Android-II

Cours Android II

13-17 avril 2015 Cesi Nanterre

Instructeur Ph.Dutron

WEB SERVICES

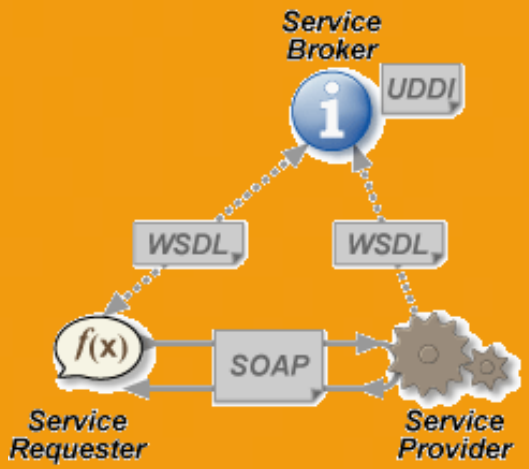
WEB SERVICES

- Overview
- A web service is:
  - a method of communication between two electronic devices over the World Wide Web.
  - a software function provided at a network address over the web or the cloud
- The W3C defined a Web Services Architecture
  - a Web service has an interface described in a machine-processable format .
  - Other systems interact with the Web service in a manner prescribed by its description using SOAP (Simple Object Access Protocol) messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.
- Most Web services do not adopt this complex architecture.
  - We can identify two major classes of Web services:
    - REST-compliant Web services
    - Arbitrary Web services, in which the service may expose an arbitrary set of operations.

WEB SERVICES

- Web services are a means of exposing an API over a technology-neutral network endpoint.
- Web services standards are still evolving.
  - Seem to converge on a handful of standards:
    - SOAP for service communication,
    - WSDL for service description,
    - UDDI for registering and discovering services,
    - BPEL(Business Process Execution Language) for service composition.
    - A plethora of WS-* specifications exists to describe the full spectrum of activities related to Web services in topics such as reliable messaging, security, privacy, policies, event processing, and coordination, to name but a few.
  - SOAP is a complex standard. Some packages do exist on Android.

Web Service Architectures



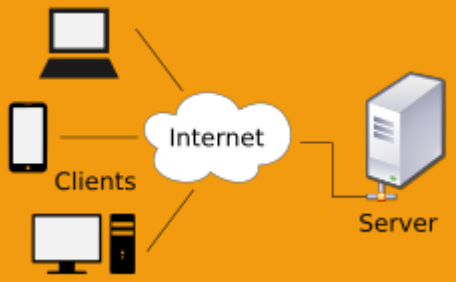
WSDL,
Web Services Description Language
UDDI
Universal Description Discovery and Infrastructure
SOAP
Simple Object Access Protocol

Concerns
non-RESTful Web services are complex and based upon large software vendors or integrators, rather than typical open source implementations.

Performance issues due to Web services'use of XML as a message format and SOAP/HTTP in enveloping and transporting.

SOAP is a protocol not an architecture

Web Service Architectures



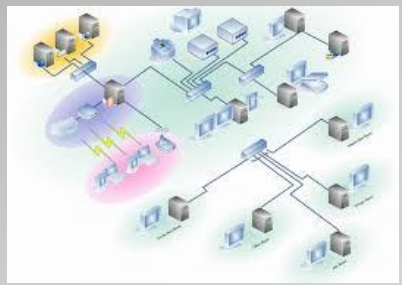
RestFull

- Client-server
- Stateless
- Cacheable
- Layered system
- Code on demand
- Uniform interface

WEB SERVICES

- Some web services directories
- http://www.publicapis.com/
- http://www.webservicex.net
- http://www.programmableweb.com/
- http://www.webservicelist.com/

WEB SERVICES



- NetWork Basics
- NetWork: group of interconnected computers

- TCP/IP
  - Node: each adressed device in a network is a node
  - Protocols: predefined and agreed-upon set of rules for communication
  - Protocols organized as layers
    - TCP/IP stack
    - Link Layer—Physical device address resolution protocols such as ARP and RARP
    - Internet Layer—IP itself, which has multiple versions, the ping protocol, and
    - ICMP, among others
    - Transport Layer—Different types of delivery protocols such as TCP and UDP
    - Application Layer—Familiar protocols such as HTTP, FTP, SMTP, IMAP, POP, DNS, SSH, and SOAP

WEB SERVICES

- IP (Internet Packet)
  - IP adress ipv4 = 32 bits, ipv6 = 48 bits
- Delivery Protocols
  - TCP UDP (user datagram protocol) delivery protocols used over TCP/IP
- Application Protocols
  - (Simple Mail Transfer Protocol) SMTP , HTTP
- Clients and servers
  - Ports
    - Well-known ports—0 through 1023
    - Registered ports—1024 through 49151
    - Dynamic and/or private ports—49152 through 65535

WEB SERVICES

- Network Status
  - Use ConnectivityManager

```
@Override
public void onStart() {
super.onStart();
ConnectivityManager cMgr = (ConnectivityManager)
this.getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo netInfo = cMgr.getActiveNetworkInfo();
this.status.setText(netInfo.toString());
}
```

