Real-time Web and Web Applications

Misc Slides #1

Real-time Web

"The real-time web is a set of technologies and practices that enable users to receive information **as soon** as it is published by its authors, rather than requiring that they or their software check a source periodically for updates [polling]"

//Wikipedia

This is **soft** real time (**as soon** ~ in a few seconds)

A Few Real-time Links

For the interested

http://blog.programmableweb.com/2011/03/17/real-time-web-or-right-time-web/

http://www.readwriteweb.com/archives/ten_useful_examples_of_the_real-time_web_in_action.php (Inspiring)

http://www.leggetter.co.uk/2012/05/31/talking-realtime-at-fluentconf.html (Ericsson-video comparing Web Sockets and Long polling, a bit down)

Real time search is **very hot** right now!!

The Web-Chat-Application Problem

In a chat application users need to be notified "as soon" as possible. No support in HTTP for notifications!

"Until now" solutions

- Let client **poll**, periodically request (using JavaScript/AJAX)
    - That's how feed-readers work (retrieving XML data at certain frequency (RSS, Atom format))
    - Normally bad...
- Comet (an umbrella specification)
    - Aka Long-polling, AJAX-push, reverse AJAX, ...
    - Somewhat "hackish" tweaking of HTTP using HTTP persistent connections, hidden HTML iframe's, sending "forever frames" **...**
    - **http://www.ibm.com/developerworks/websphere/techjournal/0711_col_burckart/0711_col_burckart.html**

New Solutions

"HTML5 area" provide new solutions to the Web-Chat-Application problem.

Server Sent Events (SSE) a W3C standard

WebSocket, another W3C standard

HTTP Push ~ HTTP Streaming ~ ... (many words, ... lack of definitions)

Websocket

Full duplex, single TCP-communication channel between client and server (like a "real" client/server application)

Initial HTTP-handshake then switch to websocket protocol (ws://.. and wss://)
- Very many protocol version; hixie-75, ... , hybi-00, ..., hybi10, RCF6455
- Last (final?) RCF6455 only supported by **latest version** of browsers

JavaScript WebSocket interface
http://www.w3.org/TR/websockets/#the-websocket-interface

Websocket JS Example

```
function connect(){    // Compact but bad style
    try{
    var socket;
    var host = "ws://localhost:8000/socket/server/...";
    var socket = new WebSocket(host);
        message('<p class="event">Socket Status: '+socket.readyState);
        socket.onopen = function(){
             message('<p class="event">Socket Status: '+socket.readyState+' (open)');
        }
        socket.onmessage = function(msg){
             message('<p class="message">Received: '+msg.data);
        }
        socket.onclose = function(){
             message('<p class="event">Socket Status: '+socket.readyState+' (Closed)');
        }
    } catch(exception){
         message('<p>Error'+exception);
    }  }
```

Server Sent Events

Push DOM-events from server to client (similar to AJAX but from other side), ... one way

Just plain old HTTP (others possible)
MIME: text/event-stream

Not supported by IE, other's browsers: late versions

JavaScript SSE interface
http://dev.w3.org/html5/eventsource/#the-eventsource-interface

SSE JS Example

```
if (!window.EventSource) { // Check if SSE supported, should do same with WebSocket
  var source = new EventSource('stream.php');  // Client registers as event handler
} else {
  //... bad luck...
}

// Bad style
source.addEventListener('message', function(e) {
  console.log(e.data);
}, false);
source.addEventListener('open', function(e) {
  // Connection was opened.
}, false);
source.addEventListener('error', function(e) {
  if (e.readyState == EventSource.CLOSED) {
    // Connection was closed.
  }
}, false);
```

Server Side

Client side no problem (iieee… if browser supports…)

Server side worse for now (in course)

GF 3.x

- **ICEFaces Push**, part of ICEFaces (JSF component library)
- **Atmosphere framework**
- Note: Must enable Comet and Websocket support in GlassFish (via NetBeans), see code samples (README file)

