Java is a trademark of Sun Microsystems, Inc.

RESTful Access to Java™ Platform, Micro Edition (Java ME Platform)

Erik Hellman

Sony Ericsson

Erik.Hellman@sonyericsson.com

We like Twitter!

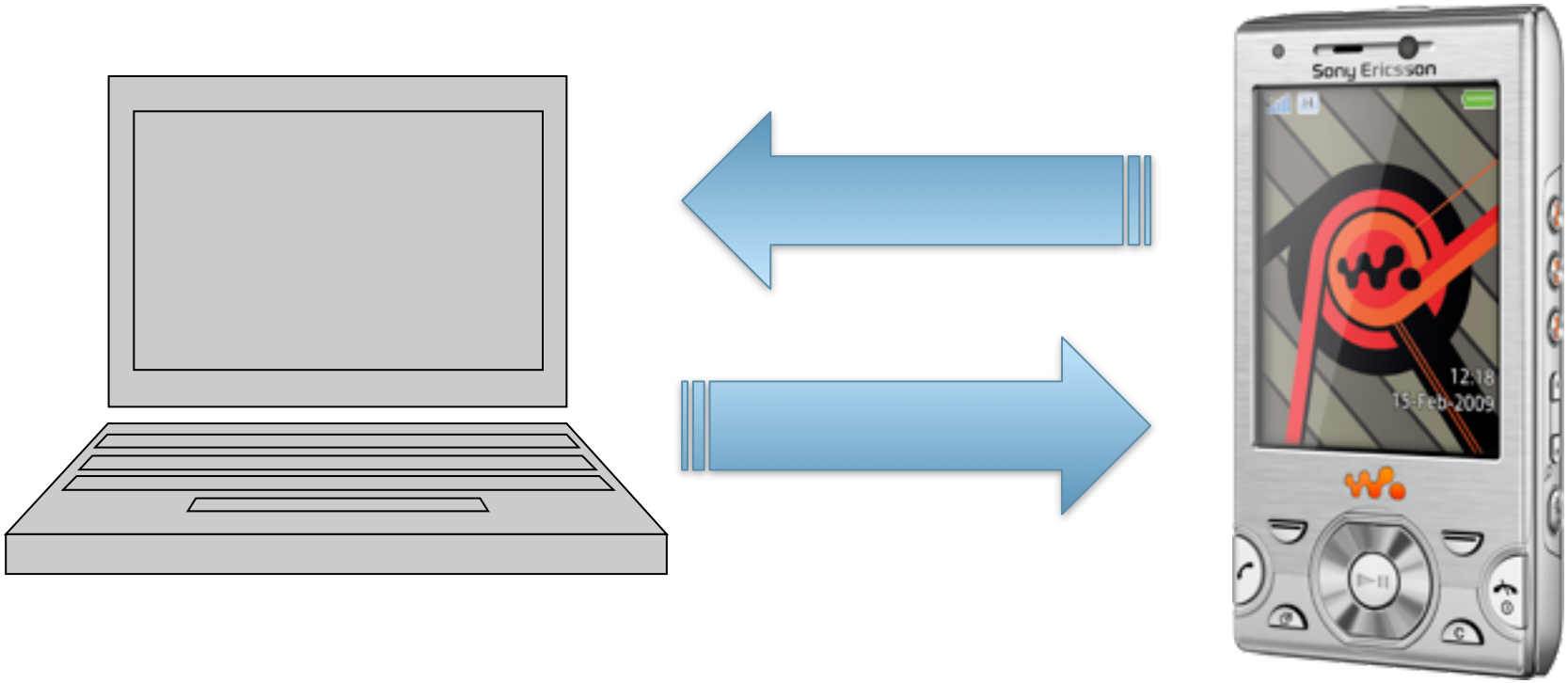
- Follow us on @SonyEricssonDev on Twitter (http://twitter.com/SonyEricssonDev)
- Use #SEMC Twitter tag to ask questions or comment on the session.
- We will follow the feed and try to address questions and feedback if possible. 

Wordle <http://www.wordle.net/>

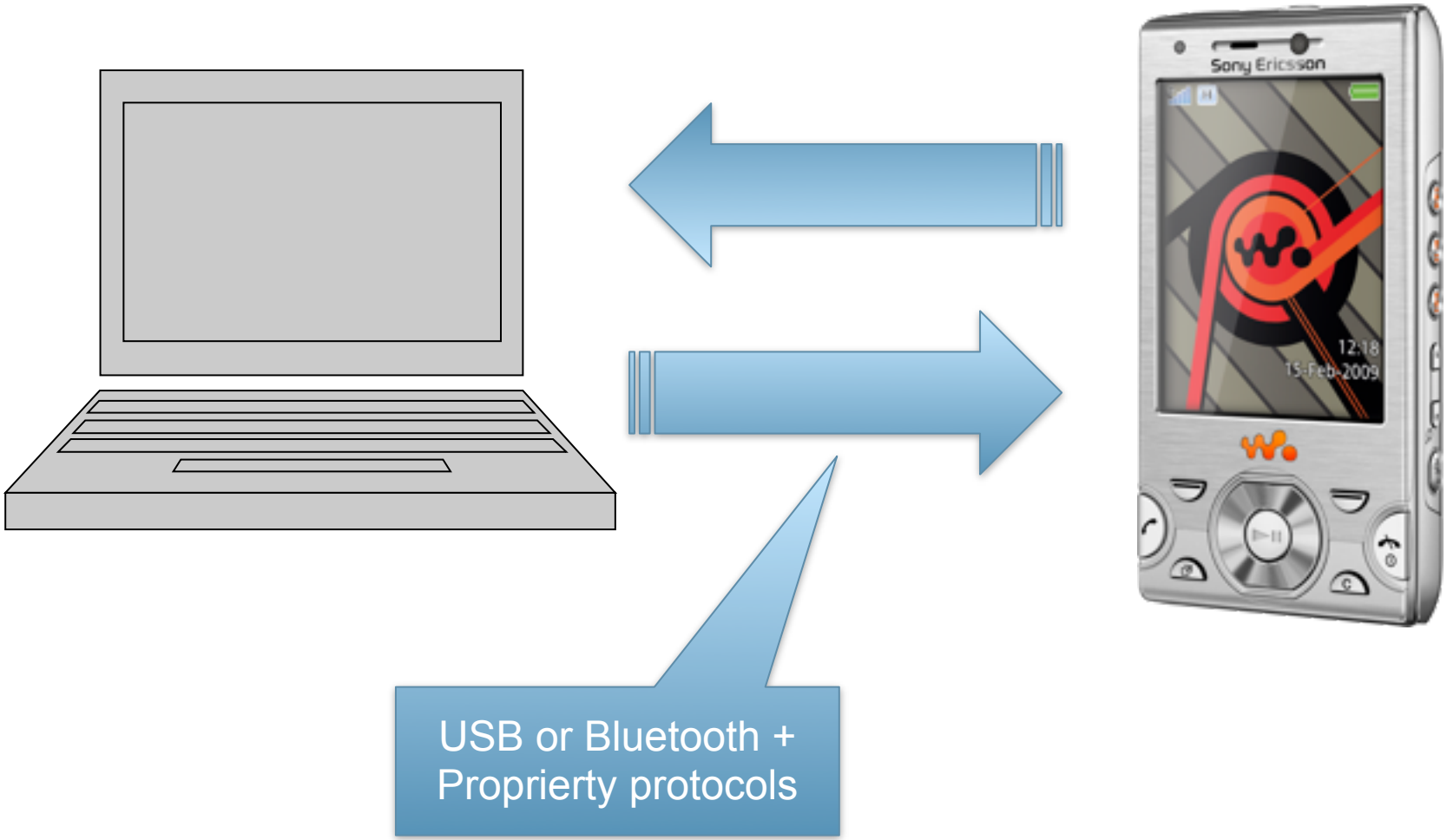
http://www.flickr.com/photos/peasap/2561252071/