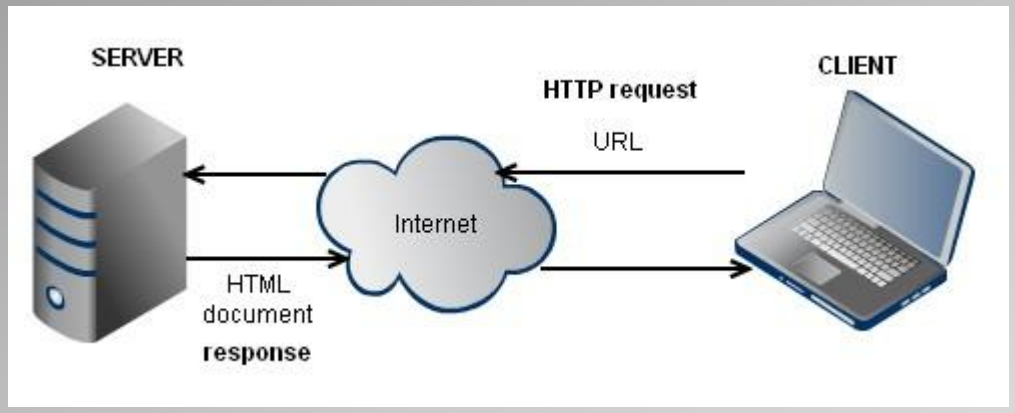
  - Once connected you can use IP Network

WEB SERVICES

- Communicating with a server socket
  - Server Socket
    - Stream to read/write raw bytes
    - get local IP using ipconfig

WEB SERVICES

- Communicating using HTTP protocol
  - http



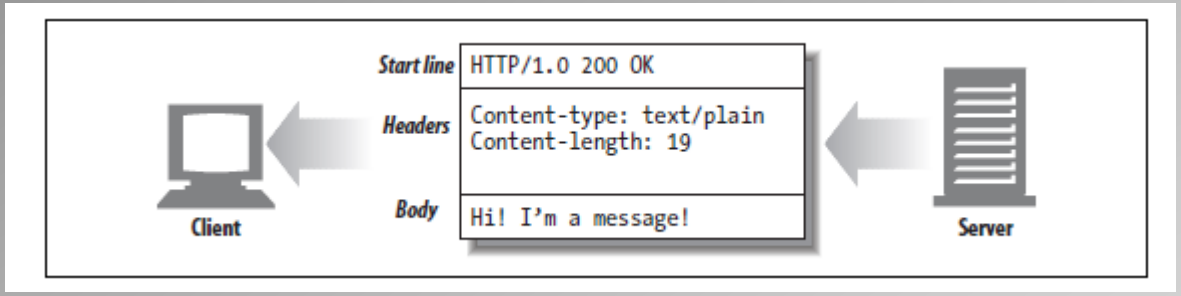
  - Servers host ressources
    - Db, files, images, html pages, Services …
    - Data types are definedby their MIME type
    - Each ressources has an identifier URI (uniform ressource identifier), URL (uniform ressource locator) gives the location

WEB SERVICES

- URL http://www.fast.hardware.com/specials/hammer.gif
  - Scheme : http://
  - Adress : www.fast.hardware.com
  - Ressource : specials/hammer.gif
- Transactions
  - Requests
  - Responses
  - Messages
  - Methods
    - GET Send named resource from the server to the client.
    - POST Send client data into a server gateway application.
    - PUT Store data from client into a named server resource.
    - DELETE Delete the named resource from a server.
    - HEADER Send HTTP headers from the response for the named resource.
  - Status codes
    - 200 Ok …

WEB SERVICES

- Messages
- 3 parts



  - VERB URL VERSION (Start line)
  - HEADERS
  - BODY

```
GET / HTTP/1.1
Accept: text/*
Host: www.microsoft.com

HTTP/1.1 200 OK
Date: Thu, 01 Jun 2006 06:34:41 GMT
Server: Microsoft-IIS/6.0
P3P: CP="ALL IND DSP COR ADM CONo CUR CUSo IVAo IVDo PSA PSD TAI TELo OUR SAMo CNT COM INT NAV ONL PHY PRE PUR UNI"
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727
Cache-Control: private
Content-Type: text/html; charset=utf-8
Content-Length: 30430
```


WEB SERVICES

- using HTTP protocol
- Android support two Http clients
- HttpURLConnection
- Or
- HttpClient (Apache)

- Pattern for usingHttpURLConnection
  - Http methods
    - Get is used by default
    - Post if setDoOutput(true)

WEB SERVICES

- Performance
  - ANR issues
  - solution
    - Perform network operations in a separate Thread
  - You cannot execute a network task in the main thread
  - You cannot update views from a background thread

FireBase

- Realtime Database
- Storage
- Reporting
- Authentication
- Cloud Messaging

Managing background activities

Executing background tasks

- Reminder
  - Java multithreading
  - Android implementations
    - asyncTask
    - Handler, messages
    - Download manager
    - IntentService

Executing background tasks

- Java Multithreading
- Two ways to implement a thread
  - Create class to Extend Thread
  - Define run()
  - Start
  - Example

```
class PrimeThread extends Thread {
      long minPrime;
      PrimeThread(long minPrime) {
         this.minPrime = minPrime;
      }

      public void run() {
         // compute primes larger than minPrime
          . . .
      }
   }
//start the thread
PrimeThread p = new PrimeThread(143);
   p.start();

//notes
thread stops
     On exit() call
     After execiting run()
A thread has a priority level
```

Executing background tasks

- Java Multithreading
  - Implement Runnable interface
    - Create class to implement runnable
    - Define run()
    - Start
    - Example

```
class PrimeRun implements Runnable {
      long minPrime;
      PrimeRun(long minPrime) {
         this.minPrime = minPrime;
      }

      public void run() {
         // compute primes larger than minPrime
         . . .
      }
   }
//start it
PrimeRun p = new PrimeRun(143);
   new Thread(p).start();
```

