Writing Real-Time Web Applications Using Google Web Toolkit and Comet

Alexandre Gomes - SEA Tecnologia
Jeanfrançois Arcand - Sun Microsystems
Ted Goddard - ICESoft

BOF-4922

Agenda

- Introduction
- Google Web Toolkit
- Introduction to Ajax Push (Comet)
- ICEFaces Demo
- Introduction to Grizzly Comet
- Grizzly Comet & GWT
- GWT Demo

To demonstrate how easy it is to combine Grizzly Comet and GWT to facilitate the creation of real-time RIA applications.

GOAL

Introduction to GWT

- What is GWT?
  - Converts Java code to HTML and JavaScript (a compiler)
- Download
  - http://code.google.com/webtoolkit/
- Install
  - tar xzvf gwt-mac-1.4.62.tar.gz
- Samples
  - samples/KitchedSink/KitchenSink-shell
  - samples/KitchedSink/KitchenSink-compile
- New app
  - applicationCreator br.com.sea.test.client.ATest

GWT - Getting Started

- New application

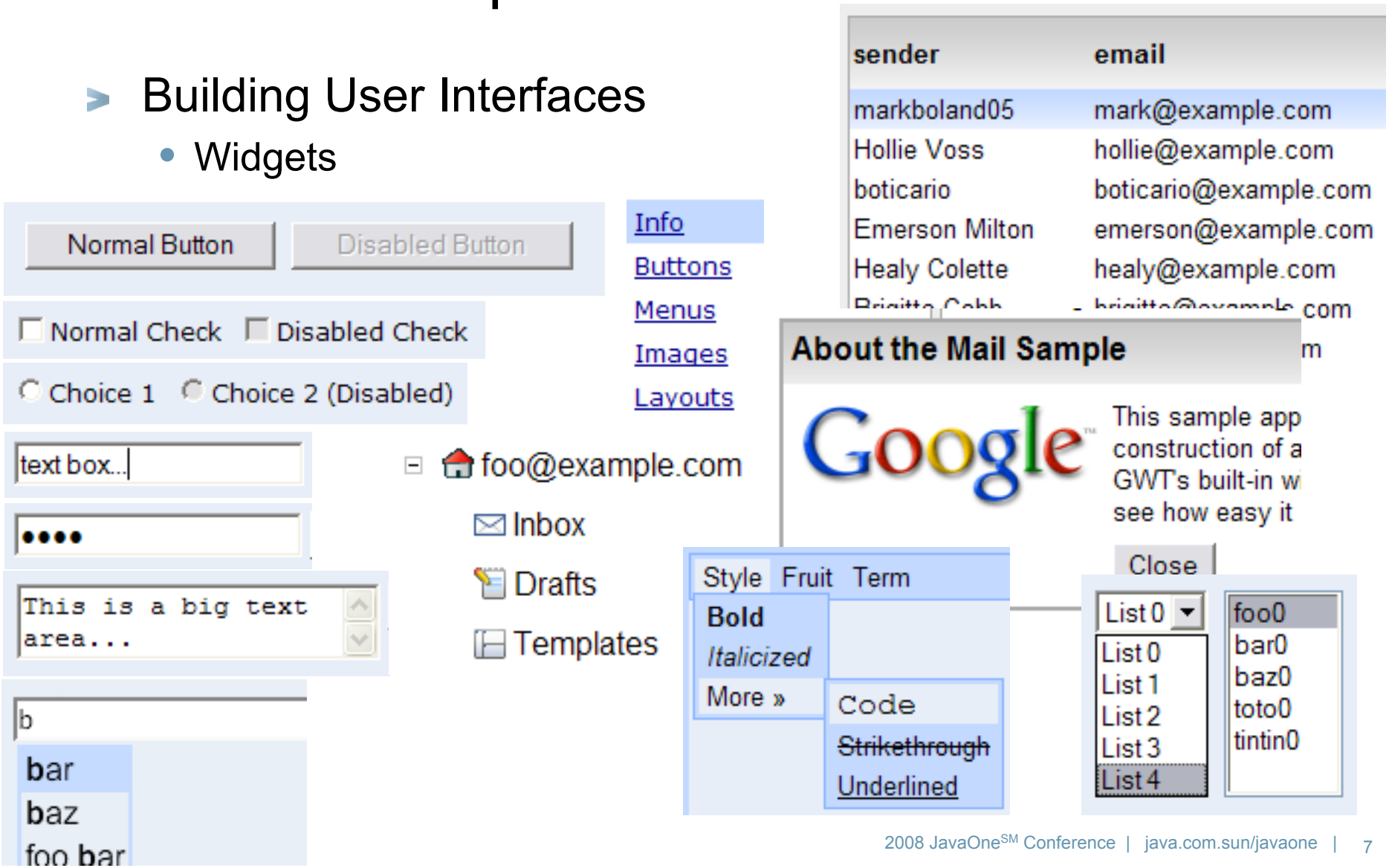
```
$ $GWT_HOME/applicationCreator br.com.sea.teste1.client.Teste1
Created directory /tmp/teste1/src
Created directory /tmp/teste1/src/br/com/sea/teste1
Created directory /tmp/teste1/src/br/com/sea/teste1/client
Created directory /tmp/teste1/src/br/com/sea/teste1/public
Created file /tmp/teste1/src/br/com/sea/teste1/Teste1.gwt.xml
Created file /tmp/teste1/src/br/com/sea/teste1/public/Teste1.html
Created file /tmp/teste1/src/br/com/sea/teste1/client/Teste1.java
Created file /tmp/teste1/Teste1-shell
Created file /tmp/teste1/Teste1-compile
$
```