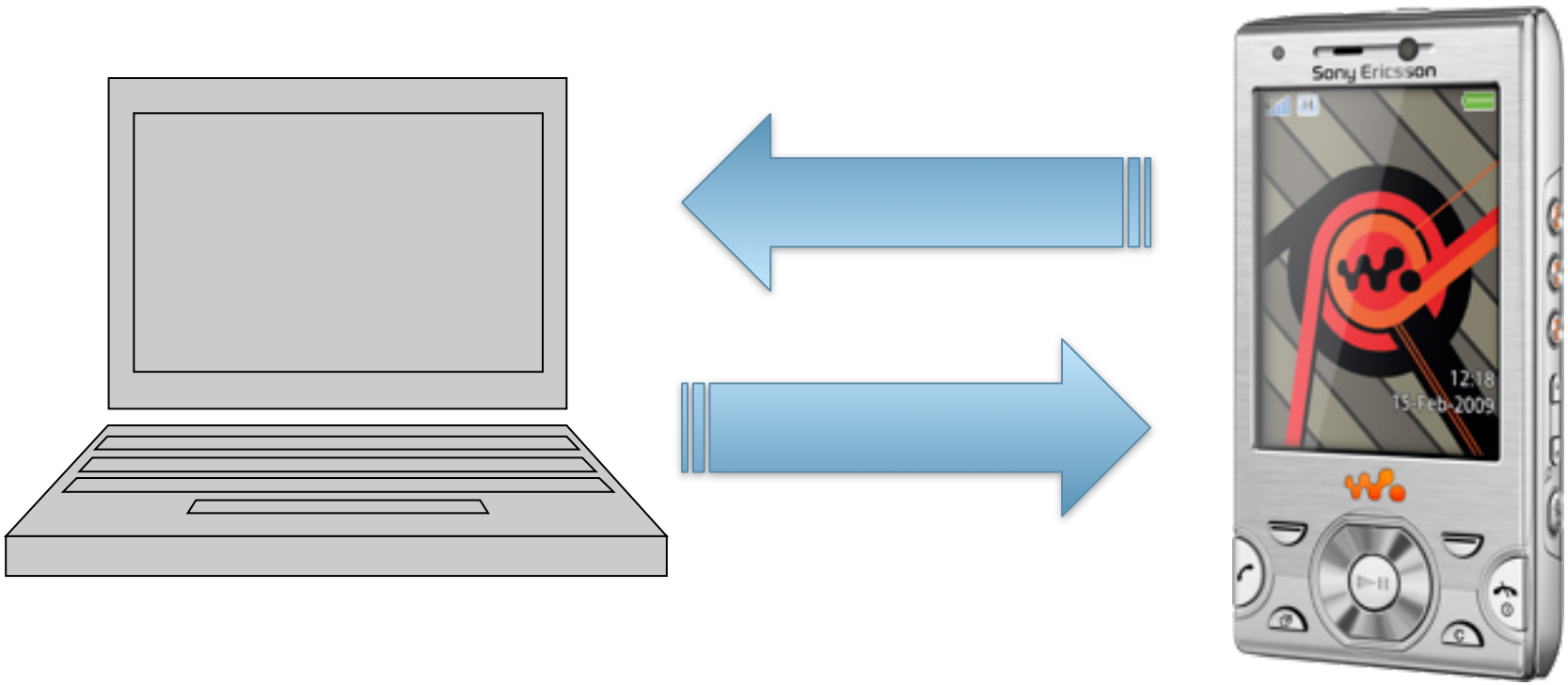
Restful?

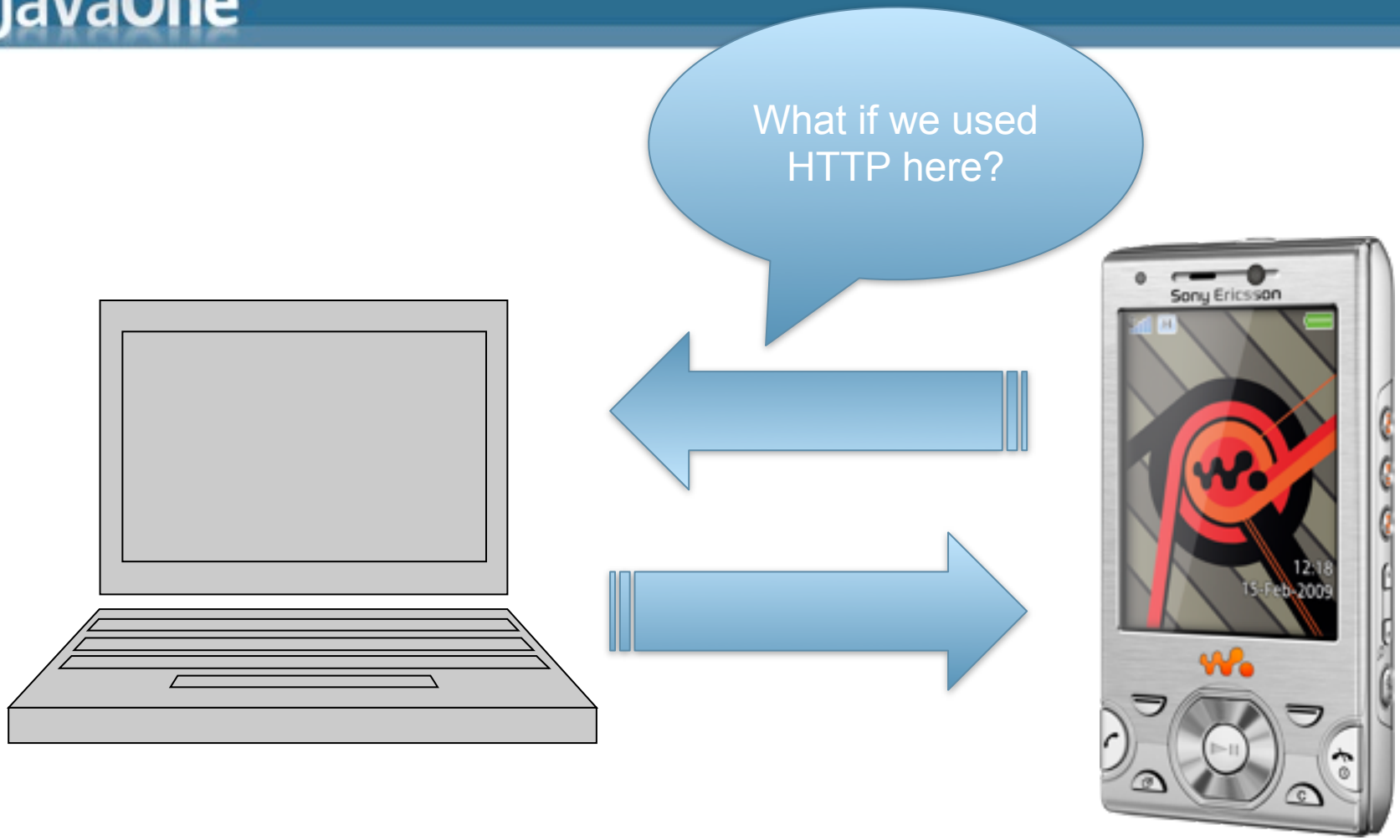
PC-to-phone communication



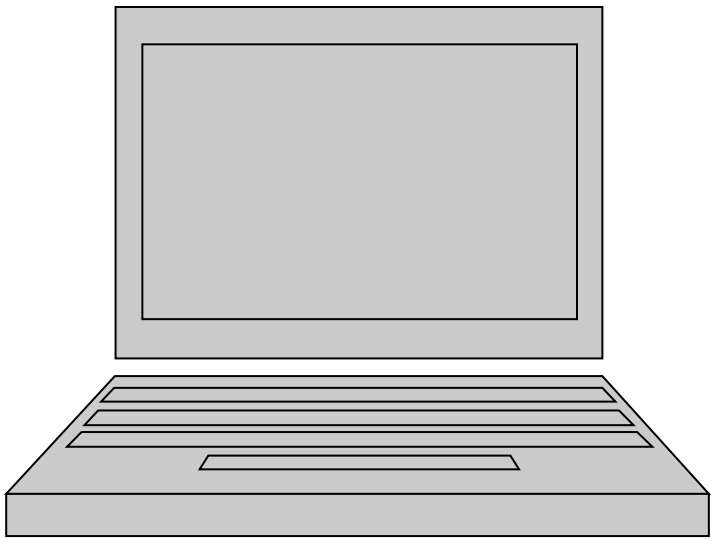
PC-to-phone communication




Sony Ericsson
12:18
15-Feb-2009

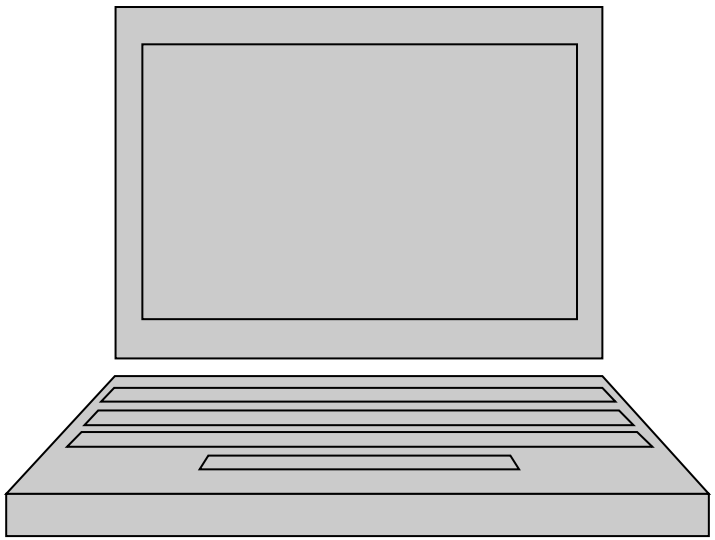

What if we used HTTP here?
Sony Ericsson
12:18
15-Feb-2009

What if we used HTTP here?



What if the PC is the client?

What if we used HTTP here?



What if the PC is the client?

What if the web server runs on the phone?