Executing background tasks

- Android
  - asyncTask, used for short operations (few seconds)
    - Allows to perform background operations
    - Publish results in UI thread
    - Uses 3 types :Parameters , Progress, Result
    - Uses 4 steps : onPreExec

```
private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {
    protected Long doInBackground(URL... urls) {
        int count = urls.length;
        long totalSize = 0;
        for (int i = 0; i < count; i++) {
            totalSize += Downloader.downloadFile(urls[i]);
            publishProgress((int) ((i / (float) count) * 100));
            // Escape early if cancel() is called
            if (isCancelled()) break;
        }
        return totalSize;
    }

    protected void onProgressUpdate(Integer... progress) {
        setProgressPercent(progress[0]);
    }

    protected void onPostExecute(Long result) {
        showDialog("Downloaded " + result + " bytes");
    }
}

//Update UI on progress triggered by publicProgress
```

Executing background tasks

- Android
  - asyncTask, 4 steps
  - onPreExecute()
    - invoked on the UI thread before the task is executed. onUpdate
  - doInBackground(Params...)
    - invoked on the background thread immediately after onPreExecute() finishes executing.
    - The result of the computation must be returned by this step and will be passed back to the last step onPostExecute().
    - This step can also use publishProgress(Progress...). Values are published on the UI thread, in the onProgressUpdate(Progress...) step.
  - onProgressUpdate(Progress...), invoked on the UI thread after a call to publishProgress(Progress...).
    - The timing of the execution is undefined.
  - onPostExecute(Result), invoked on the UI thread after the background computation finishes.

Executing background tasks

- Android
  - Handler and Messages
    - Send and process messages and Runnable
    - Processed by a message queue
    - Processed by a separate thread
    - Messages and runnable can be scheduled
    - You send a message
    - You post a runnable

Loaders

- Available to every Activity and Fragment.
- Provide asynchronous loading of data.
- Monitor the source of their data and deliver new results when the content changes.
- Automatically reconnect to the last loader's cursor when being recreated after a configuration change.

Loaders

Download Manager

- System service that handles long-running HTTP downloads.

WEBSERVICES

- Data representation/description
  - Languages
    - XML
    - JSON
    - Designed to be
      - self descriptive
      - Execution platform neutral
    - Used for data exchange, data storing
    - Commonly used to represent webservices data

JSON

- Structure

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick":
"CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

- Parsing

The same text expressed as XML:

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New"
onclick="CreateNewDoc()" />
    <menuitem value="Open"
onclick="OpenDoc()" />
    <menuitem value="Close"
onclick="CloseDoc()" />
  </popup>
</menu>
```

JSON

- Structure

```
{"widget": {
    "debug": "on",
    "window": {
        "title": "Sample Konfabulator Widget",
        "name": "main_window",
        "width": 500,
        "height": 500
    },
    "image": {
        "src": "Images/Sun.png",
        "name": "sun1",
        "hOffset": 250,
        "vOffset": 250,
        "alignment": "center"
    },
    "text": {
        "data": "Click Here",
        "size": 36,
        "style": "bold",
        "name": "text1",
        "hOffset": 250,
        "vOffset": 100,
        "alignment": "center",
        "onMouseUp": "sun1.opacity = (sun1.opacity / 100) * 90;"
    }
}}
```

The same text expressed as XML:

```
<widget>
    <debug>on</debug>
    <window title="Sample Konfabulator Widget">
        <name>main_window</name>
        <width>500</width>
        <height>500</height>
    </window>
    <image src="Images/Sun.png" name="sun1">
        <hOffset>250</hOffset>
        <vOffset>250</vOffset>
        <alignment>center</alignment>
    </image>
    <text data="Click Here" size="36" style="bold">
        <name>text1</name>
        <hOffset>250</hOffset>
        <vOffset>100</vOffset>
        <alignment>center</alignment>
        <onMouseUp>
            sun1.opacity = (sun1.opacity / 100) * 90;
        </onMouseUp>
    </text>
</widget>
```

WEBSERVICES-XML

- Data representation/description
  - XML document structure
    - Organized as a tree structure starting from root

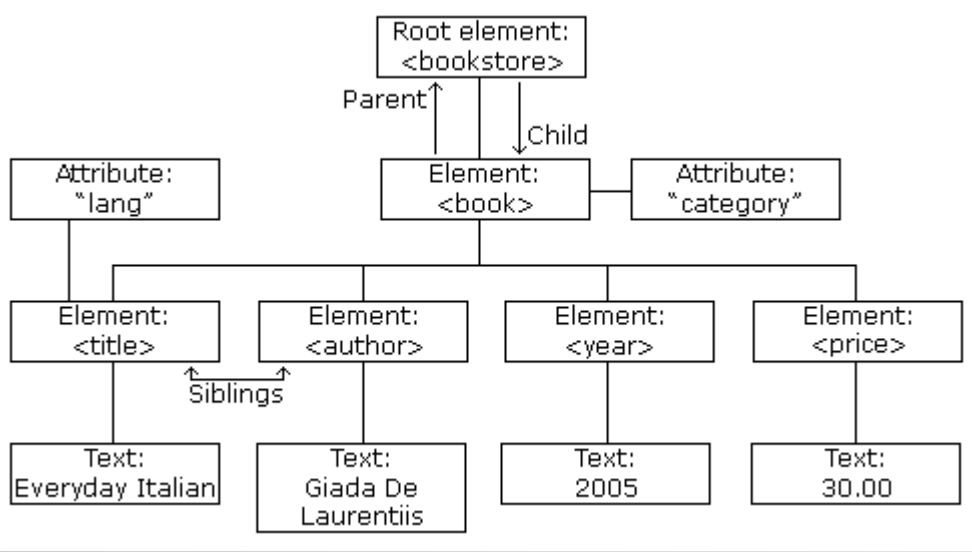
```
<?xml version="1.0" encoding="UTF-8"?>
<note>
 <dest>Marc</dest>
 <de>Jeanne</de>
 <sujet>Rappel</sujet>
 <cont>C'est l'anniversaire de Julie ce week-end!</cont>
</note>
```

  - Contains root element <note>
  - Contains 4 child elements : dest, de, sujet,cont

