Mobile Ajax for Java™ Technology

Akhil Arora & Vincent Hardy

Architects, Java ME
Sun Microsystems, Inc.

TS-5525

Mobile Ajax

Learn to apply the Ajax programming model to Java ME platform applications running outside the browser

Agenda

Demo

Introduction

Motivations

Handling Data

Presentation

Q&A

Glossary

- Ajax—Asynchronous JavaScript™ technology and XML
- JSON—JavaScript technology Object Notation
- GCF—Generic Connection Framework (MIDP)
- DOM—Document Object Model (W3C)
- SVG—Scalable Vector Graphics (W3C)

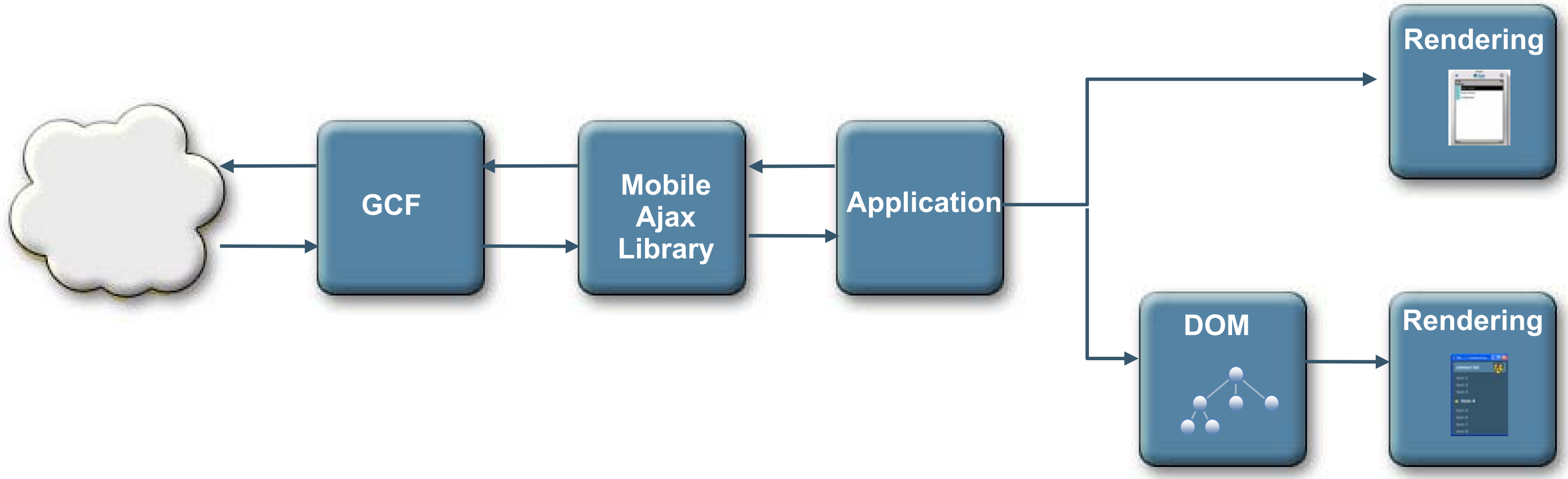
Demo

Introduction

What is Mobile Ajax?

- Traditional definition in the browser world:
  - XmlHttpRequest + XML/JSON + JavaScript technology + DOM in a browser
- A more generic definition for Java ME platform:
  - Asynchronous call to the network (GCF in MIDP)
    - Can do much more than HTTP—SMS, Bluetooth...
  - A data serialization format (XML, JSON, etc.)
    - Flexibility to roll your own format
  - Presentation—Traditional or Rich UI
    - LCDUI or DOM based UI (SVG)

An Ajax-y Interaction



Motivations

Why Ajax for Java ME applications?

