in Web Services and On-Demand technologies, Patrick Grady has guided Rearden Commerce to a commanding leadership position within the Services-On-Demand market. Five years in development, the company has a large and growing patent portfolio and enjoys first mover advantage, with a host of name-brand enterprise customers and several major technology and marketing partnerships. Grady is a sought after speaker with engagements including PC Forum, Supernova, Burton Catalyst Conference, AlwaysOn, Red Herring, Comdex, InfoWorld Symposium, Enterprise and Internet World. In addition to serving as Rearden Commerce's strategic architect, he oversees Sales, Marketing, Product Development, Finance, and Operations. Prior to founding Rearden Commerce, he spent 10 years in various venture capital, private equity, and operational roles in the technology sector. Grady led investments in successful start-ups within the software, wireless, high-performance computing, and networking sectors. He is also on the Board of Directors of the Greater Bay Area Chapter of the Juvenile Diabetes Research Foundation.

AJAXWORLD POWER PANEL

Redefining RIAs: Does AJAX Push the Browser Too Far? by Jeremy Geelan

Jeremy Geelan, currently co-Founder & President of neXplore Technologies, Inc., was from 2000-6 first editorial director and then group publisher of SYS-CON Media. He was responsible for the development of all new titles and i-Technology portals for the firm, and regularly represented SYS-CON at conferences and trade shows, speaking to technology audiences both in North America and overseas. He remains Conference Chair of SYS-CON Events such as the "Real-World Flex" One-Day Seminar and of AJAXWorld Conferenc & Expo. He also remains executive producer and presenter of "Power Panels with Jeremy Geelan" on SYS-CON.TV.

The FrameworkWars: Lightweight vs Heavyweight by Jeremy Geelan

Jeremy Geelan, currently co-Founder & President of neXplore Technologies, Inc., was from 2000-6 first editorial director and then group publisher of SYS-CON Media. He was responsible for the development of all new titles and i-Technology portals for the firm, and regularly represented SYS-CON at conferences and trade shows, speaking to technology audiences both in North America and overseas. He remains Conference Chair of SYS-CON Events such as the "Real-World Flex" One-Day Seminar and of AJAXWorld Conferenc & Expo. He also remains executive producer and presenter of "Power Panels with Jeremy Geelan" on SYS-CON.TV.

Web-Oriented Architecture: SOA + The Web + REST by Dion Hinchcliffe

Building modern SOAs and Web 2.0 sites requires an design style that goes with the "grain" of the Web. REST, a natural extension of HTTP created by the co-inventor of HTTP, is getting a lot of attention these days because it's easy to work with, aligns well with the Web, and is very high performance. Used architectural, REST is called Web-Oriented Architecture or WOA. But whither SOAP, the canoncial Web services protocol up to now? This panel will discuss that and more, including:

—REST has been with us for a while, why is it taking off now?
—What's going to happen to SOAP-based SOAs?
—What are IBM, Microsoft, and BEA doing about REST support in their products?
—REST seems to make Ajax programming much easier than SOAP, why?

AJAXWorld 2006 CEO Power Panel by Dion Hinchcliffe

More and more companies are using so-called "Web 2.0" tools to build their own mashups. They are increasingly looking at other ways to change how they communicate with customers. This special AJAXWorld 2006 CEO Power Panel will discuss the business impact of the new breed of applications that integrate content from many sources in the enterprise and Web community in as little as five minutes. Contributory factors considered will be:

—open APIs - such those now available from eBay, Google, Amazon and Yahoo!
—RSS, AJAX, Comet, blogs, Wikis, Social Interactive software and Web components

PLATINUM SPONSORS

AjaxWorld Conference and Expo

PLATINUM SPONSOR

Adobe

Adobe

The Adobe Engagement Platform enables great experiences through the following key elements: Cross-platform, cross-browser, cross-device clients, a composite-driven programming model, and server frameworks and tools that accelerate the development of compelling applications. Adobe Flex, part of the Engagement Platform, is a complete application development solution for creating cross-platform rich Internet applications. And by combining the best of AJAX and Flex, developers can now build next generation expressive, interactive applications that include vector graphics, audio, video, and charting. For more than two decades, Adobe's award-winning technologies and software have redefined business, entertainment, and personal communications by setting new standards for producing and delivering content that engages people anywhere at anytime. Now that Adobe and Macromedia are one company, Adobe is better positioned than ever to push the boundaries of the digital universe.
www.adobe.com

PLATINUM SPONSOR

CYNERGY

Cynergy Systems

Cynergy Systems delivers Web, RIA and Legacy Migration solutions to over 1,400 customers worldwide. Maintaining offices across three continents and leveraging over a decade of experience, Cynergy Systems focuses on the delivery of rich, powerful business applications to customers across dozens of verticals including Healthcare, Manufacturing, Banking, and Governmental Organizations. Partners include Adobe, Microsoft, Sybase, Percussion Software and Jive Software.For more information, visit
www.cynergysystems.com/.

PLATINUM SPONSOR

JACKBE™

JackBe

JackBe is the premier provider of software and services that combine the strengths of Ajax and SOA with reliable, optimized Web connectivity to deliver Rich Enterprise Applications (REA). Our complete REA platform enables business unit developers to quickly create desktop-like Web applications and enterprise mashups while still maintaining centralized IT governance, security, and control. JackBe solutions leverage your existing SOA and middleware investments over the complete path from server to browser to smoothly integrate internal and external services into enterprise-class Web applications.

PLATINUM SPONSOR

TIBCO®
The Power of Now®

TIBCO Software

TIBCO Software Inc. is a leading business integration and process management software company that enables real-time business. Real-time business is about giving organizations the ability to sense and respond to changes and opportunities as they arise. TIBCO has delivered the value of real-time business, what TIBCO calls The Power of Now®, to over 2,500 customers around the world and in a wide variety of industries. To learn about TIBCO's solutions for service-oriented architecture (SOA), business process management (BPM) and business optimization, contact TIBCO at +1 650-846-1000 or on the Web at www.tibco.com. TIBCO is headquartered in Palo Alto, California.

GOLD SPONSORS

GOLD SPONSOR

Backbase
Backbase offers Rich User Interface / AJAX software to enable Web developers to create a much richer user experience for their online applications. Our AJAX technology combines the “reach” of Web applications with the “richness” of desktop applications. Backbase AJAX software can be used with a normal Web browser, without the need to install plug-ins such as Flash or Java. Backbase includes a complete development and runtime environment, which makes development of AJAX applications efficient.
www.backbase.com

GOLD SPONSOR

ComponentArt
ComponentArt is a privately held software design and development firm in Toronto, Canada, specializing in the creation of software components for Microsoft’s .NET platform. Our products help developers and organizations worldwide reduce their development time and effort, save money, and gain peace of mind. ComponentArt’s current focus is on user interface elements for Web site and Web application development, as well as data visualization solutions for both Web and Windows applications.
www.componentart.com

GOLD SPONSOR

Helmi Technologies
Helmi Technologies provides an AJAX-based development platform for rapidly and cost-effectively building and deploying browser independent Rich Internet Applications (RIAs). Our object-oriented RIA platform enables developers to effectively create rich, high-performance and highly interactive Web applications by leveraging their existing development skills and development environments, such as Eclipse and JBoss. Our mission is to empower our customers to maximize their creativity and innovation, so that they can increase customer loyalty, optimize productivity, and strengthen competitive advantage.
www.helmitechnologies.com

GOLD SPONSOR

IBM
IBM is a key supporter of industry standards like AJAX and shows leadership in innovations in this emerging technology through participation in OpenAjax, an open industry collaboration dedicated to developing and expanding Ajax. Leverage the power of IBM on your AJAX development projects. Whether you’re looking for free web-based training , technical articles , or to interact and get your questions answered through forums , you can find all the resources you are looking for on IBM developerWorks.
www.ibm.com

GOLD SPONSOR

ICEsoft
ICEsoft Technologies Inc. is the leading provider of standards-compliant, AJAX-based solutions for deploying pure Java, Rich Internet Applications. ICEfaces is the first integrated AJAX application framework that enables Java EE application developers to create and deploy thin-client, rich Internet applications in pure Java. The Community Edition of ICEfaces is a fully featured product that enterprise developers can use to develop rich Java EE applications at no cost. ICEfaces delivers the features of Thin Client AJAX and AJAX Push Technology.
www.icesoft.com

GOLD SPONSOR

Infragistics
Infragistics has been the market leader in the presentation layer components industry for over 17 years, and has matured into a multi-platform Enterprise Software products and services vendor with global reach in nearly every Fortune 2000 company. Infragistics empowers developers to build great application UIs for Windows Forms, ASP.NET, Tablet PC, JavaServer Faces and Windows Presentation Foundation platforms, and additionally offers test tools, support, training and consulting services.
www.infragistics.com

GOLD SPONSOR

Laszlo Systems
Laszlo Systems is the original developer of the open source platform OpenLaszlo, and provider of Rich Internet Applications and services that advance the Web experience. OpenLaszlo is an XML-native foundation for building next generation Web applications that increase customer retention, conversion and brand loyalty. Laszlo provides comprehensive support services, education, and commercial application modules so that any company can easily make the move to Rich Internet Applications.
www.laszlosystems.com

GOLD SPONSOR

Nexaweb
Nexaweb’s Enterprise Web 2.0 solutions enable businesses to deploy Richer, Thinner, Faster applications over the Web. Specifically, applications built with Nexaweb deliver the high performance and robust functionality of client/server software with the universal reach, no-install deployment and centralized management of browser-based applications. With Nexaweb, developers can build enterprise-strength RIA solutions using Ajax and Java that access legacy, Web services, and SOA technologies in a unified declarative XML development environment.
www.nexaweb.com.

GOLD SPONSORS/EXIBITORS

GOLD SPONSOR

Parasoft

Parasoft is the leading provider of innovative solutions for automating software test and analysis and for establishing software error prevention practices as an integrated part of the software development lifecycle. Parasoft products and services enable software development and IT organizations to significantly improve visibility and control over the quality, costs and schedules of their software projects through the practice of Automated Error Prevention (AEP). Parasoft's easy-to-use, scalable and customizable software error prevention solutions span the complete software development life cycle and automatically test complex software systems from all relevant perspectives.

www.parasoft.com

GOLD SPONSOR

telerik

telerik is a leading vendor of ASP.NET UI components, content management solutions and add-ons for MCMS 2002 and SharePoint. Building on our expertise in interface development and the potential of AJAX we can help you take the richness and responsiveness of desktop applications to the Web. Our suite for Web development is the most comprehensive toolset on the market, featuring the leading grid for ASP.NET, WYSWYG editor, drop-down menu, navigation and charting controls.

www.telerik.com

ASSOCATION SPONSOR

OASIS

OASIS (Organization for the Advancement of Structured Information Standards) is a not-for-profit, international consortium that drives the development, convergence, and adoption of e-business standards. The consortium produces more Web services standards than any other organization along with standards for security, e-business, and standardization efforts in the public sector and for application-specific markets. Founded in 1993, OASIS has more than 5,000 participants representing over 600 organizations and individual members in 100 countries.

http://OASIS/home/index.php

EXHIBITORS

Apress

Apress® is a publishing company established by computer professionals, programmers, and enthusiasts just like you. Apress books are tools that will help you learn new subjects, solve problems effectively, and save time. Practical, instructional, and thought-provoking, Apress books offer the highest quality content, distinctively written with The Expert's Voice™. www.appress.com

Google

Google is a global technology leader focused on improving the way people connect with information. Google's innovations in Web search and advertising have made its Website a top Internet destination and its brand one of the most recognized in the world. Google maintains the world's largest online index of Websites and other content, and Google makes this information freely available to anyone with an Internet connection. Google's automated search technology helps people obtain nearly instant access to relevant information from its vast online index. www.google.com

ILOG

ILOG delivers software and services that empower customers to make better decisions faster and manage change and complexity. Over 2,000 global corporations and more than 400 leading software vendors rely on ILOG's market-leading visualization, business rule management system, and optimization software components to achieve dramatic returns on investment, and create market-defining products and services. The ILOG JViews family of products delivers unsurpassed displays for the web or desktop—shortening Java development, cutting costs, reducing risk—and improving the user's experience. There's a JViews product for every visualization need: data charts, Gantt charts, mapping, diagramming, networking and monitoring. At AJAX World Conference & Expo, we will be featuring ILOG JViews TGO, the industry's leading component for building network and equipment-management interfaces, enabling rapid assembly of visualization layers for the new generation of highly flexible OSSs, Tree views, Table views and more. ILOG was founded in 1987 and employs more than 730 people worldwide.

http://www.ilog.com.