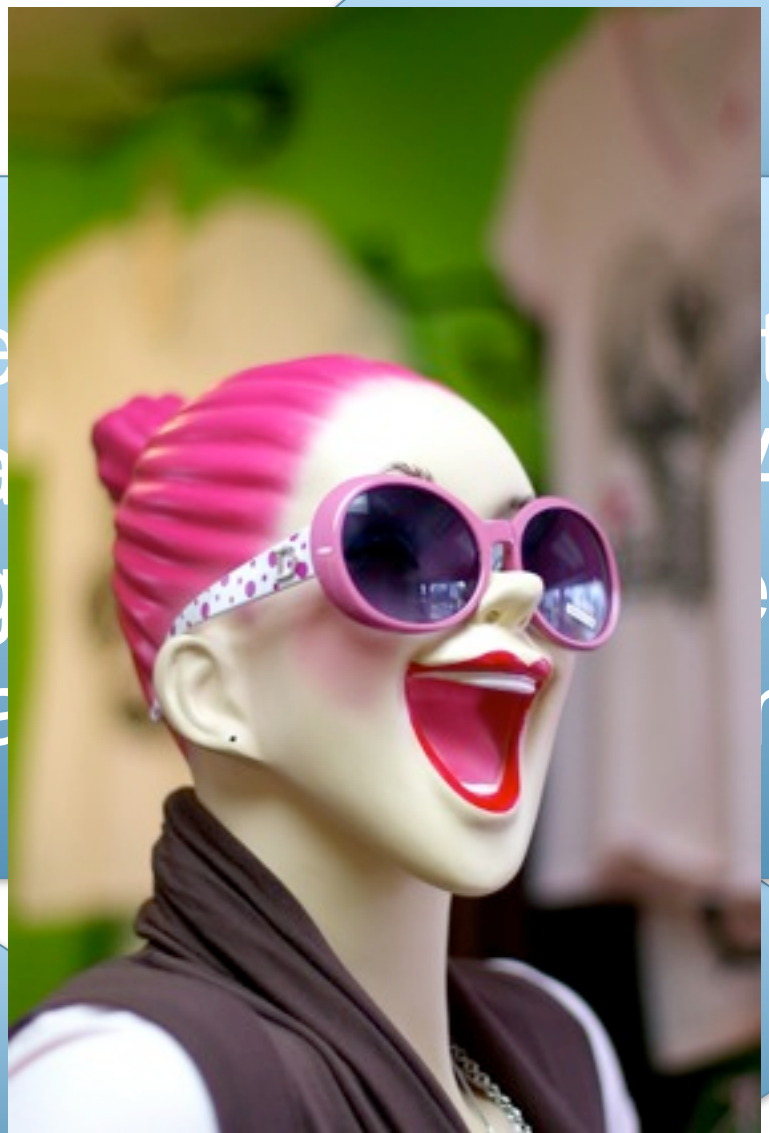
What if we used HTTP here?

Lets published all the available Java ME APIs through the web server using a RESTful design!

What if the PC is the client?

What if the web server runs on the phone?

Le
ava
throug
a
the
APIs
r using
n!



What if the PC is the client?

http://www.flickr.com/photos/houseofsims/3270370223/

RESTful recap

REST refers in the strictest sense to a collection of network architecture principles which outline how resources are defined and addressed. The term is often used more loosely to describe any simple interface which transmits domain-specific data over HTTP without an additional messaging layer such as SOAP or session tracking via HTTP cookies. These two meanings can conflict as well as overlap. It is possible to design a software system in accordance with Fielding's REST architectural style without using HTTP and without interacting with the World Wide Web.[2] It is also possible to design simple XML+HTTP interfaces which do not conform to REST principles, and instead follow a model of remote procedure call. The difference between the uses of the term "REST" therefore causes some confusion in technical discussions.

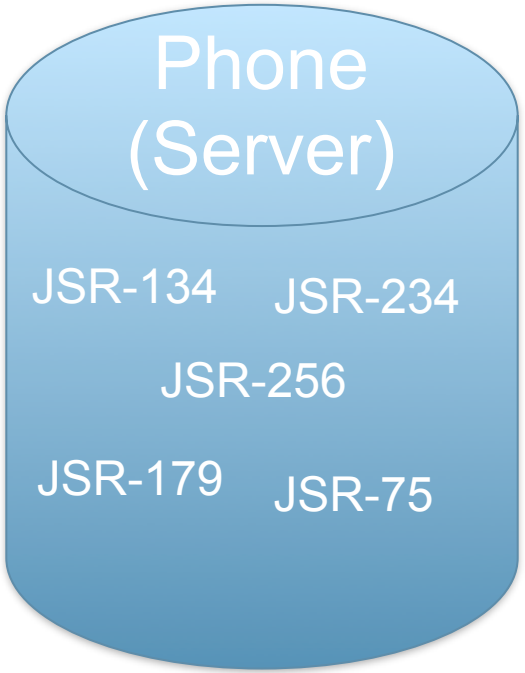
Systems which follow Fielding's REST principles are often referred to as "RESTful".

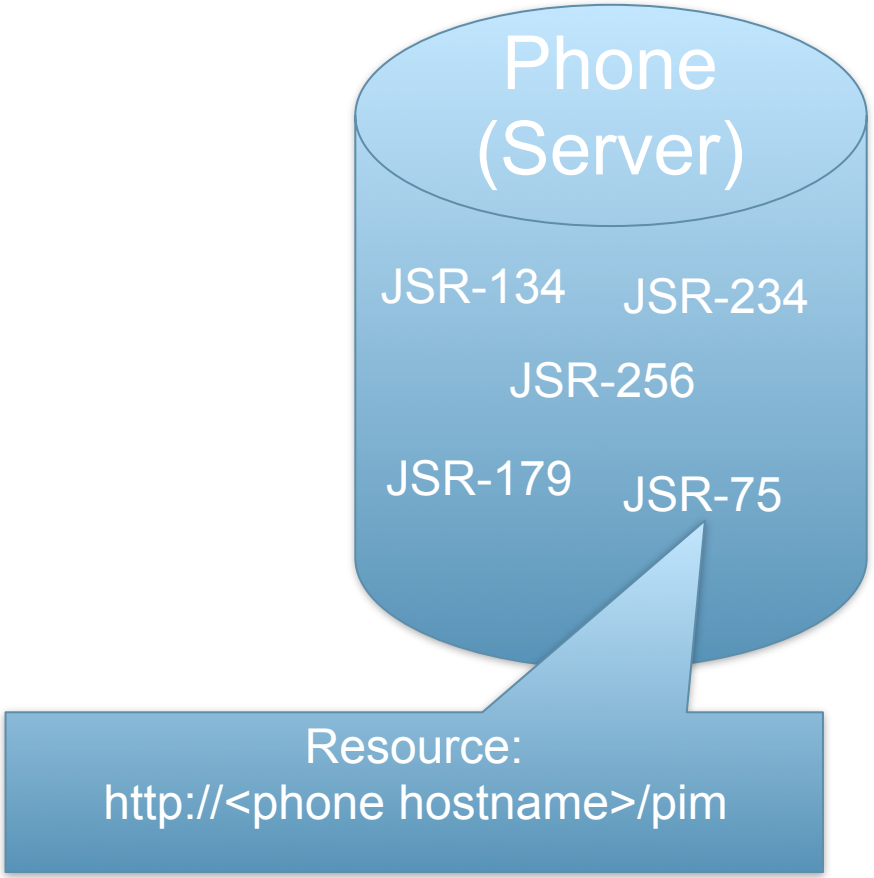
- Wikipedia, http://en.wikipedia.org/wiki/Representational_State_Transfer