- Simplicity—asynchronous vs. multi-thread
- Offer a familiar paradigm to web developers
- Abstract out low-level, data-format parsers
- Offer capabilities of the platform
  - Camera, Location, Bluetooth, Address Book, RMS, etc.
- Small library footprint

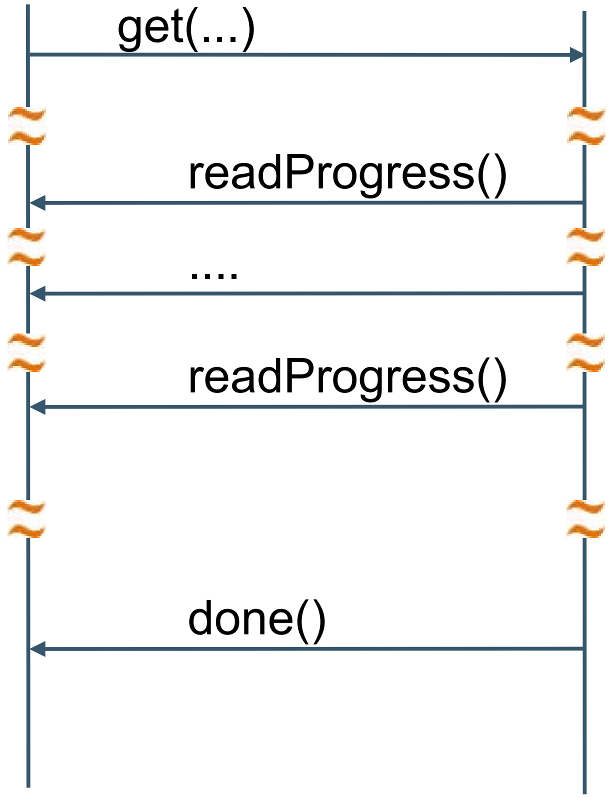
Asynchronous Requests

Building upon GCF for Web 2.0

- Asynchronous versions of HTTP Get and Post
- Progress callbacks
- HTTP Basic/Digest Authentication
- URL-encoding
- Multi-part MIME (sender-side)
- Caching

Asynchronous Get Requests

Call Sequence



Request

```
// synchronous versions
static Response get(String url, Arg[]
  inputArgs,
       Arg[] httpArgs, ProgressListener
  listener)
static Response post(String url, Arg[]
  inputArgs,
       Arg[] httpArgs, ProgressListener
  listener,
       PostData data)

// asynchronous versions
static void get(String url, Arg[]
  inputArgs, Arg[] httpArgs,
  RequestListener listener, Object
  context)
```

JavaOne

Input Args

```
String url = "http://host.com/web-api";
Arg[] args = {
  new Arg("arg1", "val1"),
  new Arg("arg2", "val2")
};

// url becomes
// "http://host.com/web-
  api?arg1=val1&arg2=val2"
```

ProgressListener

```
interface ProgressListener {
    // total will be zero if not available
    void readProgress(Object context,
                int bytes, int total);
    void writeProgress(Object context,
                int bytes, int total);
}
```

RequestListener

```
interface RequestListener extends
  ProgressListener {
     void done(Object context, Response
  result);
}
```

Response

```
class Response {
    // Result contains the parsed returned
 data
    Result getResult();

    // HTTP response code
    int getCode();

    // HTTP response headers
    Arg[] getHeaders();

    // Exception, if any
    Exception getException();
}
```

Result

```
// accessors for primitive types
boolean getAsBoolean(String
  pathToElement);
int getAsInteger(String pathToElement);
long getAsLong(String pathToElement);
double getAsDouble(String pathToElement);
String getAsString(String pathToElement);

// accessors for arrays
int getSizeOfArray(String pathToElement);
String[] getAsStringArray(String
  pathToElement);
int[] getAsIntegerArray(String
  pathToElement);
```

Parsing Data

Handling XML/JSON

- Issue: don’t want to deal with parsers
- Issue: don’t want generated code bloat
- Issue: may not have a schema
- Solution: the dynamic, declarative approach

  Based on a small expression language

  - Just “.”, “[” and “]”
  - Some example paths—

```
        statuses.status[1].text
          statuses.status[2].user.screen_name
          users.user[3].id
```

Abstracting XML and JSON

XML

```
<users>
   <user>

  <name>User
  1</name>
   </user>
   <user>

  <name>User
  2</name>
   <user>
  </users>
```