Sun Microsystems

Sun Microsystems, a creator and industry leading advocate of emerging technologies, is revolutionizing and redefining system-wide management of rich, standards-compliant, Internet applications for the Web 2.0 generation. Beyond technologies, Sun delivers a complete, interworking infrastructure for today's businesses, from state-of-the-art eco-friendly servers and storage systems to award-winning, open, integrated, and interoperable software offerings. When it comes to Web 2.0 projects, Sun provides best-of-breed solutions to enterprises worldwide for developing, deploying, and managing next-generation web applications. For more information, please visit: http://sun.com/ajax.

ThinWire

ThinWire® is an open source development framework that allows you to easily build applications for the web that have responsive, expressive & interactive user interfaces without the complexity of the alternatives. While virtually any web application can be built with ThinWire®, when it comes to enterprise applications, the framework excels with its highly interactive and rich user interface components. Use ThinWire® to handle the view-layer of your Java EE (J2EE) application and you'll be able to provide an unparalleled user experience, while at the same time completing your project faster than ever. Look over our list of features and try out our Playground Demo, we're sure you'll find our approach unique and powerful.

U7 Web Technologies

U7 Web Technologies leverages the Google Web Toolkit (GWT) to produce cutting edge AJAX based web applications. U7 has produced the U7 Web Toolkit (U7WT), which builds upon GWT. The U7WT addresses issues such as browser to browser messaging, advanced page layout and design, and component based application construction. U7 Web Technologies is a technology R&D company focused on developing advanced AJAX technology. U7WT is a wholly owned subsidiary of U1 Consulting Group.

Visible Measures

Visible Measures, headquartered in Cambridge, Massachusetts, develops and markets advanced software usage analytics solutions that show how, where, and when users interact with AJAX-enabled Rich Internet Applications (RIA). The insights gained from Visible Measures can dramatically increase the profitability of any software application development project; can help guide and inform selling and marketing strategies; and can improve customer satisfaction by enabling applications that satisfy actual user needs.

Zapatec

Zapatec is the leading provider of thin client, AJAX development tools. Zapatec targets medium-to-large enterprise customers. Its roster of customers includes HP, Boeing, Time Warner Cable and Cartier. In addition to the Zapatec AJAX Suite, Zapatec offers individual AJAX tools.

MEDIA SPONSORS

MEDIA SPONSORS

AJAX Matters
AJAX Matters provides AJAX development information including code libraries, sites using ajax, and articles on AJAX.

AJAXWorld Magazine
AJAXWorld Magazine is the pre-eminent independent vendor-neutral resource for the fastest growing new segment of the software business: entirely Web-based applications and experiences like Gmail, Google Maps, Live.com, MySpaces, and Flickr. AJAXWorld Magazine recognizes that the next-generation user-centric Web is hurtling toward us and that it's a rich-media future in which AJAX, as the most talked about of all the Rich Internet technologies, is positioned firmly at center stage. AJAXWorld Magazine provides developers, architects, IT managers, and corporate decision makers with comprehensive information and insight into the biggest paradigm shift in website design, development, and deployment since the invention of the World Wide Web fifteen years ago.

BZmedia
BZ Media LLC is a privately held, high-tech media company based in Huntington NY on Long Island's beautiful north shore. BZ Media was founded in 1999 by veterans from Miller Freeman, CMP, IDG, Newsday and Ziff-Davis.

DevtownStation.com
DevtownStation.com is an independant source portal for software developer conferences, including Linux, Java, XML, .NET, Open Source and other developer events, conferences, webinars and training.

Eclipse
Eclipse Developer's Journal is the only monthly source of quality Eclipse information.

Eclipse Review
Eclipse Review is a new quarterly magazine for IT professionals, including software developers, who use Eclipse-based tools and technologies.

Integration Developer News
Integration Developer News (IDN) believes that Enterprise Developers (Java, ASP/.NET, C/C++, COBOL, etc.) need to continually expand their skill set so they can bring together front-end and back-end systems (web, app server, legacy) to create web services and integrated solutions...

ITtoolbox
ITtoolbox is an online community, enabling peers to share professional knowledge about information technology. Since 1998, ITtoolbox has helped professionals make IT decisions and stay current in the rapidly changing IT market through peer collaboration. The ITtoolbox platform incorporates blogs, discussion groups, and a wiki, facilitating targeted community interaction that IT advertisers can participate in through a proprietary contextual matching system. This combination of community and advertising value has made ITtoolbox a leading destination for professionals and a leader in online advertising, performing for over 520 clients including Microsoft, IBM, Oracle, Dell, and HP.

Java Developer's Journal
Published monthly, Java Developer's Journal is the premier independent, vendor-neutral magazine serving the information needs of the entire community of developers in the Java programming language and the Java platform. Crammed with Internet- and Web-related features, regular columns by established Java gurus, "how-to" programming tips, product reviews, all the latest industry news and more, JDJ offers mission-critical and insightful coverage of developments occurring as a result of the fast-moving global adoption of Java's multi-platform, "write once, run everywhere" technology.

LinuxWorld.com
Published monthly, The new LinuxWorld.com, the open-source technology site for enterprise business decision-makers and implementers. While LinuxWorld.com was formerly produced by a licensee of International Data Group, the site is now owned and operated by IDG's Network World - drawing on IDG's broad information technology resources (see IDG.com).

MEDIA SPONSORS

Methods & Tools
Methods & Tools is a free magazine with PDF and text issues that provide practical knowledge and information on all topics of software development and software engineering: UML, Agile Methodologies (eXtreme Programming, Scrum, TDD, FDD,..), Software Testing, Software Configuration Management, Database Modeling, Java, .NET, RUP, Software Project Planning and Management, Test Automation, Programming, Software Analysis and Design, Quality Assurance, Software Process Assessment and Improvement, Software Development Tools, Risk Management, Refactoring, IT News, etc.

Network World
Technology leaders rely on Network Computing in print, online and at our tech-focused events because they know the content is steered by IT professionals like themselves. Our real-world IT analysts know the challenges, know the technologies and know what's important to our audience. Our peer-to-peer perspective and solid grounding in the reality of IT allows us to create a deep audience connection continuously increasing their trust in Network Computing as a valuable resource.

Open Enterprise Trends
Open Enterprise Trends (OET) believes that Open Source software is becoming more critical to the success of enterprise developers building next-generation solutions. Our mission is to "bridge the gap" that now exists between many Open Source and commercial solutions, and to provide all developers with the expertise and perspective they need to take better advantages of both technology types.

Sarbanes-Oxley Compliance Journal
Sarbanes-Oxley Compliance Journal keeps you up to date on the latest techniques to gain compliance and keep compliant as the interpretation of the Act evolves.We also keep you abreast of; Proposed Rules, Final Rules, Concept Releases, Interpretive Releases, Policy Statements and PCAOB Rulemaking. With a special focus on how they may affect you and how to deal with them. Sarbanes-Oxley Compliance Journal content is selected and reviewed by diverse editorial review process teams. During our editorial review process, our internal editors, editorial review board members and people working directly in the profession participate in a variety of ways to deliver a balanced picture of a particular part of the Act. The goal is to accurately inform the reader.

SDTimes
SD Times is the only IT trade newspaper specifically written for the software development manager. The broad-based IT newsweeklies abandoned this reader a decade ago. Now you can reach them directly, efficiently and inexpensively in SD Times.

Software Test & Performance
Software Test & Performance is offered by BZ Media LLC, publishers of SD Times, and producers of the Software Test & Performance Conference.

SOA Web Services Journal
SOA Web Services Journal is the premier publication addressing the technical and strategic depth of Web Services. Its target audience is anyone who wishes to design, administer, sell, or use Web Services - from the developer, to the ISV, to the CTO, and to the end user. It is for anyone who wishes to apply the new model for creating and using distributed applications across the Internet, utilizing common interfaces for efficient communication and high level interoperability.

SYS-CON TV
SYS-CON TV is your complete resource for LIVE i-Technology News and Views about the latest trends.

Web 2.0 Journal
Web 2.0 Journal is the premier publication covering the ongoing transformation of the Worldwide Web through Rich Internet Application (RIA) developent, AJAX and related development technologies, and service-oriented architectures (SOAs).

Web Developers & Designers Journal
For professional web developers and designers who use the award-winning products from Adobe to develop and design rich internet applications and content that engage people anywhere at anytime. WebDDJ is written by industry experts with contributed content from Adobe employees. It features articles, tutorials, success stories, sneak previews, interviews with Adobe senior staff, programming tips, and more.

