

PHÁT TRIỂN ỨNG DỤNG TRÊN THIẾT BỊ DI ĐỘNG

Hồ Văn Tú

Bộ môn Tin học ứng dụng
Khoa CNTT và truyền thông

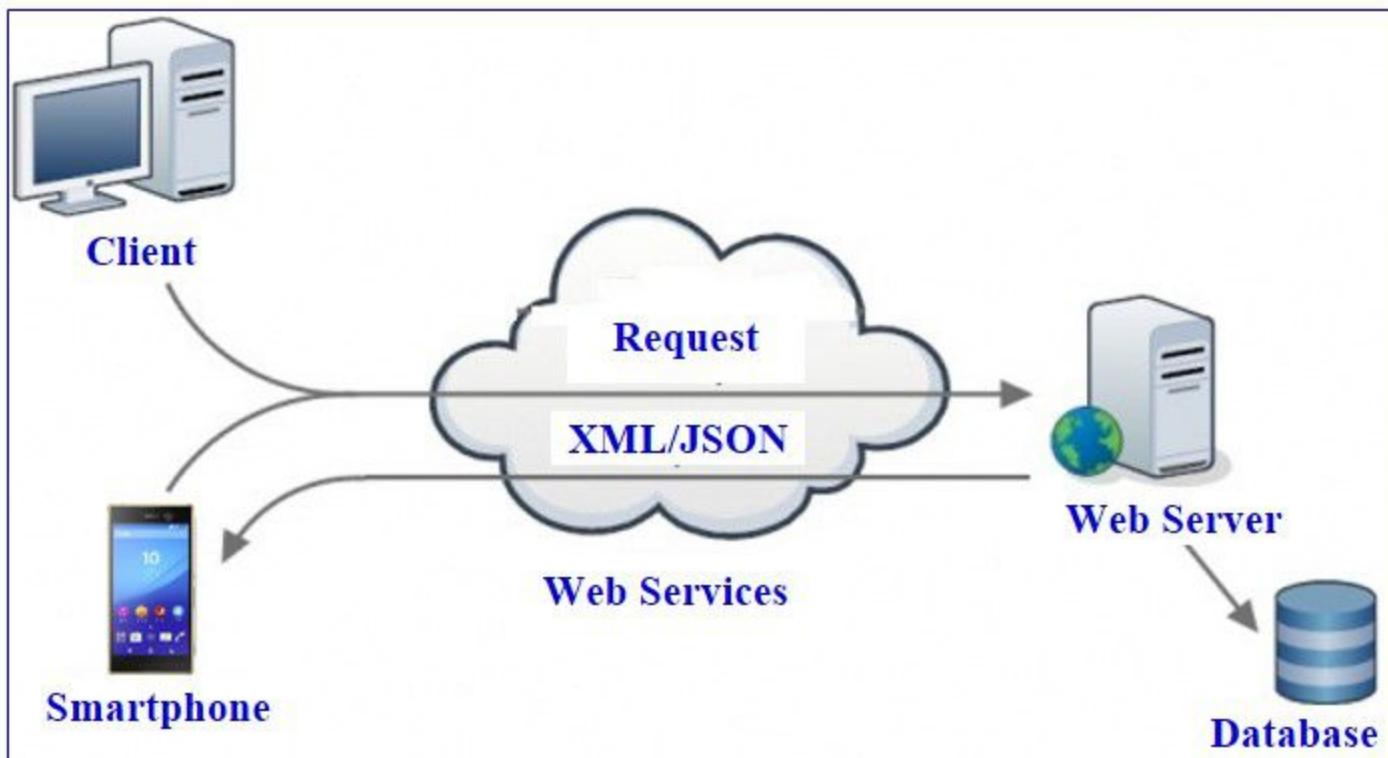
hvtu@ctu.edu.vn

Chương 6

Trao đổi dữ liệu với Web Server

NỘI DUNG

- Dữ liệu JSON
- Xử lý đa tiến trình với AsyncTask
- Trao đổi dữ liệu với Web Server