```
<root>
 <child>
  <subchild>.....</subchild>
 </child>
</root>
```

WEBSERVICES-XML

- Data representation/description
  - Elements can have text content and attributes
  - Parent, children, sibling



WEBSERVICES-XML

- Data representation/description
  - Elements

```
<bookstore>
 <book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
 </book>
 <book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
 </book>
 <book category="WEB">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
 </book>
</bookstore>
```

WEBSERVICES-XML

- Data representation/description
- Language is case sensitive
  - Well formed document
    - Must have a root element
    - Open tag must have a closing tag
    - Elements must be properly nested
    - Attributes values must be quoted
    - Predefined entity references

&lt;	<	less than
&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	apostrophe
&quot;	"	quotation mark

Comments

<!-- This is a comment -->

WEBSERVICES-XML

- Data representation/description
  - Well formed document
    - Element names must start with a letter or underscore
    - Element names cannot start with the letters xml (or XML, or Xml, etc)
    - Element names can contain letters, digits, hyphens, underscores, and periods
    - Element names cannot contain spaces

WEBSERVICES-XML

- Data representation/description
  - Attributes are quoted
    - <gangster name='George "Shotgun" Ziegler'>
    - <gangster name="George "Shotgun" Ziegler">
  - Elements versus attribute

```
<person gender="female">
 <firstname>Anna</firstname>
 <lastname>Smith</lastname>
</person>

<person>
 <gender>female</gender>
 <firstname>Anna</firstname>
 <lastname>Smith</lastname>
</person>
```

WEBSERVICES-JSON

- Data representation/description
  - JSON Java Script Open Notation
  - An alternative to XML

```
{"employees":[
   {"firstName":"John", "lastName":"Doe"},
   {"firstName":"Anna", "lastName":"Smith"},
   {"firstName":"Peter", "lastName":"Jones"}
]}
```

```
<employees>
   <employee>
      <firstName>John</firstName> <lastName>Doe</lastName>
   </employee>
   <employee>
      <firstName>Anna</firstName> <lastName>Smith</lastName>
   </employee>
   <employee>
      <firstName>Peter</firstName> <lastName>Jones</lastName>
   </employee>
</employees>
```

WEBSERVICES-JSON

- Data representation/description
  - JSON Java Script Open Notation
  - An alternative to XML
  - JSON
    - Data
      - Data: pair name/value
      - "firstName":"John«
      - Data separator ,
    - Object
      - {…}
      - {"firstName":"John", "lastName":"Doe"}
    - Array
      - […]

```
"employees":[
 {"firstName":"John", "lastName":"Doe"},
{"firstName":"Anna", "lastName":"Smith"},
{"firstName":"Peter","lastName":"Jones"}
]
```

MIME type for JSON text is "application/json

Generating XML

- Create XML data

```
public static String writeUsingNormalOperation(Study study) {
    String format =
            "<?xml version='1.0' encoding='UTF-8'?>" +
            "<record>" +
            "   <study id='%d'>" +
            "      <topic>%s</topic>" +
            "      <content>%s</content>" +
            "      <author>%s</author>" +
            "      <date>%s</date>" +
            "   </study>" +
            "</record>";
    return String.format(format, study.mId, study.mTopic, study.mContent,
study.mAuthor, study.mDate);
}
```

Generating XML

- Create XML data using DOM

```
public static String writeUsingDOM(Study study) throws Exception {
    Document doc = DocumentBuilderFactory.newInstance().newDocumentBuilder().newDocument();
    // create root: <record>
    Element root = doc.createElement(Study.RECORD);
    doc.appendChild(root);

    // create: <study>
    Element tagStudy = doc.createElement(Study.STUDY);
    root.appendChild(tagStudy);
    // add attr: id =
    tagStudy.setAttribute(Study.ID, String.valueOf(study.mId));

    // create: <topic>
    Element tagTopic = doc.createElement(Study.TOPIC);
    tagStudy.appendChild(tagTopic);
    tagTopic.setTextContent(study.mTopic);

    // create: <content>
    Element tagContent = doc.createElement(Study.CONTENT);
    tagStudy.appendChild(tagContent);
    tagContent.setTextContent(study.mContent);

    // create: <author>
    Element tagAuthor = doc.createElement(Study.AUTHOR);
    tagStudy.appendChild(tagAuthor);
    tagAuthor.setTextContent(study.mAuthor);

    // create: <date>
    Element tagDate = doc.createElement(Study.DATE);
    tagStudy.appendChild(tagDate);
    tagDate.setTextContent(study.mDate);

    // create Transformer object
    Transformer transformer = TransformerFactory.newInstance().newTransformer();
    StringWriter writer = new StringWriter();
    StreamResult result = new StreamResult(writer);
    transformer.transform(new DOMSource(doc), result);

    // return XML string
    return writer.toString();
}
```

Generating XML

- Create XML data using XML serializer

```
public static String writeUsingXMLSerializer(Study study) throws Exception {
    XmlSerializer xmlSerializer = Xml.newSerializer();
    StringWriter writer = new StringWriter();

    xmlSerializer.setOutput(writer);
    // start DOCUMENT
    xmlSerializer.startDocument("UTF-8", true);
    // open tag: <record>
    xmlSerializer.startTag("", Study.RECORD);
    // open tag: <study>
    xmlSerializer.startTag("", Study.STUDY);
    xmlSerializer.attribute("", Study.ID, String.valueOf(study.mId));