GF 4

- Supports Websockets! So no problems next year…

ICEFaces Push

Part of ICEFaces. Dependency on ice-push library, see code samples

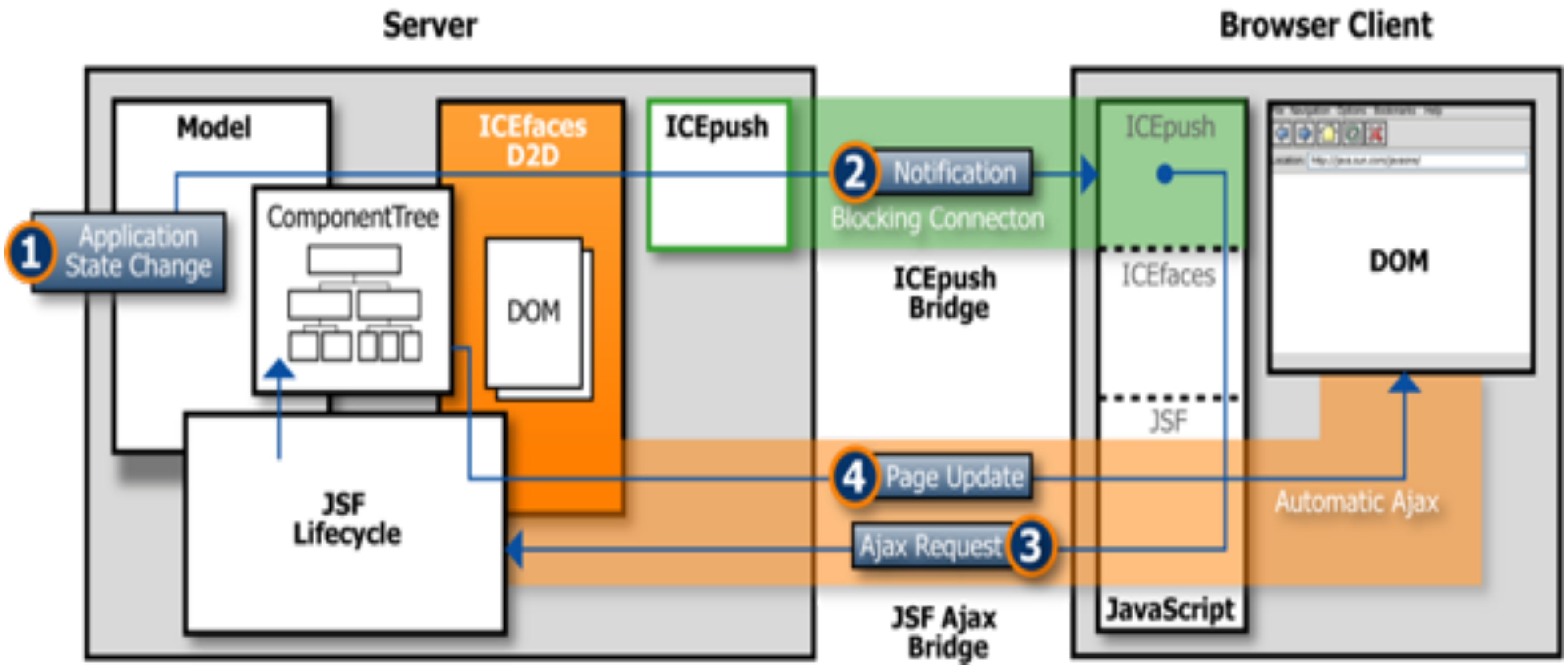
Will provide PushRender class

```
// Register for push
// The FacesContext object must be accesible for
// the PushRenderer (i.e JSF must be active (FacesContext used
// internally by PushRenderer)
String PUSH_GROUP = "...anyPreferablyUniqueString...";
PushRenderer.addCurrentSession(PUSH_GROUP);

// Do the push
PushRenderer.render(PUSH_GROUP);
```

ICEFaces Push Internals

Complex... (not visible to programmer)



Atmosphere

"The Atmosphere Framework contains client and server side components for building Asynchronous Web Application. The majority of popular frameworks are either supporting Atmosphere or supported natively by the framework. The Atmosphere Framework supports all majors Browsers and Servers"...

//Home Page...hmm, believe when I see...

"Atmosphere transparently supports WebSockets, Server Side Events (SSE), Long-Polling, HTTP Streaming (Forever frame) and JSONP"

//Home Page

Standard fallback to Long Polling

Atmosphere Design

Client Side JavaScript API (plugin to JQuery)
- Hiding native JS API's
- $.atmosphere

```
var socket = $.atmosphere;
var request = new jQuery.atmopshere.AtmosphereRequest();
var subSocket = socket.subscribe(request);
subSocket.push(data);
```

Server Side a few different API's
- AtmosphereHandler API
- Jersey API (RESTful)
- ...

Very many samples, see link course page...

Atmosphere: RESTful Server Side

```
// Also configuration in web.xml
@Path("/")
public class ResourceChat {

     //Suspend the response without writing anything back to the client.
    @Suspend(contentType = "application/json")
    @GET
    public String suspend() {
        return "";
    }

    //Broadcast the received message object to all suspended response.
    //Do not write back the message to the calling connection.
    @Broadcast(writeEntity = false)
    @POST
    @Produces("application/json")
    public Response broadcast(Message message) {
        return new Response(message.author, message.message); //utility class,
not JAX-RS
    }
}
```

Web Applications

Previously Rich Internet Application (RIA)

Application that behaves like desktop application:
Google Docs

Many new JS API's to support

- Seen a few...
- Need for offline, file, drag drop, ...
- Link to possible useful API's on course page...

… and more …

Other possible interesting things not covered in course

- Hadop-Map Reduce (enormous data sets)
- Heroku http://www.heroku.com/
- Netty https://netty.io/
- Node.js http://nodejs.org/
- …. much much more…