Asynchronous Ajax for Revolutionary Web Applications

Jeanfrancois Arcand

Ted Goddard, Ph.D.

ICEfaces GlassFish

Join the Asynchronous Web Revolution!

Easily develop multi-user collaboration features in NetBeans with Ajax Push and Comet using Dojo, DWR, or ICEfaces.

Deploy and scale on Jetty, Tomcat, or GlassFish.

Agenda

- Web2.0™
- Multi-user Ajax Demo
- Asynchronous HTTP on the Wire
- Asynchronous HTTP and the Server
- Developing Asynchronous Applications
- ICEfaces Details
- Conclusion

What sort of revolution?

"And yet it moves."

American Re

Dump everythin

Revolution

e Bastille

Scientific Revolution

Experimentation and Rationality

Web2.0™
A Web by the people, for the people.

- Documents on the web increasingly generated by users
- Out of the Information Age, into the Participation Age
- As a whole, the World Wide Web is a collaborative environment, but individual pages are only weakly so
- Are web user interfaces becoming more powerful?
- Is the user an HTTP client?

Ajax
Ajax is a state of mind.

- It was AJAX (Asynchronous JavaScript with XML)
  - or Asynchronous JavaScript with XMLHttpRequest
  - now it's Ajax (not an acronym) because many different techniques satisfied the same goals
  - coined by Jesse James Garrett in 2005 to sell an insurance company on re-writing all their software
- Is the web defined by the W3C or by browser implementers? (Ajax does not exist in W3C universe yet.)
- Ajax decouples user interface from network protocol
- Ajax is the leading edge of the user interface possible with current popular browsers
- The user experience is important

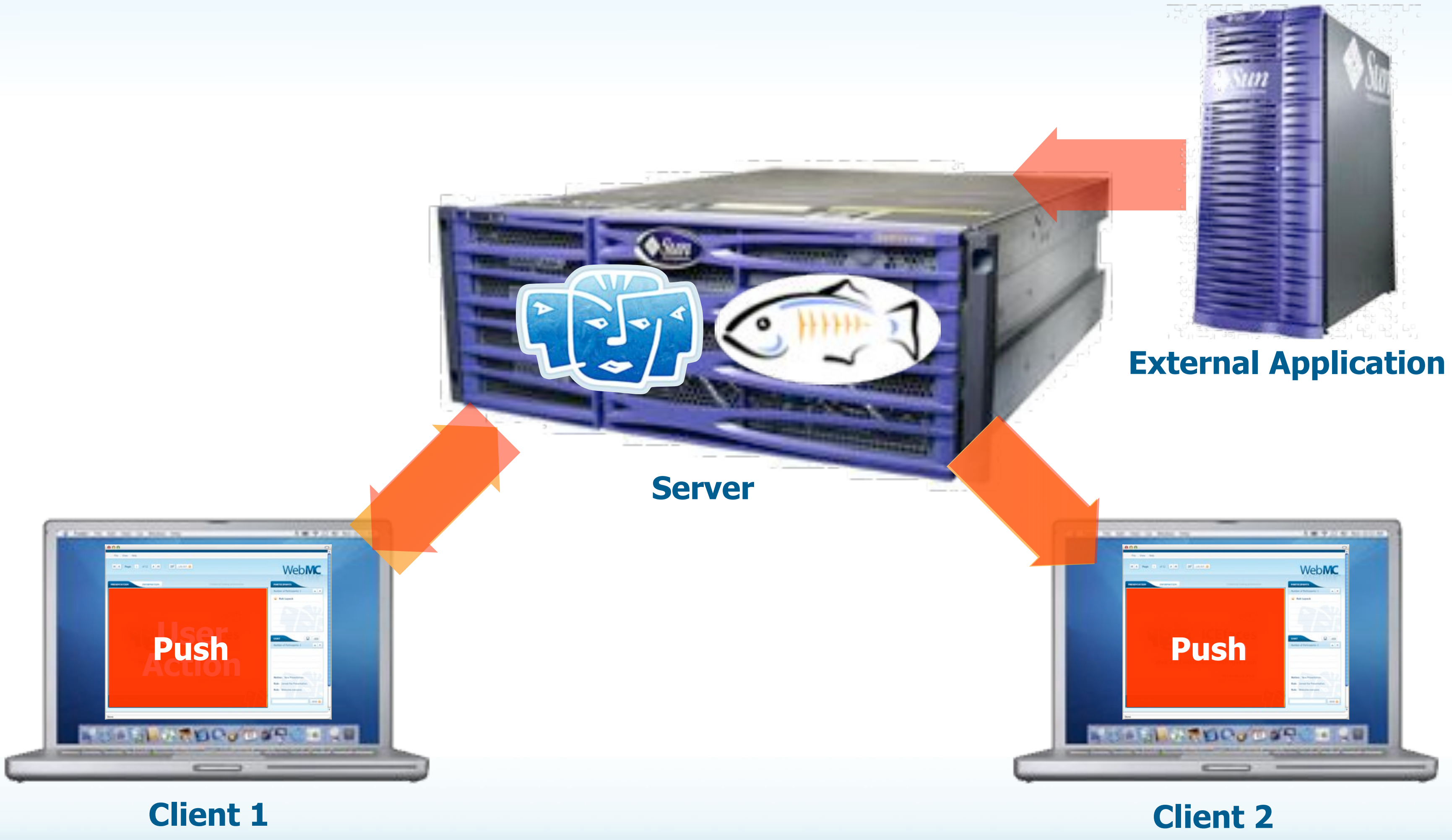
The Asynchronous Web Revolution
The Web enters the Participation Age.

