Mobilizing Enterprise Processes with XML

Isaac de la Peña

Senior Technology Manager
Nokia Enterprise Solutions
http://www.nokiaforbusiness.com/

TS-9171

Goal of the Session

What You Will Gain

Learn the best strategies to mobilize corporate processes while keeping the benefits of a Service Oriented Architecture (SOA)

Web Services and Mobility

A History of Worlds Colliding?

- Web Services has become the standard means for clients to interact with corporate backend servers, their databases and related services
- With its platform neutrality and strong industry support, XML is being used by developers to link networked clients with remote enterprise data
- An increasing number of these clients are based on the J2ME™ platform, with a broad selection of mobile phones, PDAs, and other portable devices
- As developers utilize these mobile devices more to access remote enterprise data, XML support on the J2ME platform is becoming a requirement

Agenda

The Proxy Servlet Hack

XML and Web Services API

Looking Forward: Next Generation Java™ Technology

Looking Backward: Legacy Support

Up and Beyond: Other Scenarios

Forms Case Study

The Mobile Java Service Platform

Agenda

The Proxy Servlet Hack

XML and Web Services API

Looking Forward: Next Generation Java™ Technology

Looking Backward: Legacy Support

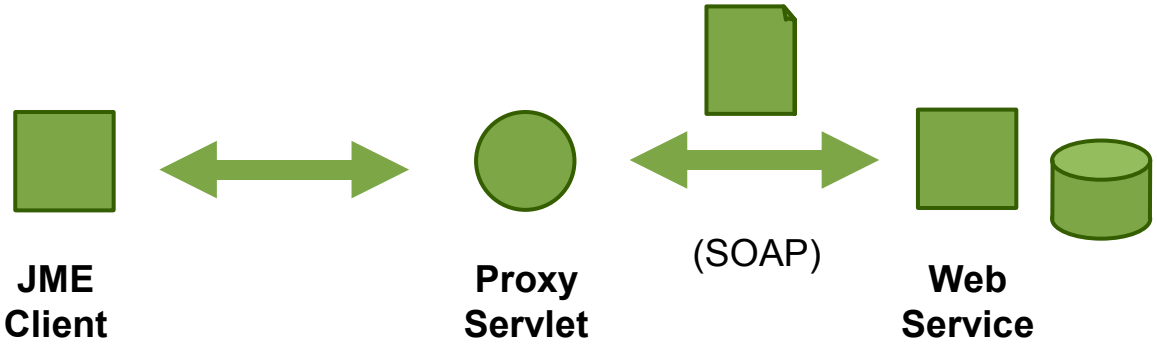
Up and Beyond: Other Scenarios

Forms Case Study

The Mobile Java Service Platform

The Proxy Servlet Hack

- Instead of accessing the Web Service directly from the Java ME technology client:
  - Create a “Proxy” Servlet to access the Web Service
  - Access the Servlet from the Java ME technology client using normal HTTP methods



HelloWorld Web Service

```
public interface HelloWs extends Remote {
  public String sayHello(String s) throws RemoteException;
}
```

```
public class HelloWsImpl implements HelloWs {
    public String message = "Hello ";
    public String sayHello(String s) {
        return message + s;
    }
}
```

HelloWorld Client

```
public static void main(String[] args) {
  ServiceFactory f = ServiceFactory.newInstance();
  Service s = f.createService(new QName("HelloWorld"));
  Call call = s.createCall(new QName("HelloPort"));
  call.setTargetEndpointAddress(
    "http://localhost:8080/hello-ws/hello");
  [...]

  call.setReturnType(new QName(
    "http://www.w3.org/2001/XMLSchema", "string"));
  call.setOperationName(new QName(
    "http://com.test/wsdl/HelloWorldWs", "sayHello"));
  String[] params = { "Audience!" };
  String result = (String) call.invoke(params);
  System.out.println(result);
}
```

HelloWorld Proxy

```
public class HelloWorldProxy extends HttpServlet {
  public void service(HttpServletRequest request [...]
    DataInputStream din =
      new DataInputStream(request.getInputStream());
    String param = din.readUTF();
    din.close();

    // Web Service Call (see previous)

    String result = (String) call.invoke(params);
    DataOutputStream out = new
      DataOutputStream(response.getOutputStream());
    out.writeUTF(result);
    out.close();
  }
}
```

HelloWorld MIDlet

```
// On its own thread...

HttpConnection con = (HttpConnection)
  Connector.open(url, Connector.READ_WRITE);
con.setRequestMethod(HttpConnection.POST);

DataOutputStream dos = con.openDataOutputStream();
dos.writeUTF("Audience!");

din = con.openDataInputStream();
String response = din.readUTF();
```

DEMO

Proxy Servlet

So What About It?