GWT - Developers Guide

- Building User Interfaces
  - Widgets
    - Contained in Panels
    - Same behavior in multiple browsers
      - e.g. Button, TextBox, Tree
  - Panels
    - Acts as a Container and a Layout Manager
      - e.g. DockPanel, HorizontalPanel, RootPanel

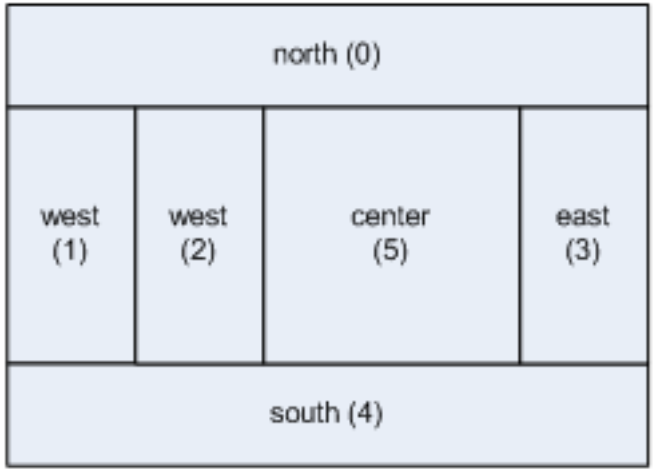
GWT - Developers Guide

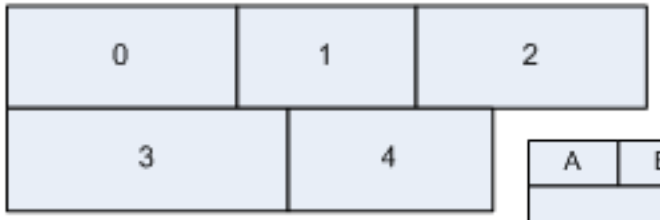
- Building User Interfaces
  - Widgets

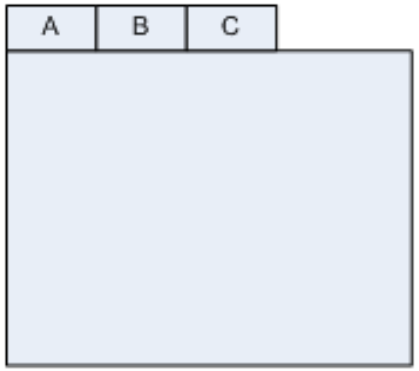