- Ajax still typically synchronous with user events
- Full asynchrony has updates pushed from server any time

- Update pages after they load
- Send users notifications
- Allow users to communicate and collaborate within the web application

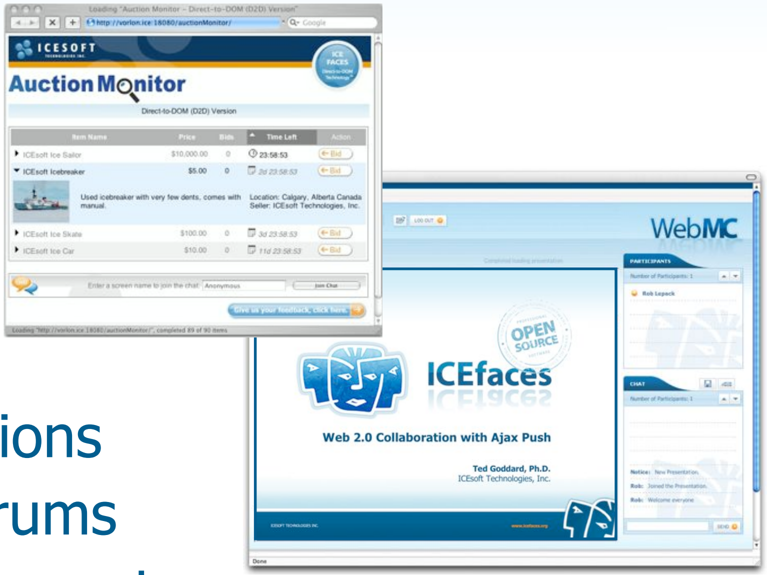
- Called "Ajax Push", "Comet", or "Reverse Ajax"
  - This is the full realization of Ajax, now fully asynchronous

Server-mediated Collaboration
The full realization of Ajax.



Applications in the Participation Age
Application-mediated communication.

- Distance learning
- Collaborative authoring
- Auctions
- Shared WebDAV filesystem
- Blogging and reader comments
- SIP-coordinated mobile applications
- Hybrid chat/email/discussion forums
- Customer assistance on sales/support pages
- Multi-step business process made collaborative
- Shared trip planner or restaurant selector with maps
- Shared calendar, "to do" list, project plan
- Games


Loading "Auction Monitor – Direct-to-DOM (D2D) Version"
http://vorlon.ice:18080/auctionMonitor/
ICESOFT
Auction Monitor
Direct-to-DOM (D2D) Version
Enter a screen name to join the chat: Anonymous
Join Chat
Give us your feedback, click here.
WebMC
PARTICIPANTS
CHAT
ICEfaces
OPEN SOURCE
Web 2.0 Collaboration with Ajax Push
Ted Goddard, Ph.D.
ICEsoft Technologies, Inc.

Item Name	Price	Bids	Time Left	Action
ICEsoft Ice Sailor	$10,000.00	0	23:58:53	Bid
ICEsoft Icebreaker	$5.00	0	2d 23:58:53	Bid
Used icebreaker with very few dents, comes with manual.			Location: Calgary, Alberta Canada Seller: ICEsoft Technologies, Inc.	
ICEsoft Ice Skate	$100.00	0	3d 23:58:53	Bid
ICEsoft Ice Car	$10.00	0	11d 23:58:53	Bid

Agenda

- Web2.0™
- Multi-user Ajax Demo
- Asynchronous HTTP on the Wire
- Asynchronous HTTP and the Server
- Developing Asynchronous Applications
- ICEfaces Details
- Conclusion