JSON

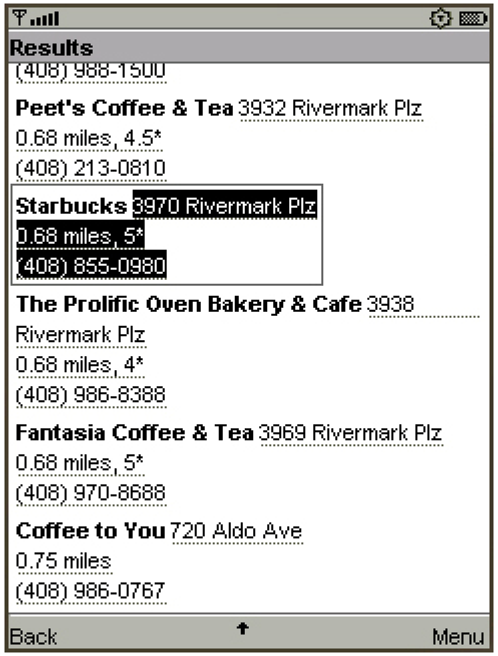
```
{ "users": {
   "user": [
   {
     "name":
  "User 1"
   },
   {
     "name":
  "User 2"
   }
  ]}}
```

```
String name =
  result.getAsString("users.user[1].n
  ame");
assert "User 2".equals(name);
```

Using Result

```
Result result = response.getResult();
int size =
  result.getSizeOfArray("users.user");
User[] friends = new User[size];
for (int i=0; i < size; i++) {
    User user = new User();
    String base = "users.user[" + i +
  "].";
    user.name = result.getAsString(base +
  "name");
    user.id = result.getAsInteger(base +
  "id");
    ...
```

Presentation



Results

(408) 988-1500

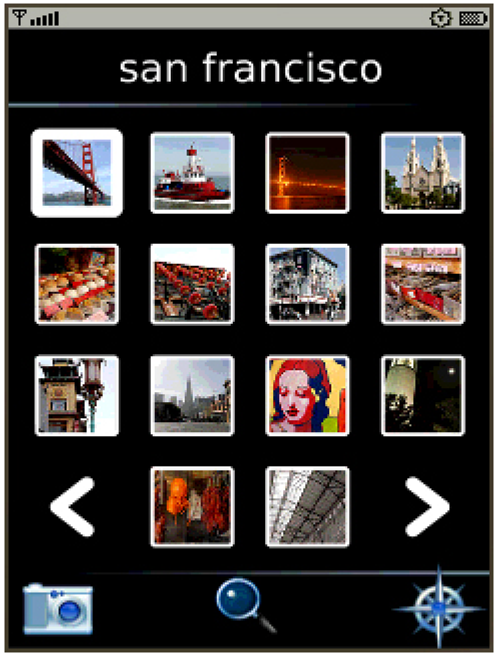
Peet's Coffee & Tea 3932 Rivermark Plz
0.68 miles, 4.5*
(408) 213-0810

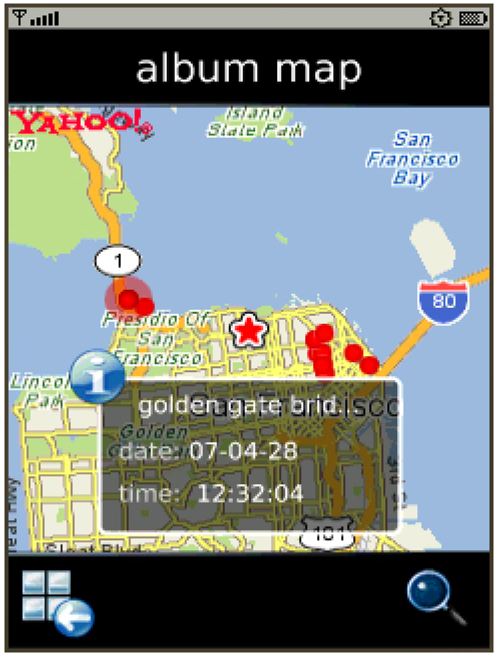
Starbucks 3970 Rivermark Plz
0.68 miles, 5*
(408) 855-0980