CONFERENCE AT-A-GLANCE

AJAXWORLD™ CONFERENCE AND EXPO

Day 1: Tuesday, October 3, 2006

	TRACK 1: Real-World AJAX	TRACK 2: Key Issues	TRACK 3: Web 2.0	TRACK 4: Flex	TRACK 5: AJAX-Compatible RIA Technologies	TRACK 6: Ruby on Rails
7:00a.m.-7:30a.m.	Registration & Breakfast					
7:30a.m.-8:15a.m.	Opening Keynote by Jesse James Garrett					
8:15a.m.-9:05a.m.	**Session 2** Presentation & Demo by Adobe at the Keynote Hall					
9:05a.m.—9:50a.m	Coffee Break / Expo Floor Open					
9:50a.m.—10:35a.m	**Session 3**					
	AJAX By Numbers: A Statistical Snapshot *John F. Andrews*	From Enriched HTML to Client/SOA: The Four Quantum States of AJAX *Kevin Hakman*	Enabling Inter-Mashup Interactions Using Composite Data Services and Workflow *Ash Parikh, Ajay Ramachandran, & Robert Smik*	Web 2.0 with Flex: Delivering ROI to Enterprise Apps *Mike Nimer*	Foundations of Atlas: Ajax programming in ASP.NET *Laurence Moroney*	Rails Reduces the Number of Decisions when Building Web Applications *Alex Bunardzic*
10:35a.m.-11:05a.m.	**Session 4** Presentation & Demo by Laszlo Systems at the Keynote Hall					
11:05a.m.-11:35a.m.	**Session 5** Presentation & Demo by Nexaweb at the Keynote Hall by David McFarlane and Coach Wei					
11:35a.m.-12:20p.m.	**Session 6**					
	The SOA Client and the Web's Copernican Revolution *David Temkin*	Ajax Debugging and Quality Assurance *Dave Johnson*	A Venture Capitalist View of Web 2.0 Value *Neil Sequiera*	Flex with Java *James Ward*	TBA *Bob Buffone*	Test Driven Design *Ryan Davis*
12:20p.m.-1:15p.m.	Lunch Break / **Session 7** Power Panel: Web-Oriented Architecture: SOA + The Web + REST / Expo Floor Open					
1:15p.m.-1:45p.m.	**Session 8** AJAX-Enabling Applications in 2 Minutes Presentation & Demo by telerik at the Keynote Hall by Sahil Malik					
1:45p.m.-2:30p.m.	**Session 9**					
	Designing for Ajax *Bill Scott*	Focusing on the "A"(rchitecture) in AJAX *John Crupi*	JSON: The Data Format of the Stars *Douglas Crockford*	Taking RIAs to the Desktop with Apollo, Adobe's Next Generation Client *Luis Polanco*	TBA	Acceptance Testing Rails Apps with Watir *Dave Hoover*
2:30p.m.-3:15p.m.	**Session 10**					
	Y! vs. Y! vs. Y!: Across the Document Application Spectrum, One Size Does Not Fit All *Nate Koechley*	Building Dynamic Flex Applications for the Enterprise *Joe Orbman*	TBA *Geoffrey Cubitt & Jeff Maling*	Creating Great Mapping Mashups Using Flex *Mansour Raad*	Architecting Great Internet Applications with AJAX *Scott Isaacs*	Behaviour Driven Development with RSpec *Steven Baker*
3:15p.m.-4:05p.m.	**Session 11** Towards a Service Oriented Application Stack Presentation & Demo by Tibco at the Keynote Hall by Matt Quinn					
4:05p.m.-4:50p.m.	**Session 12**					
	Creating Web applications that look and feel like desktop GUIs using TIBCO GI and Open Source AJAX libraries. *Luke Birdeau*	Professional JavaScript *Douglas Crockford*	Leveraging the Web 2.0 Movement *Dion Hinchcliffe*	Case Study: The JBoss Flex-based mail client *Andrew C. Oliver*	Ajax, Design, and Mobile Devices *Alex Russell*	Rails meets the Legacy-World *Robby Russell*
4:50p.m.-5:20p.m.	**Session 13** Presentation & Demo by Parasoft at the Keynote Hall					
5:20p.m.-5:50p.m.	**Session 14** AJAX Push - Transform the User Experience Presentation & Demo by ICESoft at the Keynote Hall by Steve Maryka					
5:50p.m.-6:30p.m.	Coffee Break / Expo Floor Open					
6:30p.m.-7:15p.m.	**Session 15**					
	Ajax (in) Security *Billy Hoffman*	TBA *Shanku Niyogi*	The Wider Impact of Web 2.0 Including the Use of AJAX *Ajit Jaokar*	SAP's use of Flex to change the face of enterprise applications *Boris Kabisher and Manish Jiandani*	Comet and vector graphics with Dojo *Dylan Schiemann*	Plugging into Rails *James Adam*
7:15p.m.-8:00p.m.	**Session 16** Power Panel: Redefining RIAs: Does AJAX Push the Browser Too Far					
8:00p.m.-8:45p.m.	Welcome Reception					

PLEASE NOTE: SCHEDULE AND SPEAKER LINE-UP SUBJECT TO CHANGE WITHOUT NOTICE

AJAXWORLD CONFERENCE AND EXPO

CONFERENCE AT-A-GLANCE

Day 2: Wednesday, October 4, 2006

	TRACK 1: Real-World AJAX	TRACK 2: Key Issues	TRACK 3: Web 2.0	TRACK 4: Flex	TRACK 5: AJAX-Compatible RIA Technologies	TRACK 6: Ruby on Rails
7:00a.m.-7:30a.m.	Registration & Breakfast					
7:30a.m.-8:15a.m.	Keynote by Patrick Grady					
8:15a.m.-9:05a.m.	**Session 2** Presentation & Demo by Cynergy Systems at the Keynote Hall					
9:05a.m.—9:50a.m	Coffee Break / Expo Floor Open					
9:50a.m.—10:35a.m	**Session 3**					
	AJAX: Making the Enterprise Jump *Kurt Cagle*	Real-World Scalability *Eric Hodel*	Mashing up your Ajax application *Paul Rademacher*	How OpenAjax Alliance will Accelerate Customer Success with AJAX *Jon Ferraiolo*	Real-time Location Tracking Mashup for the Enterprise *Rama Gurram, Louenas Hamdi, & Samir Raiyani*	"AJAX" - Asynchronous Cross-Domain Scripting *Zeki Mokhtarzada*
10:35a.m.-11:05a.m.	**Session 4** Presentation & Demo by ComponentArt at the Keynote Hall					
11:05a.m.-11:35a.m.	**Session 5** Presentation & Demo by Helmi Technologies at the Keynote Hall					
11:35a.m.-12:20p.m.	**Session 6**					
	Apache Derby - A Local AJAX Data Store *Francois Orsini*	Why browsers (still) matter *Hakon Wium Lie*	Paid Content in a Web 2.0 World *Dave Hanley*	Java EE 5 BluePrints for AJAX-Enabled Web 2.0 Apps *Inderjeet Singh*	Ajax and Beyond with ThinWire *Joshua Gertzen*	Google Gadgets and Gadget Discovery *Adam Sah*
12:20p.m.-1:15p.m.	Lunch Break / **Session 7** Power Panel: AJAXWorld 2006 CEO Power Panel / Expo Floor Open					
1:15p.m.-1:45p.m.	**Session 8** Presentation & Demo by JackBe at the Keynote Hal					
1:45p.m.-2:30p.m.	**Session 9**					
	How to Implement Rich Internet Applications *Steve Mulder & Riccardo La Rosa*	Expediting Application Development with the Flex Application Starter Toolkit (FAST) *John Bennett*	Enabling User Specific Mashups *Dan Theurer*	J2EE Security in Flex Applications *Rob Rusher*	AJAX and Faces: Friends or Foes? *Jonas Jacobi & John Fallows*	AJAX3D: The Open Platform for Rich 3D Web Applications *Tony Parisi*
2:30p.m.-3:15p.m.	**Session 10**					
	Ajax for Rich Internet Apps: Lessons from 1 million Ajax email clients *Florian von Kurnatowski*	Migrating Legacy Apps to Use AJAX *Dror Matalon*	Mobile Browsers: Where Next? *Martin Frid-Neilson*	Encapsulating AJAX Functionality in Java Server Faces Components *Craig R. McClanahan*	Encapsulating AJAX Functionality in Java Server Faces Components *Craig R. McClanahan*	Rich Ajax Platform (RAP) — Web 2.0 the Eclipse Way *Jochen Krause*
3:15p.m.-4:05p.m.	**Session 11** Presentation & Demo by IBM at the Keynote Hall					
4:05p.m.-4:50p.m.	**Session 12**					
	Storyboarding Ajax Applications *Eric Pascarello*	Ruby on Rails + AJAX *Michael Buffington*	QEDWiki, A Web 2.0 Application Assembler for the Mass Market *David Boloker*	A New Approach to AJAX: Asynchronous Java + XML *Coach Wei*	Developing RIA's from Front-To-Back: Turning the Process Upside Down *Dave Wolf*	Developing with AJAX for Online Widget Platforms *Alan Lewis*
4:50p.m.-5:20p.m.	**Session 13** Presentation & Demo by Infragistics at the Keynote Hall					
5:20p.m.-5:50p.m.	**Session 14** Presentation & Demo by Backbase at the Keynote Hall					
5:50p.m.-6:30p.m.	Coffee Break / Expo Floor Open					
6:30p.m.-7:15p.m.	**Session 15**					
	Exposing and Preventing Errors in AJAX Applications *Nathan Jakubiak*	Eclipse AJAX Tooling Framework *Robert Goodman and David Boloker.*	Integrating Rails into the Enterprise through SOA *Joe O'Brien*	You're Running a Database Where? — A Persistence Architecture for Local AJAX Storage using Apache Derby *David Van Couvering*	3-Tier No More: Integration-Ready Applications with AJAX and WS *Samisa Abeysinghe*	AJAX: Not The Only Game In Town *Jim Phelan*
7:15p.m.-8:00p.m.	**Session 16** Power Panel: The FrameworkWars: Lightweight vs Heavyweight					

PLEASE NOTE: SCHEDULE AND SPEAKER LINE-UP SUBJECT TO CHANGE WITHOUT NOTICE

SPEAKERS

AjaxWorld Conference and Expo

Three-Tier No More: Integration-Ready Applications with AJAX and WS
Speaker: Samisa Abeysinghe

Samisa Abeysinghe is a software architect at WSO2. Samisa pioneered the Apache Axis2/C effort, architected the core of the Apache Axis2/C Web services engine, and continues to be an active contributor. His involvement in open source projects began in 2004 with the Apache Axis C/C++ project.

Plugging into Rails
Speaker: James Adam

Dr. James Adam is the developer behind Rails Engines and the author of the upcoming 'Rails Plugins' shortcut from Addison Wesley. He has five years of experience as a contributor to the Ruby community and is a Rails early adopter and contributor. James currently works as part of a small internal development team in London, producing real-world Rails applications within a corporate environment.

AJAX by Numbers: A Statistical Snapshot
Speaker: John F. Andrews

John F. Andrews is the president of Evans Data Corporation. He has served as CEO of a variety of IT companies, both private and public. Prior to these CEO assignments he was CIO of CSX Corporation, and was also a CIO and division president at several of GTE's operations. John also has extensive experience in market research, having been the CEO of Giga Information Group before it was acquired by Forrester Research.

Behavior-Driven Development with RSpec
Speaker: Steven Baker

Steven Baker is the one of the key figures in the Ruby community when it comes to agile software development. He is the creator and lead developer of RSpec, the Behavior-Driven Development framework for Ruby, and is a featured speaker on applying agile methodologies at many of the Ruby and Ruby on Rails conferences. Steven continues to collaborate with leaders of the agile and Ruby communities. He provides training and mentoring on how organizations can improve their productivity and efficiency through workshops and private sessions. More information on Steven, his writings, and his workshops can be found at http://www.stevenbaker.ca/.

Session: TBA
Speaker: Stephane Bastain

Stephane Bastian is the founder of Otrix; a company specialized in building AJAX-enabled Java Server Faces Components. Prior to founding Otrix, Stephane held senior engineering positions at CommerceOne, and participated in building leading e-Marketplaces and e-Procurement solutions for the world's leading companies -- Boeing, Deutsche Telekom, General Motor, Schlumberger, Siemens and others. Previously, Stephane worked as a Senior Software Engineer for ComponentOne, a leading provider of components for the .net platform. He holds a master's degree in computer science and is also a member of the JSR 273 expert group (Design-Time API for JavaBeans JBDT) and JSR 276 expert group (Design-Time Metadata for Java Server Faces Components).

Expediting Application Development with the Flex Application Starter Toolkit (FAST)
Speaker: John Bennett

John Bennett leads the architect community in Adobe Consulting's Eastern region. Adobe Consulting acts as a catalyst for Flex and IDP development projects with strategic customers and partners, driving early phase development projects that demonstrate best practices in Adobe technologies.

Using Open Source Libraries and Tools to Build AJAX Apps That Rival Desktop GUIs
Speaker: Luke Birdeau

Luke Birdeau is the creator and lead software engineer for TIBCO General Interface ("GI"), the acclaimed AJAX toolkit for building Web applications that look and feel like desktop GUIs. Luke pioneered AJAX-style architectures, first releasing TIBCO GI in 2001. Several generations of releases and proven performance in global companies have led to the recognition that Luke's work is one of the most mature AJAX Rich Internet Application toolkits available.

QEDWiki, A Web 2.0 Application Assembler for the Mass Market
Speaker: David Boloker

David Boloker is a distinguished engineer and chief technical officer for Emerging Internet Technologies in the IBM Software Group. Previously, he held the position of chief technical officer for the Java Technologies in Software Group. David is recognized in and outside IBM as a technical leader in the Internet software space guiding IBM's investments as well as internal product development in the Internet space. David's responsibilities include building IBM's technical e-Business strategy, working with internal IBMers to develop the appropriate products for the Internet space, researching new areas in software design as well as guiding a group of researchers working in the Software Group. David has been recognized by IBM with numerous Outstanding Technical Achievement Awards in the areas of Java, Secure Internet Gateways, Dynamic I/O Configuration and the Remote Distribution and Control of S/370, 4300, S/390 and AS/400 Architecture processors.