Good and Bad Points

- Good
  - Works! :-)
  - Client-side straightforward
  - Lightweight implementation
- Bad
  - More network components
    - Added complexity in server-side
    - Increased dev and maintenance costs
  - Reduced flexibility in adding new services on the fly
    - Fast service composition is at the heart of SOA

What SOA Is All About

A Different Perspective

"SOA is not just a market thing, and it's not interfaces that you need to change. If it was that easy then we'd have all done it 10 years ago. It's a complete change in the way you deliver services. We no longer think about building monolithic applications and integrating them. We think about building services and composing them."

…Ron Schmelzer, senior analyst at ZapThink LLC

Agenda

The Proxy Servlet Hack

XML and Web Services API

Looking Forward: Next Generation Java™ Technology

Looking Backward: Legacy Support

Up and Beyond: Other Scenarios

Forms Case Study

The Mobile Java Service Platform

XML and Web Services API

JSR 172

- Java ME technology infrastructure to:
  - Provide basic XML processing capabilities
  - Enable reuse of Web Service concepts when designing Java ME technology clients to enterprise processes
  - Programming model consistent with that of Java SE technology
- Platform API
  - Small footprint
  - Low complexity
- Use Carbide.j IDE
  - http://forum.nokia.com

XML Optional Package

Subset of JSE JAXP 1.2 API

- Support for SAX 2.0
- Support for XML namespaces
- Support for at least UTF-8 and UTF-16
- No support for Document Object Model (DOM)
- No support for XSLT
- Optional DTD validation

Source: Nokia

Web Services Optional Package

Subset of JSE JAX-RPC 1.1 API

- Mostly oriented to use SOAP as transport
  - No support for attachments
- Only literal message representation
- No support for extensible type mapping
- Interoperability with non JAX-RPC services: JSR 172 mandates that implementations follow the WS-I Basic Profile recommendations

Source: Nokia

HelloWorld MIDlet

```
// Use Carbide.j for Stub Generation from WSDL

HelloWsBinding_Stub stub = new HelloWsBinding_Stub();
stub._setProperty(
  Stub.ENDPOINT_ADDRESS_PROPERTY, endPoint);
HelloWs ws = (HelloWs) stub;

try {

    String response = ws.sayHello("Audience!");

} catch (RemoteException e) { }
```

DEMO

XML and Web Services API

Agenda

The Proxy Servlet Hack

XML and Web Services API

Looking Forward: Next Generation Java™ Technology

Looking Backward: Legacy Support

Up and Beyond: Other Scenarios

Forms Case Study

The Mobile Java Service Platform

Next Generation Mobile Java

JSR 249

- Additional APIs
  - XML and Web Services API (JSR 172) et alia
- Rich set of UI widgets based in eSWT
- Shared libraries
- Component services (OSGi framework)
  - Customized XML Configuration and Deployment
  - Local and Remote Manageability
  - Many learnings from Apache Avalon
- Full Java VM and Java Security
- Superset of JSR 248 (“MIDP 3.0”)

Source: Nokia

Impact On Web Service Mobilization

So What?

- Leverage the platform APIs, and
- Componentize your application
- Decouple the Web Service-processing routines from the applications consuming these services, the presentation layer...
- Reduces bundle footprint
- Increases maintainability and reusability

Source: Nokia

HelloWorld Service

```
// Decoupling Service Inteface from Implementation

public interface HelloService {
  public String getMsg();
}

public class HelloServiceImpl implements HelloService {
  public String getMsg() {

    // Web Service Call (see previous)

    return response;          // "Hello Audience!"
  }
}
```

HelloWorld Application (I)

```
// Code embeddd in standard MIDlet application

ApplicationContext ctx =
    Framework.getApplicationContext(this);
if (ctx != null) {
  HelloService hello = (HelloService)
    ctx.locateService("HelloService");
  if (hello != null)
    System.out.println(hello.getMsg());
  }
}

/* You can access as many services as needed, both custom
and platform (e.g. Logging, Configuration...)
(see next slide) */
```

