DEVOXX
Guest Speaker
DEVOXX
Guest Speaker
DEVOXX
Guest Speaker

Adding Comet to your GWT application in less than 10 minutes!

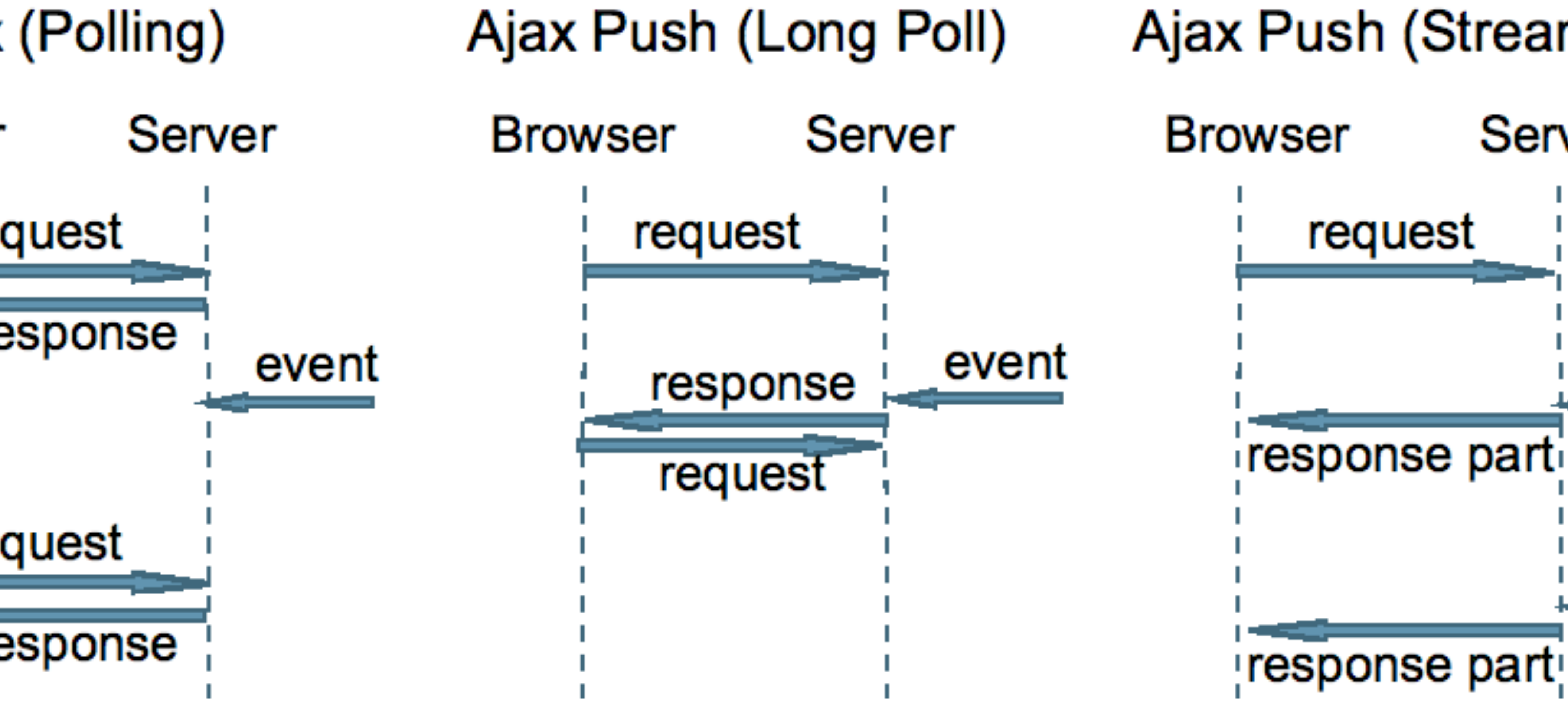
Jeanfrancois Arcand
Senior Staff Engineer
Sun Microsystems

Goal

Because Ajax-based applications are almost becoming the de facto echnology for designing web-based applications, it is more and more mportant that such applications react on the fly, or in real time, to bot client and server events.

This Quickie highlights how to build GWT-based applications that car take advantage of the Ajax push (a.k.a. Comet) technique to deliver real-time rich Internet applications (RIAs).

oll, Long Poll and Streamin



omet Techniques

end a request to the server every X seconds.

ne response is “empty” if there is no update.

g Poll:

end a request to the server, wait for an event to happen, then send the respon

ne response is never empty.

TTP specification satisfied: indistinguishable from “slow” server

Streaming:

end a request, wait for events, stream multi-part/chunked response, and then
r the events.

ne response is continually appended to.