AJAXWorld "Power Panel" - Redefining RIAs
Speaker: Robert Brewin

Robert Brewin, a Sun Distinguished Engineer and co-CTO of Sun Software, is responsible for developer products and application platforms, which includes the Java platform, mobility, enterprise software and business integration products. Within this role, some of his key areas of responsibility include Sun's ever-expanding role in Web 2.0 technologies as well as generally improving the developer experience, alignment and integration of our platforms, technologies, and tools in a next-generation Web world. Bob will be bringing his extensive customer- and developer-focused experience to bear since prior to taking on this new role, he was the chief architect for Sun's developer tools portfolio, where he was a principal driver behind a number of key initiatives and projects. These included strategic enhancements to the award-winning NetBeans IDE, architectural changes, and alignment of the Studio family of tools and add-ons. Prior to joining Sun over 13 years ago, Bob worked at Taumetric Corporation, a supplier of compiler technology to many industry-leading companies.

Ruby on Rails + AJAX
Speaker: Michael Buffington

Michael Buffington has been an entrepreneur and Web application developer for over 10 years. He cofounded Price.com Inc. in 1998, authored an advanced-level book on Macromedia's ColdFusion, and helped architect measuremap.com, Adaptive Path's new Weblog analytics service. Most recently, Michael has become notable in the Rails community for developing and managing the open source llor.nu online game, a massive and addictive game with some simple underlying principles.

Introduction to Apache XAP
Speaker: Bob Buffone

Bob Buffone, Chief Architect, is responsible for platform and tool technology at Nexaweb Technologies Inc, a provider of the Nexaweb Platform enabling enterprise class rich Internet applications (RIAs). Bob is also a committer on the Apache XAP project(http://incubator.apache.org/xap) which provides an extensible framework for declaratively creating AJAX applications. Before Nexaweb, Bob was with Trakus, a technology company focused on tracking sports in real time. Bob possesses deep experience and background with Java and Windows technologies and is a leading expert in User Interface design. Bob is a regular speaker at industry events such as JavaOne, Server-Side Symposium and EclipseCon and has published multiple articles on tool and application development.

Rails Reduces the Number of Decisions When Building Web Applications
Speaker: Alex Bunardzic

Alex Bunardzic is a seasoned software developer, with 16 years of full-time experience building comprehensive software solutions. He specializes in delivering high-quality software products that are focused on helping users and businesses achieve their goals. To achieve and maintain high standards of delivery, Alex advocates the Less Technology/Less Infrastructure approach. This is why he embraces Ruby on Rails and AJAX, as these technologies allow him to deliver dramatically improved products in only a fraction of the time usually needed to supply such solutions.

AJAX: Making the Enterprise Jump
Speaker: Kurt Cagle

Kurt Cagle is a developer and author, with nearly 20 books to his name and several dozen articles. He writes about Web technologies, open source, Java, and .NET programming issues. He has also worked with Microsoft and others to develop white papers on these technologies. He is the owner of Cagle Communications and a co-author of Real-World AJAX: Secrets of the Masters (SYS-CON books, 2006).

Power Panel: Does AJAX Push the Browser Too Far?
Speaker: Jeremy Chone

Jeremy Chone is director, product management & strategy, Enterprise & Developer Solutions Business Unit, at Adobe. He focuses on the Flex and ColdFusion product families and is responsible for defining product strategy and a roadmap to help Adobe establish itself as the leading Rapid and Rich Internet Application platform provider.

He joined Adobe in 2006. Prior to that he was with Oracle, for six years, serving most recently as director of the Oracle Open Client engineering team, focused on working with the open community to develop the next generation of cross-platform collaborative client applications. Jeremy speaks at user conferences and industry events on topics such as rich Internet applications, collaboration, mobile computing, and software architecture.

High-Definition User Experience: The Convergence of Documents, Applications, and Media
Speaker: Christophe Coenraets

Christophe Coenraets currently works as a senior technical evangelist at Adobe. Before joining Adobe, Christophe was an evangelist at Macromedia, focusing on Rich Internet Applications and Enterprise integration. Prior to Macromedia, he was the head of Java and J2EE Technical Evangelism at Sybase, where he started working on Java Enterprise projects in 1996. Before joining Sybase in the U.S., Christophe held different positions at Powersoft in Belgium, including principal consultant for PowerBuilder, and manager of the Professional Services organization. Before joining Powersoft, Christophe worked as a developer and architect on several retail and BPM projects. He has been a regular speaker at conferences worldwide for the last 10 years.

PLEASE NOTE: SCHEDULE AND SPEAKER LINE-UP SUBJECT TO CHANGE WITHOUT NOTICE

Professional JavaScript / JSON: The Data Format of the Stars

Speaker: Douglas Crockford

Douglas Crockford, creator of the JSON data interchange format, is a developer who currently works for Yahoo!. He is known for his work in video game design, including the porting of Maniac Mansion. He maintains a Website called Crockford's Wrrrld Wide Web devoted to language, technology, programming, and games. He's also the author of JSLint, the JavaScript Verifier.

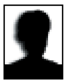
Today's Platform Just Won't Cut It - The Need for the Rich Enterprise Application Platform (REAP) for AJAX/SOA

Speaker: John Crupi

John Crupi is the CTO of JackBe Corporation. As CTO he is entrusted with understanding market forces and business drivers to drive JackBe's technical vision and strategy. John has 20 years of experience in OO and enterprise distributed computing. Previously, John spent eight years with Sun Microsystems, serving as a Distinguished Engineer and CTO for Sun's Enterprise Web Services Practice. Mr. Crupi is co-author of the highly popular Core J2EE Patterns book, has written many articles for various magazines and is a well-known speaker around the globe. He is a frequent blogger and was selected to join the International Advisory board for Ajax Developers Journal. John was also named as a member of the Software Development Magazine's Dream Team.

Test Driven Design

Speaker: Ryan Davis



Ryan Davis has been using Ruby since 2000 and is a founding member of the Seattle Ruby Brigade. He has worked on and released coco/ruby, ParseTree, ruby2c, RubyInline, ZenHacks, ZenTest, and ZenWeb. He has not yet released BFTS, metaruby, or zero2rails but they are coming out RSN.

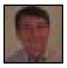
AJAX & JavaServer Faces

Speaker: John Fallows

John Fallows, former lead developer for Oracle ADF Faces Rich Client, has been working in distributed systems for over a decade. After five years spent focused on designing and developing the JavaServer Faces standard to provide AJAX functionality, playing a leading role in the Oracle ADF Faces team, he recently joined an AJAX start-up. Originally from Northern Ireland, John graduated from Cambridge University in the United Kingdom and has worked in the software industry for more than 10 years. Prior to joining Oracle, he worked as a research scientist for British Telecommunications Plc. His new book "Pro JSF and Ajax: Building Rich Internet Components" was released by Apress in February, 2006.

How OpenAJAX Alliance Will Accelerate Customer Success with AJAX

Speaker: Jon Ferraiolo



Jon Ferraiolo is a Web architect within IBM's Emerging Technologies group where he plays a leadership role in the OpenAjax initiative. Before joining IBM, he worked at Adobe for 13 years where he was an architect, engineering manager and product manager on multiple products and participated in various standards activities.

Mobile Browsers and AJAX: What's Next?

Speaker: Martin Frid-Nielsen

Martin Frid-Nielsen is CEO and Founder of SoonR, Inc., and an inventor on four patents for data synchronization and Web authoring technologies. Formerly, he was at Merant (acquired by Serena), and prior to that he was vice president of R&D at NetObjects. Before that he was at Borland as a member of the team that created Sidekick.

AJAX and Beyond with Open Source ThinWire Framework

Speaker: Joshua Gertzen

Joshua Gertzen is a professional computer programmer with over 10 years of experience in software development and architecture. His knowledge and experience range from utilizing differing programming languages, development frameworks, and architectural patterns, to troubleshooting complex architectural issues. Over the last six years, he has played a key role in building the technology infrastructure at Custom Credit Systems (http://www.customcreditsystems.com), a Dallas-based software company. His primary focus over that time has been building Web architectures that utilize DHTML and/or AJAX programming techniques. Additionally, he has been the primary architect behind building, maintaining, and enhancing the ThinWire AJAX Framework, which was recently open sourced. Joshua regularly posts updates to his blog (http://www.truecode.org).

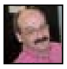
Client-centric Web Programming with ComponentArt Web.UI

Speaker: Milos Glisic

Milos Glisic is a senior developer with ComponentArt Inc. (www.componentart.com), the creators of AJAX-enabled controls with the richest client-side capabilities in the industry. Milos has been with the company since its inception and has been instrumental in developing and refining the unique patent-pending technologies found in ComponentArt's Web.UI component suite. He has many years of experience working with Web-based user interface technologies and holds a Hon.B.Sc. in Computer Science from the University of Toronto.

Eclipse AJAX Tooling Framework

Speaker: Robert Goodman



Robert (Bob) Goodman is a Software Engineer in Emerging Technologies division of the IBM software Group. He started his career at IBM working on Object Oriented Programming and IBM WebSphere. More recently he has work on Web Services for pervasive devices and tooling for PHP. He is now project leader of the AJAX Tools Framework (ATF) on Eclipse.

Real-time Location Tracking Mashup for the Enterprise

Speaker: Rama Gurram

Rama Gurram is a Research Scientist in the SAP Research Center at SAP Labs, Palo Alto, CA. In his current role at SAP, he is involved in the research and architecture of digital communities, creating voice interfaces to SAP enterprise applications, and RFID-related research projects that provide an integration layer between RFID data and SAP enterprise applications. Rama's primary areas of expertise include the architecture and design of Internet and intranet applications, service-oriented architectures (SOA), Enterprise Application Integration (EAI), and Internationalization and Human Computer Interaction (HCI). He has been involved in the architecture, design and development of enterprise scale applications since 1995. He has a strong background in software development methodologies and the entire product development life cycle. Prior to joining SAP, Rama worked as a software architect in project lead roles at Hewlett-Packard (HP). At HP, he led the system design and development of the enterprise server product called Watson, responsible for architecting solutions using HP's Web services platform called e-speak. Rama's current interests include Web services, SOA, Messaging, J2EE, .NET and to experiment with open source application development frameworks. Rama received his Masters degree in Computer Science from Birla Institute of Technology & Science (BITS - Pilani), India.

From Enriched HTML to Client/SOA: The Four Quantum States of AJAX

Speaker: Kevin Hakman

Kevin Hakman is co-founder of TIBCO Software Inc.'s General Interface product. His vision for software applications that run in a standard Web browser led him to co-found General Interface Corp. in 2001. TIBCO acquired General Interface Corp. in 2004. In the early 1990s he created software for print-on-demand catalogs and then, based on the success of that software, started the online direct marketing company eMergingMedia in 1995.

Real-time Location Tracking Mashup for the Enterprise

Speaker: Louenas Hamdi

Louenas Hamdi joined the SAP Research Team in Montreal in January 2004. He is currently involved in different projects within the Real-World Concentration Area and the mobility initiatives. He works actively and supervises a number of interns on the following projects: Digital Communities (Automatic Vehicle Location), Occasionally Connected applications, Nokia collaborations, advanced mobility scenarios for A1S, OSGi evaluation, and related European projects. He has five years of work experience in research and development in the field of software engineering in Algeria and Canada (CRIM & SAP). Louenas received his master's degree in software engineering from ETS (Ecole de Technologie Supérieure), Montreal, Canada. He also holds an engineering diploma in computer science from Université de Tizi-Ouzou, Algeria.

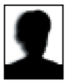
Session: TBA

Speaker: Thomas Hammell

Thomas Hammell is a product manager with Infragistics and manages the team responsible for developing JavaServer Faces and Swing components. Tom has over 20 years of experience developing software. He has published a book, "Test-Driven Development: A J2EE Example," as well as numerous articles on Java topics ranging from Swing development to unit testing. Tom also lectures frequently on Java topics and holds a bachelor's degree in electrical engineering and a master's degree in computer science from Stevens Institute of Technology.