HelloWorld Application (II)

```
ConfigurationAdmin service = (ConfigurationAdmin)
  ctx.locateService("ConfigService");
if (service != null) {
  try {
    Configuration config = service.getConfiguration(
      "simple.client.HelloWorldMIDlet.PID");
    Dictionary props = config.getProperties();
    if (props == null) {
      props = new Hashtable();
      props.put("MyParam", "MyValue");
      config.update(props);
    } else {
      System.out.println(props.get("MyParam"));
    }
  } catch (IOException e) {  }
}
```

Configuration Descriptors

The Essence of Declarative Services

- Use Carbide.j IDE Future Edition
  - http://pro.forum.nokia.com (Restricted Access)
- Create a Service Descriptor File (.srvcfg) each:
  - Service Interface
  - Service Implementation
- Create an Application Descriptor File (.appcfg)
  - Application
- Create a Deployment Descriptor File (.dpcfg)
  - Deployment Package (Bundle)

Source: Nokia

Sample Configuration File

```
<!- apps.xml -->

<?xml version="1.0" encoding="UTF-8"?>
<descriptor xmlns="http://www.osgi.org/xmlns/app/v1.0.0">
  <application class="sample.HelloWorldMIDlet">
    <reference interface="sample.HelloWorldService"
name="HelloWorldService"/>
    <reference interface="org.osgi.service.log.LogService"
name="LogService"/>
    <reference
interface="org.osgi.service.cm.ConfigurationAdmin"
name="ConfigService"/>
  </application>
</descriptor>
```

DEMO

Next Generation Mobile Java

Agenda

The Proxy Servlet Hack

XML and Web Services API

Looking Forward: Next Generation Java™ Technology

Looking Backward: Legacy Support

Up and Beyond: Other Scenarios

Forms Case Study

The Mobile Java Service Platform

kXML and kSOAP

http://ksoap.org

- kSOAP by Enhydra
  - http://www.enhydra.org
- Open Source
- Robust, simple and well documented
- Small footprint
  - Less than 42 KB in a single JAR
- Support for static typing
  - Implement KvmSerializable

Source: Enhydra

HelloWorld kSOAP MIDlet

```
SoapObject rpc = new SoapObject
  ("http://com.test/wsdl/HelloWorldWs", "sayHello");

rpc.addProperty ("String_1", "Audience!");

HttpTransport tr = new HttpTransport
  ("http://localhost:8080/hello-ws/hello",
   "http://com.test/wsdl/HelloWorldWs#sayHello");

String result = "" + tr.call(rpc);
   // org.ksoap.SoapPrimitive
```

HelloWorld (Revisited)

```
// if not org.ksoap.SoapPrimitive...

public class HelloWorld {

  private String msg = null;

  public String getMsg() {
    return message;
  }

  public void setMsg(String msg) {
    this.msg = msg;
  }

}
```

Adding kSOAP to HelloWorld (I)

```
// implements org.kobjects.serialization.KvmSerializable

private static int PROP_COUNT = 1;
private static PropertyInfo PI_msg = new
  PropertyInfo("msg", ElementType.STRING_CLASS);

public Object getProperty(int param) {
  if (param == 0) return getMsg();
  else return null;
}

 public void setProperty(int param, Object obj) {
  if (param == 0) setMsg( (String) obj );
}
```

Adding kSOAP to HelloWorld (II)

```
private static PropertyInfo[] PI_PROP_ARRAY = {PI_msg};

public int getPropertyCount() {
  return PI_PROP_ARRAY.length;
}

public void getPropertyInfo(int param, PropertyInfo in) {
  in.name = PI_PROP_ARRAY[param].name;
  in.nonpermanent = PI_PROP_ARRAY[param].nonpermanent;
  propertyInfo.copy(PI_PROP_ARRAY[param]);
}
```