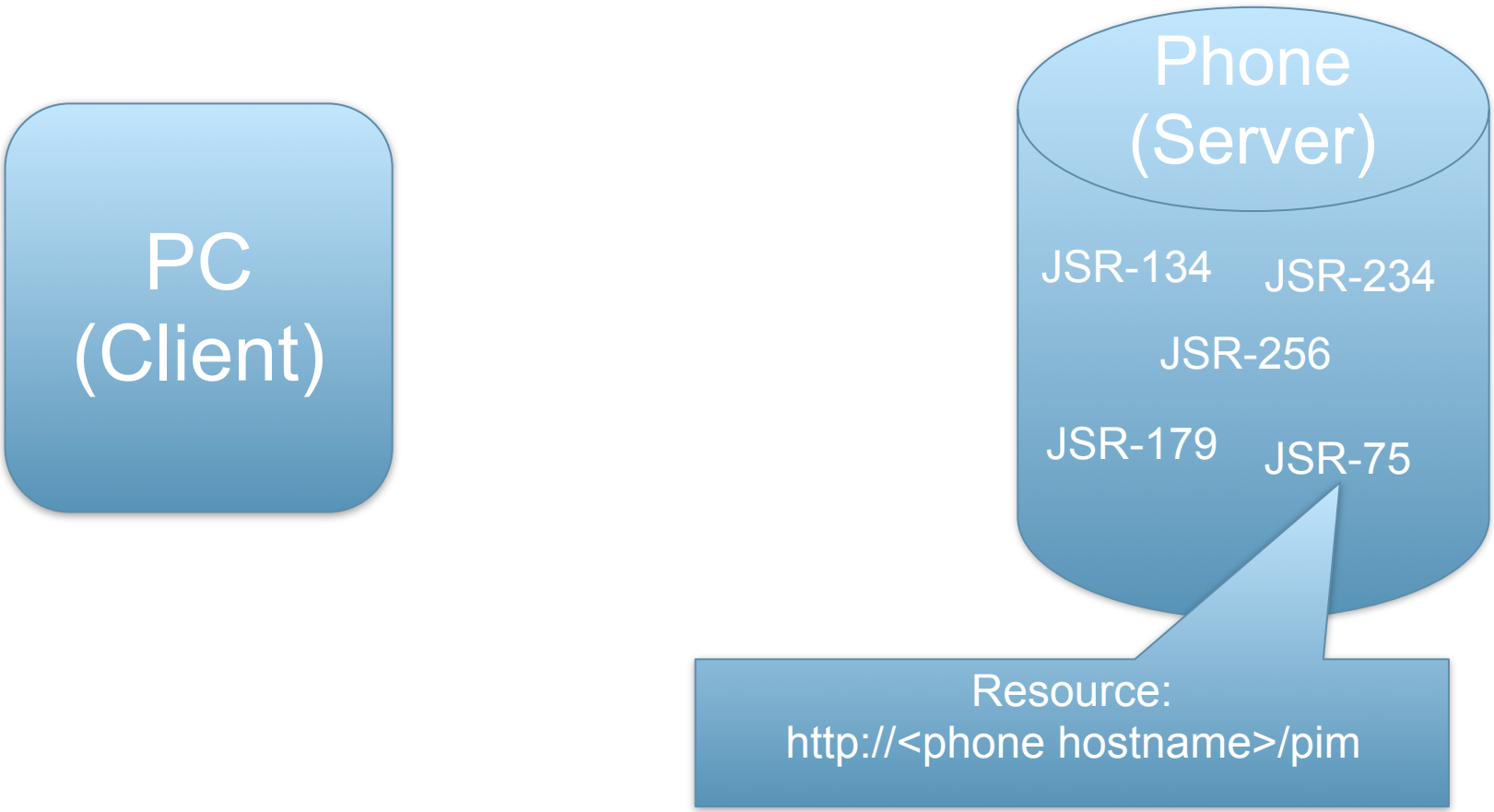
RESTful recap

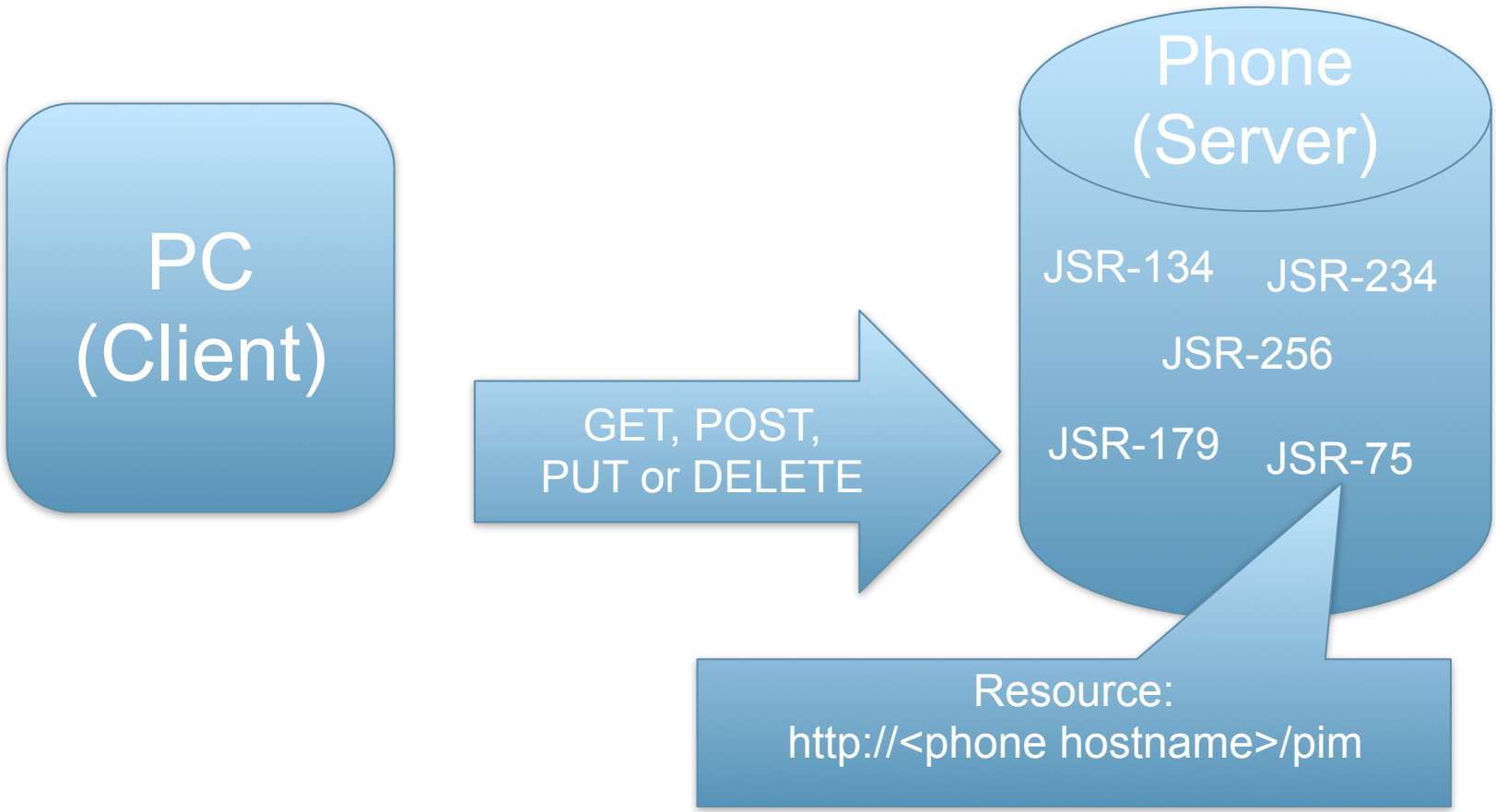
The difference between the uses of the term "REST" therefore causes some confusion in technical discussions.

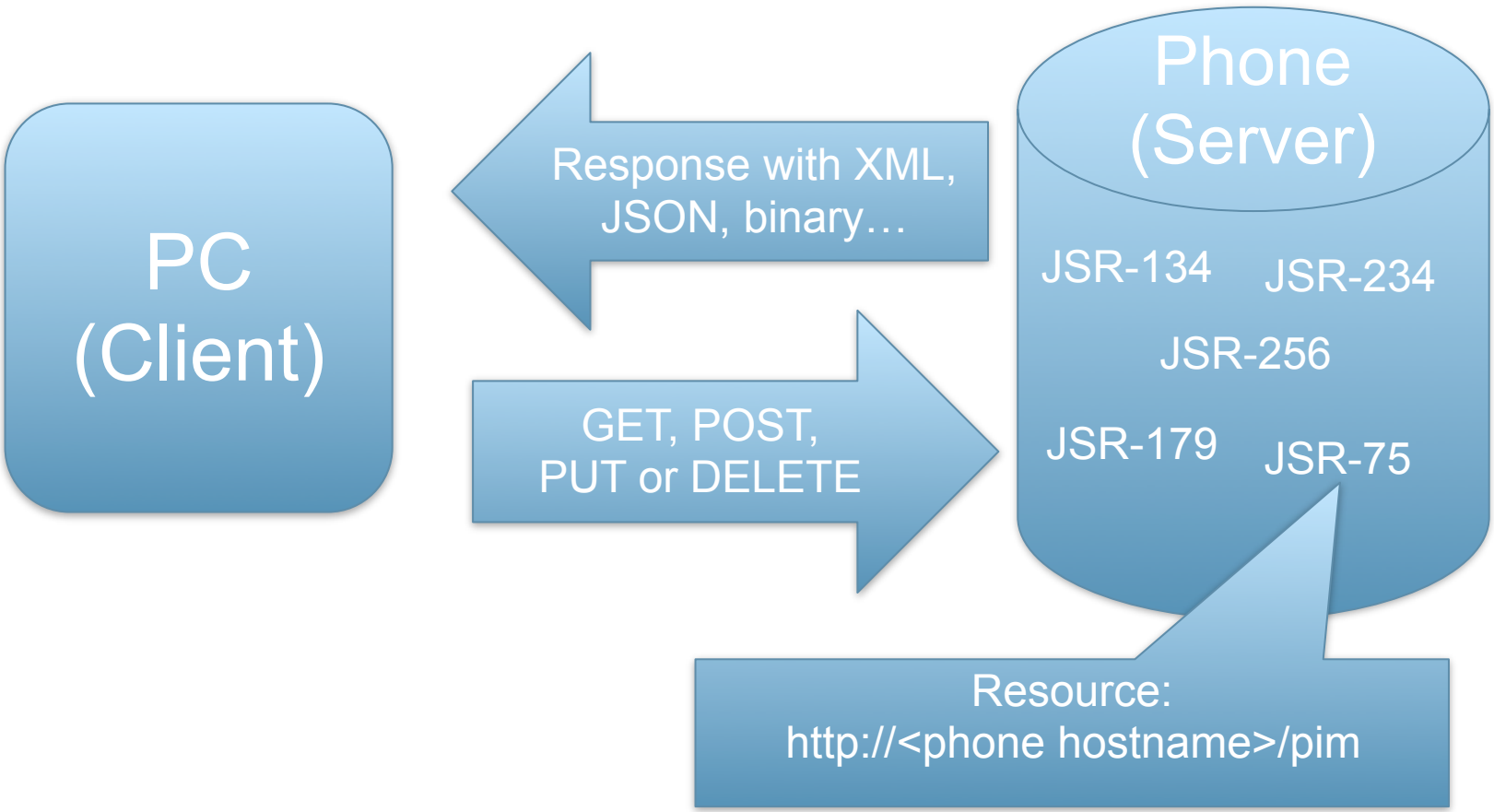
Systems which follow Fielding's REST principles are often referred to as "RESTful".