Dữ liệu JSON (1)

```
{  
    "msnv": 1,  
    "hoten": "Trần Xuân Tùng"  
}  
  
[  
    {  
        "msnv": 1,  
        "hoten": "Trần Xuân Tùng"  
    },  
    {  
        "msnv": 2,  
        "hoten": "Trần Quốc Toản"  
    }  
]
```

```
{  
    "Name": "John Smith",  
    "age": 25,  
    "address": {  
        "street": "21 2nd Street",  
        "city": "New York"  
    },  
    "phoneNumbers": [  
        {  
            "type": "home",  
            "number": "212 555-1234"  
        },  
        {  
            "type": "office",  
            "number": "646 555-4567"  
        }  
    ]  
}
```

Dữ liệu JSON (2)

- JSON (JavaScript Object Notation): chuẩn hoán vị dữ liệu nhanh
- Dễ tạo và dễ truy xuất, lưu trữ gọn nhẹ
- Hỗ trợ trao đổi dữ liệu giữa các ứng dụng
- Được xây dựng chủ yếu từ 2 dạng cấu trúc
 - JSONObject: tập hợp các giá trị theo cặp name-value
 - JSONArray: tập hợp các giá trị có thứ tự: mảng, vector

JSONObject (1)

- Thuộc lớp org.json.JSONObject, kế thừa java.lang.Object
- Đại diện cho đối tượng trong chuỗi JSON
- Thường sử dụng với các kiểu dữ liệu: int, long, String, double, Boolean

JSONObject (2)

- Phương thức thông dụng
 - new JSONObject(): khởi tạo JSONObject
 - new JSONObject(String): khởi tạo JSONObject với tham số là chuỗi JSON
 - put(name, value): gán giá trị value cho JSONObject có từ khóa nhận dạng name
 - getXXX(name): nhận giá trị có từ khóa nhận dạng name từ JSONObject

Kiểu dữ liệu XXX như int, long, String, double, Boolean, JSONObject, JSONArray

JSONObject (3)

- VD: Tạo đối tượng JSONObject bằng cách gán giá trị và nhận giá trị từ JSONObject

```
JSONObject jsonNV1 = new JSONObject();
try {
    jsonNV1.put("msnv", 1);
    jsonNV1.put("hoten", "Trần Xuân Tùng");
    int msnv = jsonNV1.getInt("msnv");
    String strHoTen = jsonNV1.getString("hoten");
} catch (JSONException e) {
}
```

JSONObject (4)

- VD: Tạo đối tượng JSONObject từ chuỗi JSON

```
String strNV2 = "{\"msnv\": 2, \"hoten\": \"Trần Quốc Toản\"}";  
JSONObject jsonNV2 = new JSONObject();  
try {  
    jsonNV2 = new JSONObject(strNV2);  
} catch (JSONException e) {  
}  
}
```

JSONArray (1)

- Thuộc lớp org.json.JSONArray, kế thừa java.lang.Object
- Đại diện cho mảng các JSONObject
- Phương thức thông dụng
 - new JSONArray(): khởi tạo JSONArray
 - new JSONArray(String): khởi tạo JSONArray với tham số là chuỗi JSON

JSONArray (2)

- Phương thức thông dụng
 - put([index,] value): gán giá trị value cho JSONArray tại chỉ số index

Nếu đã có giá trị tại index: thay thế bằng giá trị mới
Nếu không có tham số index: thêm vào cuối mảng
 - length(): kích thước mảng
 - getXXX(index): nhận giá trị trong mảng tại chỉ số index
Kiểu dữ liệu XXX: int, long, String, double, Boolean, JSONObject, JSONArray

JSONArray (3)