The Prolific Oven Bakery & Cafe 3938 Rivermark Plz
0.68 miles, 4*
(408) 986-8388

Fantasia Coffee & Tea 3969 Rivermark Plz
0.68 miles, 5*
(408) 970-8688

Coffee to You 720 Aldo Ave
0.75 miles
(408) 986-0767





DOM-Based Presentation

- As in Ajax applications, apply the results to a DOM tree
- Currently: use Java Specification Request (JSR) 226 to manipulate rich, animated 2D graphics
- JSR 226 is part of the Mobile Service Architecture (MSA, JSR 248)
- In the future: use JSR 287 (Java SVG Tiny Viewer 1.2) or JSR 290 (Compound Document Formats, XHTML, SVG, CSS and ECMAScript)

DOM-Based Presentation

Rendering and playing SVG content

```
import javax.microedition.m2g.SVGImage;

  SVGImage image = SVGImage.createImage(url,
  null);

  // Play the image
  SVGAnimator animator =
  SVGAnimator.createAnimator(image);

  Canvas canvas = (Canvas)
  animator.getTargetComponent();
  getDisplay().setCurrent(canvas);
  canvas.play();

  // Can also use
  javax.microedition.m2g.ScalableGraphics
  // for 'one-shot' SVGImage rendering into a
  custom
  // MIDP Canvas
```

DOM-Based Presentation

Progress bar example 1/3

```
<svg ...>
    <rect id="bkg" width="240" height="320" fill="white" />
    <rect id="progress"
      x="20" y="200" width="1" height="30" fill="blue"/>
    <animateTransform id="doneAnimation"
        attributeName="transform" type="translate"
        values="0,0;400,0" begin="indefinite" dur="0.5s" />

    <text id="progressText" x="120" height="240">0%</text>
</svg>
```

DOM-Based Presentation

Progress bar example 2/3

```
class ProgressBar implements ProgressListener {
                 SVGAnimationElement doneAnimation;
                 SVGLocatableElement progress;
                 SVGElement progressText;

                 public ProgressBar(Document doc) {
                     doneAnimation = (SVGAnimationElement)
               doc.getElementById(“doneAnimation”);
                     progress = (SVGLocatableElement)
                         doc.getElementById(“progress”);
                     progressText = (SVGElement)
                         doc.getElementById(“progressText”);
                 }
                 // See next slides ...
               }
```

DOM-Based Presentation

Progress bar example 3/3

```
public void readProgress(int bytes, int total)
  {
     float pos = (bytes / (float) total);

     // Scale the progress bar graphic
       SVGMatrix scale =
  computeScaleMatrix(pos);
       progress.setMatrixTrait("transform",
  scale);

       progressText.setTrait("#text",
               (int) Math.ceil(pos * 100) +
  "%");
  }


void done(Object context, Response result) {
       doneAnimation.beginElementAt(0);
}
```

Summary

- Ajax-y layers over the base platform's networking and parsers with DOM-based presentation
  - Simplify application development
  - Provide high separation of concerns between
    - Presentation
    - Application logic
    - Data services
  - Yield high flexibility
    - Data sources and formats can change independently
    - User experience can change independently

For More Information

- ME Application Developer Project: https://meapplicationdevelopers.dev.java.net/
- **TS-5628:** Developing Flashy Mobile Applications, Using SVG and JSR 226
- **TS-5743:** Graphical, Scripted and Animated User Interfaces on Java Platform, Microedition (Java ME)
- Java SVG Tiny Viewer 1.1 user interfaces with JSR 226: http://jcp.org/en/jsr/detail?id=226
- Java SVG Tiny Viewer 1.2 user interfaces with JSR 287: http://jcp.org/en/jsr/detail?id=287
- CDF user interfaces with JSR 290: http://jcp.org/en/jsr/detail?id=290

Q&A

Akhil Arora
Vincent Hardy

Mobile Ajax for Java™ Technology

Akhil Arora & Vincent Hardy

Architects, Java ME
Sun Microsystems, Inc.

TS-5525