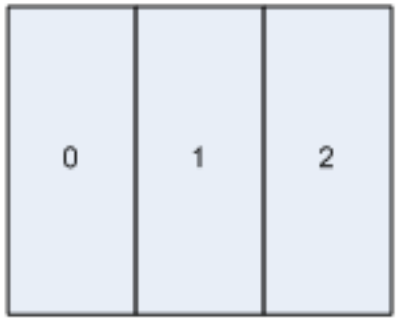
GWT - Developers Guide

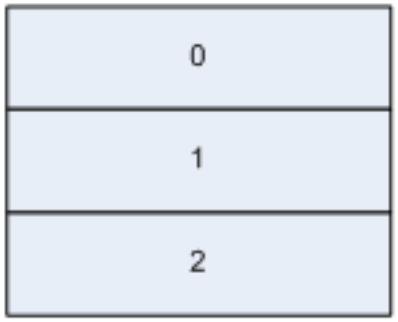
- Panels

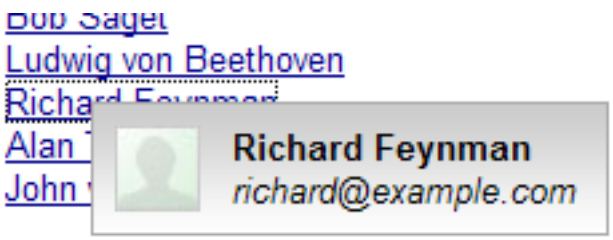


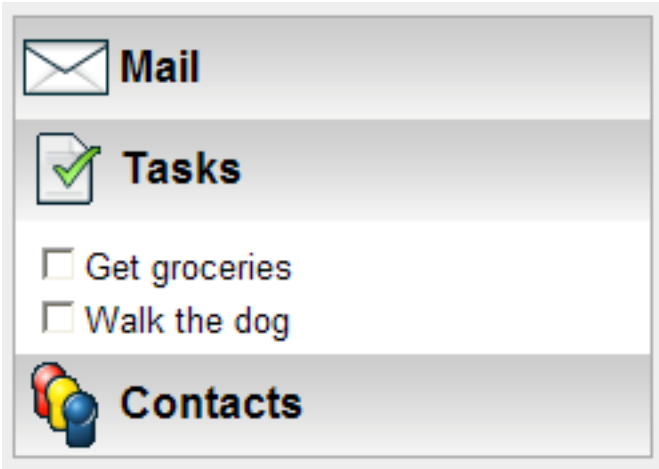






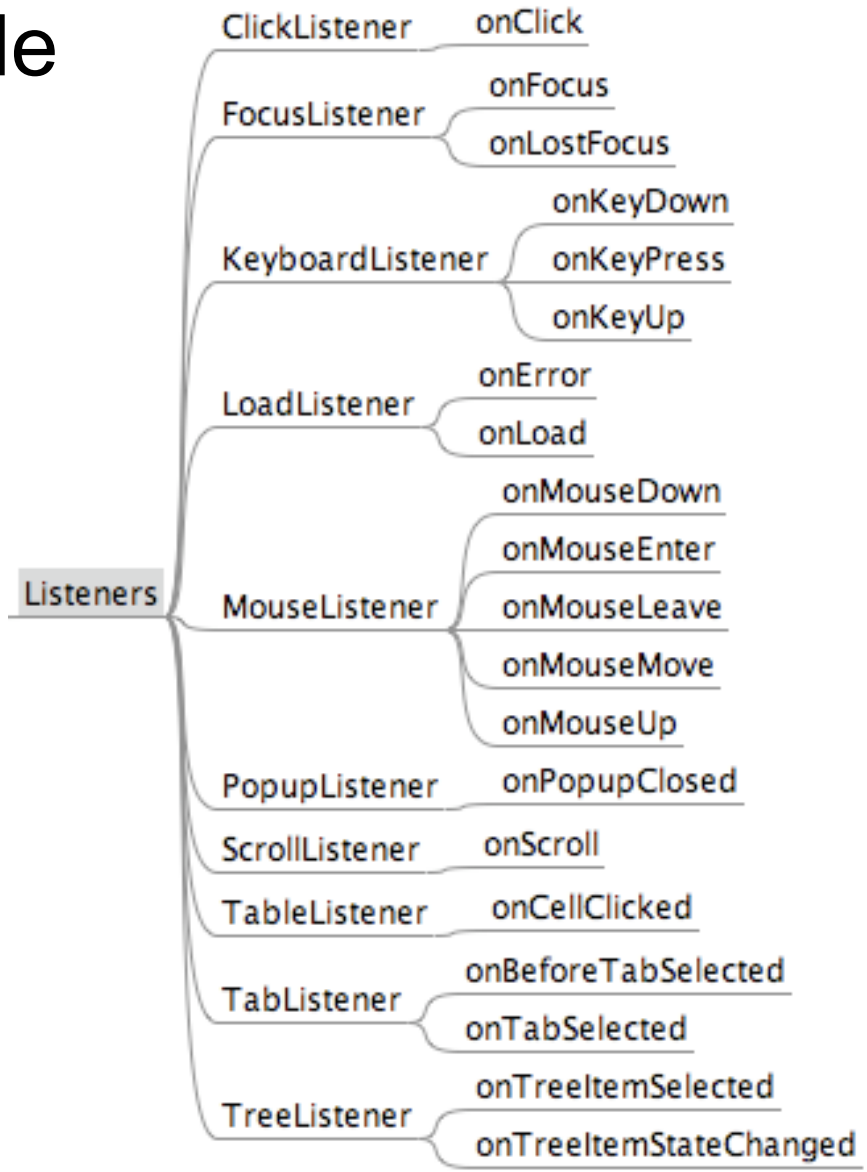






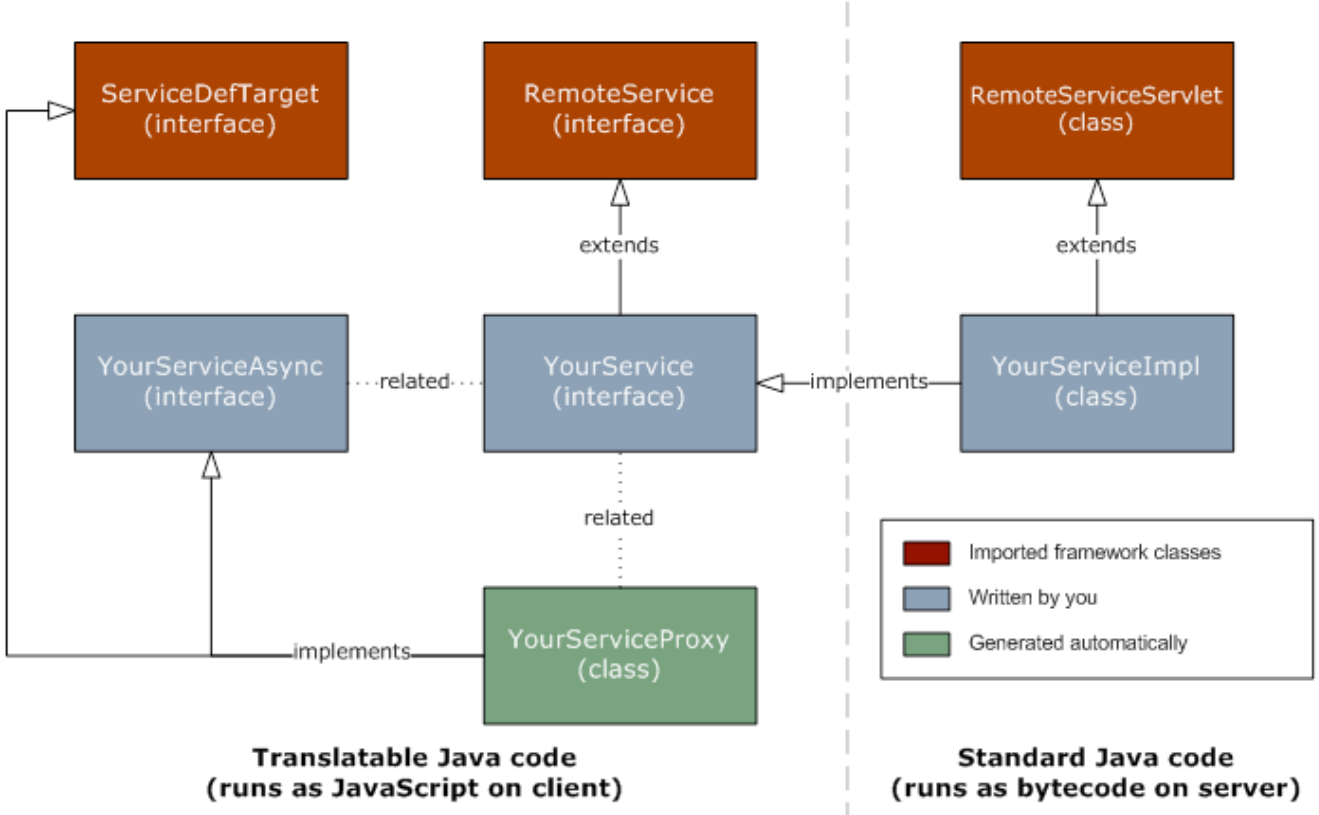
GWT - Developers Guide

- Events and Listeners
  - AWT/Swing-like



GWT - Developers Guide

- Remote Procedure Call



GWT - Show me the code! (server)