- VD: Tạo JSONArray từ chuỗi JSON và thêm phần tử JSONObject vào mảng

```
String strNV3 = "[ {\"msnv\": 3, \"hoten\": \"Kim Dung\"} ]";  
JSONArray jsonArrNV = new JSONArray();  
try {  
    jsonArrNV = new JSONArray(strNV3);  
    jsonArrNV.put(jsonNV2); // Thêm vào cuối mảng  
    jsonArrNV.put(0, jsonNV1); // Thay thế phần tử  
} catch (JSONException e) {  
}
```

JSONArray (4)

- VD: Nhận giá trị từ JSONArray, hiển thị tại Logcat

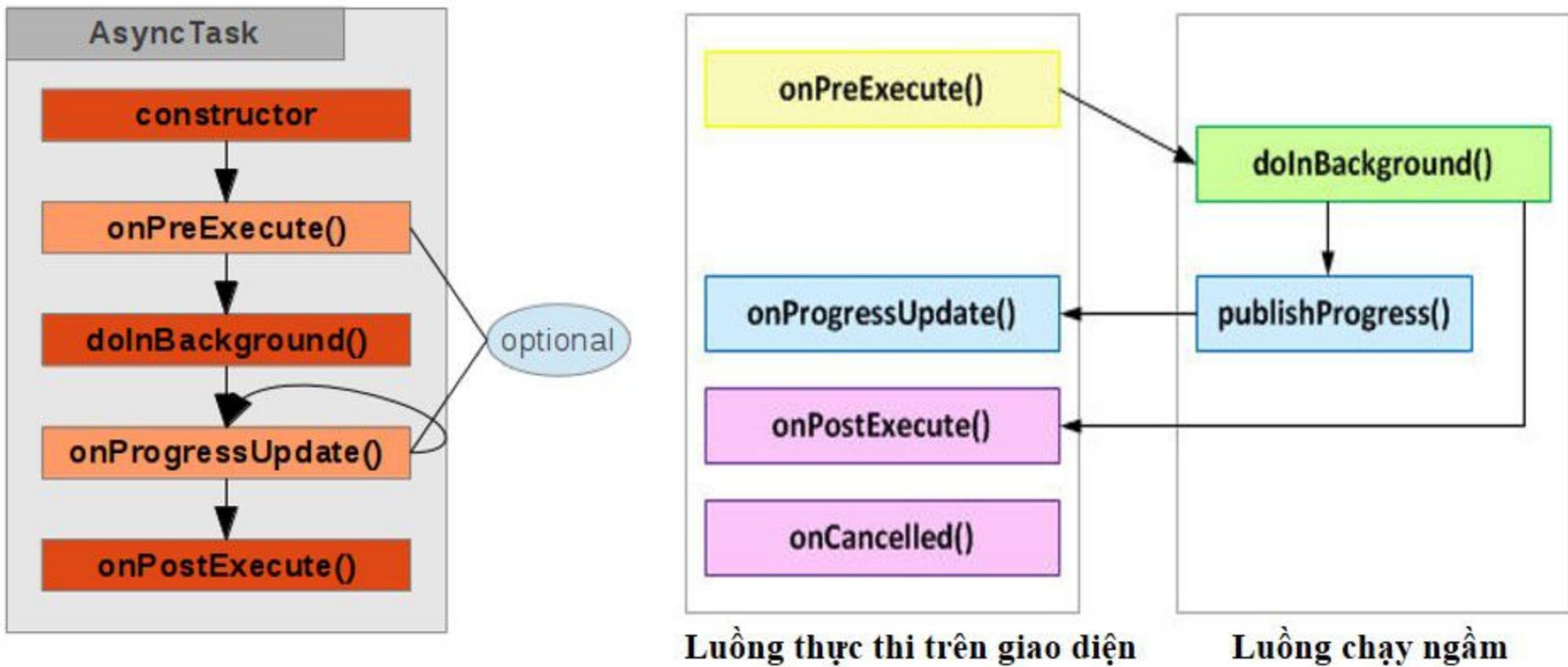
```
try {  
    for (int i = 0; i < jsonArrNV.length(); i++) {  
        JSONObject jsonNV = jsonArrNV.getJSONObject(i);  
        int msnv = jsonNV.getInt("msnv");  
        String strNV = jsonNV.getString("hoten");  
        Log.i("Thông tin NV", msnv + " " + strNV);  
    }  
} catch (JSONException e) {  
}
```