Phone
(Server)
JSR-134
JSR-234
JSR-256
JSR-179
JSR-75


Phone
(Server)
JSR-134
JSR-234
JSR-256
JSR-179
JSR-75
Resource:
http://<phone hostname>/pim

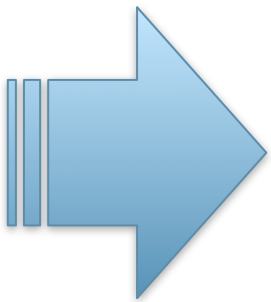

PC
(Client)
Phone
(Server)
JSR-134
JSR-234
JSR-256
JSR-179
JSR-75
Resource:
http://<phone hostname>/pim


PC
(Client)
GET, POST,
PUT or DELETE
Phone
(Server)
JSR-134
JSR-234
JSR-256
JSR-179
JSR-75
Resource:
http://<phone hostname>/pim


PC
(Client)
Response with XML,
JSON, binary…
GET, POST,
PUT or DELETE
Phone
(Server)
JSR-134
JSR-234
JSR-256
JSR-179
JSR-75
Resource:
http://<phone hostname>/pim

A Web Server on my Java ME phone?

http://www.flickr.com/photos/yatra/1098663076/



Some code!

Some code!

```
// HTTP server on port 8080
```

Some code!

```
// HTTP server on port 8080
String CONNECTION_URL = "socket://:8080";
```

Some code!

```
// HTTP server on port 8080
String CONNECTION_URL = "socket://:8080";
```

Some code!

```
// HTTP server on port 8080
String CONNECTION_URL = "socket://:8080";

// Open server socket
```

Some code!

```
// HTTP server on port 8080
String CONNECTION_URL = "socket://:8080";

// Open server socket
ServerSocketConnection ss;
```

Some code!

```
// HTTP server on port 8080
String CONNECTION_URL = "socket://:8080";

// Open server socket
ServerSocketConnection ss;
ss = (ServerSocketConnection) Connector.open
   (serverConnectionURL);
```

Some code!

```
// HTTP server on port 8080
String CONNECTION_URL = "socket://:8080";

// Open server socket
ServerSocketConnection ss;
ss = (ServerSocketConnection) Connector.open
   (serverConnectionURL);
```

Some code!

```
// HTTP server on port 8080
String CONNECTION_URL = "socket://:8080";

// Open server socket
ServerSocketConnection ss;
ss = (ServerSocketConnection) Connector.open
   (serverConnectionURL);

// Listen for incoming connections and spawn worker thread
```

Some code!

```
// HTTP server on port 8080
String CONNECTION_URL = "socket://:8080";

// Open server socket
ServerSocketConnection ss;
ss = (ServerSocketConnection) Connector.open
   (serverConnectionURL);

// Listen for incoming connections and spawn worker thread
while(runServer) {
```

Some code!

```
// HTTP server on port 8080
String CONNECTION_URL = "socket://:8080";

// Open server socket
ServerSocketConnection ss;
ss = (ServerSocketConnection) Connector.open
   (serverConnectionURL);

// Listen for incoming connections and spawn worker thread
while(runServer) {
   SocketConnection sc = serverSocket.acceptAndOpen();
```

Some code!

```
// HTTP server on port 8080
String CONNECTION_URL = "socket://:8080";

// Open server socket
ServerSocketConnection ss;
ss = (ServerSocketConnection) Connector.open
   (serverConnectionURL);

// Listen for incoming connections and spawn worker thread
while(runServer) {
   SocketConnection sc = serverSocket.acceptAndOpen();
   new Thread(new HttpWorkerThread(sc)).start();
```

Some code!

```
// HTTP server on port 8080
String CONNECTION_URL = "socket://:8080";

// Open server socket
ServerSocketConnection ss;
ss = (ServerSocketConnection) Connector.open
   (serverConnectionURL);

// Listen for incoming connections and spawn worker thread
while(runServer) {
   SocketConnection sc = serverSocket.acceptAndOpen();
   new Thread(new HttpWorkerThread(sc)).start();
}
```

Some more code!

```
public class HttpWorkerThread implements Runnable {
   private SocketConnection sc;

   public void run() {
      is = sc.openInputStream();
      os = sc.openOutputStream();
      String[] requestLine = readRequestLine(is);
      Hashtable headers = readHeaders(is);
      Servlet servlet = getServlet(requestLine);
      HttpRequest request = new HttpRequest(requestLine,
  headers, is);
      HttpResponse response = new HttpResponse(os);
      servlet.handleRequest(request, response);
   }
}
```

Even more code!

```
public class HelloWorldServlet implements Servlet {

    public void handleRequest(HttpRequest request,
                                        HttpResponse
   response);
        OutputStream os = response.getOutputStream();

        os.write(("Hello World at "
              + new Date().toString()).toBytes("UTF-8");
        os.flush();

        response.setResponseCode(HTTP_OK);
    }

}
```

Demo 1:

Hello World from
Java ME ”Servlet” container!

RESTful design of Java ME APIs

http://<phone ip>/pim/contacts/Erik+Hellman

JSR-75/PIM Example

```
// Vector for storing contacts from the PIM API
Vector contacts = new Vector();

// Open the ContactList
ContactList contactList = null;
contactList = (ContactList) PIM.getInstance().
   openPIMList(PIM.CONTACT_LIST, PIM.READ_WRITE);

// Read all Contacts to the Vector
Enumeration en = contactList.items();
while(en.hasMoreElements()) {
   Contact contact = (Contact) en.nextElement();
   contacts.addElement(contact);
}
```

Transforming to XML with KXML

Transforming to XML with KXML

```
KXmlSerializer xmlSerializer = new KXmlSerializer();
```

Transforming to XML with KXML

```
KXmlSerializer xmlSerializer = new KXmlSerializer();
xmlSerializer.setOutput(outputStream, "UTF-8");
```

Transforming to XML with KXML

```
KXmlSerializer xmlSerializer = new KXmlSerializer();
xmlSerializer.setOutput(outputStream, "UTF-8");
xmlSerializer.startDocument("UTF-8", Boolean.TRUE);
```

Transforming to XML with KXML

```
KXmlSerializer xmlSerializer = new KXmlSerializer();
xmlSerializer.setOutput(outputStream, "UTF-8");
xmlSerializer.startDocument("UTF-8", Boolean.TRUE);
xmlSerializer.startTag(null, “contacts”);
```

Transforming to XML with KXML

```
KXmlSerializer xmlSerializer = new KXmlSerializer();
xmlSerializer.setOutput(outputStream, "UTF-8");
xmlSerializer.startDocument("UTF-8", Boolean.TRUE);
xmlSerializer.startTag(null, “contacts”);
```

Transforming to XML with KXML

```
KXmlSerializer xmlSerializer = new KXmlSerializer();
xmlSerializer.setOutput(outputStream, "UTF-8");
xmlSerializer.startDocument("UTF-8", Boolean.TRUE);
xmlSerializer.startTag(null, “contacts”);

for(int i = 0; i < contacts.size(); i++) {
```

Transforming to XML with KXML

```
KXmlSerializer xmlSerializer = new KXmlSerializer();
xmlSerializer.setOutput(outputStream, "UTF-8");
xmlSerializer.startDocument("UTF-8", Boolean.TRUE);
xmlSerializer.startTag(null, “contacts”);

for(int i = 0; i < contacts.size(); i++) {
   Contact c = contacts.elementAt(i);
```

Transforming to XML with KXML

```
KXmlSerializer xmlSerializer = new KXmlSerializer();
xmlSerializer.setOutput(outputStream, "UTF-8");
xmlSerializer.startDocument("UTF-8", Boolean.TRUE);
xmlSerializer.startTag(null, “contacts”);

for(int i = 0; i < contacts.size(); i++) {
   Contact c = contacts.elementAt(i);
   xmlSerializer.starTag(null, ”contact”);
```

Transforming to XML with KXML

```
KXmlSerializer xmlSerializer = new KXmlSerializer();
xmlSerializer.setOutput(outputStream, "UTF-8");
xmlSerializer.startDocument("UTF-8", Boolean.TRUE);
xmlSerializer.startTag(null, “contacts”);

for(int i = 0; i < contacts.size(); i++) {
   Contact c = contacts.elementAt(i);
   xmlSerializer.starTag(null, ”contact”);
   xmlSerializer.starTag(null, ”name”);
```

Transforming to XML with KXML

```
KXmlSerializer xmlSerializer = new KXmlSerializer();
xmlSerializer.setOutput(outputStream, "UTF-8");
xmlSerializer.startDocument("UTF-8", Boolean.TRUE);
xmlSerializer.startTag(null, “contacts”);

for(int i = 0; i < contacts.size(); i++) {
   Contact c = contacts.elementAt(i);
   xmlSerializer.starTag(null, ”contact”);
   xmlSerializer.starTag(null, ”name”);
   xmlSerializer.text(contact.
          getString(Contact.FORMATTED_NAME));
```

Transforming to XML with KXML

```
KXmlSerializer xmlSerializer = new KXmlSerializer();
xmlSerializer.setOutput(outputStream, "UTF-8");
xmlSerializer.startDocument("UTF-8", Boolean.TRUE);
xmlSerializer.startTag(null, “contacts”);

for(int i = 0; i < contacts.size(); i++) {
   Contact c = contacts.elementAt(i);
   xmlSerializer.starTag(null, ”contact”);
   xmlSerializer.starTag(null, ”name”);
   xmlSerializer.text(contact.
          getString(Contact.FORMATTED_NAME));
   xmlSerializer.endTag(null, ”name”);
```

Transforming to XML with KXML

```
KXmlSerializer xmlSerializer = new KXmlSerializer();
xmlSerializer.setOutput(outputStream, "UTF-8");
xmlSerializer.startDocument("UTF-8", Boolean.TRUE);
xmlSerializer.startTag(null, “contacts”);

for(int i = 0; i < contacts.size(); i++) {
   Contact c = contacts.elementAt(i);
   xmlSerializer.starTag(null, ”contact”);
   xmlSerializer.starTag(null, ”name”);
   xmlSerializer.text(contact.
          getString(Contact.FORMATTED_NAME));
   xmlSerializer.endTag(null, ”name”);
   // Write the remaining field…
```