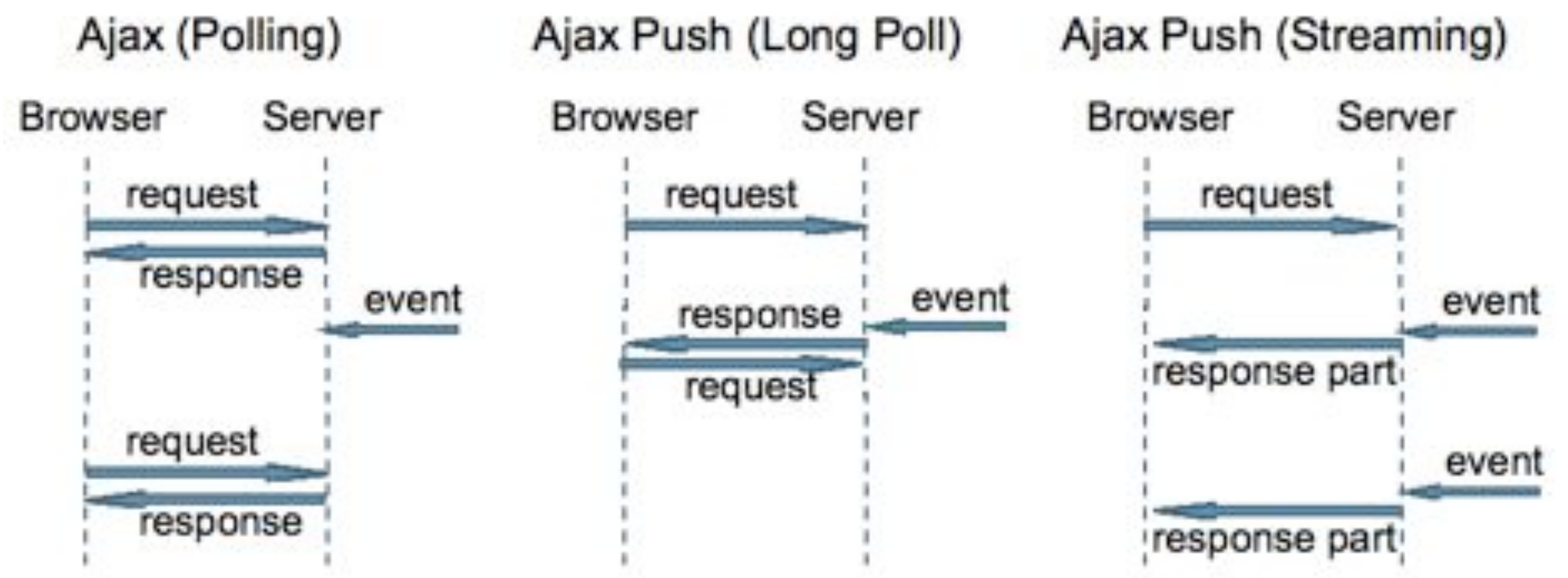
Asynchronous Ajax Demo with ICEfaces and GlassFish Grizzly

http://webmc.icefaces.org

Agenda

- Web2.0™
- Multi-user Ajax Demo
- Asynchronous HTTP on the Wire
- Asynchronous HTTP and the Server
- Developing Asynchronous Applications
- ICEfaces Details
- Conclusion

Ajax Poll vs Ajax Push
Bending the rules of HTTP.



Ajax Poll vs Ajax Push
Bending the rules of HTTP.

- **Poll:**
  - Send a request to the server every X seconds.
  - The response is “empty” if there is no update.
- **Long Poll:**
  - Send a request to the server, wait for an event to happen, then send the response.
  - The response is never empty.
  - HTTP specification satisfied: indistinguishable from “slow” server
- **Http Streaming:**
  - Send a request, wait for events, stream multi-part/chunked response, and then wait for the events.
  - The response is continually appended to.

HTTP Polling
Regularly checking for updates.

```
GET /chatLog HTTP/1.1
Accept: */*
Connection: keep-alive

<message>One</message>
```

- Uses the HTTP protocol in a standard way, but requests are regularly invoked

```
setTimeout('poll()', 10000);
```

Asynchronous HTTP Streaming
The long response.

```
GET /chatLog HTTP/1.1
Accept: */*
Connection: keep-alive

<message>One</message>
<message>Two</message>
<message>Three</message>
<message>Four</message>
```

- Parse most recent message in JavaScript (not shown here)
- The original 1999 “Push” technique (Netscape 1.1)

Ajax Push
HTTP message flow inversion.

```
GET /auctionMonitor/block/receive-updates?icefacesID=1209765435 HTTP/1.1
Accept: */*
Cookie: JSESSIONID=75CF2BF3E03F0F9C6D2E8EFE1A6884F4
Connection: keep-alive
Host: vorlon.ice:18080

HTTP/1.1 200 OK
Content-Type: text/xml;charset=UTF-8
Content-Length: 180
Date: Thu, 27 Apr 2006 16:45:25 GMT
Server: Apache-Coyote/1.1

<updates>
  <update address="_id0:_id5:0:chatText">
    <span id="_id0:_id5:0:chatText">Howdy</span>
  </update>
</updates>
```

Chat message "Howdy"

Agenda

- Web2.0™
- Multi-user Ajax Demo
- Asynchronous HTTP on the Wire
- Asynchronous HTTP and the Server
- Developing Asynchronous Applications
- ICEfaces Details
- Conclusion

Architecture Challenge
Can Push scale?