Paid Content in a Web 2.0 World

Speaker: Dave Hanley



Dave Hanley is director of product management for Rhapsody Web Services, where he focuses on helping partners, developers, and publishers make innovative music products. He has worked in the digital media industry for three years, and most recently oversaw the creation and launch of Rhapsody.com. Dave is a Fulbright scholar and holds bachelors and masters degrees in pubic policy from Brigham Young University, and an MBA from Stanford.

Leveraging the Web 2.0 Movement

Speaker: Dion Hinchcliffe

Dion Hinchcliffe is co-founder and Tier-I Architect for the premier enterprise architecture firm Sphere of Influence Inc. (www.sphereofinfluence.com/), founded in 2001 in McLean Virginia. A veteran of software development, Dion works with leading-edge technologies to accelerate project schedules and raise the bar for software quality. He is highly experienced with enterprise technologies and he designs, consults, thinks, and writes prolifically. His latest work is with enterprise-scale service-oriented architecture, where he blends advanced standards-based engineering with Agile Development Processes. Dion actively consults with enterprise IT clients in the federal government and Fortune 500. Dion also speaks and publishes about Web 2.0 and SOA on a frequent basis.

Scaling 43 Things

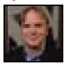
Speaker: Eric Hodel

Eric Hodel has been using Ruby for almost five years. Currently he works as a programmer and sysadmin for The Robot Co-op, which built the Web site 43 Things. In his time with Ruby, he has written numerous packages, including ParseTree with Ryan Davis, which is used in the Rails Template Optimizer, performance analysis tools for Rails that included a log analyzer and profiler. In his spare time, Eric commits documentation patches to Ruby and tinkers with RDoc and ri.

SPEAKERS

AJAX (in) Security

Speaker: Billy Hoffman



Billy Hoffman is a lead security researcher for SPI Dynamics (www.spidynamics.com). At SPI Dynamics, Billy focuses on automated discovery of Web application vulnerabilities and crawling technologies. He has been a guest speaker at Black Hat Federal, Toorcon, Shmoocon, O'Reilly's Emerging Technology Conference, The 5th Hope, and several other conferences. His work has been featured in Wired, Make magazine, Slashdot, G4TechTV, and in various other journals and Web sites. Topics have included reverse engineering law and techniques, ATMs, XM Radio and magstripe projects. In addition, Billy is a reviewer of white papers for the Web Application Security Consortium (WASC), and is a creator of Stripe Snoop, a suite of research tools that captures, modifies, validates, generates, analyzes, and shares data from magstripes. He also spends his time contributing to OSS projects and writes articles under the handle Acidus.

Acceptance Testing Rails Apps with Watir

Speaker: Dave Hoover

Dave Hoover is the lead agile practices consultant at Obtiva Corp and author of the upcoming "Acceptance Testing Rails" shortcut from Addison Wesley. He enjoys learning about and contributing to the craft of software development.

Architecting Great Internet Applications with AJAX

Speaker: Scott Isaacs

Scott Isaacs is an architect on Microsoft Windows Live, responsible for the Web-client architecture and frameworks used across MSN/Windows Live properties and Windows Live DHTML Gadgets. He's an Internet veteran known widely as "The Father of DHTML" since during the mid- to late-90s. He authored the original DHTML specification and represented Microsoft in the W3C, helping define and drive many Internet standards.

AJAX & JavaServer Faces

Speaker: Jonas Jacobi

Jonas Jacobi is a J2EE technology evangelist at Oracle. A native of Sweden, he has worked in the software industry for more than 15 years. Prior to joining Oracle, Jonas worked at several major Swedish software companies in management, consulting, development, and project management roles. For the past three years, he has been responsible for the product management of JavaServer Faces, Oracle ADF Faces, and Oracle ADF Faces Rich Client in the Oracle JDeveloper team. His new book "Pro JSF and Ajax: Building Rich Internet Components" was released by Apress In February, 2006.

Exposing and Preventing Errors in AJAX Applications

Speaker: Nathan Jakubiak

Nathan Jakubiak is a software engineer at Parasoft. He currently manages the development of Parasoft WebKing (an automated error prevention and detection suite for Web applications) and is the lead developer of Parasoft's AJAX testing solution. He has extensive experience developing testing tools for Web-based technologies, and has led the development of customized Web testing solutions for enterprises such as IBM. His Web-related R & D experience has resulted in a number of pending patents in the area of automated regression testing. Jakubiak holds a BS in mathematics from Harvey Mudd College.

Mobile AJAX

Speaker: Ajit Jaokar

Ajit Jaokar is the author of the book 'Mobile Web 2.0' (http://www.mobileweb20.futuretext.com/) and is also a member of the web2.0 workgroup (http://www.web20workgroup.com/). Currently, he plays an advisory role to a number of mobile start-ups in the UK and Scandinavia. He also works with the government and trade missions of a number of countries including South Korea and Ireland. He is a regular speaker at technology events including Real-World AJAX (June 2006 - New York www.ajaxseminar.com) and AJAXWorld Conference & Expo (Oct 2006 Santa Clara http://www.ajaxworldexpo.com). He was also featured recently on CNN money (http://money.cnn.com/2006/07/13/technology/web_2.0/index.htm). Ajit chairs Oxford University's Next generation mobile applications panel (www.forumoxford.com) and is the founder and CEO of a publishing company – Futuretext (www.futuretext.com) Ajit lives in London, UK, but has three nationalities (British, Indian and New Zealander) and is proud of all three. He is a member of the RSA(http://www.thersa.org/) – The Royal Society for the encouragement of Arts, Manufactures and Commerce.

SAP's Use of Flex to Change the Face of Enterprise Applications

Speaker: Manish Jiandani

Manish Jiandani is a senior solution manager at SAP Labs with the xApp Analytics team. He focuses on building analytical applications for Discrete Industries. Before joining SAP, Manish was a product manager with the Siebel Analytics Platform team.

AJAX Debugging and Quality Assurance

Speaker: Dave Johnson

Dave Johnson is the co-founder, CTO, and self-proclaimed XML evangelist of Nitobi Software, a Vancouver-based AJAX component vendor. He focuses his energy on architecting and building high-performance AJAX components for Web-based applications.

SAP's Use of Flex to Change the Face of Enterprise Applications

Speaker: Boris Kabisher

Boris Kabisher is an engineer in SAP Visual Composer development team, where he works on the User Interface Modeling framework. Prior to joining the Visual Composer team, Boris was a founder and chief architect of number of technology startups, focusing on state-of-the-art GUI and user experience.

Y! vs Y! vs Y!: Across the Document-Application Spectrum, One Size Does Not Fit All

Speaker: Nate Koechley

One of the first Web developers at Yahoo!, Nate Koechley has been instrumental in creating and defining the practice of Web development and front-end engineering. Through evolving roles as developer, manager, and evangelist on both the development and user experience and design sides of the company, Nate has championed modern standards-based Web development, a commitment to accessibility, code and pattern library creation, and open source and blogging initiatives. Through it all, Nate focuses on the intersection and coordination of design and development, helping teams understand "why" in addition to "how".

Rich AJAX Platform (RAP) - Web 2.0 the Eclipse Way

Speaker: Jochen Krause

Jochen Krause is CEO of Innoopract Information Systems and a member of the Eclipse Board of Directors. At Innopract, he is responsible for the Eclipse Strategy and the Yoxos Eclipse Distribution solution. He is a member of the Web Tools Project Management Committee and the lead of the RAP project.

Make AJAX Work for Your Site and Your Users: A Case Study on Improving Product Selection

Speaker: Riccardo La Rosa

Riccardo La Rosa joined Molecular in October 1998. As a principal consultant, technical architect Riccardo applies his 10+ years of technology experience in business analysis, requirements gathering, and solutions architecture for companies including Fleet/Bank of America, Fidelity, Staples, McGraw-Hill, and Analog Devices. In his current role, Riccardo is responsible for designing solutions on Microsoft .NET technologies that meet his clients' business goals. Over the years, Riccardo has developed a deep knowledge in the financial, retail, and manufacturing industries. As a Microsoft Certified Software Developer, Riccardo also leads the Microsoft practice group at Molecular where he fosters continuous learning and a passion for new technologies. Currently he is working to grow Molecular expertise in service-oriented architectures (SOA) and AJAX applications. Riccardo leads presentations and seminars on all aspects of SOA, from planning to implementation, often speaking at conferences, most recently at the 2005 Microsoft Code Camp. Before joining Molecular, Riccardo worked as a consultant for companies including Massachusetts General Hospital and Malden Mills. He holds a master of science degree in electronic engineering from Università di Padova, Italy. During that time he wrote his thesis on "High Power Factor Electronic Ballasts" at Brown University.

Using AJAX and Web Services with Widgets

Speaker: Alan Lewis

Alan Lewis is a technical evangelist with the eBay Developers Program. He works with third parties who want to use the eBay Web Services platform to develop applications that help users sell, search, and buy on eBay. Alan has a BA in philosophy from UCSB. He blogs at alanlewis.typepad.com.

Why Browsers (Still) Matter

Speaker: Hakon Wium Lie

Hakon Wium Lie is CTO of Opera Software, where he has worked since 1999. Known as "The Father of CSS," he proposed the concept of Cascading Style Sheets in 1994. He has worked for, among others, the W3C, INRIA, CERN, the MIT Media Lab and Norwegian telecom research in Televerket.

AJAX-Enabling Applications in Two Minutes: A Practical Demonstration

Speaker: Sahil Malik

Sahil Malik is a telerik evangelist, Microsoft MVP [C#], INETA Speaker, and author of a bestselling ADO.NET 2.0 book. He is also an ADO.NET trainer for Keystonelearning.com and a consultant, trainer and mentor in various Microsoft technologies.

AJAX Push - Transform the User Experience

Speaker: Stephen Maryka

Stephen Maryka, chief technical officer at ICEsoft Technologies Inc., leads the development of all AJAX-based technologies at ICEsoft. He is responsible for ICEfaces product development, and all AJAX-related R&D. Prior to joining ICEsoft in 2004, Stephen was co-founder of AudeSi Technologies where he served as VP of Technology and led Java product development for Internet appliances. After Wind River's acquisition of AudeSi, Stephen served as a principal technologist for Wind River, working on embedded Java, device management, and high-availability technologies. Stephen has been involved with Java technologies since 1997, when he engaged with Nortel Networks as the chief software architect for the world's first commercial embedded Java telephone. Stephen earned his BSc in mathematics and computer science from the University of Victoria in 1984.

PLEASE NOTE: SCHEDULE AND SPEAKER LINE-UP SUBJECT TO CHANGE WITHOUT NOTICE

AjaxWorld Conference and Expo

SPEAKERS

Migrating Legacy Applications to Use AJAX

Speaker: Dror Matalon

Dror Matalon, a 20-year technology industry veteran, founded Zapatec to create cost-effective, Web-based applications for all businesses. He brings a wide range of software, application development, product cycle, and Internet expertise to the Zapatec team. Prior to founding Zapatec, Dror founded DNAI in 1994, creating a customer-focused ISP based on strong team unity and technical knowledge. In August 1999, DNAI was acquired by RCN, the nation's first and largest U.S. provider of bundled phone, cable television, and high-speed Internet services. Before his DNAI days, Dror served as the director of software development at Real Time Solutions, where he managed the product life cycle of the "Pick to Light" warehouse automation system. There, his responsibilities included product design, management of the development team, development, implementation and after-sales customer support. Dror began his career in development at Informix Software, Inc., as a member of the tool development team responsible for creating Informix Isql and Informix-4gl.