izzly Comet Framework

e Framework contains the classes required to add pport for Comet in a Web Application

ain classes to interact with (details next):

ometEngine

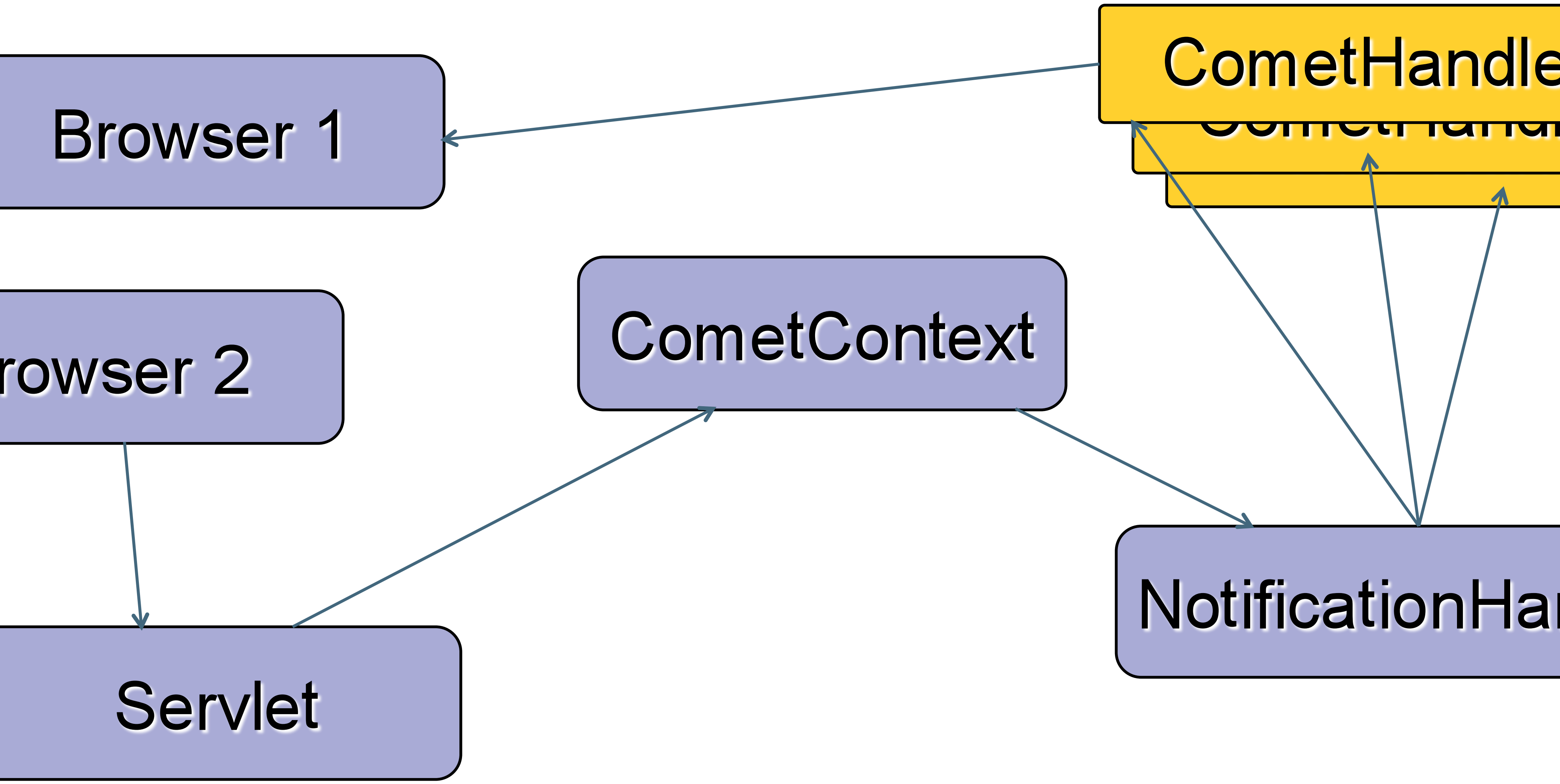
ometContext

ometHandler

otificationHandler

ometReader

ow it works


Browser 1
rowser 2
Servlet
CometContext
CometHandle
NotificationHa

ometContext

A CometContext is a distribution mechanis 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

ometHandler

e CometHandler is the master piece of a Grizzly Con
sed application.

CometHandler contains the business logic of what w
 pushed back to the browser.

CometHandler might be invoked by the Container:

/hen a push operation happens

/hen a I/O operations are ready to be process
synchronous read or write)

/hen the browser close the connection.

ometHandler

e CometHandler is the master piece of a Grizzly Con
sed application.

CometHandler contains the business logic of what w
 pushed back to the browser.

CometHandler might be invoked by the Container:

'hen a push operation happens

'hen a I/O operations are ready to be process
synchronous read or write)

'hen the browser close the connection.

ree really simple steps

Extends RemoteServiceServlet, register CometContext

Implement CometHandler

Implement RemoteService, invoke CometContext.notify()

Grizzly Comet & GWT

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.

Next, create another RemoteServiceServlet which implement your RemoteService

That's it!!!

tend RemoteServiceServle

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

nd RemoteServiceServlet

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

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

eate your CometHandler

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

public void onEvent(CometEvent ce) throws IOExcepti

      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);
```

nk your RemoteService

```
 Update the connected client.
ivate void sendNewBid(AuctionItem item,
                               TextBox myBid, Label message

    ….

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

metContext

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

EMO

/T Auction Demo

Conclusion

Writing GWT application is simple
The Asynchronous Web will revolutionize human
nteraction
Adding Comet/Ajax Push support is even simple
sing Grizzly Comet.

Q&A

Follow us on **http://twitter.com/project_grizzly**

Getting Started with GlassFish and Comet

http://grizzly.dev.java.net

http://weblogs.java.net/blog/jfarcand/archive/
2008/11/writing_a_twitt.html

Project Grizzly mailing lists,

dev@grizzly.dev.java.net &
users@dev.grizzly.java.net

My blog: http://weblogs.java.net/jfarcand