HelloWorld kSOAP MIDlet (Revisited)

```
ClassMap cm = new ClassMap();
cm.addMapping("http://com.test/wsdl/HelloWorldWs",
"HelloWorld", new HelloWorld().getClass());

SoapObject rpc = new SoapObject
  ("http://com.test/wsdl/HelloWorldWs", "getMsg");
HttpTransport tr = new HttpTransport
  ("http://localhost:8080/hello-ws/hello",
   "http://com.test/wsdl/HelloWorldWs#sayHello");
tx.setClassMap(cm);
HelloWorld world = (HelloWorld) tx.call(rpc);

System.out.println(world.getMsg());
   // System.out.println(tx.requestDump);
   // System.out.println(tx.responseDump);
```

DEMO

Legacy Support with kSOAP

Agenda

The Proxy Servlet Hack

XML and Web Services API

Looking Forward: Next Generation Java™ Technology

Looking Backward: Legacy Support

Up and Beyond: Other Scenarios

Forms Case Study

The Mobile Java Service Platform

Now That We Talk About It

Why Using XML only for Web Services?

- XML is a powerful data representation schema
- Not limited to data transport over HTTP
- Brings structure and clarity to many places
- The resources are already there!
  - XML Manipulation Sample with kXML

Source: Nokia

To Start with…a Simple XML File

```
<!-- hi.xml -->

<story>
    <msg>Hello</msg>
    <msg>World</msg>
</story>
```

kXML "SAX"—Like Manipulation

```
KXmlParser p = new KXmlParser();
p.setInput(class.getResourceAsStream("/hi.xml"), null);
p.nextTag();
p.require(XmlPullParser.START_TAG, null, "story");
while (p.nextTag () != XmlPullParser.END_TAG) {
  p.require(XmlPullParser.START_TAG, null, "msg");
  String text = p.nextText();
  vector.addElement(text);
  p.require(XmlPullParser.END_TAG, null, "msg");
}
p.require(XmlPullParser.END_TAG, null, "story");
p.next();
p.require(XmlPullParser.END_DOCUMENT, null, null);
```

kXML "DOM"—Like Manipulation

```
Document doc = new Document();
doc.parse(p);
Element root = doc.getRootElement();
int msgCount = root.getChildCount();
for (int i = 0; i < msgCount; i++) {
  if (root.getType(i) == Node.ELEMENT) {
    Element msg = root.getElement(i);
    if ("msg".equals(msg.getName())) {
      int msgCount = msg.getChildCount();
      for (int j = 0; j < titleCount; j++)
        if (msg.isText(j))
          vector.addElement(msg.getText(k));
      }
    }
  }
}
```

Up and Beyond

What About XML In

- Configuration files?
- Language files?
- RMS data structures?
- Inter-process data exchange?
- UI “Forms” rendering?
  - Case Study

Source: Please add the source of your data here

Agenda

The Proxy Servlet Hack

XML and Web Services API

Looking Forward: Next Generation Java™ Technology

Looking Backward: Legacy Support

Up and Beyond: Other Scenarios

Forms Case Study

The Mobile Java Service Platform

Mobile Forms Concept

Turn Your Java MIDlet into an Extendable Browser

- Using XML documents (“Forms”) to encapsulate the data model, handling and presentation
- Combined ideas from MIDP, HTML and XForms
- Total application flexibility: not only data model but also layout, look and feel, navigation... decided at runtime
- The rendering capabilities of a simple browser, with the extensibility of Java
- Lightweight (around 30 KB in our sample)

Source: Nokia

XML Form (Data Model)

```
<mform name=" Say Hello" uid="com.nokia.forms@1234">

<model>
  <instance>
    <person>
      <name/>
      <msg>Hello World!</msg>
    </person>
  </instance>

  <bind id=" you" nodeset="person/name" />
  <submission id="on_line_post" method="post"
    action="http://localhost/submit.jsp" />
  <submission id="on_line_xml" method="xml"
    action="http://localhost/submit.jsp" />
</model>
```

XML Form (Presentation)

```
<group>
  <img src="http://localhost/logo_nokia.gif"/>
  <p>Please say hello to the world:</p>
  <hr/>
  <input bind="you"><label>Name: </label></input>
  <input ref="person/msg"><label>Msg: </label></input>

  <submit submission="on_line_xml">
    <label>Submit as XML</label>
  </submit>
  <submit submission="on_line_post">
    <label>Submit as POST</label>
  </submit>
</group>

</mform>
```