- A blocking, synchronous technology will result in a blocked thread for each open connection that is “waiting”
- Every blocked thread will consume memory
- This lowers scalability and can affect performance
- To get the Java Virtual Machine (JVM™) to scale to 10,000 threads and up needs specific tuning and is not an efficient way of solving this

- Servlets 2.5 are an example of blocking, synchronous technology

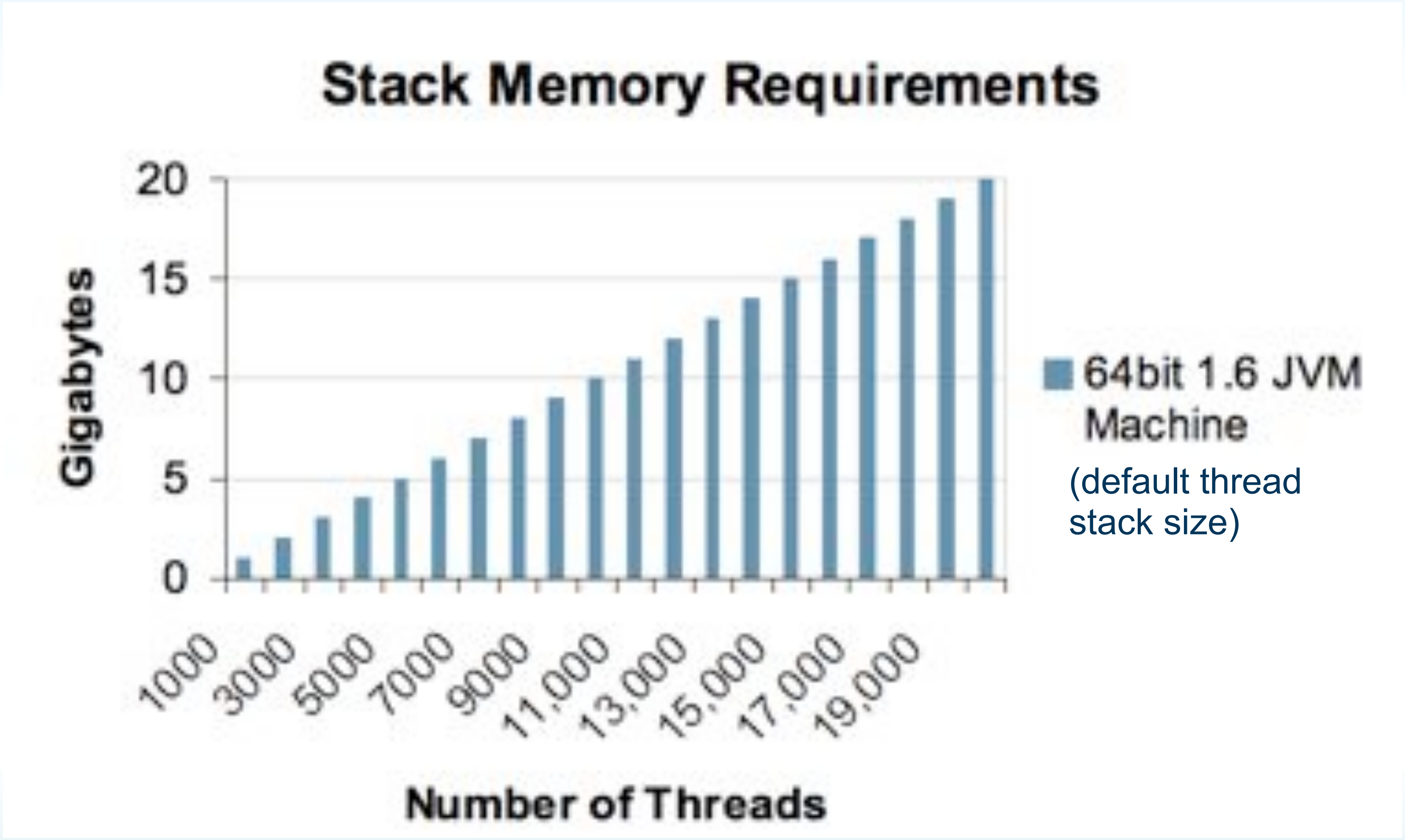
Servlet Thread Catastrophe
Strangled by a thread for every client.

GET /updates HTTP/1.1
Connection: keep-alive

GET /updates HTTP/1.1
Connection: keep-alive

GET /updates HTTP/1.1
Connection: keep-alive

Architecture Challenges
The serious effect of blocking threads.



(default thread stack size)

Server-side Ajax Push: Who supports what
The asynchronicity matrix.

