Filling The Gap: Going Mobile With JBoss Technologies Today

Jay Balunas
Principal Software Engineer
JBoss, by Red Hat Inc.

Wesley Hales
Senior Software Engineer
JBoss, by Red Hat Inc.
Jay Balunas

- RichFaces Project Lead
- JBoss Core Developer
- JSF 2.2 Expert Group
- Red Hat W3C Member
- http://in.relation.to/Bloggers/Jay
- http://twitter.com/tech4j
- jbalunas@jboss.org
Wesley Hales

• JBoss Portlet Bridge Lead
• Core GateIn Developer
• JSR–301/329 Red Hat Rep.
• Committer on Mozilla, Apache, Richfaces....
• Multiple series on InfoQ, Dzone, Refcards, personal blog, etc...

• [www.wesleyhales.com](http://www.wesleyhales.com)
• @wesleyhales
Check It Out For Yourself

Wireless SSID:

tweetstream

Local URL:

http://192.168.0.101:8080/tweetstream/

Matching Tweets Tags

#jboss #JBW #JUDCon
TweetStream

- Built to support mobile webkit devices
- No 3rd party front end frameworks
- Completely driven by JBoss tech (from front to back)
TweetStream

- Built to support mobile webkit devices
- No 3rd party front end frameworks
- Completely driven by JBoss tech (from front to back)
TweetStream

- Built to support mobile webkit devices
- No 3rd party front end frameworks
- Completely driven by JBoss tech (from front to back)
TweetStream Breakdown

• Filling The Gap...

- RichFaces
- Weld
- Infinispan
- Seam
- HornetQ
TweetStream Architectural Overview

**TwitterSource** gets JUDCon tweet history based on "tracks", on application start.

**CacheBuilder** stores different key value sets into Infinispan.

**TweetListenerBean** listens for real time tweets based on hashtag. Triggers refresh of **CacheBuilder**.

**CacheListener** listens for Infinispan events and publishes refreshed view.

**a4j:push** subscribes to topic and gets updates as needed.
Tweetstream
Architecture Details
TweetStream Twitter4j

- Open source project
- Java API for easy access
- OAuth support
- Streaming feeds
- Historical feeds
TweetStream Infinispan

- JBoss open source
- Highly scalable datagrid
- In memory & disk options
- Easily clusters out
- Management facilities
TweetStream HornetQ & RichFaces

- JBoss open source
- Reliable, high performant
- Flexible clustering
- POJO based design
- Perfect for RichFaces Push!

Tuesday, May 3, 2011
From Standard To Mobile Web
From standard to mobile web
From standard to mobile web
From standard to mobile web
Considerations

• Device and feature detection
• Using RichFaces push effectively
• HTML 5 & CSS3
• Touch & gesture support
• Optimizations & tweaks
Device Detection
Header Inspection

HTTP_HOST == myproxylists.com
HTTP_USER_AGENT == Mozilla/5.0 (iPad; U; CPU OS 4_3_2 like Mac OS X; en-us) AppleWebKit/533.17.9 (KHTML like Gecko) Version/5.0.2 Mobile/8H7 Safari/6533.18.5
HTTP_ACCEPT == application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*;q=0.5
HTTP_ACCEPT_LANGUAGE == en-us
HTTP_ACCEPT_ENCODING == gzip, deflate
HTTP_CONNECTION == keep-alive
REMOTE_ADDR == xx.xxx.x.xxx
REMOTE_PORT == 53298
REQUEST_METHOD == GET
REQUEST_URI == /my-http-headers
REQUEST_TIME == 1304015649
REMOTE_HOST == pool-xx-xxx-x-xxx.fios.verizon.net
COUNTRY == [US] UNITED STATES
Help Is Out There

- MobileESP
- WURFL APIs

Wireless Universal Resource File

Single Class User-Agent Processor
For Tweetstream

- MobileESP
- WURFL APIs

Wireless Universal Resource File

Single Class User-Agent Processor
@Named("userAgent")
@RequestScoped
class UserAgentProcessor
{
    @PostConstruct
    public void init()
    {
        FacesContext context = FacesContext.getCurrentInstance();
        HttpServletRequest request = (HttpServletRequest)context.getExternalContext().getRequest();
        userAgentStr = request.getHeader("user-agent");
        httpAccept = request.getHeader("Accept");
        uAgentTest = new UAgentInfo(userAgentStr, httpAccept);
    }

    public boolean isPhone()
    {
        //Detects a whole tier of phones that support similar functionality as the iphone
        return uAgentTest.detectTierIphone();
    }

    public boolean isTablet()
    {
        //Detect ipads, xooms, blackberry tablets, but not galaxy - they use a strange user-agent
        return uAgentTest.detectTierTablet();
    }
}
Single Point of Entry