    // open tag: <topic>
    xmlSerializer.startTag("", Study.TOPIC);
    xmlSerializer.text(study.mTopic);
    // close tag: </topic>
    xmlSerializer.endTag("", Study.TOPIC);

    // open tag: <content>
    xmlSerializer.startTag("", Study.CONTENT);
    xmlSerializer.text(study.mContent);
    // close tag: </content>
    xmlSerializer.endTag("", Study.CONTENT);

    // open tag: <author>
    xmlSerializer.startTag("", Study.AUTHOR);
    xmlSerializer.text(study.mAuthor);
    // close tag: </author>
    xmlSerializer.endTag("", Study.AUTHOR);

    // open tag: <date>
    xmlSerializer.startTag("", Study.DATE);
    xmlSerializer.text(study.mDate);
    // close tag: </date>
    xmlSerializer.endTag("", Study.DATE);

    // close tag: </study>
    xmlSerializer.endTag("", Study.STUDY);
    // close tag: </record>
    xmlSerializer.endTag("", Study.RECORD);
```

Generating JSON

- Use JSON class
  - Data
  - Object
  - Array

```
"employees":[
 {"firstName":"John", "lastName":"Doe"},
{"firstName":"Anna", "lastName":"Smith"},
{"firstName":"Peter","lastName":"Jones"}
]
JSONObject main = new JSONObject();
JSONArray arrayEmployee = new JSONArray();
for (int i = 0; i<=4; i++)
{
JSONObject employee = new JSONObject();
//La classe de base
try {
employee.put("firstName","Prenom"+"_"+Integer.toString(i));
employee.put("lasttName","Nom"+"_"+Integer.toString(i));
//ajout de la classe à l'array
arrayEmployee.put(employee);
} catch (JSONException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
main.put("Employees", arrayEmployee);
return main.toString();
}
```

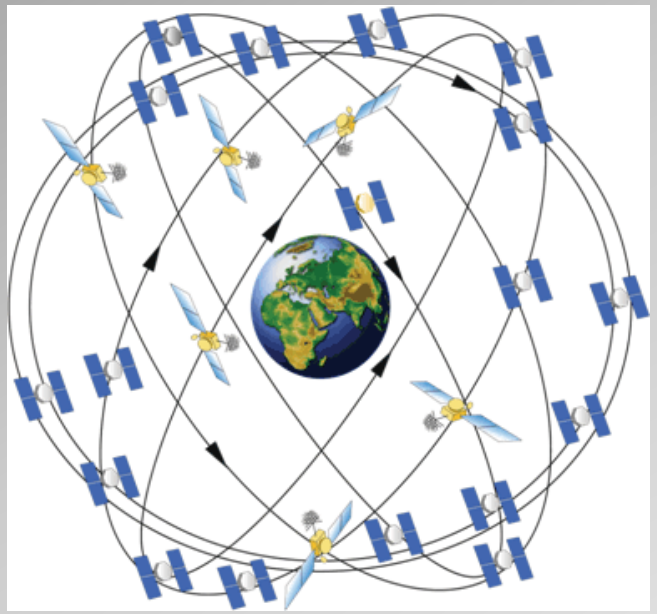
Parsing - XML

- **Dom Parser** - Loads the complete contents of the document and creates its complete hiearchical tree in memory.
- **SAX Parser** - Does not load the complete document into the memory. Parses the document on event based triggers.
- **JDOM Parser** - Parses the document in similar fashion to DOM parser in a easier way
- **XPath Parser** - Parses the XML based on expression.
- **XmlPullParser** recommended by Goggle

ANDROID LOCATION SERVICE

Location Service

- Overview
  - Location providers
    - GPS Provider



    - Four satellites visible anytime over 27 orbiting

LOCATION SERVICE

- Ephemeris and Almanach Date handled by receivers to determine long/lat
- Getting the Almanach is a slow process, limiting GPS use for mobile applications
  - Work around
  - A-GPS (Assisted GPS), Almanach sent using the ground mobile network
  - S-GPS (Standard GPS)

– Network Provider

- Use Wi-FI
- Use Cellular Network, towerCell ID

LOCATION SERVICE

- Location API
  - Package android.location
  - Classes:
    - LocationManager : class providing access to the system location services
    - LocationProvider : abstract superclass for location providers
    - Location : data class representing a geographic location
    - Criteria : criteria for selecting a location provider
  - Interfaces:
    - LocationListener: Used for receiving notifications from the LocationManager when the location has changed.
- Google location API (recommended)

LOCATION SERVICE

- Geocoder
  - Geocoding : address -> other format (i.e Lat,Long)
  - Reverse geocoding location-> address
    - isPresent to test if backend service does exist.

LOCATION SERVICE

- Maps API
  - Mapview : UI to display maps
  - MapActivity : extend this class to use Mapview
  - Overlay : class used to annotate maps
  - MapController : class used to control the map display
  - MyLocationOverlay : layer to display the current position
  - ItemizedOverlay : layer to add drawables to MapViews

LOCATION SERVICE

- Maps API
  - Drawing on a map
  - Markers
  - Window information
  - Shapes
  - Ground overlay (images)
  - StreetView

LOCATION SERVICE

- MapFragment
  - Markers
  - MapType : terrain, hybrid, satellite
  - IndoorMaps
  - Polylines

LOCATION SERVICE

- Maps API
  - You need a Key set to use this API
  - Get it from Google : https://console.developers.google.com/project
  - Import Google Play Services into Eclipse
  - Write your application extends mapviewactivity
  - Set-up the manifest with metadata (api key) and use permissions
    - ACCESS_NETWORK_STATE
    - INTERNET
    - WRITE_EXTERNAL_STORAGE
    - ACCESS_COARSE_LOCATION
    - ACCESS_FINE_LOCATION
    - OpenGL ES V2

    - Your terminal (Mobile phone) may need google display services update

LOCATION

– Simulating the location in the emulator

CONTENT PROVIDER

- Manages access to a central repository of data
- Applications access a provider to handle repository data.
- Applications use a provider client object
- Ex native Android content provider
  - Contact
  - Agenda
- Data is presented to external applications as one or more tables (Relational DB like, row columns)

word	App_id	frequency	locale	_ID
Kodie	user1	100	En_US	1
completude	user2	45	Fr_FR	2

.

CONTENT PROVIDER

- Application get access the data from a provider with a contentresolver client object
- The ContentResolver expose the basic "CRUD" (create, retrieve, update, and delete) methods of persistent storage access.
- Use permission are usually required for the application to use a content provider
  - ex using User Dictionary Provider
    - call ContentResolver.query().
    - Returns a cursor

CONTENT PROVIDER

- Contentresolver.query
  - mCursor = getContentResolver().query(
    UserDictionary.Words.CONTENT_URI, // The content URI of the words table
    mProjection, // The columns to return for each row
    mSelectionClause, // Selection criteria
    mSelectionArgs, // Selection criteria
    mSortOrder); // The sort order for the returned rows
  - Compared to Sql

query() argument	SELECT keyword/parameter	Notes
Uri	FROM table_name	Uri maps to the table in the provider named table_name.
projection	col,col,col,...	projection is an array of columns that should be included for each row retrieved.
selection	WHERE col = value	selection specifies the criteria for selecting rows.
selectionArgs	(No exact equivalent. Selection arguments replace? placeholders in the selection clause.)	
sortOrder	ORDER BY col,col,...	sortOrder specifies the order in which rows appear in the returned Cursor.

CONTENT PROVIDER

- URI
- content://user_dictionary/words
  - user_dictionary is provider's authority
  - Words is the table to access
  - content:// scheme is allways present , indicates this is a provider
- You append an ID to the URI to get one item
- Otherwise you get the full set of data

- Utils classes to build a URI
  - Uri
  - Uri.Builder
  - ContentUris (to append data to uri)

CONTENT PROVIDER

- To retrieve Data
  - Request read access to provider (use in application manifest)
  - Write code that sens a query to the provider
- Constructing a query
  - ```
    // A "projection" defines the columns that will be returned for each row
    String[] mProjection =
    {
       UserDictionary.Words._ID,    // Contract class constant for the _ID column name
       UserDictionary.Words.WORD,   // Contract class constant for the word column name
       UserDictionary.Words.LOCALE  // Contract class constant for the locale column name
    };
    ```
  - ```
    // Defines a string to contain the selection clause
    String mSelectionClause = null;
    ```
  - ```
    // Initializes an array to contain selection arguments
    String[] mSelectionArgs = {""};
    ```
    ```