JavaOne

```
public interface Clock extends RemoteService {

    public String getCurrentTime();

}
```

```
public interface ClockAsync {

    public void getCurrentTime(AsyncCallback c);
}
```

```
public class ClockImpl extends RemoteServiceServlet implements Clock {

    public String getCurrentTime() {
        Date now= GregorianCalendar.getInstance().getTime();
        SimpleDateFormat ft = new SimpleDateFormat();
        ft.applyPattern("dd/MM/yyyy hh:mm:ss");
        return ft.format(now);
    }
```

```
<!-- URL Entry Point do servico remoto  -->
<servlet path='/clock' class='br.com.seatecnologia.gwt.teste.server.ClockImpl'/>
```

GWT - Show me the code! (client)

```
final Button button = new Button("Obter a hora atual");
final Label label = new Label();

// Proxy cliente
final ClockAsync clock = (ClockAsync) GWT.create(Clock.class);

// URL do servico
ServiceDefTarget endpoint = (ServiceDefTarget) clock;
String moduleRelativeURL = GWT.getModuleBaseURL() + "clock";
endpoint.setServiceEntryPoint(moduleRelativeURL);

// Callback
final AsyncCallback callback = new AsyncCallback() {
    public void onSuccess(Object result) {
        label.setText((String) result);
    }

    public void onFailure(Throwable caught) {
        label.setText("A hora atual nao pode ser obtida: " + caught);
    }
};

// Quando o botao for clicado, invocar o servico remoto
button.addClickListener(new ClickListener() {
    public void onClick(Widget sender) {
        // Invocacao do servico
        clock.getCurrentTime(callback);
    }
});

RootPanel.get().add(button);
RootPanel.get().add(label);
```

Web 2.0

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™ Technology with XML)
  - or Asynchronous JavaScript technology 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

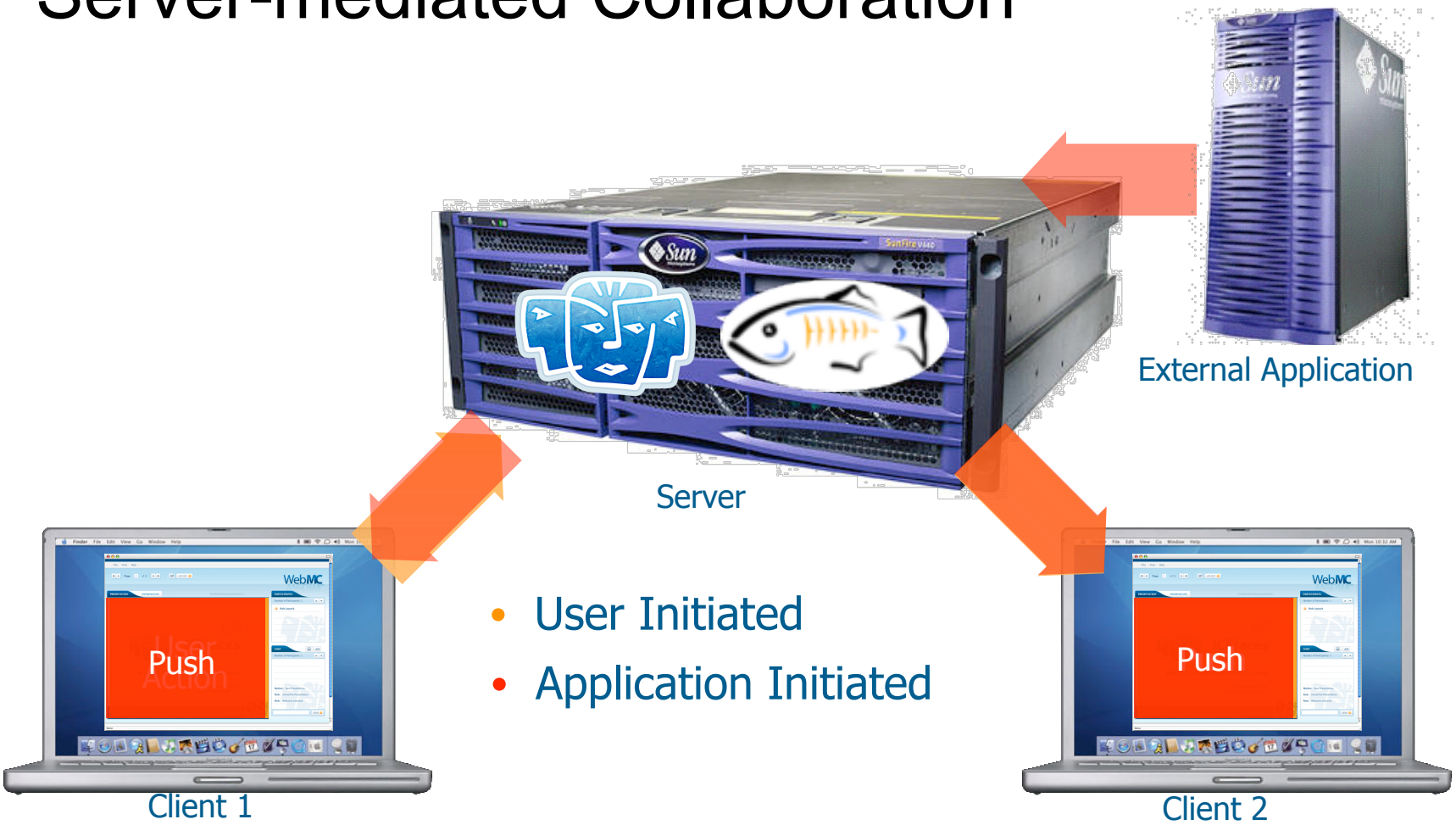
The Web enters the Participation Age.