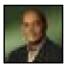
Encapsulating AJAX Functionality in JavaServer Faces Components

Speaker: Craig R. McClanahan

Craig R. McClanahan is a programmer and original author of the Apache Struts framework for building Web applications. He was part of the expert group that defined the servlet 2.2, 2.3 and JSP 1.1, 1.2 specifications. He is also the architect of Tomcat's servlet container Catalina.

Session: TBA

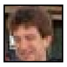
Speaker: David McFarlane



David McFarlane's career spans more than 20 years of experience in operations, global sales and marketing management, business development, and information technology. As the chief operating officer (COO) at IM Logic, Inc, he successfully led the global expansion, driving triple-digit revenue growth and growing the customer roster to more than 400 and worldwide users to 400,000. Prior to IM Logic, Inc, David served as president and COO at Exchange Applications, Inc. where he drove revenues from $6m to $80m in four years, resulting in a successful IPO and peak valuation of $1.6 billion. In addition, David was vice president International and Alliances at MRO Software, Inc. Before MRO Software, Inc, David was an Executive Manager with Plessey Avionics Ltd. of England. David graduated with honors from Bath University in England, and also holds a master of engineering degree from the University.

AJAX in Moderation: Getting Excited About the Web's New Middle Ground

Speaker: Eric Miraglia



Eric Miraglia, one of the world's leading experts on "advanced JavaScript utilities and widgets," works for Yahoo!'s Presentation Platform Team. He plays a critical role in helping product teams realize their forward-reaching development goals. He also teaches regular classes for Yahoo! Web developers. Eric has been involved in the creation of social Web applications since 1995, when he began developing interactive writing spaces for universities; his Speakeasy Studio & Café was used by more than 100 universities between 1997 and 2004. Since 2003, he has been a part of Yahoo!'s Web development community. When he's not trying to convince pixels to do what they're told, he can sometimes be found at Stanford University, where he teaches writing as a visiting lecturer. He holds a PhD in the strange hybrid discipline of Technology and Rhetoric.

JJAX: Asynchronous Cross-Domain Scripting

Speaker: Zeki Mokhtarzada

Zeki Mokhtarzada co-founded Freewebs.com in 2001, and he now serves as CTO and sits on the board of directors. As CTO and head of Freewebs' engineering team, he is the architect behind the scalable Freewebs back end and server architecture supporting tens of millions of visitors. Prior to co-founding Freewebs, he was employee number one at WebOS, where he was instrumental in some of the leading-edge innovations in the on-demand applications space. Zeki double-majored in mathematics and computer science at the University of Maryland. In addition to his work in the technology field, he is an experienced speaker and author on topics concerning neuroscience.

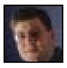
Foundations of Atlas: AJAX Programming in ASP.NET

Speaker: Laurence Moroney

Laurence Moroney is the director for technology evangelism for Mainsoft, the cross-platform company. He is the author of several books on ASP.NET, security and J2EE interoperability.

Make AJAX Work for Your Site and Your Users: A Case Study on Improving Product Selection

Speaker: Steve Mulder



Steve Mulder is a principal consultant in user experience at Molecular, where he specializes in customer research, personas, information architecture, and usability. Prior to joining Molecular, he was manager of user experience at Terra Lycos, where he led the information architecture and usability efforts, including the global rollout of user-centered design processes. He has worked with clients such as Morgan Stanley, PC Connection, Talbots, VistaPrint, 3M, Estee Lauder Companies, and Gloss.com. Steve is currently writing a book on personas, and he speaks regularly at conferences on topics such as user behavior patterns, customer research, personas, search interfaces, and Web site navigation.

AJAX with Java

Speaker: Greg Murray

Greg Murray is the Servlet 2.5 specification lead and now the AJAX architect for Sun Microsystems. He has been involved with the Java Enterprise Edition Web tier since the creation of the platform. Greg started a grass roots effort at Sun promoting the use of AJAX with Java technologies. He contributed to the upcoming Java EE AJAX/Web 2.0 reference application and is the creator of a new technology to enable JavaScript and Java to work together (both of which will be formally announced soon). Greg is also working the Java BluePrints and other teams at Sun on developing the best practices for enterprise AJAX and looking for ways to better integrate the Java and AJAX worlds. Before taking on his current responsibilities, he was a member of the Java BluePrints team for which he was responsible for the recommendations and guidelines for the Java Enterprise Edition Web technologies. He contributed to the design and implementation of the original Java Pet Store Demo and Java Adventure Builder reference applications. Prior to working with the BluePrints team, Greg worked on internationalization tools for the globalization group at Sun.

Flex: Delivering ROI to Enterprise Apps

Speaker: Mike Nimer

Mike Nimer has recently joined forces with fellow guru Jeff Tapper to form Tapper, Nimer and Associates to provide expert guidance on Rich Internet Application (RIA) development and mentoring. Mike is a formerly a member of the ColdFusion engineering team, responsible for the Flex2 integration with ColdFusion and a number of other features in ColdFusion, such as Flash Forms. Before joining the engineering team he spent three years working as a senior consultant with the Macromedia consulting group, where he provided on-site assistance to customers around the world with their architecture planning, code reviews, performance tuning, and general code issues.

Session: TBA

Speaker: Shanku Niyogi

Shanku Niyogi is product unit manager of the UiFX Team, Microsoft. He is responsible for ASP.NET, WinForms, Atlas, and many of Microsoft's future platform investments.

Integrating Rails into the Enterprise Through SOA

Speaker: Joe O'Brien

Joe O'Brien ia a co-founder of EdgeCase, LLC (theedgecase.com), a software development company based in central Ohio specializing in Ruby, Rails, and Web 2.0 application development and training. Previously he was a developer with ThoughtWorks and spent much of his time working with large J2EE and .NET systems for Fortune 500 companies. He has spent his career as a developer, project manager, and everything in between. Joe is a passionate member of the open source community. Recently, he founded the Columbus Ruby Brigade and has helped organize the Chicago Area Ruby Users Group. His passions are agile development in the enterprise, Ruby, and demonstrating to the Fortune 500 the elegance and power of this incredible language. Joe is currently working on a book for the Pragmatic Programmers on Ruby and SOA.

Case Study: The JBoss Flex-based Mail Client

Speaker: Andrew C. Oliver

Andrew C. Oliver is a professional cat herder who moonlights as a software developer. He's been developing in Java since 1998, primarily as a consultant for large companies with the inevitable dot-com thrown in between. He is a former member of the Apache Software Foundation, former member of the Apache Jakarta PMC, and founder of the Jakarta POI project. When he's not off globetrotting to provide training, consulting and support to JBoss, Inc. customers, you can find him bit-twiddling with some obscure file format or protocol, or JBoss Collaboration Server.

Apache Derby - A Local AJAX Data Store

Speaker: Francois Orsini

Francois Orsini, a senior staff engineer working in the Database Technology group at Sun Microsystems, has 18 years' experience in databases and infrastructure development. His expertise is in distributed data management systems, security, resource management, HA cluster solutions, and connectivity services. Francois spent eight years at Sybase as a senior engineer working on the SQL Server core engine. He then worked at Cloudscape, Inc., as a technical lead, where he designed and implemented connectivity, security, and middleware services for the Cloudscape Java database. Francois holds a Bachelor's Degree in Civil Engineering and Computer Science from the Paris Institute of Technology.

Enabling Inter-Mashup Interactions Using Composite Data Services and Workflow

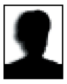
Speaker: Ash Parikh

Ash Parikh is a director of development and technology of Raining Data's XML-Centric Applications and Platforms group. He has over 15 years of software vision, architecture, and development experience and combines the ability to drive product vision and architecture to reality by coupling focused evangelism with real customer validation and market analysis.

PLEASE NOTE: SCHEDULE AND SPEAKER LINE-UP SUBJECT TO CHANGE WITHOUT NOTICE

SPEAKERS

AJAX3D: The Open Platform for Rich 3D Web Applications
Speaker: Tony Parisi


Tony Parisi, president and CEO of Media Machines, is a technology pioneer and accomplished entrepreneur. He is co-creator of the Virtual Reality Modeling Language (VRML), the ISO standard for 3D graphics on the World Wide Web, and is widely recognized as an expert in standards, technologies, and emerging markets for interactive rich media. In 1995 Tony founded Intervista Software, an early innovator in real-time, networked 3D graphics technology and developed WorldView, the first real-time VRML viewer for Microsoft Windows. In 1998 Intervista was purchased by PLATINUM technology, inc., and Tony joined the company to lead business affairs for its 3D visualization group. Tony founded Media Machines in 2001 and is spearheading the development of FLUX, a real-time 3D technology that continues to push the envelope in interactive graphics for the Web. He is also a lead editor and co-chair of the Extensible 3D (X3D) Specification, the new standard for Web3D graphics being developed by the Web3D Consortium.

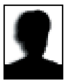
Storyboarding AJAX Applications
Speaker: Eric Pascarello

Eric Pascarello is coauthor of AJAX in Action.

AJAX: Not the Only Game in Town
Speaker: Jim Phelan

Jim Phelan is vice president of development and chief architect for Stream57, a New York City-based firm specializing in communication solution development for the enterprise. His expertise in creating solutions for consolidation and collateralization of business communications has allowed his team to create applications for the management and delivery of live and on-demand rich media content. Jim is a strong proponent of the Adobe Flash Platform and is a member of the editorial board of Web Developer's & Designer's Journal.

Building Dynamic Flex Applications for the Enterprise
Speaker: Mark Piller


Mark Piller is the founder and CEO of Midnight Coders (www.themidnightcoders.com), a software development company specializing in client/server connectivity solutions for Rich Client Applications. With more than a decade of experience in developing distributed computing applications, Mark has applied his deep technical acumen to produce the state of the art presentation server platform known as WebORB. Prior to Midnight Coders, Mark worked at webMethods and was responsible for architecting the company's SOA solution. Prior experience includes leading the design and implementation of multiple features of Glue, a web-services platform from The Mind Electric that was acquired by webMethods, as well as senior technical positions with ObjectSpace, MCI and SABRE.

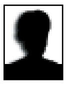
Taking RIAs to the Desktop with Apollo, Adobe's Next-Generation Client
Speaker: Luis Polanco

Luis Polanco is the senior product manager for Apollo at Adobe. Luis has managed a variety of next-generation client and server software, including desktop applications and enterprise software products. Prior to the Apollo team, Luis worked on Adobe's Rich Client Initiative. Most recently, Luis has led product development for an Operation Intelligence product suite targeted to Fortune 100 companies, which leveraged Flash and Java technologies, and a Business Process Management platform for large enterprises.

Toward a Service-Oriented Application Stack
Speaker: Matt Quinn

Matt Quinn heads product strategy and product management for TIBCO Software Inc., a leading business integration and process management software company. TIBCO provides the popular TIBCO General Interface AJAX Rich Internet Application toolkit.

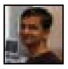
Creating Great Mapping Mashups Using Flex
Speaker: Mansour Raad


Mansour Raad has over 20 years of experience in the IT field. As the senior software architect of ArcWeb Services at Environmental System Research Institute (ESRI), he is using his command of Internet technologies to design the next-generation Internet solutions. ESRI is a leader in Geographic Information Systems (GIS). Currently, Mansour is the Flex evangelist within ESRI and is leading the Flex-based ArcWeb Explorer solution. He has been using Flex since its Royal days. In addition, while at ESRI, he was a team lead in architecting and implementing ArcIMS, the premier mapping solution on the internet. Mansour assisted several large companies in implementing and integrate mapping solution in their enterprise. He graduated from Boston University with a masters degree in Aerospace Engineering. With the combination of his IT experience, he designed and implemented an airport noise and operation monitoring system (ANOMS) that is currently used in over 30 airports world wide.