Content Provider

- Contacts provider
- Entities

Content Provider

- Create a content provider

Data Persistence

- Shared Preferences
  - Allows to save/read key/values pairs into/from a file
- Content Provider

Telephony

- Terms
  - GSM based on Time Division Multiple Access(TDMA)
  - Use a Suscriber Identity Module (SIM) card to store important user/carrier data
  - *Integrated Circuit Card Identifier (ICCID)*—Identifies a SIM card; also known as a SIM Serial Number, or SSN.
  - *International Mobile Equipment Identity (IMEI)*—Identifies a physical device. ThevIMEI number is usually printed underneath the battery.
  - *International Mobile Subscriber Identity (IMSI)*—Identifies a subscriber (and thevnetwork that subscriber is on).
  - *Location Area Identity (LAI)*—Identifies the region within a provider network that's occupied by the device.
  - *Authentication key (Ki)*—A 128-bit key used to authenticate a SIM card on a provider network.

Telephony

- Terms
  - CDMA other technology used in USA and Asia countries
  - No SIM card
  - Mobile Equipment Identifier (MEID)—Identifies a physical device. It corresponds to GSM's IMEI.
  - Electronic Serial Number (ESN)—The predecessor to the MEID, this number is shorter and identifies a physical device.
  - Pseudo Electronic Serial Number (pESN)—A hardware identifier, derived from the MEID, that's compatible with the older ESN standard. The ESN supply was exhausted several years ago, so pESNs provide a bridge for legacy applications built around ESN. A pESN always starts with 0x80 in hex format or 128 in decimal format.
  - If your application needs telephony the manifest.xml file should include
  - <uses-feature android:name="android.hardware.telephony"="true"/>

Telephony

- TelephonyManager
  - Get telephony information
  - Phone Network state
  - Attach a PhoneStateListener event listener to be aware of state changes
- Retrieve telephony properties
  - TelephonyManager telMgr = (TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE);

```
String callStateString = "NA";
int callState = telMgr.getCallState();
switch (callState) {
case TelephonyManager.CALL_STATE_IDLE:
callStateString = "IDLE";
break;
case TelephonyManager.CALL_STATE_OFFHOOK:
callStateString = "OFFHOOK";
break;
case TelephonyManager.CALL_STATE_RINGING:
callStateString = "RINGING";
break;
}
```