Container	Asynchronous IO	Suspendible Request/ Response	Delivery Guarantee
Jetty		X	
Tomcat	X	X	
GlassFish	X	X	X
Resin		X	
WebLogic		X	

Jetty
service() will resume shortly.

```
import org.mortbay.util.ajax.Continuation;

service(request, response) {
    Continuation continuation = ContinuationSupport
        .getContinuation(request, this);
    ...
    continuation.suspend();
    response.getWriter().write(message);
}
```

Asynchronously and elsewhere in the application ...

```
message.setValue("Howdy");
continuation.resume();
```

Tomcat 6
Eventful Comet.

```
import org.apache.catalina.CometProcessor;

public class Processor implements CometProcessor {

public void event(CometEvent event)  {
    request = event.getHttpServletRequest();
    response = event.getHttpServletResponse();

    if (event.getEventType() == EventType.BEGIN) { ...
    if (event.getEventType() == EventType.READ)  { ...
    if (event.getEventType() == EventType.END)   { ...
    if (event.getEventType() == EventType.ERROR) { ...
}
```

Asynchronously and elsewhere in the application ...

```
message.setValue(“Howdy”);
response.getWriter().write(message);
event.close();
```

Resin
Suspend, Wake, and Resume with Resin.

```
public class CometServlet extends GenericCometServlet {
    public boolean service(ServletRequest request,
                           ServletResponse response,
                           CometController cometController)
      ...
      return true;
    }

    public boolean resume(ServletRequest request,
                          ServletResponse response,
                          CometController cometController)
      PrintWriter out = res.getWriter();
      out.write(message);
      return false;
}
```

Suspend

Resume

Asynchronously and elsewhere in the application ...

```
message.setValue("Howdy");
cometController.wake();
```

WebLogic
doRequest() and doResponse() separated by notify().

```
import weblogic.servlet.http.AbstractAsyncServlet;
import weblogic.servlet.http.RequestResponseKey;

class Async extends AbstractAsyncServlet {

boolean doRequest(RequestResponseKey rrk) {
    ... = rrk;
    return false;
}

void doResponse(RequestResponseKey rrk, Object message) {
    rrk.getResponse().getWriter.write(message);
}
```

Asynchronously and elsewhere in the application ...

```
message.setValue("Howdy");
AbstractAsyncServlet.notify(rrk, message);
```

GlassFish
Suspend with Grizzly.

```
CometContext context =
    CometEngine.getEngine().register(contextPath);
context.setExpirationDelay(20 * 1000);
SuspendableHandler handler = new SuspendableHandler();
handler.attach(response);
cometContext.addCometHandler(handler);
```

```
class SuspendableHandler implements CometHandler  {
    public void onEvent(CometEvent event) {
        response.getWriter().println(event.attachment());
        cometContext.resumeCometHandler(this);
    }
```

Asynchronously and elsewhere in the application ...

```
message.setValue("Howdy");
cometContext.notify(message);
```

Servlet 3.0
Future Asynchronous Standard.

- Defined by JSR-315 Expert Group
- DWR, Jetty, Tomcat, GlassFish and ICEfaces participants
- Standard asynchronous processing API being defined
  - Asynchronous I/O
  - Suspendible Request
  - Not included: Delivery Guarantee

- Will improve portability of DWR, Cometd, and ICEfaces
- (But unless you write Servlets today, this API will be hidden by your chosen Ajax framework.)

Agenda

- Web2.0™
- Multi-user Ajax Demo
- Asynchronous HTTP on the Wire
- Asynchronous HTTP and the Server
- Developing Asynchronous Applications
- ICEfaces Details
- Conclusion

JavaScript Polling
Are we there yet? Are we there yet? Are we there yet? ...

```
function poll()  {
  setTimeout(‘poll()’, 10000);
  req = new XMLHttpRequest();
  req.onreadystatechange = update();
  req.open(“POST”, “http://server/getMessage.jsp”);
}

function update() {
  chatLog.innerHTML = req.responseText;
}

poll();
```

Cometd
Distributed, loosely coupled scripting

JavaScript

```
function update(message) {
  chatLog.innerHTML = message.data.value;
}
...
  cometd.subscribe("chat", remoteTopics, "update")
  cometd.publish("chat", message)
```

Java

```
import dojox.cometd.*;

Channel channel = Bayeux.getChannel("chat", create);
channel.subscribe(client);
```

Asynchronously and elsewhere in the application ...

```
message.setValue("Howdy");
channel.publish(client, message, "chat text");
```

Bayeux/Cometd
JSON Pub/Sub.

```
[
  {
    "channel": "/some/name",
    "clientId": "83js73jsh29sjd92",
    "data": { "myapp" : "specific data", value: 100 }
  }
]
```

- JSON Messages are published on specified channels
- Channel operations: connect, subscribe, unsubscribe, etc.
- Multiple transports: polling, long-polling, iframe, flash
- Server implementations: Perl, Python, Java
- Server-side reflector with no server-side application possible

Grizzly Messages Bus

- The Grizzly Messages Bus implements the Grizzly Comet Protocol (GCP).
- The GCP protocol is a very basic protocol that can be used by browser to share data, using the comet technique, between several clients without having to poll for it.
- The protocol is very simple. First, a client must subscribe to a topic:
  - http://host:port/contextPath?subscribe=[topic name]&cometTechnique=[polling|log-polling|http-streaming]&message=[text]
- When issuing the URL above, the connection will be automatically suspended based on the cometTechnique specified

Grizzly Messages Bus

- To share data between applications, a browser just need to send the following request:
  - http://host:port/contextPath?publish=[topic name]&message=[text]
- The Servlet can be used as it is or extended to add extra features like filtering messages, security, login, etc.
- Quite easy to write games using the Grizzly Messages Bus. No server side implementation required, just client side!