- Ajax is 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



- User Initiated
- Application Initiated

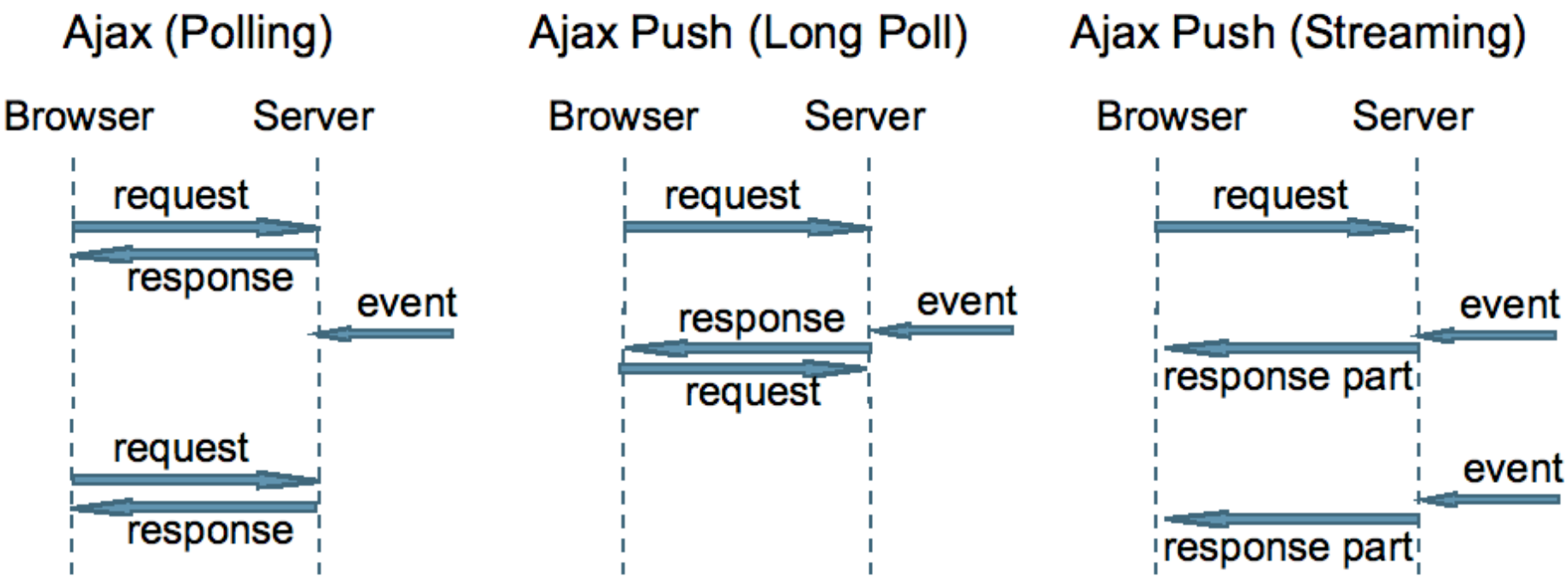
What is Ajax Push, exactly?

Responsive, low-latency interaction for the web.

- highly responsive, event driven browser applications
  - Keep clients up-to-date with data arriving or changing on the server, without frequent polling
- Pros
  - Lower latency, not dependent on polling frequency
  - Server and network do not have to deal with frequent polling requests to check for updates
- Example Applications
  - GMail and GTalk
  - Meebo
  - Many more …
  - 4homemedia.com (using GlassFish project's Comet)
  - JotLive
  - KnowNow

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.

How Push works

Keep an open connection.

