



What the Bayeux?

Filip Hanik

SpringSource Inc

Keystone, Colorado, 2008





Who am I?

- Breathing air since 1975
- Tomcat user since 1999
- Tomcat developer since 2001
- Happily married since 2003
- Proud father since 2006
- Coloradan since 2007
- Speaker at this Summit since 2008



What we will cover

- Introduction to AJAX and Comet
 - AJAX yesterday, today and tomorrow
 - Polling methods
 - Client support

- The Bayeux Protocol
 - Introduction to JSON
 - Publish/Subscribe



What we will cover

- The Bayeux Protocol
 - Message exchange

- Bayeux clients
 - Server side
 - Client side/browser



What we will cover

- Tomcat Bayeux API
 - Server side framework for bayeux web applications
- Implementing a simple app
- What now, buzz or bull?



Introduction to AJAX

- What is AJAX
 - Aynchronous Javascript and XML
 - Web development technique
 - Term coined in 2005
 - Technique originated from Microsoft
 - 1996 iframe
 - 1999 XMLHttpRequest object



Introduction to AJAX

- AJAX on the client side
 - Javascript running in browser
 - Making HTTP requests in the background
 - Ability to update without refresh
 - Must be tailored to the browser
 - Lots of frameworks out there

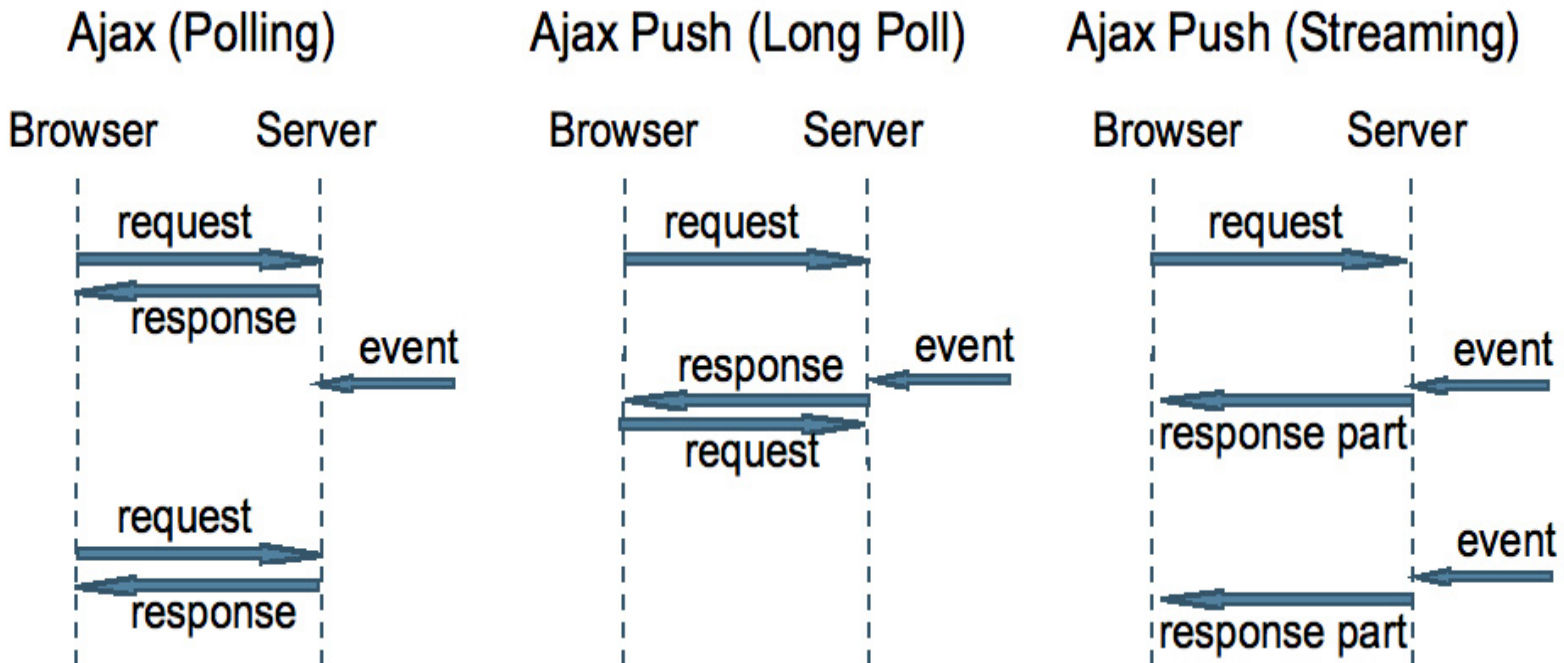


Introduction to AJAX

- AJAX Today
 - Has become a common solution
 - Used by more and more large sites
- AJAX on the server side
 - Scalability has been an issue
 - Polling is expensive
 - Works best when server can do asynchronous processing
 - Tomcat Comet is such asynchronous

Introduction to AJAX

- Polling and how it works





Introduction to AJAX

- Client support
 - Various Javascript frameworks
 - Communication with browser is still achieved with

BIG TIME HACKS!!!