DWR
JavaScript RPC

```
import org.directwebremoting.proxy.dwr.Util;

scriptSessions =
    webContext.getScriptSessionsByPage(currentPage);
    util = new Util(scriptSessions);
```

To “Reverse Ajax” and invoke arbitrary JavaScript:

```
util.addScript(ScriptBuffer script);
```

Asynchronously and elsewhere in the application ...

```
util.setValue(“form:chat:_id3”, “Howdy”);
```

ICEfaces
Preserve MVC with JSF and Transparent Ajax.

PageBean.java

```
public class PageBean  {
    String text;

    public String getText()  {
        return text;
    }

    public void setText(String text)  {
            this.text = text;
    }

}
```

Presentation Model

Page.xhtml

```
<f:view
        xmlns:f="http://java.sun.com/jsf/core"
        xmlns:h="http://java.sun.com/jsf/html"
     >
  <html>
    <body>
      <h:form>
        <h:inputText value="#{pageBean.text}" /
     >
      </h:form>
    </body>
  </html>
</f:view>
```

Declarative User Interface

A language for Ajax Push that preserves Designer and Developer roles

ICEfaces
High level push.

```
import org.icefaces.application.SessionRenderer;
```

One line of code for basic Ajax Push in standard JSF:

```
SessionRenderer.render(SessionRenderer.ALL_SESSIONS);
```

Or to keep track of groups of users:

```
SessionRenderer.addCurrentSession("chat");
```

Asynchronously and elsewhere in the application ...

```
message.setValue("Howdy");
SessionRenderer.render("chat");
```

The JSF lifecycle runs and each user's page is updated from the component tree.

SessionRenderer Details
Framework-managed Ajax Push.

- Sessions are removed from groups upon expiry

- Session groups are created upon first join
- Session groups are removed when empty

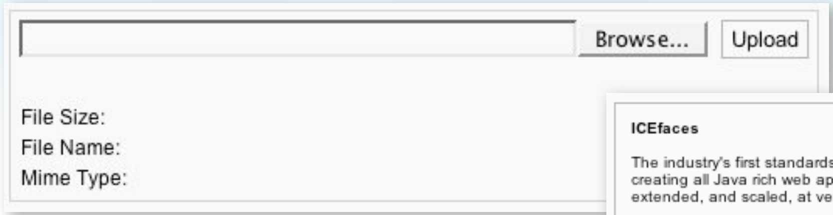
- Inefficient to render all windows in a session?

- All views but the caller are rendered

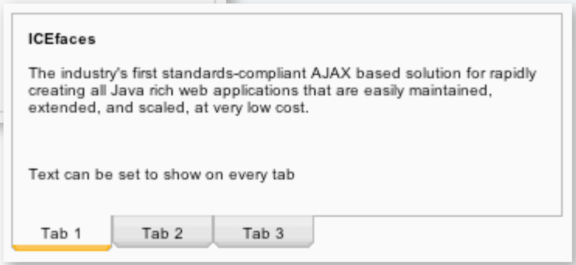
Agenda

- Web2.0™
- Multi-user Ajax Demo
- Asynchronous HTTP on the Wire
- Asynchronous HTTP and the Server
- Developing Asynchronous Applications
- ICEfaces Details
- Conclusion

ICEfaces Open Source Ajax Components



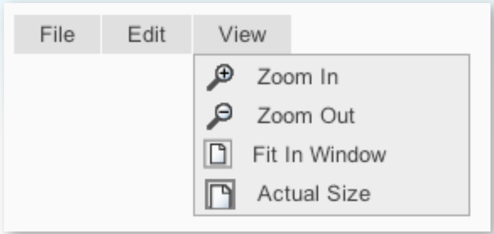
File Size:
File Name:
Mime Type:

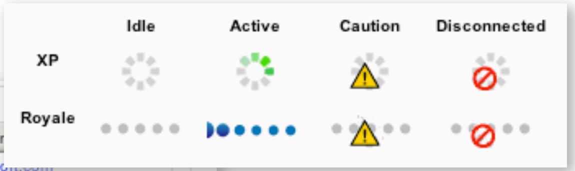


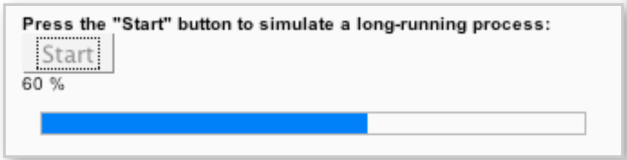
ICEfaces

The industry's first standards-compliant AJAX based solution for rapidly creating all Java rich web applications that are easily maintained, extended, and scaled, at very low cost.

