Mobile Application Development Platforms and Mobio’s Technology

Adam Blum
Vice President of Engineering
Mobio Networks

Agenda

• Approaches to developing connected apps on mobile devices
  – Webwise
  – Local apps
• What’s new and different now
• So… what Mobio built
• Demo
Approaches to Mobile Development of Third Party Apps

• “the mobile web”
  – WAP
  – WAP/WML 2.0/XHTML Mobile Profile
• Mobile optimized operating systems
  – Symbian
  – Palm
  – Windows Mobile
  – BREW
• J2ME

---

WAP

• Open standard for wireless applications
  – Browse mobile-optimized websites with your cellphone

- Wireless Application Environment
- Wireless Session Protocol
- Wireless Transaction Protocol
- Wireless Transport Layer Security
- Wireless Datagram Protocol

Any wireless data network
The WAP
Wireless Application Environment

- The primary language is Wireless Markup Language (WML)
  - XML compliant
- A WML document is a “deck”
- Decks are composed of “cards” representing single interaction with the user
- Compressed into WBXML
- WML browsers
  - Operate
  - WinWAP
  - Klondike
  - OpenWave
  - NeoMar => Good Technology
- Almost all of these browsers are moved to either XHTML or HTML

Few seriously say that WAP per se has a future

Other Mobile Web “Standards”

- Openwave, formerly phone.com, need Unwired Planet)
  - Handheld Device Markup Language
  - Influenced WAP/WML
- NTT DoCoMo
  - iMode (based on proprietary Compact HTML)
The Vision of WML2

- XHTML Basic
  - Basic Forms
  - Hypertext
  - Basic Tables
  - CSS – important for now graphical devices
- More features from XHTML
  - acronym, address, br, b, big, hr, i, small, dl, fieldset, optgroup
- Features of WML not in XHTML
  - Navigation aids, onenter events, elements, attributes
  - All prefixed with “wml:”

WML2: The Compromise

- WML2 compliance ended up being defined as either
  - supporting true WML2
  - OR supporting XHTML
- MP+ WML 1.x
- In this muddy state of affairs, vendors implemented what they wanted to
  - Not necessarily either of these scenarios
WML2 Current Status

• Implementations
  – Nokia – XHTML MP, no WML namespace
  – Openwave – uses WML namespace
  – Access Compact NetFront 3.0, Samsung - only XHTML basic
  – Opera supports most of the WML 2 (all of XHTML MP) with some exceptions of WML controls

• Usability on the XHTML basic implementations is possibly worse on these noncompliant implementations than even WAP

So How Do You Write an XHTML MP App?

Create some XHTML content
• <html>
  • <body>
  • <h1>an example</h1>
  • <p>item 1</p>
  • <p>item 2</p>
  • </body>
  • </html>

Style it with CSS
• h1 {font-size:x-large;color:#4040ff;text-align:center; text-decoration:underline;}
• p {border:1px #6060ff solid;background:#f0f0ff; text-align:center;font-size:medium;padding:4px;}
• a {color:black}
• body {background:#c0c0ff;}

Set the mime type to application/xhtml+xml
But How Do You Handle When The Device Can Support WML Extensions?

WURFL and XSLT Invocation

- <? header("Content-type: text/vnd.wap.wml");
  require_once('wurfl_class.php');
  $device=new wurfl_class($HTTP_USER_AGENT);
  if($device->browser_is_wap){
      $xhtmlfile="file://D:/localhost/somecontent.html";
      $xslfile="file://D:/localhost/wmltransform.xsl";
      $xslt=xslt_create();
      $xslresult=xslt_process($xslt,$xhtmlfile,$xslfile);
      print($xslresult);
      xslt_free($xslt);
  }
- ?>

The XSLT Itself

- <wml>
  <card id="menu" title="{html/head/title}">
    <p align="center"> <b> <xsl:value-of select="html/body/h1"/></b> </p>
    <xsl:for-each select="html/body/p">
      <p align="center">
        – <a href="{a/@href}">
          – <xsl:value-of select="a"/></a>
      </p>
    </xsl:for-each>
  </card>
- </wml>
Relevant Specifications

• JSR 37 – MIDP 1.0 – September 2000
  – Launched a wave of mostly failed wireless companies at the height of Bubble 1.0
• JSR 118 – MIDP 2.0 – June 2005
  – Enhanced UI, multimedia and game functionality, greater connectivity options, OTA provisioning, end-to-end security
• JSR 139: Connected Limited Device Configuration 1.1 – March 2003

MIDP 2.0

Covers
• Application delivery and billing
• Application signing model and privileged domains security model
• End-to-end transactional security (https)
• MIDlet push registration (server push)
• Networking
• Persistent storage
• Sound (from wav to MID)
• Timers
• User Interface (including requirements for games)