Xử lý đa tiến trình với AsyncTask (1)

- Thuộc android.os.AsyncTask<Params, Progress, Result>
- Cho phép chạy ngầm công việc
- Gồm 1 tiến trình chạy ngầm và một giao diện cập nhật kết quả khi tiến trình kết thúc
- Sử dụng: công việc có thời gian xử lý tương đối ngắn: download/upload, thao tác CSDL, Web Services, ...

Xử lý đa tiến trình với AsyncTask (2)

- Cấu trúc và quá trình thực thi: Khai báo lớp kế thừa AsyncTask<Params, Progress, Result>



Xử lý đa tiến trình với AsyncTask (3)

- Cấu trúc và quá trình thực thi

```
private class NhanDuLieu extends AsyncTask<String, Void, Integer> {

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        // Khởi tạo giá trị (nếu có)
    }

    @Override
    protected Integer doInBackground(String... strings) {
        // Công việc chạy ngầm cần thực hiện
        return ketqua;
    }
}
```

Xử lý đa tiến trình với AsyncTask (4)

- Cấu trúc và quá trình thực thi

```
@Override  
protected void onProgressUpdate(Void... values) {  
    super.onProgressUpdate(values);  
    // Cập nhật giao diện khi đang thực thi  
}  
  
@Override  
protected void onPostExecute(Integer integer) {  
    super.onPostExecute(integer);  
    // Xử lý kết quả trả về  
}  
}
```

Xử lý đa tiến trình với AsyncTask (5)

- Cấu trúc và quá trình thực thi
 - ***onPreExecute()***: công việc (không bắt buộc) thực thi trên giao diện trước khi thực thi luồng chạy ngầm
 - ***doInBackground(Params...)***
 - Công việc (**bắt buộc**) thực thi ở luồng chạy ngầm, có thể tồn tại lâu dài
 - Lời gọi `publishProgress(Progress...)` dùng để yêu cầu cập nhật tiến độ lên giao diện (nếu cần)
 - Kết quả trả về sẽ được chuyển qua `onPostExecute`

Xử lý đa tiến trình với AsyncTask (6)

- Cấu trúc và quá trình thực thi
 - ***onProgressUpdate(Progress...)***
 - Cập nhật tiến độ chạy ngầm trên giao diện (không bắt buộc)
 - Sử dụng khi trong doInBackground có lời gọi thực thi publishProgress(Progress...)
 - ***onPostExecute(Result)***
 - Nhận kết quả và tiếp tục công việc trên giao diện (không bắt buộc)
 - Nếu trong doInBackground có isCancelled() là true thì onCancelled(...) sẽ thực thi thay vì onPostExecute(...)

Xử lý đa tiến trình với AsyncTask (7)

- Kiểu dữ liệu chung (Generic types)
 - **Params**: kiểu dữ liệu của các tham số truyền vào, là tham số trong phương thức doInBackground
 - **Progress**: kiểu dữ liệu lưu trữ khi tiến trình đang thực thi, là tham số trong phương thức onProgressUpdate
 - **Result**: kiểu dữ liệu kết quả, là kết quả trả về trong phương thức doInBackground và là tham số trong phương thức onPostExecute

Các kiểu dữ liệu thường được sử dụng: Integer, Long, String, Double, Void

Xử lý đa tiến trình với AsyncTask (8)

- Thực thi AsyncTask
 - Khai báo và yêu cầu thực thi trong luồng giao diện
 - Sử dụng phương thức execute(Params) để yêu cầu thực thi

VD: Yêu cầu thực thi lớp cục bộ NhanDuLieu (thuộc MyActivity) với tham số chuỗi strGiaTri xác định trước

```
String strGiaTri = ...;  
NhanDuLieu nhanDuLieu = new NhanDuLieu();  
nhanDuLieu.execute(strGiaTri);
```