Transforming to XML with KXML

```
KXmlSerializer xmlSerializer = new KXmlSerializer();
xmlSerializer.setOutput(outputStream, "UTF-8");
xmlSerializer.startDocument("UTF-8", Boolean.TRUE);
xmlSerializer.startTag(null, “contacts”);

for(int i = 0; i < contacts.size(); i++) {
   Contact c = contacts.elementAt(i);
   xmlSerializer.starTag(null, ”contact”);
   xmlSerializer.starTag(null, ”name”);
   xmlSerializer.text(contact.
          getString(Contact.FORMATTED_NAME));
   xmlSerializer.endTag(null, ”name”);
   // Write the remaining field…
```

Transforming to XML with KXML

```
KXmlSerializer xmlSerializer = new KXmlSerializer();
xmlSerializer.setOutput(outputStream, "UTF-8");
xmlSerializer.startDocument("UTF-8", Boolean.TRUE);
xmlSerializer.startTag(null, “contacts”);

for(int i = 0; i < contacts.size(); i++) {
   Contact c = contacts.elementAt(i);
   xmlSerializer.starTag(null, ”contact”);
   xmlSerializer.starTag(null, ”name”);
   xmlSerializer.text(contact.
          getString(Contact.FORMATTED_NAME));
   xmlSerializer.endTag(null, ”name”);
   // Write the remaining field…

   xmlSerializer.endTag(null, ”contact”);
```

Transforming to XML with KXML

```
KXmlSerializer xmlSerializer = new KXmlSerializer();
xmlSerializer.setOutput(outputStream, "UTF-8");
xmlSerializer.startDocument("UTF-8", Boolean.TRUE);
xmlSerializer.startTag(null, “contacts”);

for(int i = 0; i < contacts.size(); i++) {
   Contact c = contacts.elementAt(i);
   xmlSerializer.starTag(null, ”contact”);
   xmlSerializer.starTag(null, ”name”);
   xmlSerializer.text(contact.
          getString(Contact.FORMATTED_NAME));
   xmlSerializer.endTag(null, ”name”);
   // Write the remaining field…

   xmlSerializer.endTag(null, ”contact”);
}
```

Transforming to XML with KXML

```
KXmlSerializer xmlSerializer = new KXmlSerializer();
xmlSerializer.setOutput(outputStream, "UTF-8");
xmlSerializer.startDocument("UTF-8", Boolean.TRUE);
xmlSerializer.startTag(null, “contacts”);

for(int i = 0; i < contacts.size(); i++) {
   Contact c = contacts.elementAt(i);
   xmlSerializer.starTag(null, ”contact”);
   xmlSerializer.starTag(null, ”name”);
   xmlSerializer.text(contact.
          getString(Contact.FORMATTED_NAME));
   xmlSerializer.endTag(null, ”name”);
   // Write the remaining field…

   xmlSerializer.endTag(null, ”contact”);
}
```

Transforming to XML with KXML

```
KXmlSerializer xmlSerializer = new KXmlSerializer();
xmlSerializer.setOutput(outputStream, "UTF-8");
xmlSerializer.startDocument("UTF-8", Boolean.TRUE);
xmlSerializer.startTag(null, “contacts”);

for(int i = 0; i < contacts.size(); i++) {
   Contact c = contacts.elementAt(i);
   xmlSerializer.starTag(null, ”contact”);
   xmlSerializer.starTag(null, ”name”);
   xmlSerializer.text(contact.
          getString(Contact.FORMATTED_NAME));
   xmlSerializer.endTag(null, ”name”);
   // Write the remaining field…

   xmlSerializer.endTag(null, ”contact”);
}

xmlSerializer.endTag(null, ”contacts”);
```

Transforming to XML with KXML

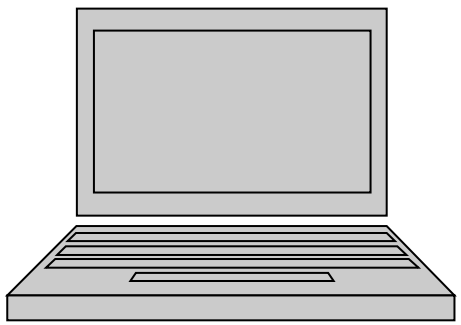
```
KXmlSerializer xmlSerializer = new KXmlSerializer();
xmlSerializer.setOutput(outputStream, "UTF-8");
xmlSerializer.startDocument("UTF-8", Boolean.TRUE);
xmlSerializer.startTag(null, “contacts”);

for(int i = 0; i < contacts.size(); i++) {
   Contact c = contacts.elementAt(i);
   xmlSerializer.starTag(null, ”contact”);
   xmlSerializer.starTag(null, ”name”);
   xmlSerializer.text(contact.
          getString(Contact.FORMATTED_NAME));
   xmlSerializer.endTag(null, ”name”);
   // Write the remaining field…

   xmlSerializer.endTag(null, ”contact”);
}

xmlSerializer.endTag(null, ”contacts”);
xmlSerializer.endDocument();
```

Demo 2:

PC client for phone contacts.

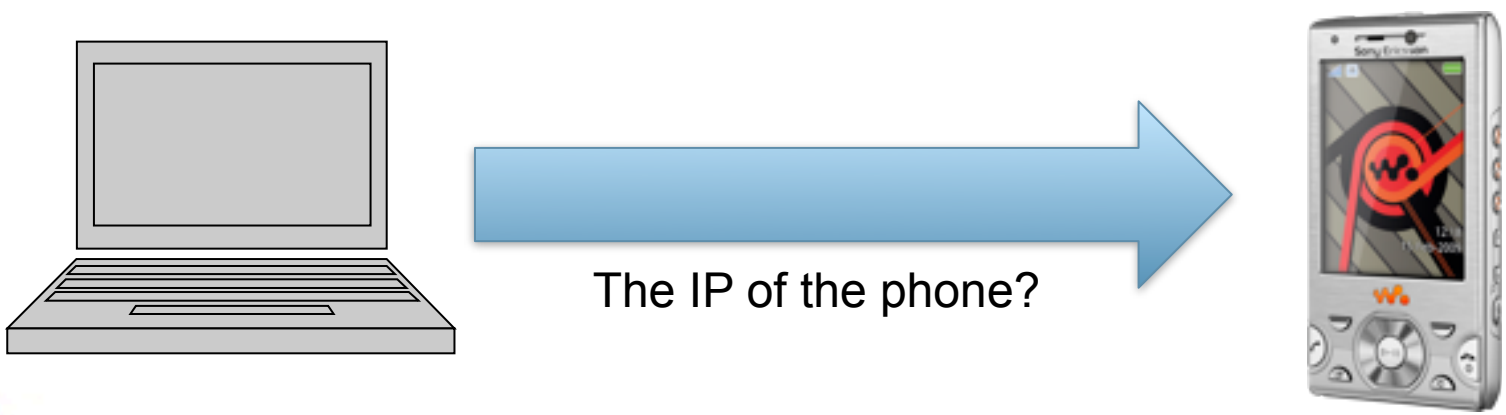
JSR-75 (PIM API)