Text can be set to show on every tab

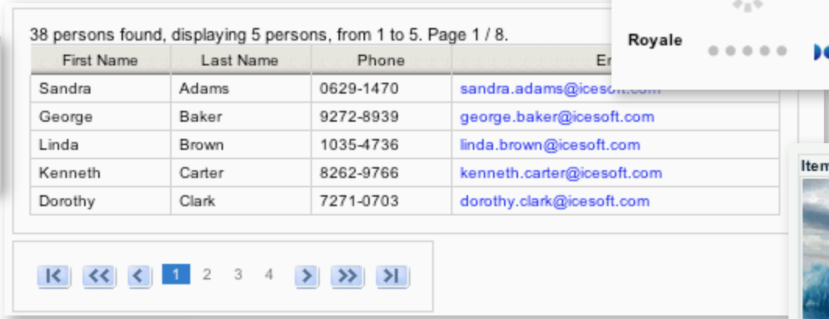






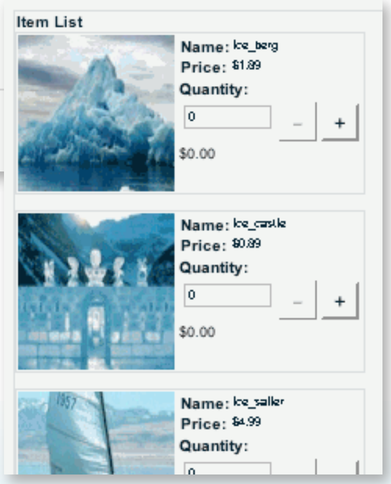
Press the "Start" button to simulate a long-running process:

60 %



38 persons found, displaying 5 persons, from 1 to 5. Page 1 / 8.

First Name	Last Name	Phone	Em
Sandra	Adams	0629-1470	sandra.adams@iceso
George	Baker	9272-8939	george.baker@icesoft.com
Linda	Brown	1035-4736	linda.brown@icesoft.com
Kenneth	Carter	8262-9766	kenneth.carter@icesoft.com
Dorothy	Clark	7271-0703	dorothy.clark@icesoft.com

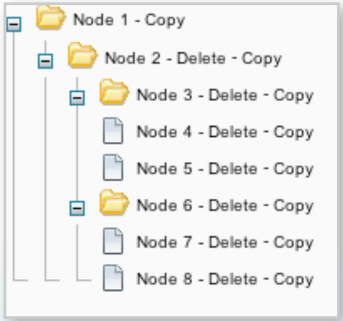


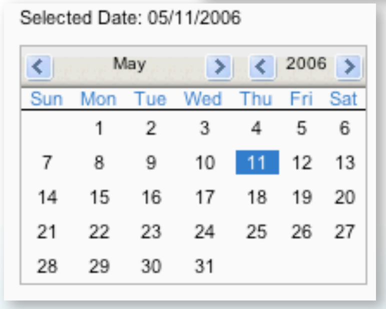
Item List

Name: Ice_berg
Price: $1.89
Quantity: 0
$0.00

Name: Ice_castle
Price: $0.89
Quantity: 0
$0.00

Name: Ice_sailer
Price: $4.99
Quantity: 0

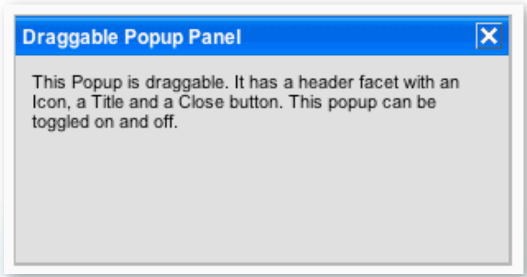




Selected Date: 05/11/2006

May 2006

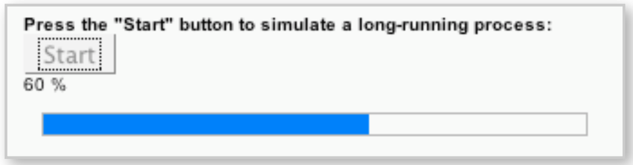
Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			



Draggable Popup Panel

This Popup is draggable. It has a header facet with an Icon, a Title and a Close button. This popup can be toggled on and off.