<c:choose>
  <c:when test="#{userAgent.phone}"/>
  <ui:include src="tabletHome.xhtml"/>
  <c:otherwise>/n
    <!--ui:include src="desktopHome.xhtml"/>
    <ui:include src="tabletHome.xhtml"/>
  </c:otherwise>
</c:choose>

Detection Made By Form-Factor
Feature Detection

- WURFL (server-side)
- Modernizr (client-side)
- CSS 3 & HTML 5
For TweetStream

- WURFL (server-side)
- Modernizr (client-side)
- CSS 3 & HTML 5
/* catch tablet portrait orientation */
@media all and (orientation: portrait) {}

/* catch tablet landscape orientation */
@media all and (orientation: landscape) {}

/* iphone landscape screen width */
@media screen and (max-device-width: 480px) and (orientation: landscape) {}

/* iphone portrait screen width */
@media screen and (max-device-width: 320px) and (orientation: portrait) {}/
Modernizr Example

Simple JavaScript File to load

```
.multiplebgs div p {
   /* properties for browsers that
      support multiple backgrounds */
}
.no-multiplebgs div p {
   /* optional fallback properties
      for browsers that don't */
}
```
RichFaces Push
RichFaces Push

- Integrates with Atmosphere
  - Comet/WebSockets
  - Graceful degradation

- Java Messaging Service (JMS)
  - Reliability, flexibility, etc....
Getting The Message Out

• RichFaces TopicsContext
• Direct JMS
For TweetStream

- RichFaces TopicsContext
- Direct JMS
- CDI Events*
try {
    log.debug("Pushing Update notification, no data needed");
    TopicsContext.lookup().publish(new TopicKey("twitter", "content"), null);
} catch (Exception e) {
    log.error(e.getMessage(), e);
}
//Standard JMS connection setup not shown

```java
topicConnection = cf.createConnection("guest", "guest");
Topic topic = (Topic) ic.lookup("/topic/twitter");
```

```java

```
message = session.createTextMessage();
//Publishing update notification, no data needed
message.setText(null);

//Set the subtopic for the message to "content"
message.setStringProperty("rf_push_subtopic", "content");

producer.send(message);
```
Client Side Push Options

- Direct DOM Updates
- Partial Page Rendering
For TweetStream

- Direct DOM Updates
- Partial Page Rendering
Partial Page Rendering

Topic & Sub-Topic

```xml
<a4j:push address="content@twitter"
  onerror="alert('Error processing push')">  
  <a4j:ajax event="dataavailable"
    render="tweeters tops"
    execute="@none"/>
</a4j:push>
```

Event Triggered

What to Re-Render

Tuesday, May 3, 2011
DOM Updates With Data Push

```
<a4j:push address="tweets@twitter"
onerror="alert(event.rf.data)"
ondataavailable="processData(event.rf.data)"/>
```

- JMS Message Types
  - ObjectMessage (JSON)
  - TextMessage (String)

- Direct DOM processing in processData function

```
event.rf.data on client side
```

Tuesday, May 3, 2011
Using JSF for Mobile Dev

• Sacrifices, sacrifices...
• Mobile HTML5 apps live or die by the UI
• Making components work with touch events
Understanding The Problems

- We know today’s components are built for desktop
- What exists for mobile JSF today?
- The future
User Interface/Experience

- CSS3 transitions
- Orientation Detection / Media Queries
- Page layout approach
- Understanding Touch Events and Gestures
- Performance and Tweaks
User Interaction

@media screen and (max-device-width: 320px) and (orientation:portrait) {
  .stage-left {
    opacity: 0;
    left: -320px;
  }
  .stage-right {
    opacity: 0;
    left: 320px;
  }
  .transition {
    -moz-transition-duration: 1s;
  }
}
User Interaction

```html
<div id="page-container">
  <div class="iphone-page" id="home-page"></div>
  <div class="iphone-page stage-right hide" id="tweet-detail-page"></div>
  <div class="iphone-page stage-right hide" id="top-tweeters-page"></div>
  <div class="iphone-page stage-right hide" id="top-hashtags-page"></div>
  <div class="iphone-page hide" id="about-page"></div>
</div>

function swap(id, classString) {
  document.getElementById(id).className = classString;
}
```
User Interaction