Mashing Up Your AJAX Application
Speaker: Paul Rademacher

Paul Rademacher is the creator of HousingMaps.com, which combined craigslist and Google Maps for the first Web mashup. He holds a PhD in computer science from UNC-Chapel Hill, and worked as an R&D engineer at Dreamworks Animation on such movies as Shrek 2 and Madagascar. Since creating HousingMaps, Paul is now at Google.

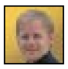
Real-time Location Tracking Mashup for the Enterprise
Speaker: Samir Raiyani


Samir Raiyani is the director of Homeland Security Research at SAP Research in Palo Alto. He heads the Digital Communities project, which aims to deliver new services for cities, citizens and businesses that deploy high-speed wireless (WiFi/WiMax) data networks. Samir's areas of expertise are mobile middleware, applications and user interfaces. At SAP Research, Samir has supervised a number of research projects. He led research in multimodality for mobile devices, which involves using different types of input modalities (speech, keyboard, RFID scanners) and output modalities (speech, visual) for mobile devices. He has collaborated with researchers at some of world's leading universities and corporate research labs, as well as with large and small enterprises in a variety of areas. His research in mobility, multimodality and RFID has resulted in product offerings from SAP in the supply chain area. He has also provided strategic input to various product and industry groups within SAP. He has authored a number of patent applications and has presented his results at various conferences around the world. Prior to SAP, he founded a mobile healthcare applications startup called MediSpark (iScribe). He has a Masters degree in Computer Science from Stanford University.

Enabling Inter-Mashup Interactions Using Composite Data Services
Speaker: Ajay Ramachandran

Ajay Ramachandran is the CTO and vice president of Raining Data's XML-Centric Applications and Platforms group. His expertise includes management consulting, information technology, biotechnology, product management, sales, and marketing. His career includes cofounding and executive-level positions at several Internet services and software companies. His has degrees in molecular cellular biology (genetic engineering) and organizational communications.

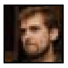
Session: TBA
Speaker: Juho Risku


Juho Risku is the founder of Helmi Technologies, Inc., a provider of an industry-leading platform for building AJAX-based Rich Internet Applications (RIAs). His experience with the RIA space and AJAX technologies dates back to 1997 when he began working extensively with cross-browser DHTML. Juho is passionate about great user experience and this drive has helped him to develop innovative technologies over the last 10 years to raise the Web experience to the next level. Juho is a serial entrepreneur, who started his first company at age 20. In addition to Helmi, he is also the Chairman, owner and former CEO of Visualway Design, Inc. (1996), an innovative web design agency with high profile clients in Europe. Additionally, Juho Risku has held several board memberships in other media and creative industry related organizations.

J2EE Security in Flex Applications
Speaker: Rob Rusher

Rob Rusher is a certified ColdFusion MX instructor and one of only a few certified Flex instructors in the world. He is the co-author on ColdFusion articles and books, including the "Advanced Web Application Construction Kit." Always entertaining and chock-full of knowledge, Rob is a frequent speaker on ColdFusion and Flex technologies. He has worked with dozens of Adobe's largest and most strategic customers since 1997. His dynamic personality and wealth of experience help to architect, build, tune, and troubleshoot some of the largest and most critical ColdFusion and Flex applications in the world.

AJAX, Design, and Mobile Devices
Speaker: Alex Russell


Alex Russell is project Lead for the Dojo Toolkit and director of R&D for SitePen, a consulting firm specializing in responsive Web applications and the open source tools that enable them. He also serves as president of the Dojo Foundation, an organization that supports development of several high-quality open source JavaScript projects and distributes them under liberal terms. Before joining SitePen, Alex was a senior engineer at JotSpot and Informatica where he helped both companies build highly interactive, Web interfaces. His earlier open source involvement included stints as editor of the OWASP Guide to Building Secure Web Applications and primary author of the netWindows DHTML toolkit.

Rails Meets the Legacy World
Speaker: Robby Russell

Robby Russell founded PLANET ARGON in 2002, and it has now grown to be one of the most well-known Rails development, consulting, and hosting firms in operation. He is also currently finishing his long anticipated book, "Programming Rails," for the technical book publisher, O'Reilly Media. Robby is well known in the Rails community, thanks to his popular Rails-related blog, robbyonrails.com. When not in the office or traveling on business, Robby can usually be found playing music and has a few music projects in the works.

Google Gadgets and Gadget Discovery
Speaker: Adam Sah

Adam Sah is the architect of Google Gadgets and the Gadget Content Directory. Prior to Google, he was a founding engineer at several startups, among them Inktomi and Sensage, where he is a member of the board. He holds several patents in databases and Web systems.

Comet and Vector Graphics with Dojo
Speaker: Dylan Schiemann

As CEO and co-founder of SitePen, and co-founder of the Dojo Toolkit, Dylan Schiemann is best known for building web applications that make use of JavaScript/Ajax, Dojo, Comet, and other standard web development technologies. Previously, he has developed web applications for Renkoo, Informatica, Security FrameWorks, and Vizional Technologies, to name a few.

Adobe® Flex™ 2

Go Beyond Ajax with Flex

AJAXWORLD Conference & Expo, Tuesday, October 3

8:15am – 9:05am	**High Definition User Experience: The Convergence of Documents, Applications, and Media** **Christophe Coenraets**, Adobe
7:15pm – 8:00pm	**Redefining RIAs: Does AJAX Push the Browser Too Far?** Power Panel – **Jeremy Chone**, Adobe **Scott Isaacs**, Microsoft **Bob Brewin**, Sun **Jesse James Garrett**, Adaptive Path **Coach Wei**, Nexaweb

Flex Track

9:50am – 10:35am	**Web 2.0 with Flex: Delivering ROI to Enterprise Apps** **Mike Nimer**, Tapper, Nimer and Associates
11:35am – 12:20pm	**Flex with Java** **James Ward**, Adobe
1:45pm – 2:30pm	**Taking RIAs to the Desktop with Apollo, Adobe's Next Generation Client** **Luis Polanco**, Adobe
2:30pm – 3:15pm	**Creating Great Mapping Mashups Using Flex** **Mansour Raad**, ESRI
4:05pm – 4:50pm	**Case Study: The JBoss Flex-based Mail Client** **Andrew Oliver**, Consultant
6:30pm – 7:15pm	**SAP's Use of Flex to Change the Face of Enterprise Applications** **Boris Kabisher and Manish Jiandani**, SAP

Visit Adobe at Booth 101

Adobe®

SPEAKERS

Designing for AJAX

Speaker: Bill Scott

Bill Scott is an AJAX evangelist and design pattern curator at Yahoo!, where he spreads the rich and sane AJAX design. Before Yahoo!, Bill led user experience at Sabre and co-founded Rico (an open source AJAX framework – openrico.org). For 20 years Bill has designed and created interfaces in a variety of areas (including video games). His musings can be found at http://looksgoodworkswell.com.

A Venture Capitalist View of Web 2.0 Value

Speaker: Neil Sequiera

Neil Sequiera invests in both new and existing technology businesses. Areas of special interest include Internet and new media; software; consumer services; and network infrastructure. He is currently a board member of Eons and ViTrue, a board observer for BrightCove and Qumas, and is actively involved with BlackDuck, ITA Software, and Maven. Neil joined General Catalyst Partners from Time Warner where he was most recently managing director, technology for Time Warner Investments, formerly AOL Time Warner Ventures, the early stage private investment vehicle for the world's largest media company. During his four years at Time Warner, Neil worked closely with various operating groups including AOL, HBO, Time Inc., Time Warner Cable, Turner and Warner Bros. to identify investment opportunities. Neil sourced, led and was actively involved with the boards of several of the companies within the Time Warner Investments portfolio including Arroyo Video Solutions, BigBand Networks, Entropic, Goldpocket Interactive (Tandberg), N2Broadband (Tandberg) and Waterfront Media. Prior to joining Time Warner, Neil led Mergers and Acquisitions for CMGI, a diversified technology operating company and venture capital firm. While at CMGI, Neil worked on a variety of acquisitions and investments in software, media and networking infrastructure and was actively involved with CMGI companies including Alta Vista (YHOO), Engage, NaviSite (NAVI), Ubid (CCSR), and Yesmail (IUSA). Previously, Neil was an Associate with Goldman, Sachs & Co. Prior to Goldman, Neil was a senior consultant with Accenture, where he managed project teams on technology focused strategic and process initiatives for Fortune 500 companies. In addition, Neil was an early employee at Goldenvoice Presents, a diversified media company, and was a contributor to the founding of OpenRatings, a venture-backed software start-up based in Boston. He has spoken at numerous media and venture capital conferences and is a judge for the Harvard, MIT, and Wharton Business Plan Competitions. Neil holds a BA in economics from the University of California, Santa Barbara, and an MBA from the Harvard Business School.

Java EE 5 BluePrints for AJAX-Enabled Web 2.0 Apps

Speaker: Inderjeet Singh

Inderjeet Singh is a senior staff engineer with Sun where he is the architect for the Java BluePrints program. He has been involved with the Java BluePrints since its inception. He is the primary author of the Addison-Wesley Java-series books, "Designing Web Services with the J2EE 1.4 Platform" and "Designing Enterprise Applications with the Java 2 Platform, Enterprise Edition" (now in its second edition). He is a regular speaker on enterprise application design. Prior to joining Java BluePrints, he designed and developed the proxy server in the JavaWebServer product. In the past, he designed fault-tolerance software for large-scale distributed telecommunications switching systems.

Enabling Inter-Mashup Interactions Using Composite Data Services

Speaker: Robert Smik

Robert Smik is a director of solutions engineering of Raining Data's XML-Centric Applications and Platforms group. He has been involved in application and software design and development for more than 15 years. His experience includes design and development of highly complex database systems; architecting multitier Web environments; and architecting and developing various connectivity solutions, products, and smart cards, in addition to SOA and data aggregation tools. RObert is an active member of HL7 and CDISC. He has also co-authored articles in XML-Journal and JavaWorld.

Using Google Web Toolkit

Speaker: Bret Taylor

Bret Taylor is the Senior Product Manager for Google's developer programs. He joined Google in early 2003, where he has been responsible for over 25 product launches, including Google Maps, the Google Maps API, and Google Local. Prior to Google, Bret worked as a software engineer at Reactivity, a startup incubator in Silicon Valley. Bret holds an MS and BS in Computer Science from Stanford University.

AJAX and the Web's Copernican Revolution

Speaker: David Temkin

A globally recognized pioneer of Rich Internet Applications, David Temkin is CTO and Founder of Laszlo Systems. In this role, he has positioned the company to become the next technology standard for Rich Internet Applications. Under his direction, Laszlo developed its patent-pending open source product suite and extended operations to both coasts of the United States. Before founding Laszlo, David was senior director of engineering at Excite@Home, where he led a team of 55 engineers, designers, and technical writers responsible for developing the company's consumer software. Prior to Excite@Home, he was an engineering manager in the Newton division at Apple Computer and developed enterprise software at EDS. He graduated from Brown University with a double major in computer science and history, and is named on four software patents.

Enabling User Specific Mashups

Speaker: Dan Theurer

Dan Theurer is a technical evangelist for the Yahoo! Developer Network, where he spreads news about Yahoo! Web services, promotes API adoption, and creates and supports developer communities. Before joining Yahoo!, he worked at eBay, evangelizing Web integration technologies. Prior to that, he worked as a software consultant for LBBW, one of the five largest banks in Germany, where he led database and Web services projects developed primarily in Java. He also collaborated with the mobile database application development team at IBM's Silicon Valley Lab. Dan has an MS in computer ccience from the University of Applied Sciences Esslingen in Germany.

You're Running a Database Where? -- A Persistence Architecture for Local AJAX Storage using Apache Derby

Speaker: David Van Couvering

David Van Couvering has spent his engineering career crossing the bridge between databases and the middle-tier world of application servers, Java, and distributed systems. He was the original architect for the Sybase J2EE application server and for the first release of the clustered Sun Java Application Server Enterprise Edition. Currently he is involved in database technology at Sun, and is a committer for Apache Derby and a member of the Apache DB PMC.