Asynchronous Progress Bar

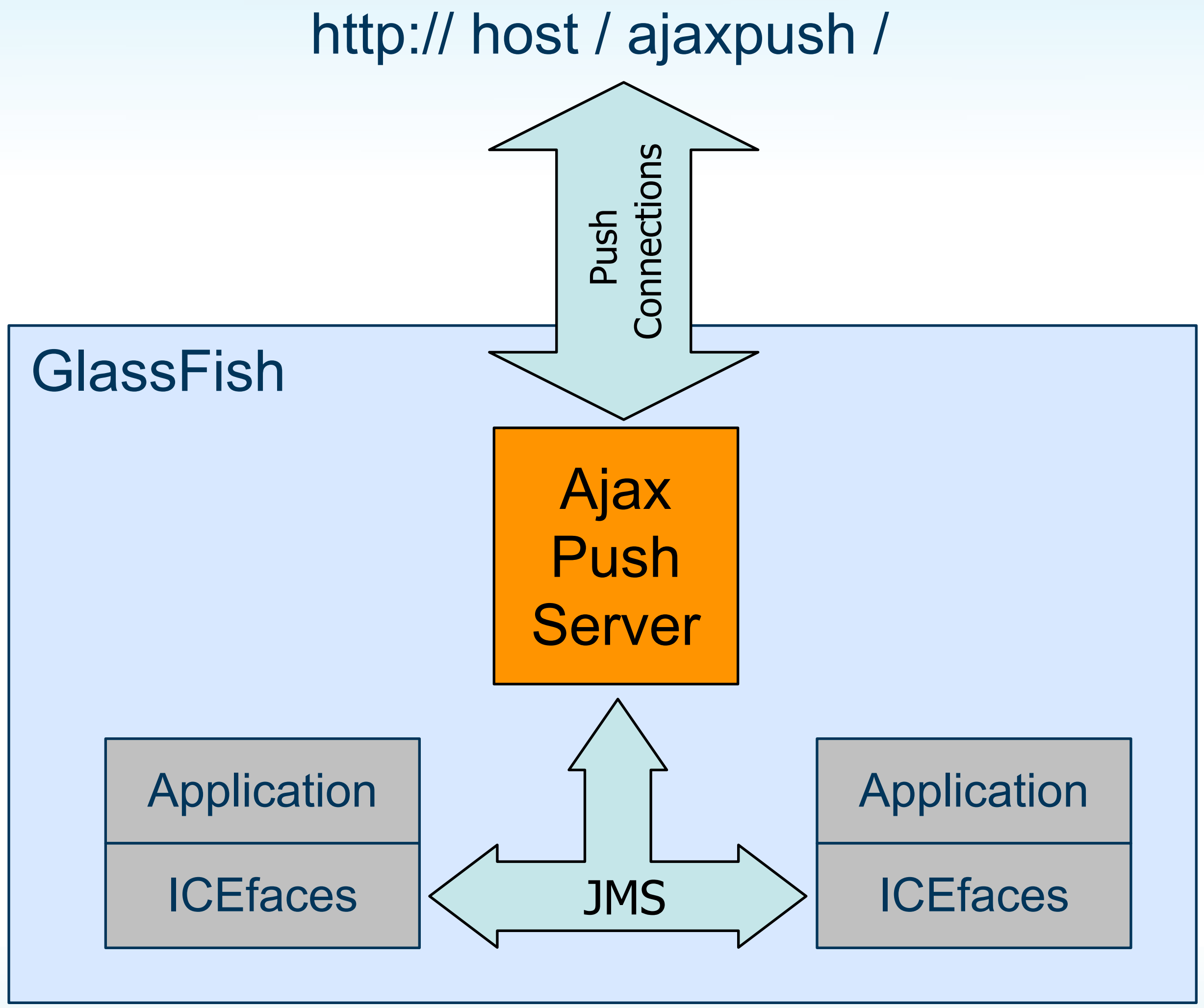

Press the "Start" button to simulate a long-running process:

RFC 2616: HTTP 1.1

```
  Clients that use persistent connections SHOULD limit the number of
simultaneous connections that they maintain to a given server. A
single-user client SHOULD NOT maintain more than 2 connections with
any server or proxy. ... These guidelines are intended to improve
HTTP response times and avoid congestion.
```

- Two-connection limit is a guideline
- Is a "client" a browser or a window?
  - windows have isolated JavaScript memory spaces
- "Share" a single connection across windows
  - notify windows of updates via cookie polling
  - easier to implement with postMessage()

Ajax Push Server



Agenda

- Web2.0™
- Multi-user Ajax Demo
- Asynchronous HTTP on the Wire
- Asynchronous HTTP and the Server
- Developing Asynchronous Applications
- ICEfaces Details
- Conclusion

Summary
The Asynchronous Web Revolution is Now

- The Asynchronous Web will revolutionize human interaction
- Push can scale with Asynchronous Request Processing
- With ICEfaces, GlassFish, and Grizzly, the revolution begins with your applications today
- Get ready for Servlet 3.0

Asynchronous Ajax for Revolutionary Web Applications

TedGoddard@icesoft.com

http://www.icefaces.org

Jeanfrancois.Arcand@sun.com

http://glassfish.dev.java.net

Thank You