Does Not Cover
• Power management
• Low level security
High Level Architecture

Hardware Requirements

- Screen size: 96 x 54
- Display depth: 1 bit
- Pixel shape: 1:1
- Input: keyboard or touchscreen
- 256kb non-volatile memory for MIDP implementation
- 8 kb non-volatile memory for app-created persistent data
- 128 kb volatile for Java runtime (heap)
- Two way wireless
- Sound (including tones via MIDI)
Software Requirements

• Minimal kernel to manage underlying hardware, at least one schedulable entity to run the JVM
• Read and write from nonvolatile memory
• Read and write access to device’s wireless networking
• Time base for timestamping records written to permanent storage
• Write to bitmapped display
• Capture user input
• Application lifecycle management

Specification Requirements

• MIDP 1.0 and 2.0 MIDlets
• All classes, interface specified
• OTA provisioning
• Push
• Visual indication of network usage
• Serial ports
• http 1.1
• https
• PNGs
• MIDI and WAV
• X.509
• UTF-8 for character encoding
Package Summary

- User Interface
  - javax.microedition.lcdui
  - javax.microedition.lcdui.game
- Application Lifecycle Package
  - javax.microedition.midlet
- Persistence Package
  - javax.microedition.rms
- Networking Package
  - javax.microedition.io
- Public Key Package
  - javax.microedition.pki
- Sounds and tone media
  - javax.microedition.media
  - javax.microedition.media.control
- Core Packages
  - java.lang
  - java.util

Success of J2ME

Pros
- Over 1Bn devices predicted by end of next year
- Applications are relatively portable
- It’s a successful gaming platform

Issues
- Very challenging to manage typical device memory footprints
- It still does require manual intervention and code analysis to port between devices
- Hasn’t really taken off for enterprise or informational applications
Other Mobile OSes

- BREW
  - Qualcomm only
  - C++ is a high bar for app (not system) developers
- Symbian
  - Nokia only, not big in US
- Windows Mobile
  - Very small marketshare (though Q may change that)
- PalmOS
  - Palm not committed to it, PalmSource (now Access) is now more Linux focused

What’s Different Now

- J2ME devices are ubiquitous
  - 50 million Razors
- Data plans are finally taking off in the US (20% penetration)
  - And being promoted aggressively by carriers
  - Still no widespread informational applications
- Web services (of one form or another) from consumer websites actually exist now
The Mobio Platform

Motivation and Vision

- Allow building of rich, thick client, device-optimized mobile consumer applications very rapidly
- Leverage existing technology where possible
- Use standards where convenient
Assumptions and Scope

- There is an existent WS-I compliant SOAP-based XML web service
  - Simplifies the problem enormously (witness Oracle Mobile XForms and Interactive Wireless Corp)
- Initial implementations of XForms on J2ME compliant devices
  - But will support Windows Mobile, BREW, Symbian, Linux eventually

What is an mApp

- XForms
- XFS
  - Server-side logic embedded into client-side script
- XSLT transformations
- (optional) BPEL scripts
- web services inventory
XFS

- JSP (server-side tags) but only
  - Root, element, attribute, if, choose, include
- Built-in objects
  - Preferences
    • `<c:forEach var="prefEntry" items="${preferences}">`
    • `<preference>`
    • `<c:out value="${prefEntry.key}"/> : ${prefEntry.value.valuesAsString}`
    • `<preference>`
    • `<c:forEach>`
  - Device
    • `${device.memory}`

Why Rich Client?

- Highly Interactive Dynamic Forms
- Code Execution and Data Processing on the Device (“the Ajax Effect”)
- “Three Clicks to Pick”

**Counterexample:**
*Try wap.fandango.com*
Why XForms

- **Separation of data and presentation**
- Leverage XForms tools
  - Orbeon
  - Visual XForms Designer
- Align with industry standards
- True chance for “write once-run anywhere”
- Allows multiple apps to fit on limited devices (the runner approach with highlevel app descriptions enables this)

Future:
- Even assuming Ajax-capable mobile browsers, XForms can generate HTML+Ajax Javascript on server and is much easier authoring experience