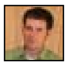
AJAX for Rich Internet Apps: Lessons from One Million AJAX E-mail Clients

Speaker: Florian von Kurnatowski

Florian von Kurnatowski has over a decade of experience implementing and deploying large-scale advanced messaging and collaboration systems for hundreds of organizations. He has deep experience with one of the most widely used AJAX applications in the world, Scalix Web Access, which combines the power of desktop e-mail and collaboration with the convenience of a browser. Florian is Scalix's participant in the OpenAJAX initiative and is an experienced industry thought leader. He is a regular speaker at Linux events and LinuxWorld, especially in Europe.

Flex with Java

Speaker: James Ward



James Ward is a Technical Evangelist for Flex at Adobe. Much like his love for climbing mountains he enjoys programming because it provides endless new discoveries, elegant workarounds, summits and valleys. His adventures in climbing have taken him many places. Likewise, technology has brought him many adventures, including: Pascal and Assembly back in the early 90's; Perl, HTML, and JavaScript in the mid 90's; then Java and many of it's frameworks beginning in the late 90's. Today he primarily uses Flex to build beautiful front ends for Java based back ends. Prior to Adobe, James built a rich marketing and customer service portal for Pillar Data Systems.

Web 2.0: The State of Confusion?

Speaker: Coach Wei

Coach combines in-depth IT industry expertise with extensive education and research experience at MIT to drive technology innovation and business direction for Nexaweb. He founded Nexaweb in 2000 and served as CEO until summer 2003. Before founding Nexaweb, Coach architected and designed enterprise software for managing storage networks at EMC Corporation. As a graduate researcher at MIT, Coach developed software and hardware systems for non-destructive evaluation as well as signal/image processing algorithms. Coach was a finalist in the 1999 MIT $50K entrepreneurship competition and holds several U.S. patents. An accomplished writer and speaker, Coach has published numerous articles on topics including: AJAX, J2EE and .NET, RIA development, XML, signal/image processing, composite materials and ultrasonic imaging. He is a frequent speaker at top industry events, such as JavaOne and Web Services Edge. Coach holds an MS in information technology from MIT and maintains a blog at http://www.coachwei.com.

Developing RIAs from Front-to-Back: Turning the Process Upside Down

Speaker: Dave Wolf

Dave Wolf is vice president of consulting at Cynergy Systems, Inc. Prior to joining Cynergy, he has held senior management positions at several major software firms including Sybase and Microsoft, where he was responsible for the development, marketing, and sales of several enterprise-class software products. A sought-after public speaker, he has presented at major technology conferences such as JavaOne and Microsoft TechEd. At Cynergy, he oversees consulting operations worldwide and spends most of his time interacting directly with the field and working in every time zone but his own.

Delivering Data to Your AJAX Solutions

Speaker: Bob Zurek

Bob Zurek is director of Advanced Technologies and Product Strategy in IBM's Information Integration Solutions group and is an internationally known expert in development tools, integration and middleware technologies. Prior to joining IBM, he was vice president of product management and advanced technologies at Ascential Software and was instrumental in developing and driving Ascential's enterprise data integration software strategy along with mergers and acquisitions. Before joining Ascential, Bob was a senior technology research and strategy advisory with Forrester Research, and was founder of three technology startups - Linc Systems, Infinity Systems, and Lumapath - and has held senior executive management positions at Sybase and Powersoft where he was responsible for helping build one of the first enterprise client/server software development tools and enterprise Web application development tools. He has consulted with hundreds of major corporations on topics of technology strategy, open source and design, and development of enterprise computing systems. Bob holds a BS from Keene State College. He has been a guest lecturer at the MBA programs of Brigham Young University and Salem (Mass.) State College, as well as a speaker at the Harvard University Cyberposium and Duke's Graduate School.

REGISTER TODAY AND $AVE!

Rich Internet Applications: AJAX, Flash, Web 2.0 and Beyond...

www.AjaxWorldExpo.com

AJAXWORLD™ EAST CONFERENCE & EXPO

NEW YORK CITY

THE ROOSEVELT HOTEL LOCATED AT MADISON & 45th

SYS-CON Events is proud to announce the AjaxWorld East Conference 2007!

The world-beating Conference program will provide developers and IT managers alike with comprehensive information and insight into the biggest paradigm shift in website design, development, and deployment since the invention of the World Wide Web itself a decade ago.

The terms on everyone's lips this year include "AJAX," "Web 2.0" and "Rich Internet Applications." All of these themes play an integral role at AjaxWorld. So, anyone involved with business-critical web applications that recognize the importance of the user experience needs to attend this unique, timely conference – especially the web designers and developers building those experiences, and those who manage them.

BEING HELD APRIL 2-3 2007!

We are interested in receiving original speaking proposals for this event from i-Technology professionals. Speakers will be chosen from the co-existing worlds of both commercial software and open source. Delegates will be interested in learning about a wide range of RIA topics that can help them achieve business value.

AJAX and Simpler Times

We live in the eternal present, yet think mostly about the future and the past

Roger Strukhoff

We live in the eternal present, yet think mostly about the future and the past. When we are able to stop time and consider what's going on "right now" or "these days," we often think about how our lives and times used to be simpler. How often do you recount stories from a "simpler, more innocent time?"

The truth is, there has never been a simpler, more innocent time. One doesn't need to be overly acquainted with history to realize this. But the temptation is there, to remember fondly how life used to be, somehow, easier. In the IT industry, this feeling is often invoked when thinking about an earlier age when IBM was dominant in all phases of the industry, when it was IBM being attacked by the U.S. government for alleged monopolistic practices, when no one got fired for buying IBM.

But from today's perspective, a period as recently as two years ago may seem to represent a simpler time, when WebSphere had clearly distanced itself from competitive products in the web development application space, and the only J2EE environment worth considering came from Big Blue.

Then came Open Source. It was no longer enough for IBM to put the hurt to BEA and Microsoft in this space, but now the company had to take open source web app development seriously. Sure, it seemed a remote possibility that major government agencies or Fortune 50 companies were going to take open source seriously. But the steady drumbeat of the open source movement has converted government administrators at many levels, educational institutions, large segments of the SMB market, and is percolating upward into the largest companies in the world as well, on a global basis. Then came AJAX. Not to confuse or equate the terms, but it is a reality that the mindset that is attracted to Open Source is also attracted to AJAX, which uses two widely deployed languages (JavaScript and XML) to create the websites of the present and future.

IBM hardly ignored this, loosing its Open AJAX initiative on the world a couple of months ago. But Open AJAX is, in fact, a bit too open in that it does not have a structure, a specific mission or goals, membership requirements, or milestones that will define its success.

IBM's David Boloker is a clear leader in this movement, though, and he is a compelling figure around whom any number of companies (and developers) should rally in coming months. Boloker will be speaking at SYS-CON Media's Real-World AJAX seminar in New York in June 5-6, during which time he is expected to bring attendees up to speed on IBM's vision of an Open AJAX future. The Open Source and AJAX movements can all lead one to believe that previous times were simpler. What are organizations with entrenched WebSphere application development initiatives supposed to make of all this?

Fortunately, there seems not to be a simple, binary answer to this question, as IBM seems intent on embracing all levels of budget and application complexity with its family of application development products and approaches. However, in the short, medium, and long term the company must (and will) realize that AJAX is a true genie, now out of its bottle.

Open Source is as much a political movement as a technical one. AJAX, on the other hand, is pure practicality. The idea that specific elements of a page can communicate directly with the server-rather than the old way of either encoding functionality on the desktop through JavaScript or sending the whole bloody page back to the server-is launching us into an entirely new Web 2.0 era of web design and functionality. Organizations of any size will want to, will be required to, embrace AJAX, to make their websites look better, act better, and perform better. IT managers will need to get a grip on new load balancing and overall storage management requirements. And the world will continue to seem more complex than it was in the good old day. ■

Roger Strukhoff is President of www.wdva.com. He spent 15 years with Miller Freeman Publications and The International Data Group (IDG), then co-founded CoverOne Media, a custom publishing agency that he sold in 2004. His work has won awards from the American Business Media, Western Press Association, Illinois Press Association, and the Magazine Publishers Association. Read his blog at http://www.rssblog.Linux.SYS-CON.com. roger@sys-con.com

VISIBLEMEASURES

Next Generation Analytics

See what's beneath

Coming Fall 2006

Sneak preview at:

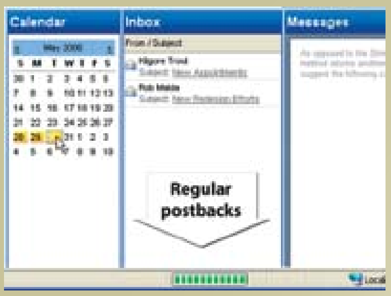
www.visiblemeasures.com/ajaxworld

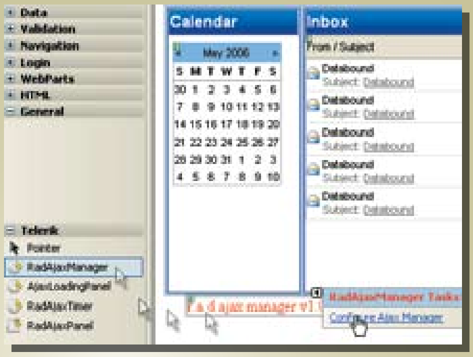
Visible Measures uncovers insights into how customers interact with increasingly complex Internet-enabled applications to empower informed decisions and accelerate business goals.

telerik
deliver more than expected

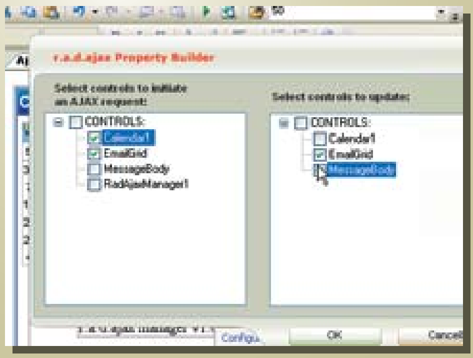
1min < AJAX-enable any application < 2 min

Reporting Solution | ASP.NET UI Suite | AJAX Framework | WinForms UI Suite

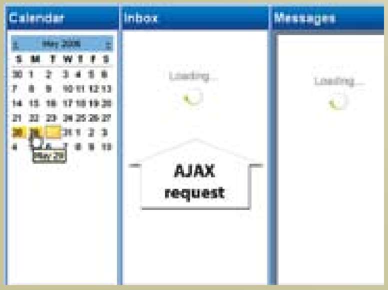




Step 1:
Drag the AJAX Manager control on the WebForm
Total time: ~ 5 sec.


r.a.d.ajax Property Builder
Select controls to initiate an AJAX request:
Select controls to update:
CONTROLS:
Calendar1
EmailGrid
MessageBody
RadAjaxManager1
OK
Cancel

Step 2:
Use the single dialog to set:
• which controls initiate AJAX
• which controls are updated with AJAX
Total time: < 1.55 min.



• **No Previous AJAX Experience Needed**
r.a.d.ajax does not require any prior experience with AJAX. You don't need to learn an "AJAX-specific" way of building new applications.

• **No Modifications to Your Application Necessary**
No need to place Callback Panels around areas that need to be updated. No need to set triggers or manually invoke AJAX requests.

• **Intuitive and Centralized Management of AJAX Relations**
All relations between controls which initiate AJAX and those that are updated with AJAX are managed from a single dialog.

• **Codeless Development**
Drag-and-drop the AJAX Manager control on the page, tick the respective checkboxes in the dialog, and hit F5.

Watch a video at: www.telerik.com/ajaxvideos
Total time: 1.37 min.

www.telerik.com/radajax