Online access to phone contacts

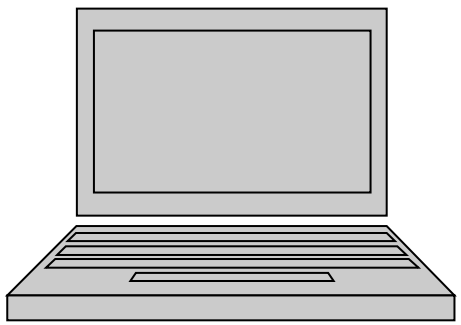


Online access to phone contacts

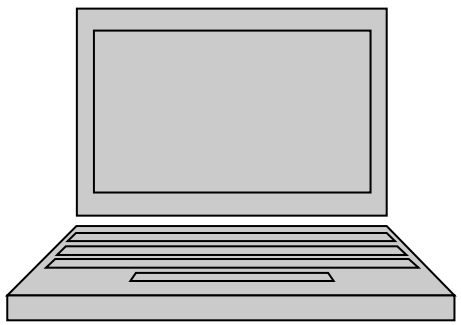
Online access to phone contacts



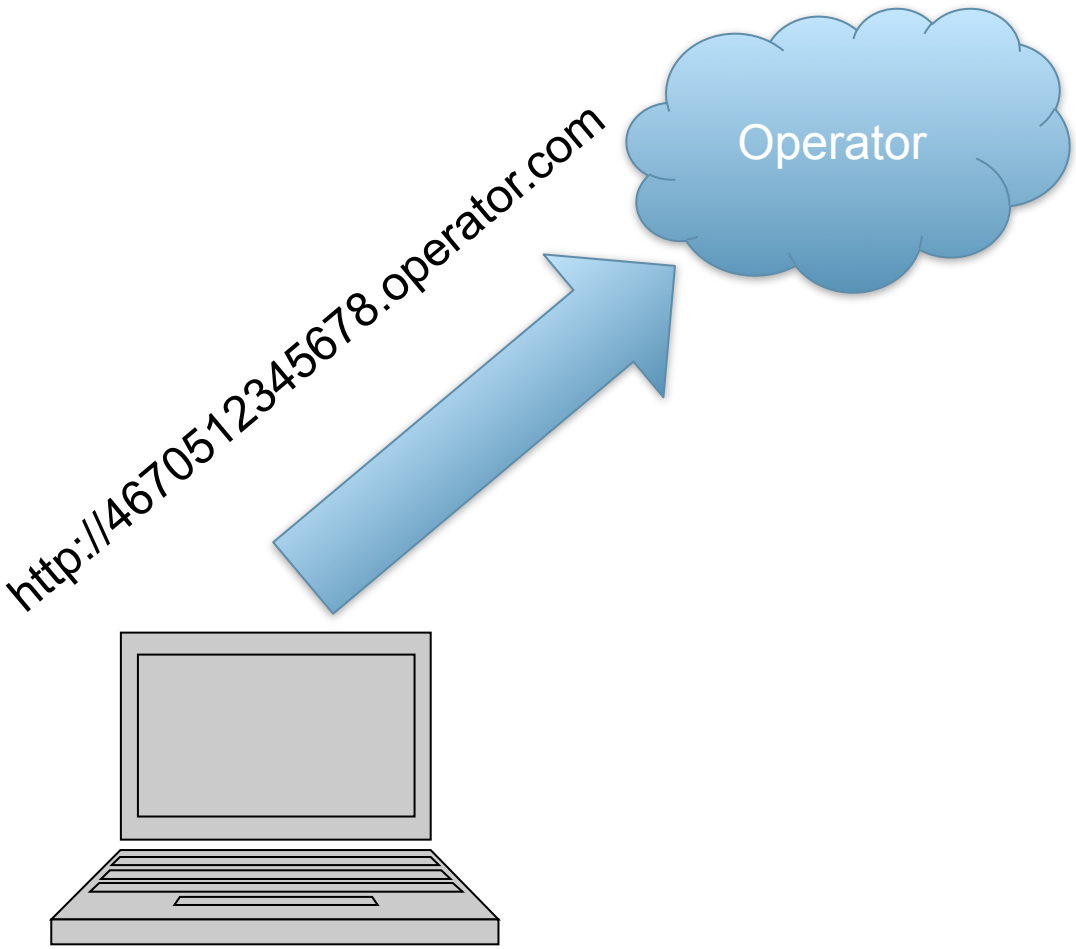
Online access to phone contacts



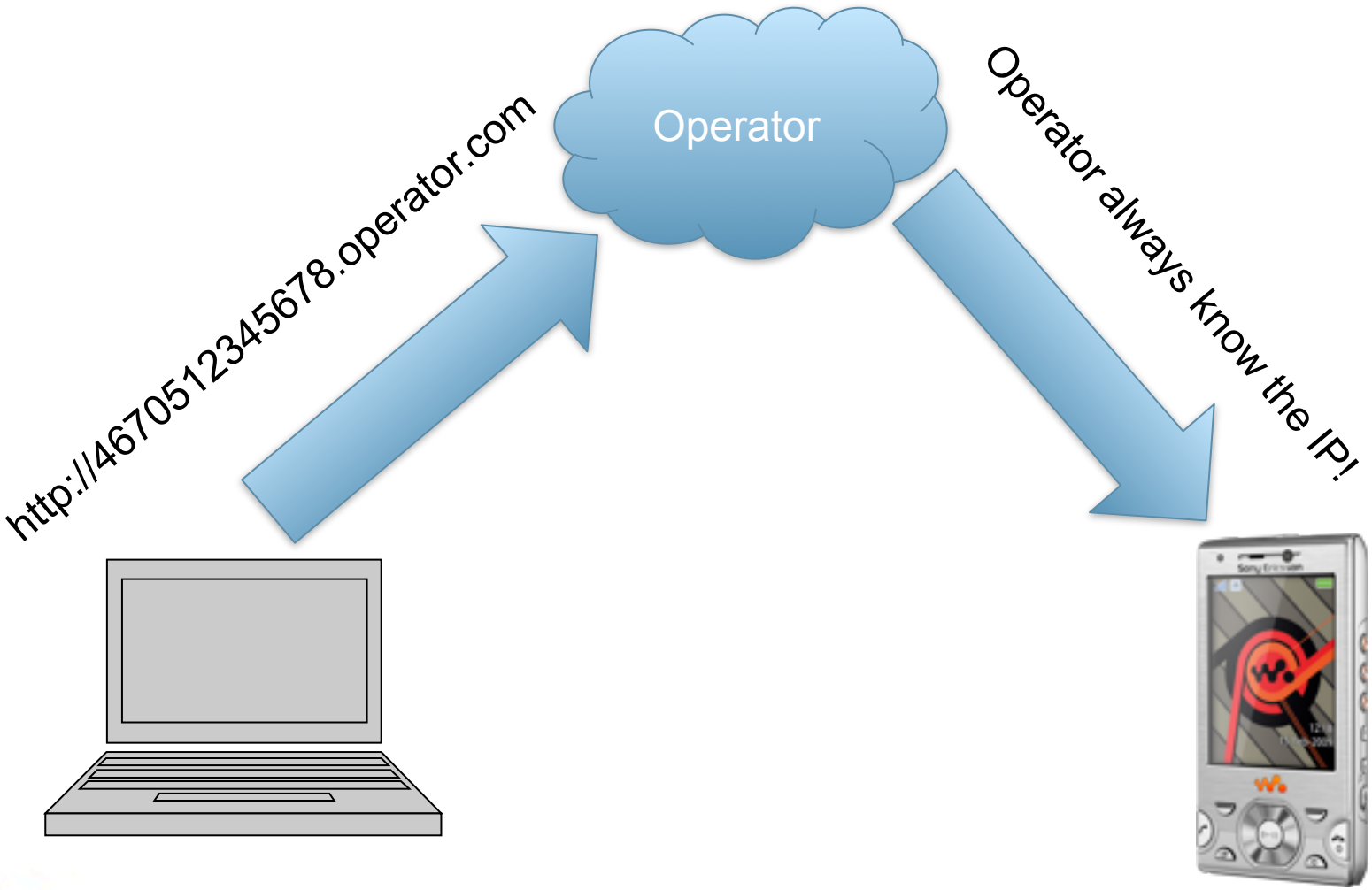
Online access to phone contacts



Online access to phone contacts



Online access to phone contacts



Demo 3:

Web-based file browser.

JSR-75 (FileConnection)

Code: File access

```
FileConnection fc = (FileConnection) Connector.open
   ("file:///e:" + requestPath);
if(!fc.isDirectory()) {
    String name = fc.getName();
    if(name.endsWith(".jpg")) {
       response.addHeader("Content-Type","image/jpeg");
       response.addHeader("Content-Length",
                          fc.fileSize());
    }

    InputStream is = fc.openInputStream();
    // Write the file content to the HttpResponse…
}
```

Demo 4:

My Mobile Web Cam!

RESTful access to JSR-135

Code: Taking snapshots

Code: Taking snapshots

```
// Create a new Player object connected to video capture
```

Code: Taking snapshots

```
// Create a new Player object connected to video capture
Player player = Manager.createPlayer("capture://video");
```

Code: Taking snapshots

```
// Create a new Player object connected to video capture
Player player = Manager.createPlayer("capture://video");
player.realize();
```

Code: Taking snapshots

```
// Create a new Player object connected to video capture
Player player = Manager.createPlayer("capture://video");
player.realize();

// Create a new VideoControl to take snapshots
```

Code: Taking snapshots

```
// Create a new Player object connected to video capture
Player player = Manager.createPlayer("capture://video");
player.realize();

// Create a new VideoControl to take snapshots
VideoControl  vc = (VideoControl) player.getControl
   ("VideoControl");
```

Code: Taking snapshots

```
// Create a new Player object connected to video capture
Player player = Manager.createPlayer("capture://video");
player.realize();

// Create a new VideoControl to take snapshots
VideoControl  vc = (VideoControl) player.getControl
   ("VideoControl");
```

Code: Taking snapshots

```
// Create a new Player object connected to video capture
Player player = Manager.createPlayer("capture://video");
player.realize();

// Create a new VideoControl to take snapshots
VideoControl  vc = (VideoControl) player.getControl
   ("VideoControl");

player.start();
```

Code: Taking snapshots

```
// Create a new Player object connected to video capture
Player player = Manager.createPlayer("capture://video");
player.realize();

// Create a new VideoControl to take snapshots
VideoControl  vc = (VideoControl) player.getControl
   ("VideoControl");

player.start();
```

Code: Taking snapshots

```
// Create a new Player object connected to video capture
Player player = Manager.createPlayer("capture://video");
player.realize();

// Create a new VideoControl to take snapshots
VideoControl  vc = (VideoControl) player.getControl
   ("VideoControl");

player.start();

// take snapeshot with default encoding
```

Code: Taking snapshots

```
// Create a new Player object connected to video capture
Player player = Manager.createPlayer("capture://video");
player.realize();

// Create a new VideoControl to take snapshots
VideoControl  vc = (VideoControl) player.getControl
   ("VideoControl");

player.start();

// take snapeshot with default encoding
byte[] imageData = vc.getSnapshot(null);
```

Code: Taking snapshots

```
// Create a new Player object connected to video capture
Player player = Manager.createPlayer("capture://video");
player.realize();

// Create a new VideoControl to take snapshots
VideoControl  vc = (VideoControl) player.getControl
   ("VideoControl");

player.start();

// take snapeshot with default encoding
byte[] imageData = vc.getSnapshot(null);
```

Code: Taking snapshots

```
// Create a new Player object connected to video capture
Player player = Manager.createPlayer("capture://video");
player.realize();

// Create a new VideoControl to take snapshots
VideoControl  vc = (VideoControl) player.getControl
   ("VideoControl");

player.start();

// take snapeshot with default encoding
byte[] imageData = vc.getSnapshot(null);

// Write imageData to the HttpResponse
```

Demo 5:

Where Is My Phone?