Telephony

```
CellLocation cellLocation = (CellLocation)telMgr.getCellLocation();
String cellLocationString = null;
if (cellLocation instanceof GsmCellLocation)
{
cellLocationString = ((GsmCellLocation)cellLocation).getLac()
+ " " + ((GsmCellLocation)cellLocation).getCid();
}
else if (cellLocation instanceof CdmaCellLocation)
{
cellLocationString = ((CdmaCellLocation)cellLocation).
getBaseStationLatitude() + " " +
((CdmaCellLocation)cellLocation).getBaseStationLongitude();
}
String deviceId = telMgr.getDeviceId();
String deviceSoftwareVersion =
telMgr.getDeviceSoftwareVersion();
String line1Number = telMgr.getLine1Number();
String networkCountryIso = telMgr.getNetworkCountryIso();
String networkOperator = telMgr.getNetworkOperator();
String networkOperatorName = telMgr.getNetworkOperatorName();
String phoneTypeString = "NA";
int phoneType = telMgr.getPhoneType();
switch (phoneType) {
case TelephonyManager.PHONE_TYPE_GSM:
phoneTypeString = "GSM";
break;
case TelephonyManager.PHONE_TYPE_CDMA:
phoneTypeString = "CDMA";
break;
case TelephonyManager.PHONE_TYPE_NONE:
phoneTypeString = "NONE";
```

Telephony

- Attach listener.

```
final TelephonyManager telMgr =
(TelephonyManager) getSystemService(
Context.TELEPHONY_SERVICE);
PhoneStateListener phoneStateListener =
new PhoneStateListener() {
public void onCallStateChanged(
int state, String incomingNumber) {
telMgrOutput.setText(getTelephonyOverview(telMgr));
}
};
telMgr.listen(phoneStateListener,
PhoneStateListener.LISTEN_CALL_STATE);
String telephonyOverview = getTelephonyOverview(telMgr);
telMgrOutput.setText(telephonyOverview);
```

Telephony

- Interacting with phone
  - Using intents to make calls
    - Use Intent.ACTION_CALL action and the tel: Uri.
    - Use Intent.ACTION_DIAL action

```
dialintent = (Button) findViewById(R.id.dialintent_button);
dialintent.setOnClickListener(new OnClickListener() {
public void onClick(View v) {
Intent intent =
new Intent(Intent.DIAL_ACTION,
Uri.parse("tel:" + NUMBER));
startActivity(intent);
}
});
```

Telephony

- Interacting with phone
- Android permissions

android.permission.CALL_PHONE
Initiates a phone call without user confirmation in dialer
android.permission.CALL_PRIVILEGED
Calls any number, including emergency, without confirmation in dialer
android.permission.MODIFY_PHONE_STATE
Allows the application to modify the phone state: for example, to turn the radio on or off
android.permission.PROCESS_OUTGOING_CALLS
Allows the application to receive broadcast for outgoing calls and modify
android.permission.READ_PHONE_STATE
Allows the application to read the phone

Telephony

- Interacting with phone
  - To parse numbers
    - Use PhoneNumberUtils class
  - Intercepting outbounds calls

```
public class OutgoingCallReceiver extends BroadcastReceiver {
public static final String ABORT_PHONE_NUMBER = "1231231234";
@Override
public void onReceive(Context context, Intent intent) {
if (intent.getAction().equals(
Intent.ACTION_NEW_OUTGOING_CALL)) {
String phoneNumber =
intent.getExtras().getString(Intent.EXTRA_PHONE_NUMBER);
if ((phoneNumber != null)
&& phoneNumber.equals(
OutgoingCallReceiver.ABORT_PHONE_NUMBER)) {
Toast.makeText(context,
"NEW_OUTGOING_CALL intercepted to number "
+ "123-123-1234 - aborting call",
Toast.LENGTH_LONG).show();
abortBroadcast();
}
}
}
}
```

Telephony

- Interacting with phone
  - Working with SMS (Short Message Service)
  - Receiving SMS

```
public class SmsReceiver extends BroadcastReceiver {
private static final String SMS_REC_ACTION =
"android.provider.Telephony.SMS_RECEIVED";
@Override
public void onReceive(Context context, Intent intent) {
if (intent.getAction().
equals(SmsReceiver.SMS_REC_ACTION)) {
StringBuilder sb = new StringBuilder();
Bundle bundle = intent.getExtras();
if (bundle != null) {
Object[] pdus = (Object[])
bundle.get("pdus");
for (Object pdu : pdus) {
SmsMessage smsMessage =
SmsMessage.createFromPdu
((byte[]) pdu);
sb.append("body - " + smsMessage.
getDisplayMessageBody());
}
}
Toast.makeText(context, "SMS RECEIVED - "
+ sb.toString(), Toast.LENGTH_LONG).show();
}
}
}
```

Telephony

- Interacting with phone
  - Working with SMS (Short Message Service)
  - Sending SMS

```
private Button smsSend;
private SmsManager smsManager;
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.smsexample);
// . . . other onCreate view item inflation omitted for brevity
smsSend = (Button) findViewById(R.id.smssend_button);
smsManager = SmsManager.getDefault();
final PendingIntent sentIntent =
PendingIntent.getActivity(
this, 0, new Intent(this,
SmsSendCheck.class), 0);
smsSend.setOnClickListener(new OnClickListener() {
public void onClick(View v) {
String dest = smsInputDest.getText().toString();
if (PhoneNumberUtils.
isWellFormedSmsAddress(dest)) {
smsManager.sendTextMessage(
smsInputDest.getText().toString, null,
smsInputText.getText().toString(),
sentIntent, null);
Toast.makeText(SmsExample.this,
"SMS message sent",
Toast.LENGTH_LONG).show();
} else {
Toast.makeText(SmsExample.this,
"SMS destination invalid - try again",
Toast.LENGTH_LONG).show();
}
}
});
}
```

Ermissions

```
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.READ_SMS" />
<uses-permission android:name="android.permission.WRITE_SMS" />
<uses-permission android:name="android.permission.SEND_SMS" />
```


Notifications and alarms

- Alarm
  - Simple mecanism for time based operations
    - Fire intents
    - Start services … using broadcast receivers
    - Works outside of your application, when device is asleep
    - Inside your application use a timer and handler
    - Alarm specifics
      - Alarm Type: elapsed time or realtime
      - Trigger time
      - Alarm interval
      - Pending intent