The Bayeux Protocol

- Publish Subscribe Model
 - A JSON based publish/subscribe protocol
 - Development lead by Dojo Foundation
 - Approach is similar to JMS topics
 - Still in somewhat of a trial stage
 - Idea is to have someone else take over the specification



The Bayeux Protocol

- Publish Subscribe Model
 - Client and Server side frameworks
 - Remove complexity of cometsd/ajax from web developers
 - Instead of pub/sub, one can think of it as listening for events



The Bayeux Protocol

- Introduction to JSON
 - JavaScript Object Notation
 - Portable serialization format
 - Not a markup language – no tags
 - Fast serialization and deserialization



The Bayeux Protocol

- Introduction to JSON

```
{  
  "firstName": "John",  
  "lastName": "Smith",  
  "address":  
  {  
    "streetAddress": "21 2nd Street",  
    "city": "New York",  
    "state": "NY",  
    "postalCode": 10021  
  },  
}
```



The Bayeux Protocol

- Introduction to JSON
 - JSON compared to XML?
 - Long (and war like) debate
 - JSON doesn't have tags
 - Much less verbose
 - Parsing libraries much smaller and more efficient
 - JSON is JavaScript! No parsing needed!
 - XML with AJAX has been said to be slow.



The Bayeux Protocol

- Introduction to JSON
 - Cyclic references are supported in custom format
 - <http://www.sitepen.com/blog/2008/06/17/json-referencing-in-dojoo/>



The Bayeux Protocol

- Message exchanges
 - All message exchange is done using JSON
 - Very simple
 - Establish client
 - Subscribe to channel
 - Publish events
 - Receive events
 - Two connection operation (optional)
 - Allows send and receive at the same time



The Bayeux Protocol

- Message exchanges
 - Content type for messaging is
 - text/json
 - text/json-comment-filtered
 - Comment filtered
 - JSON message encapsulated in script comments `/* ... */`
 - Prevents AJAX hi-jacking



The Bayeux Protocol

- Message exchanges
 - Client establishment is done using handshake

```
[ {  
  "channel": "/meta/handshake",  
  "version": "1.0",  
  "minimumVersion":  
  "1.0beta",  
  "supportedConnectionTypes":  
    [ "long-polling",  
      "callback-polling",  
      "iframe" ]  
}]
```



The Bayeux Protocol

- Message exchanges
 - Client establishment is followed by a connect request

```
[ {  
  "channel": "/meta/connect",  
  "clientId": "Un1q31d3nt1f13r",  
  "connectionType": "long-polling"  
}]
```

The Bayeux Protocol

- Message exchanges
 - To disconnect, simply send disconnect message

```
[ {  
  "channel": "/meta/disconnect",  
  "clientId": "Un1q31d3nt1f13r",  
}]
```

- Server will also have some sort of timeout in case disconnect message is not received
 - Similar to HTTP sessions



The Bayeux Protocol

- Message exchanges
 - Channel subscription is easy

```
[ {  
  "channel": "/meta/subscribe",  
  "clientId": "Un1q31d3nt1f13r",  
  "subscription": "/foo/**"  
} ]
```

- Wild card patterns are supported

The Bayeux Protocol

- Message exchanges
 - Unsubscribing is equally

```
[ {  
    "channel": "/meta/unsubscribe",  
    "clientId": "Un1q31d3nt1f13r",  
    "subscription": "/foo/individual-  
channel"  
} ]
```

- Wild card patterns are supported



The Bayeux Protocol

- Message exchanges
 - meta channels are used to negotiate between client and server
 - The only other exchange is sending and receiving events (data)

The Bayeux Protocol

- Message exchanges
 - Unsubscribing is equally simple

```
[ {  
  "channel": "/meta/unsubscribe",  
  "clientId": "Un1q31d3nt1f13r",  
  "subscription": "/foo/some-channel"  
} ]
```

- Wild card patterns are supported



The Bayeux Protocol

- Message exchanges
 - Messages are simple key value pair objects

```
public interface Message  
    extends Map<String, Object>
```