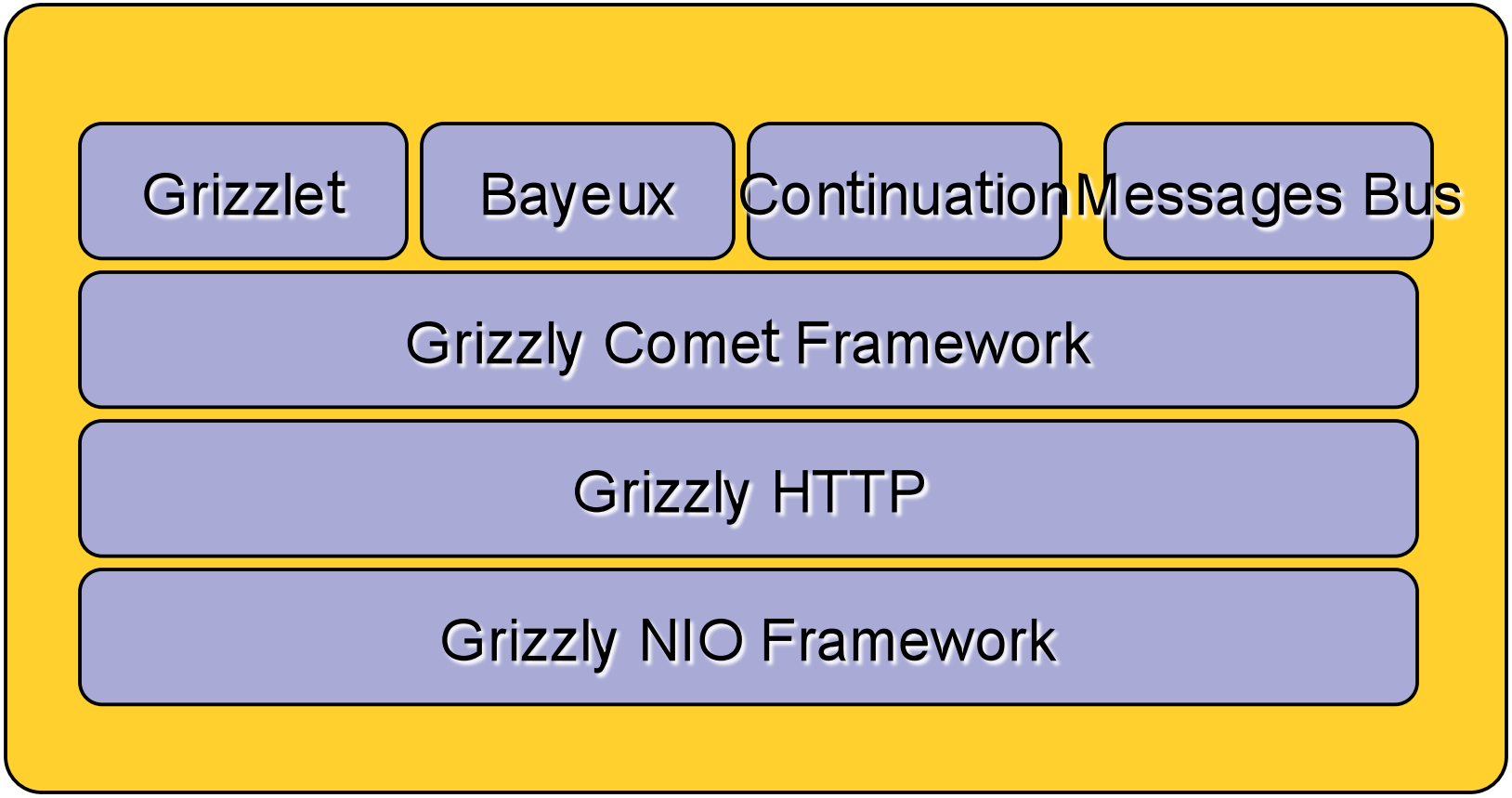
- Deliver data over a previously opened connection
- Always “keep a connection open”
  - do not respond to the initiating request until event occurs
- Streaming is an option
  - send response in multiple parts without closing the connection in between