Form Renderer (I)

```
int childCount = root.getChildCount();
for (int i = 0; i < childCount; i++) {
  Element node = root.getElement(i);
  String nodeName = node.getName()
  if ("model".equals(nodeName)) {
    // process the model
    // instance = model.getElement(i);
    // bindings.addElement(binding);
    // submissions.addElement(submission);
  } else if ("group".equals(nodeName)) {
    // process the layout.
    // items.addElement(item);
    // commands.addElement(command);
  }
}
```

Form Renderer (II)

```
Form form = new Form(formName);

// ...
if ("p".equals(widgetName)) {
  item = new StringItem(null, getText(widget));
} else if (else if ("input".equals(ctlName) {
   item = new TextField(
     getLabel(widget), getText(widget),
     getMaxSize(widget), getFlags(widget));
}
form.append(item);
// form.addCommand(command);
// ...

display.setCurrent(form);
```

Form Listener (I)

```
// Leverage the MIDP Event Model
public void commandAction(Command cmd, Displayable dis) {
  if (cmdBack.equals(cmd) || cmdExit.equals(cmd)) {
    // ...
  } else {
    Enumeration list = command.elements();
    while (list.hasMoreElements()) {
      Command command = (Command) list.nextElement();
      if (command.equals(cmd)) {
        // Bussiness logic here (use wrappers)
        // Form-based operations
          // or your own custom Java procedures
      }
    }
  }
}
```

Form Listener (II)

```
// Sample of POST XML submission

Document doc = new Document();
doc.addChild(Node.ELEMENT, instance);

HttpConnection con = (HttpConnection)
  Connector.open(actionURL, Connector.READ_WRITE);
con.setRequestMethod(HttpConnection.POST);
OutputStream out = con.openOutputStream();
KXmlSerializer serializer = new KXmlSerializer ();
serializer.setOutput(out, doc.getEncoding();
doc.write (serializer);

// Now get the server response through an InputStream...
```

DEMO

Form Case Study

Agenda

The Proxy Servlet Hack

XML and Web Services API

Looking Forward: Next Generation Java™ Technology

Looking Backward: Legacy Support

Up and Beyond: Other Scenarios

Forms Case Study

The Mobile Java Service Platform

What Is This Hype of Web 2.0?

From Publication Media to Distributed Services

"Web 2.0 is the network as a platform, spanning all connected devices; Web 2.0 applications are those that make the most of the platform advantages: delivering software as a continually-updated service that gets better the more people use it, consuming and remixing data from multiple sources"