```javascript
function swap(id, classString) {
    document.getElementById(id).className = classString;
}
```
UX / Orientation Detection

- CSS3 Media Queries
  @media screen and (max-device-width: 320px) and (orientation:portrait) {

- Viewport Settings
  <meta name="viewport" content="width=device-width, minimum-scale=1.0, maximum-scale=1.0"/>
Using Touch Events and Gestures

- Duck punch approach
- onmousedown / onmousemove measurements for swipe
Network Detection (WiFi, 3G, etc..)

- `navigator.connection` (Android 2.2+)

```javascript
// contents of navigator.connection object
{
    "type": "3",
    "UNKNOWN": "0",
    "ETHERNET": "1",
    "WIFI": "2",
    "CELL_2G": "3",
    "CELL_3G": "4"
}
```
Tweaks and Optimization

- Getting close to a native feel...
- Start with chrome or firebug profiling (SpeedTracer, yslow, etc...)
- Access the GPU for smooth transitions:
  - webkit-transform: translateZ(0);
Other Helper Frameworks

- SenchaTouch
- JQuery*
- Phone Gap
- ...

* denotes a specific framework or tool
Questions?

The Source:

https://github.com/richfaces/tweetstream
Page Layout / Optimization

• Reducing Latency over cell networks:
  • One huge page o’ markup = less file requests
  • Dealing with external file requests
  • What about client side caching?
  • Paving the way for future components...
Reducing Latency

![Image](image.png)

<table>
<thead>
<tr>
<th>Name</th>
<th>Size Transfer</th>
<th>Time Latency</th>
<th>Timeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>android_normal.png</td>
<td>2.65KB, 223B</td>
<td>1.05s</td>
<td>597ms</td>
</tr>
<tr>
<td>home.jsf</td>
<td>41.23KB, 41.49KB</td>
<td>287ms, 275ms</td>
<td>895ms</td>
</tr>
<tr>
<td>lavoroinrete_normal.png</td>
<td>4.02KB, 223B</td>
<td>49ms</td>
<td>1.19s</td>
</tr>
<tr>
<td>default_profile_2.png</td>
<td>1.35KB, 149B</td>
<td>49ms</td>
<td>1.49s</td>
</tr>
<tr>
<td>JobHitsIT64x64.png</td>
<td>1.62KB, 149B</td>
<td>48ms</td>
<td></td>
</tr>
</tbody>
</table>
Prefetching / Proxy

• One possible solution would be:
  • Base64 images to javascript using canvas
  • Infinispan Storage
// Base64 an image
function getBase64Image(img) {
    // Create an empty canvas element
    var canvas = document.createElement("canvas");
    canvas.width = img.width;
    canvas.height = img.height;

    // Copy the image contents to the canvas
    var ctx = canvas.getContext("2d");
    ctx.drawImage(img, 0, 0);

    // Get the data-URL formatted image
    // Firefox supports PNG and JPEG. You could check img.src to guess the
    // original format, but be aware the using "image/jpg" will re-encode
    // the image.
    var dataURL = canvas.toDataURL("image/png");

    return dataURL.replace(/\^data:image\/(png|jpg);base64,/, "");
}
Client Side Caching

- What about HTML5...
  - localStorage
  - sessionStorage
  - cacheManifest
TweetStream / JSF Improvements

Network Utilization

- Combine external JavaScript (4)
  There are multiple resources served from same domain. Consider combining them into as few files as possible.
  4 JavaScript resources served from localhost.

- Enable gzip compression (1)
  Compressing the following resources with gzip could reduce their transfer size by about two thirds (~27.49KB):
  - home.jsf could save ~27.49KB

- Leverage browser caching (9)
  The following resources are missing a cache expiration. Resources that do not specify an expiration may not be cached by browsers:
  - home.jsf
  - iquery.js.jsf
  - richfaces.js.jsf
  - richfaces-queue.js.jsf
  - SMrgbeps.png
  - default_profile_2_normal.png
  - judcon-logo.gif
  The following resources are explicitly non-cacheable. Consider making them cacheable if possible:
  - skinning.ecss.jsf
  - isf.js.jsf

- Leverage proxy caching (11)
  Consider adding a "Cache-Control: public" header to the following resources:
Questions?

The Source:
https://github.com/richfaces/tweetstream