ICEFaces Demo

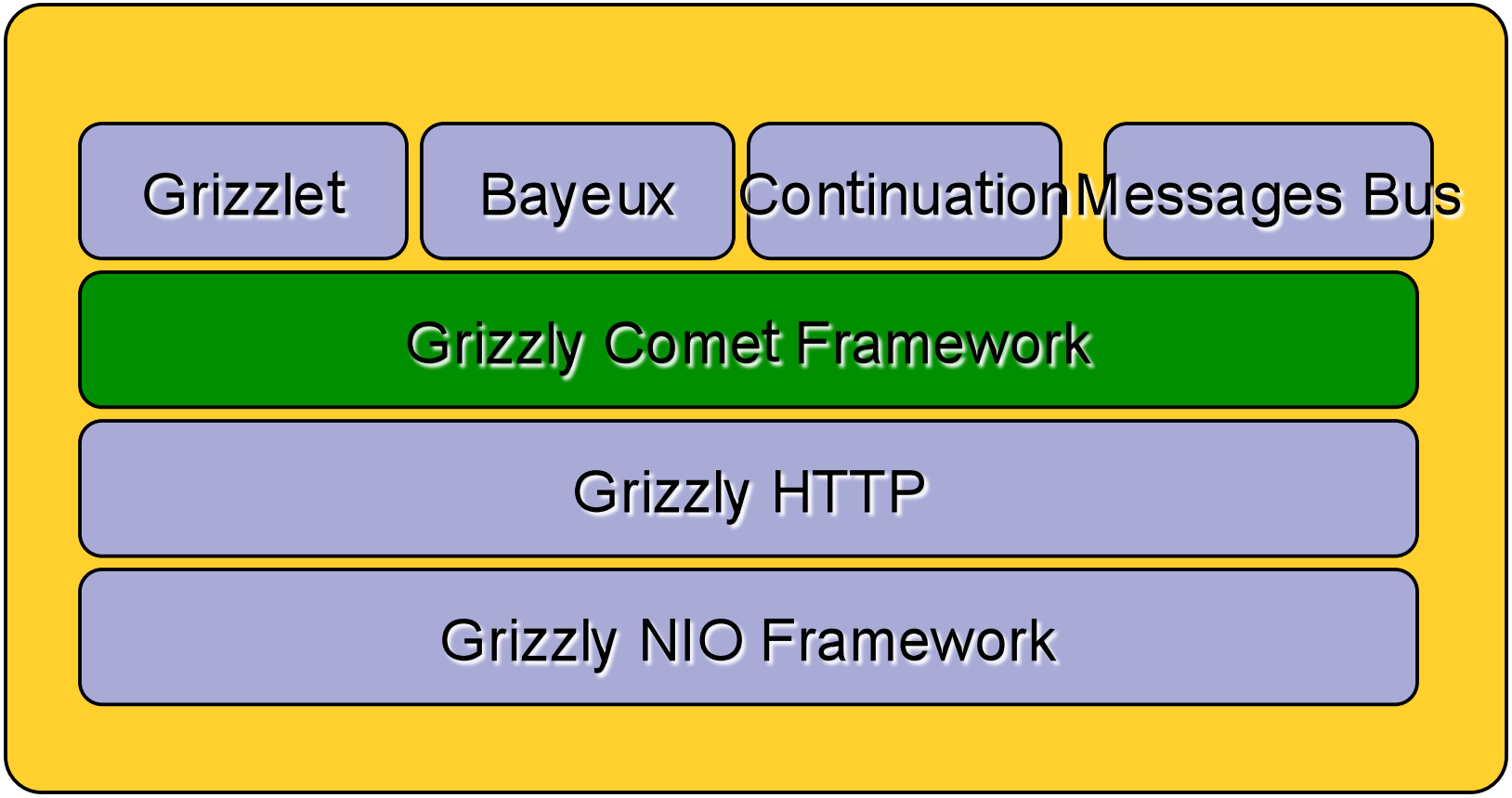
Introduction to Grizzly Comet

- Grizzly Comet is a framework that ship with GlassFish v1|2|3, and can also be embedded into any application using the Grizzly Embed interface (no fish involved).
  - In June, the code will be moved to a new project called Atmosphere (Atmosphere.dev.java.net)
  - Grizzly Comet running on *all* Containers supporting Comet.
- The Grizzly Comet Framework includes a set of components that can be used for building Comet based application:
  - Grizzly Comet, Continuation, Grizzlet, Messages Bus, Bayeux support

Grizzly Comet Components



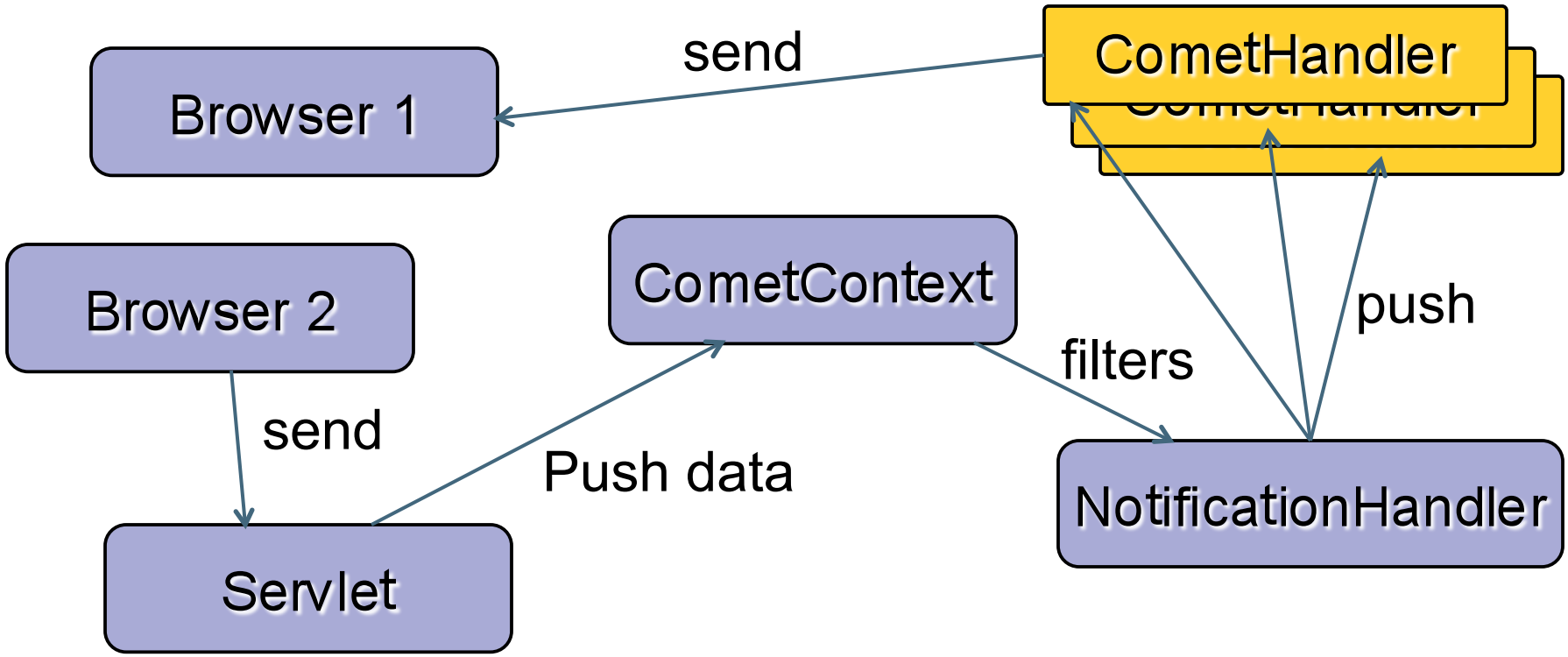
Grizzly Comet Components



Grizzly Comet Framework

- The Framework contains the classes required to add support for Comet in a Web Application
- Main classes to interact with (details next):
  - CometEngine
  - CometContext
  - CometHandler
  - NotificationHandler
  - CometReader
  - CometWriter

How it works


send
CometHandler
Browser 1
Browser 2
CometContext
push
filters
send
Push data
NotificationHandler
Servlet

CometContext

- A CometContext is a distribution mechanism for pushing messages that are delivered to multiple subscribers called CometHandler.
- All connections registered to a CometContext automatically becomes suspended, waiting for an event (a push) to happens.
- A browser receives only those messages published after the client “register “ to a CometContext.
- Its contains references to all suspended connections (encapsulated inside a CometHandler)

CometContext - Example

```
//Create a CometContext for my Chat application
CometContext chatContext
    = CometEngine.getEngine().register("chatroom");

// Suspend the request
ChatRoomHandler() chr = new ChatRoomHandler();
chatContext.addCometHandler(chr);

// Push welcome message
chatContext.push("Ted is entering the room");

 // Push bye bye message
chatContext.push("Alexandre is leaving the room");

// Later, resume the request
chatContext.resumeCometHandler(chr);
```

CometHandler

- The CometHandler is the master piece of a Grizzly Comet based application.
- A CometHandler contains the business logic of what will be pushed back to the browser.
- A CometHandler might be invoked by the Container:
  - When a push operation happens
  - When a I/O operations are ready to be process (asynchronous read or write)
  - When the browser close the connection.