Bayeux actors

- Clients
 - Server side and client side
 - Subscribe to channel
 - Publish and receive events from channels

- Browser side clients
 - Only implementation is the Dojo Toolkit
 - Have to handshake
 - Supports different polling methods



Bayeux actors

- Server side clients (Java)
 - Implemented in several platforms
 - Tomcat, Jetty, Glassfish
 - All three use different server side API
 - Dojo Foundation has been the hinder for a common Java API
 - Lack of process around infrastructure
 - Lack for process around community development



Bayeux actors

- Server side clients (other)
 - Effort has been put in place to add APIs in other languages
 - Perl
 - Python
 - .NET



Tomcat Bayeux API

- Server Side API
- Goal is to reduce complexity
- Derived from the original Dojo Java API
 - Spaghetti references removed
 - Redundant/ambiguous API removed
 - More object oriented, instead of converting from string-to-object and object-to-string over and over again



Tomcat Bayeux API

- API found at

`org.apache.cometd.bayeux`

- Implementation found at

`org.apache.tomcat.bayeux`

- Built on top of Tomcat's CometProcessor



Tomcat Bayeux API

- Configured through web.xml

```
<servlet>
  <servlet-name>cometd</servlet-name>
  <servlet-class>
    org.apache.tomcat.bayeux.BayeuxServlet
  </servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>cometd</servlet-name>
  <url-pattern>/cometd/*</url-pattern>
</servlet-mapping>
```

Tomcat Bayeux API

- Create a client

```
Bayeux bayeux = ServletContext.getAttribute(  
    "Bayeux.DOJOX_COMETD_BAYEUX");
```

```
Client client =  
    bayex.newClient("client-id", callback);
```

- Callback is an implementation of the `org.apache.cometd.bayeux.Listener` interface



Tomcat Bayeux API

- Subscribe to a channel

```
Channel channel =  
    bayeux.getChannel("/chat/demo", true);
```

```
channel.subscribe(client);
```



Tomcat Bayeux API

- Send a message

```
Message msg = bayeux.newMessage(client);
```

➤ Client is the “sender”

```
channel.publish(msg);
```

➤ Puts the message into the queue for all subscribed clients

Tomcat Bayeux API

- Send a message

```
Message msg = bayeux.newMessage(client);
```

➤ Client is the “sender”

```
channel.publish(msg);
```

➤ Puts the message into the queue for all subscribed clients

Tomcat Bayeux API

- Receive a message

```
public void deliver(Message[] msgs) {
```

- Messages can come in batches

- You can reply directly to a client

```
Client sender = msgs[i].getClient();  
Message reply = bayeux.newMessage(...);  
sender.deliver(reply);
```



Tomcat Bayeux API

- Building Bayeux

```
svn  
  co  
  http://svn.apache.org/repos/asf/tomcat/trunk  
  tctrunk
```

```
cd tctrunk
```

Tomcat Bayeux API

- Building Bayeux

```
ant download
```

```
ant
```

- Tomcat has now been built, build the Bayeux extensions

```
ant -f extras.xml bayeux
```

- output/extras contains JARs and sample WAR



Tomcat Bayeux API

- Simple API
- Built using Tomcat's CometProcessor
 - Scalable
 - No thread per connection limit
 - Requires NIO or APR connectors



Dojo Toolkit

- Using the Dojo Toolkit

```
dojo.require("dojox.cometd");
```

```
dojox.cometd.init("/cometd/cometd");
```

➤ This is the URL of your Bayeux servlet

A graphic consisting of overlapping colored squares (yellow, red, blue) and a black crosshair.

Dojo Toolkit

- Using the Dojo Toolkit

```
dojox.cometd.subscribe("/chat/demo", onMsgEvent);
```

```
var evt =  
  { 'data': { 'msg': trim(message) + '|' + msg } };
```

```
dojox.cometd.publish("/chat/demo", evt.data);
```

- We've subscribed to a channel and sent a message
 - All complexity is behind the scenes



Buzz or Bull?

- A little bit of both
- Dojo lacks some of policy, process and infrastructure that ASF has
 - Good at building user community
 - Harder to build development community
- Client side still focuses on the AJAX hacks



Buzz or Bull?

- Once something like 'WebSockets' come in HTML/JavaScript
 - We can probably expect to see more protocols and frameworks
- IMHO – Bayeux is still early, it provides some nice features
 - But we lack more client frameworks
 - And server API's vary a lot



And we're done

- Thank you!
- Questions and hopefully Answers
- filip.hanik@springsource.com