Examples: Google, Yahoo, Amazon…

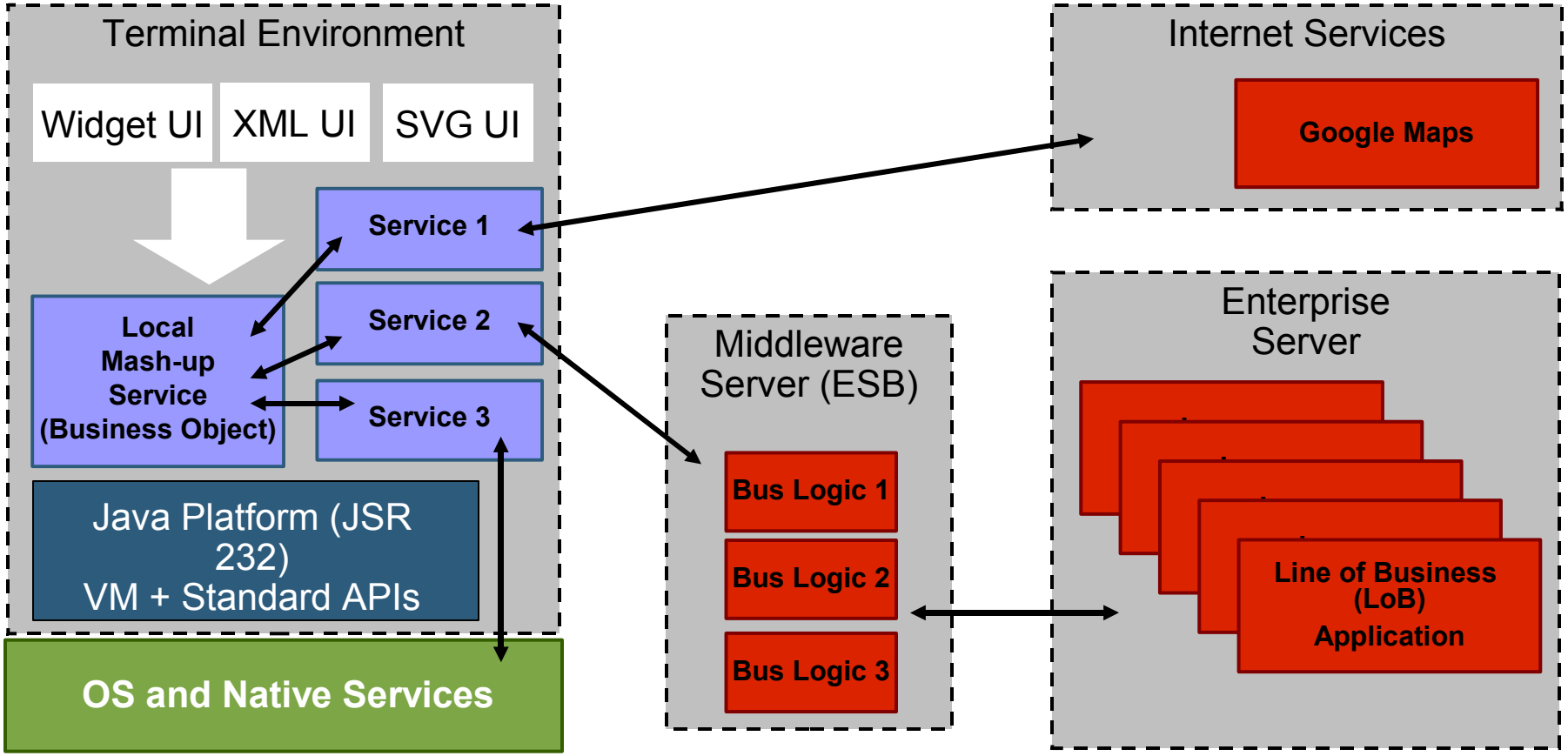
Mobile Java Service Platform

The Required Terminal Evolution

- 1st Generation: MIDP
  - Static APIs to Basic Terminal Features
  - UI, Network Access, Simple Persistence...
- 2nd Generation: JSR 248
  - Static APIs to Services running in the native OS
- 3rd Generation: JSR 232
  - Simple, dynamic local APIs reflecting network services
  - Shield the app developer from implementation details
  - Any WEB 2.0 service or composition of services, regardless of the technology, can quickly and easily be turned into a MJSP service

Source: Nokia

Web 2.0 Mobile Mash-Ups



Source: Nokia

Sample Case: Google (I)

Source: Nokia

Sample Case: Google (II)

- Developers utilizing Next Generation Mobile Java:
  - Take the Google Java Service API from Google's site
  - Adapt as a Mobile Java Service with all communication self contained including mobile specific QOS
  - Push to Java-enabled terminal using remote management features and publish the Service API to developers
- Result: The Google service would be instantly available to any application running in the Java environment, with any customized UI the application designer wants
- Cost: Several hours of skilled programmer time
- Benefit: Huge market advantage to react at WEB speed!

Source: Nokia

Summary

- SOA is a cornerstone of modern enterprise IT
  - Mobile services must comply to it
- XML and Web Services API (JSR 172)
- Next Generation Mobile Java for flexibility
- kSOAP for legacy support
- Don’t be constrained to data transport
  - Experiment by yourself!
- Be ready for Mobile Java Service Platform!

For More Information

- Other JME Technical Sessions @ JavaOne
  - TS-4936—Mobile Services Architecture Initiative, JSR 248
  - TS-3757—Introduction to JSR 232
- http://www.forum.nokia.com
  - http://pro.forum.nokia.com (Restricted Access)
- http://www.jcp.org
  - Mobile Web Services API (JSR 172)
  - Mobile Service Architecture (JSR 249)
  - Mobile Java Service Platform (JSR 232)
- http://www.ksoap.org
  - http://www.kxml.org

Q&A

Isaac de la Peña
isaac.delapena@nokia.com

Mobilizing Enterprise Processes with XML

Isaac de la Peña

Senior Technology Manager
Nokia Enterprise Solutions
http://www.nokiaforbusiness.com/

TS-9171