*But don’t stick with “from the standard” widgets*

---

Mobio Platform

- **Service Creation Toolkit**
- **User Profile Web Interface**
- **Mobio Apps Thin Client**
- **Network Service Adaptation Manager**
- **Device Adaptation Manager**
- **Service Processing Platform**
  - XML Native Service Processing
  - Blending of Web Services
- **Web Service Adaptation Manager**
Sample mApp: The Model

```xml
<xf:model>
  <xf:instance id="search_data">
    <BugQuery xmlns=""">
      <assigned_to>${preferences['owner'].value}@mobio.net</assigned_to>
    </BugQuery>
  </xf:instance>
  <xf:instance id="BugQueryResponse">
    <BugQueryResponse xmlns=""/>
  </xf:instance>
  <xf:instance id="admin">
    <data xmlns="">
      <index>1</index>
    </data>
  </xf:instance>
  <xf:instance id="search_output">
    <data xmlns=""/>
  </xf:instance>
  <xf:submission action="/mobio/invoker/Bugs/BugService/BugQuery" method="post" id="search_for_bugs" replace="instance" instance="search_output" ref="instance('search_data')"/>
  <xf:submission action="bugs.xfs" method="get" id="home" includenamespaceprefixes="" />
</xf:model>
```
Sample mApp: The Search Form

```html
<body class="bg">
  <xf:switch>
    <xf:case id="search_form">
      <div class="color1 dotted-underline">Search by Owner</div>
      <xf:input ref="instance('search_data')/assigned_to">
        <xf:label>Owner:<xf:label>
      </xf:input>
      <xf:action ev:event="xforms-submit-done"
        ev:observer="search_for_bugs">
        <xf:toggle case="show_bugs"/>
      </xf:action>

      <!--softkey menu-->
      <xf:send submission="search_for_bugs" ev:observer="root"
        ev:event="menu-1-action"/>
      <xf:select1 ref="instance('home_menu')/selected"
        class="right_menu">
        <xf:label>Options</xf:label>
        <xf:item>
          <xf:label>Get Bugs</xf:label>
          <xf:value>menu-1-action</xf:value>
        </xf:item>
      </xf:select1>
    </xf:case>
  </xf:switch>
</body>
```

Sample App: The Results Form

```html
<xf:case id="show_bugs"/>
  <xf:group ref="instance('search_output')/BugQueryResult" class="inline">
    <xf:repeat nodeset="Bug" id="bug-repeat" number="4">
      <xf:trigger class="link">
        <xf:label ref="bug_id"/>
        <xf:action ev:event="DOMActivate">
          <!-- set the index for the "clicked" bug so that the next page knows which bug to show -->
          <xf:setvalue ref="instance('admin')/index" value="index('bug-repeat')"/>
          <xf:toggle case="bug_description"/>
        </xf:action>
      </xf:trigger>
      <xf:output class="color2 inline" ref="short_desc"/>
    </xf:repeat>
  </xf:group>
  <xf:trigger id="new_search" class="soft_button">
    <xf:label>New Search</xf:label>
    <xf:toggle case="search_form" ev:event="DOMActivate"/>
  </xf:trigger>
</xf:case>
```
Sample mA.pp File

- <?xml version="1.0" encoding="UTF-8"?>
- <mapp name="Bugs" parent="Base Application"
  xmlns="http://mobio.net/appsdev/mapp">
  <author>Adam Blum</author>
  <version>1.0</version>
  <services>
    <service name="BugService" wsdl="http://clientdev1.mobio.net/Bugs/BugService.asmx?wsdl"/>
  </services>
  <preferences>
    <preference name="owner" multi-valued="false"/>
  </preferences>
</mapp>

mA.pp Schema
The MovieFinder App

• If you have Razor or other popular J2ME phone (Samsung i700, etc.), download from http://www.getmobio.com/customers/universal/

Challenges with the Platform

• Getting mobile XForms client running quickly and in limited memory on lowend devices
  – Such as Moto Razors
  – This is an ongoing issue
• Debugging applications with so many moving parts
• Teaching developers XForms
  – Easier in concept, but noone has knowledge out of the box
• Limitations of XForms as programmatic tool
Other Similar Proprietary Technology Platforms

- **UI Evolution**
  - UJML – procedural language wrapped in XML tags
  - J2EE app server with custom tag library
  - But
    - Not truly declarative
    - Not standards based
    - Not oriented to multiple apps running on client
    - Limited server services

- **MFoundary**
  - At a high level, very similar to what we do
  - MIL – declarative language for describing client apps that is even MVC
  - Server which provides provisioning, compression, data brokering, carrier integration, tracking/reporting
  - Eclipse-based IDE
  - But
    - No standards-based stuff (MIL vs. XForms)
    - No XFS (server-scripting)
    - No preferences console or personalization
    - No transformation on server

Future Directions for the Mobio Platform

- **More devices**
  - All J2ME
  - Windows Mobile
  - BREW
  - All Ajax capable browser (Opera) devices with server-side generation of HTML+Javascript via Orbeon

- **IDE**
  - Drag and drop XForms
  - Point to web services
  - Command completion for preferences and capabilities in XFS

- **Support for building backend service**
  - First class use of Rails?

- **REST-based web services on backend as well (or primarily?)**

- **Efficiencies between XForms MVC interfaces and MVC-based backend web services?**
References