RESTful access to JSR-179 (Location API and GPS)

Code: Location API

Code: Location API

```
// init the LocationProvider with default criteria
```

Code: Location API

```
// init the LocationProvider with default criteria
locationProvider = LocationProvider.getInstance(new Criteria());
```

Code: Location API

```
// init the LocationProvider with default criteria
locationProvider = LocationProvider.getInstance(new Criteria());
// Get the currently last known location
```

Code: Location API

```
// init the LocationProvider with default criteria
locationProvider = LocationProvider.getInstance(new Criteria());
// Get the currently last known location
currentLocation = LocationProvider.getLastKnownLocation();
```

Code: Location API

```
// init the LocationProvider with default criteria
locationProvider = LocationProvider.getInstance(new Criteria());
// Get the currently last known location
currentLocation = LocationProvider.getLastKnownLocation();
// Listen for updates with default parameters
```

Code: Location API

```
// init the LocationProvider with default criteria
locationProvider = LocationProvider.getInstance(new Criteria());
// Get the currently last known location
currentLocation = LocationProvider.getLastKnownLocation();
// Listen for updates with default parameters
locationProvider.setLocationListener(this, -1, -1, -1);
```

Code: Location API

```
// init the LocationProvider with default criteria
locationProvider = LocationProvider.getInstance(new Criteria());
// Get the currently last known location
currentLocation = LocationProvider.getLastKnownLocation();
// Listen for updates with default parameters
locationProvider.setLocationListener(this, -1, -1, -1);
```

Code: Location API

```
// init the LocationProvider with default criteria
locationProvider = LocationProvider.getInstance(new Criteria());
// Get the currently last known location
currentLocation = LocationProvider.getLastKnownLocation();
// Listen for updates with default parameters
locationProvider.setLocationListener(this, -1, -1, -1);

// Callback function for location updates
```

Code: Location API

```
// init the LocationProvider with default criteria
locationProvider = LocationProvider.getInstance(new Criteria());
// Get the currently last known location
currentLocation = LocationProvider.getLastKnownLocation();
// Listen for updates with default parameters
locationProvider.setLocationListener(this, -1, -1, -1);

// Callback function for location updates
public void locationUpdated(LocationProvider locationProvider,
   Location location) {
```

Code: Location API

```
// init the LocationProvider with default criteria
locationProvider = LocationProvider.getInstance(new Criteria());
// Get the currently last known location
currentLocation = LocationProvider.getLastKnownLocation();
// Listen for updates with default parameters
locationProvider.setLocationListener(this, -1, -1, -1);

// Callback function for location updates
public void locationUpdated(LocationProvider locationProvider,
   Location location) {
    if (location != null &&
        location.getQualifiedCoordinates() != null &&
        location.isValid()) {
```

Code: Location API

```
// init the LocationProvider with default criteria
locationProvider = LocationProvider.getInstance(new Criteria());
// Get the currently last known location
currentLocation = LocationProvider.getLastKnownLocation();
// Listen for updates with default parameters
locationProvider.setLocationListener(this, -1, -1, -1);

// Callback function for location updates
public void locationUpdated(LocationProvider locationProvider,
   Location location) {
    if (location != null &&
        location.getQualifiedCoordinates() != null &&
        location.isValid()) {
        currentLocation = location;
```

Code: Location API

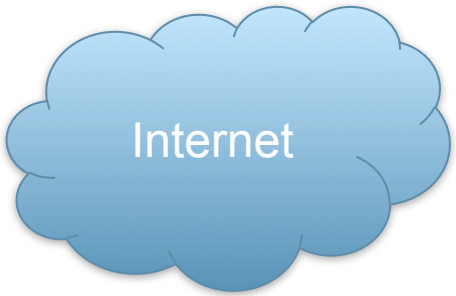
```
// init the LocationProvider with default criteria
locationProvider = LocationProvider.getInstance(new Criteria());
// Get the currently last known location
currentLocation = LocationProvider.getLastKnownLocation();
// Listen for updates with default parameters
locationProvider.setLocationListener(this, -1, -1, -1);

// Callback function for location updates
public void locationUpdated(LocationProvider locationProvider,
   Location location) {
    if (location != null &&
        location.getQualifiedCoordinates() != null &&
        location.isValid()) {
        currentLocation = location;
    }
```

Code: Location API

```
// init the LocationProvider with default criteria
locationProvider = LocationProvider.getInstance(new Criteria());
// Get the currently last known location
currentLocation = LocationProvider.getLastKnownLocation();
// Listen for updates with default parameters
locationProvider.setLocationListener(this, -1, -1, -1);

// Callback function for location updates
public void locationUpdated(LocationProvider locationProvider,
   Location location) {
    if (location != null &&
        location.getQualifiedCoordinates() != null &&
        location.isValid()) {
        currentLocation = location;
    }
}
```

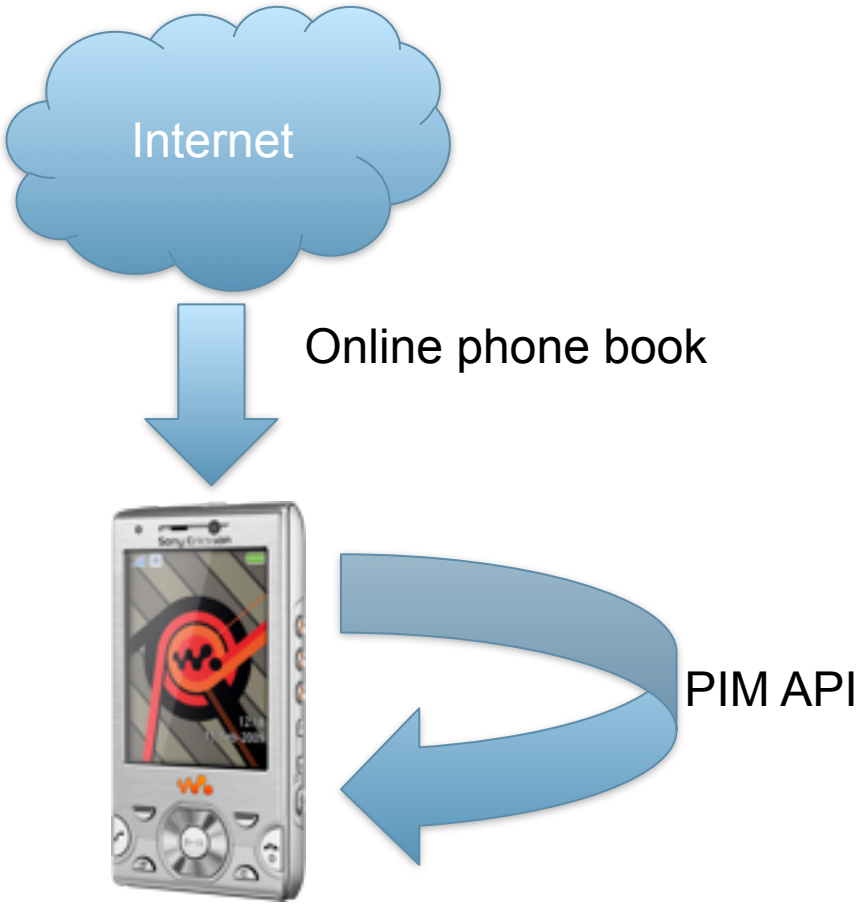
Mixing local and online content

Mixing local and online content



Mixing local and online content

Mixing local and online content



Demo 6:

Web Enabled Home Screen!

A word on Same-Origin-Policy

- Only relevant for JavaScript in browsers
- Add support for a callback function when necessary

What about security?

What about security?

- Access control options
  - IP and domain whitelist
  - API Key
  - Client-side certificates
  - Manual authorization per session
  - Manual verification of shared secret
  - OAuth?

  - All of the above?

Additional use-cases

- Multi-vendor, multi-platform tools and applications
- Remote Synchronization of Media (music, video and photos)
- Rapid enabling of additional platform runtimes (e.g., Ruby, Scala, Perl, Python etc.)
- Easy content sharing (phonebook, media etc.)
- Your own personal, mobile blog! 

Obstacles

- IP communication to phone not always possible.
- Sensitive to DoS attacks…
- Security!
- Highly constrained devices – very limited throughput and memory available.

Conclusions

Conclusions

- Easy integration - Web is an ubiquitous platform.

Conclusions

- Easy integration - Web is an ubiquitous platform.
- Access phone services anywhere.

Conclusions

- Easy integration - Web is an ubiquitous platform.
- Access phone services anywhere.
- Enable new types of services and runtimes.

Conclusions

- Easy integration - Web is an ubiquitous platform.
- Access phone services anywhere.
- Enable new types of services and runtimes.

Conclusions

- Easy integration - Web is an ubiquitous platform.
- Access phone services anywhere.
- Enable new types of services and runtimes.

Conclusions

- Easy integration - Web is an ubiquitous platform.
- Access phone services anywhere.
- Enable new types of services and runtimes.

- All demos are available at http://developer.sonyericsson.com

Erik Hellman
erik.hellman@sonyericsson.com
http://twitter.com/ErikHellman

http://developer.sonyericsson.com