CometHandler

- The CometHandler is the master piece of a Grizzly Comet based application.
- A CometHandler contains the business logic of what will be pushed back to the browser.
- A CometHandler might be invoked by the Container:
  - When a push operation happens
  - When a I/O operations are ready to be process (asynchronous read or write)
  - When the browser close the connection.

CometHandler API

```
//Invoked when CometContext.notify() is called
public void onEvent(CometEvent ce);

// Invoked when the browser close a suspended
// connection or when the suspend timeout expire.
public void onInterrupt(CometEvent ce);

// Invoked when the request is suspended
public void onInitialize(CometEvent ce);

// Attach an object
// most probably the HttpServletResponse}
public void attach(E e);
```

CometHandler Example

```
public void onEvent(CometEvent<String> ce){
    // The message send by CometContext.notify()
    // operations.
    String pushMessage = ce.attachment();

   // Flush the response back to the browser, and
   // keep the connection suspended.
    httpServletRequest.getWriter().write
                  ("window.parent.app.update(…..)");
 }
```

NotificationHandler

- The NotificationHandler object is the ***masterpiece*** when writing Comet application
  - This is inside that object that you will decide to what to do with the push operation:
    - Throttle: If too many push occurs simultaneously, should we delay them?
    - Aggregate: Should we cache push operations and aggregate them to avoid overloading the network?
    - Filter: Should all messages by pushed back to the client.
    - Should a thread pool be used to improve the push speed operation? Should a JMS backed be used to deliver the message?
- The DefaultNotificationHandler push all messages.

NotificationHandler - Example

```
   public void notify(CometEvent<String>
ce,List<CometHandler> l){
       // Escape any XSS scripting attack.
       escape(ce);

       // Do not push wrong statement like "Grizzly is slow"
       discard(ce);

   }
```

Asyncronous I/0 – Read and Write

- A CometHandler can be notified for asynchronous read and write operations
- Useful when reading or writing large chunk (like file upload.
- Bring NIO to Servlet indirectly ☺

```
public void onEvent(CometEvent<String> ce){
    if (ce.getType() == CometEvent.READ){

      }
 }
```

Grizzlet, Continuation, Messages Bus, Bayeux

- What to learn what are those ready to use components? Stay awake and come later tonight to see:

Using Comet to Create a Two-Player Web Game

20:30 - 21:20

Esplanade 307-310

Grizzly Comet & GWT

Three really simple steps

- Extends RemoteServiceServlet, register CometContext
- Implement CometHandler
- Implement RemoteService, invoke CometContext.notify()

Grizzly Comet & GWT

Extend RemoteServiceServlet

- First, create a Servlet that extends RemoteServiceServlet. Let's call it GrizzlyCometGWTServlet
- Inside the init(), register your CometContext.
- Inside the doGet(), creates CometHandler and add them to the CometContext
- By default, all GET request will be suspended.

Grizzly Comet & GWT

Extend RemoteServiceServlet

```
 // Create the CometContext associated with the
 // application
@Override
 public void init() throws ServletException {
     CometEngine ce = CometEngine.getEngine();
      cc = ce.register(”AuctionTopic");
      cc.setBlockingNotification(true);
      cc.setExpirationDelay(keepAliveTimeout);
  }
```

Grizzly Comet & GWT

Extend RemoteServiceServlet

```
// Suspend the connection
@Override
protected void doGet(HttpServletRequest request,
                     HttpServletResponse response)
                 throws ServletException, IOException {
    response.setContentType("text/
                      html;charset=ISO-8859-1");

    GWTCometHandler ch = new GWTCometHandler();
    ch.attach(response);
    cc.addCometHandler(ch);
}
```

Grizzly Comet & GWT

Create your CometHandler

```
private class GWTCometHandler implements
                    CometHandler<HttpServletResponse>{

  public void onEvent(CometEvent ce) throws IOException {
        GWTEvent event = (GWTEvent)ce.attachment();
        StringBuffer stream = new StringBuffer();
        writeCallback(stream,
                      event.queueName, event.message);

        writeToStream(res.getOutputStream(),
                      stream.toString());
        if (count++ > numberOfIteration){
           cc.resumeCometHandler(this);
        }
  }
```

Grizzly Comet & GWT

Link your RemoteService to our CometContext

```
// Update the connected client.
private void sendNewBid(AuctionItem item,
                        TextBox myBid, Label message) {

    ….

    cometService.updateClient(TOPIC, message);
}
```

Grizzly Comet & GWT

Link your RemoteService to our CometContext

```
// Update the connected client.
public void updateClient(String topic,
                         String message){
        try {
            CometEngine.getEngine()
                 .getCometContext("AuctionTopic")
                    .notify(
                       new GWTEvent(topic, message));
        } catch (IOException ex) {
        }
```

GWT Grizzly Comet Demo

Conclusion

- Writing GWT application is simple
- The Asynchronous Web will revolutionize human interaction
- Push can scale with Asynchronous Request Processing
- Adding Comet/Ajax Push support is even simple using Grizzly Comet.

THANK YOU

Writing Real-Time Web Applications, Using Google Web Toolkit and Comet

Alexandre Gomes - SEA Tecnologia

Jeanfrançois Arcand - Sun Microsystems

Ted Goddard - ICESoft

BOF-4922