Notifications and alarms

- Notification
  - Notification visible in the window bar
  - Notification specifics
    - A title
    - An icon
    - A message
    - An action
- Use steps
  - Ask for a notificationManager (getSystemService)
  - instanciate NotificationCompat.Builder to build the notification
  - notificationManager.notify to send the notification

Contacts

Cours

- Day 1
  - Intro to Web services
  - Socket
  - HTTP GET POST
  - Check for connection
  - Performance issues
  - Reminder
    - Thread
    - AsyncTask
    - Handler and message
    - -intentService
    - Fragments
- Day 2
  - XML
  - JSON
  - Create XML
  - Create JSON
  - Parse XML
  - Parse JSON
  - Connect to a feed
  - Decode a feed
- Day 3
  - Decode USGS feed
  - Decode MyWeather2 feed
  - Data persistence
  - Shared preferences
  - Preferences Framework
- Day 4
  - ContactContract
  - Agenda
  - Localisation
  - Geo-codage
  - MapViews
- Day 5
  - Telephony
  - Send SMS
  - Alarms
  - Broadcast receiver
  - Notifications
  - Sensors
  - P2P

Cours

- Day 1
  - Service IPC
  - Location
  - Maps
- Day 2
  - Sockets, XML, data persistence
- Day 3
  - Web Service
- Day 4
  - Data provider, Prefrences, Contact manager
- Day 5
  - Sensors

Cours

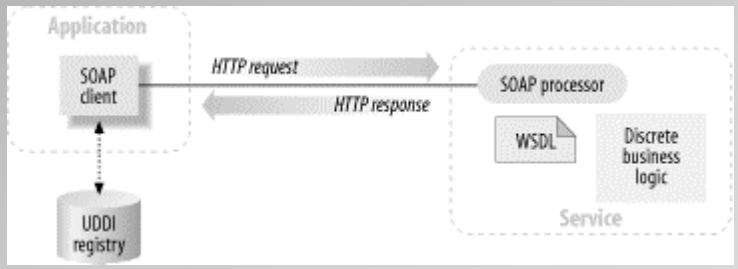
- Day 1
  - Web Services
  - Intro
  - Client/Serveur
  - Format de données
  - Transport des données
  - Intro XML
  - Intro JSON
  - Parsers XML,JSON
  - exos
- Day 2
  - Transport des données
  - Sockets,
  - HTTP
  - Rappels sur lse threads
  - asyncTask
  - Handlers
  - DownloadManager
  - HttpURLConnection
  - Connection à des web services
  - URLMatcher

Exercices

- Rappel sur les bundle
- Rappel sur asyncTask
- Rappel sur les Handlers

WEB SERVICES

- Simple Object Access Protocol (SOAP)
- Web Service Description Language (WSDL)
- Universal Description, Discovery, and Integration (UDDI)



WEB SERVICES

- SOAP
- SOAP Message structure
- POX (Plain Old XML)
- REST (Representational State Transfer)

URI

- Classe URI
- content://authority/optionalPath/optionalId
- URI Matcher

WEB SERVICES

- using HTTP protocol
- Android support two Http clients
- HttpURLConnection
- Or
- HttpClient (Apache)

- Pattern for usingHttpURLConnection
  - Http methods
    - Get is used by default
    - Post if setDoOutput(true)
  - Supports Ipv6
  - Response Caching
    - Android 4.0 (Ice Cream Sandwich, API level 15) includes a response cache.

Certificats

- certificate signing requests (CSRs)
- Public Key Infrastructure (PKI)
- Binds public keys with users identities by means of cerificates authority(CA)
- Registration authority
  - A public key infrastructure (PKI) is a system for the creation, storage, and distribution of digital certificates which are used to verify that a particular public key belongs to a certain entity. The PKI creates digital certificates which map public keys to entities, securely stores these certificates in a central repository and revokes them if needed.[5][6][7]
  - A PKI consists of:[6][8][9]
  - A *certificate authority* (CA) that both issues and verifies the digital certificates
  - A *registration authority* which verifies the identity of users requesting information from the CA
  - A *central directory*—i.e., a secure location in which to store and index keys
  - A *certificate management system*[*clarification needed*]
  - A *certificate policy*

Certificats

- Crypto asymetrique
- La cle publique permet de coder l'info
- La cle privée de décoder
- MD5
- SHA-1
- Eclipse -> preferences->build get md5

Cle API

- Cle debug
- Récupérer le md5 d'éclipse
- Aller sur google console
- Obtenir la cle API
- Importer google play dans le workspace
- Creer l'appli
- Mettre à jour le manifeste de l'appli avec la cle API
  - <!-- Goolge Maps API Key -->
  - <meta-data

android:name="com.google.android.maps.v2.API_KEY"
android:value="AIzaSyBZMlkOv4sj-M5JO9p6wksdax4TEjDVLgo" />

URI

- URI.builder
- Construire
  - http://lapi.transitchicago.com/api/1.0/ttarrivals.aspx?key=[redacted]&mapid=`value`.
  - Uri.Builder builder = new Uri.Builder();
    builder.scheme("http")
    .authority("www.lapi.transitchicago.com")
    .appendPath("api") .appendPath("1.0")
    .appendPath("ttarrivals.aspx")
    .appendQueryParameter("key", "[redacted]")
    .appendQueryParameter("mapid", value);

Google Plateformes

- Maps
  - API de geolocalisation
- FireBase
  - Real Time DB No Sql , Json
  - Auhentification
  - Sauvegardes
  - Cloud Messaging
  - …
- Services payants, certains ont un accès gratuit sous conditions