Xử lý đa tiến trình với AsyncTask (9)

- Xử lý lỗi rò rỉ bộ nhớ (Memory Leak)

Người dùng đóng Activity khi AsyncTask đang thực hiện và chờ nhận kết quả cập nhật giao diện

- Khai báo lớp kế thừa AsyncTask ở dạng lớp tĩnh (static)
- Khai báo thuộc tính Activity đang thực hiện là đối tượng tham chiếu WeakReference<T>
- Trong phương thức ghi đè onPostExecute(Result): kiểm tra sự tồn tại của Activity trước khi cập nhật giao diện

Xử lý đa tiến trình với AsyncTask (10)

- Xử lý lỗi rò rỉ bộ nhớ (Memory Leak)

```
Private statics class NhanDuLieu  
        extends AsyncTask<String, Void, Integer> {  
  
    private WeakReference<MyActivity> myActivity;  
  
    public NhanDuLieu(MyActivity activity) {  
        myActivity = new WeakReference<>(activity);  
    }  
  
    @Override  
    protected void onPreExecute() {  
        super.onPreExecute();  
        // Khởi tạo giá trị (nếu có)  
    }  
}
```

Xử lý đa tiến trình với AsyncTask (11)

- Xử lý lỗi rò rỉ bộ nhớ (Memory Leak)

```
@Override  
protected Integer doInBackground(String... strings) {  
    ...  
}  
  
@Override  
protected void onProgressUpdate(Void... values) {  
    ...  
}  
  
@Override  
protected void onPostExecute(Integer integer) {  
    super.onPostExecute(integer);  
    if (myActivity.get() != null) {  
        // Xử lý kết quả trả về  
    }  
}
```

Xử lý đa tiến trình với AsyncTask (12)

- Xử lý lỗi rò rỉ bộ nhớ (Memory Leak)
 - Thực thi AsyncTask: thêm đối tượng Activity đang thực hiện là đối tượng tham chiếu trong phương thức khởi tạo VD: Yêu cầu thực thi lớp cục bộ NhanDuLieu (thuộc MyActivity) với tham số chuỗi strGiaTri xác định trước

```
String strGiaTri = ...;  
NhanDuLieu nhanDuLieu = new NhanDuLieu(MyActivity.this) ;  
nhanDuLieu.execute(strGiaTri) ;
```

Xử lý đa tiến trình với AsyncTask (13)

- **Lưu ý khi sử dụng**

- Tạo class cục bộ kế thừa AsyncTask với (các) phương thức cần thực hiện (`doInBackground` là bắt buộc)
- Thể hiện của Task và lời gọi thực thi phải được tạo và chạy trong tiến trình chính (có UI)
- Không gọi thủ công các phương thức của lớp
- Task chỉ có thể được chạy 1 lần (nếu khởi chạy lần thứ 2 sẽ có Exception)

Xử lý đa tiến trình với AsyncTask (14)

- VD: lớp NhanKhachHang nhận danh sách khách hàng theo khu vực xác định trước

```
private class NhanKhachHang extends  
        AsyncTask<Integer, Void, Integer>{  
  
    @Override  
    protected Integer doInBackground(Integer... arg0) {  
        int soKhachHangMoi;  
        int khuvuc = arg0[0];  
        // Công việc chạy ngầm cần thực hiện  
        return soKhachHangMoi;  
    }  
}
```

Xử lý đa tiến trình với AsyncTask (15)

- VD: lớp NhanKhachHang nhận danh sách khách hàng theo khu vực xác định trước

```
@Override  
protected void onPostExecute(Integer result) {  
    if (result >= 0) {  
        // Công việc cập nhật giao diện (nếu có)  
        Toast.makeText(getApplicationContext(),  
            "Nhận được " + result + " khách hàng mới",  
            Toast.LENGTH_SHORT).show();  
    }  
    finish();  
}
```

Xử lý đa tiến trình với AsyncTask (16)

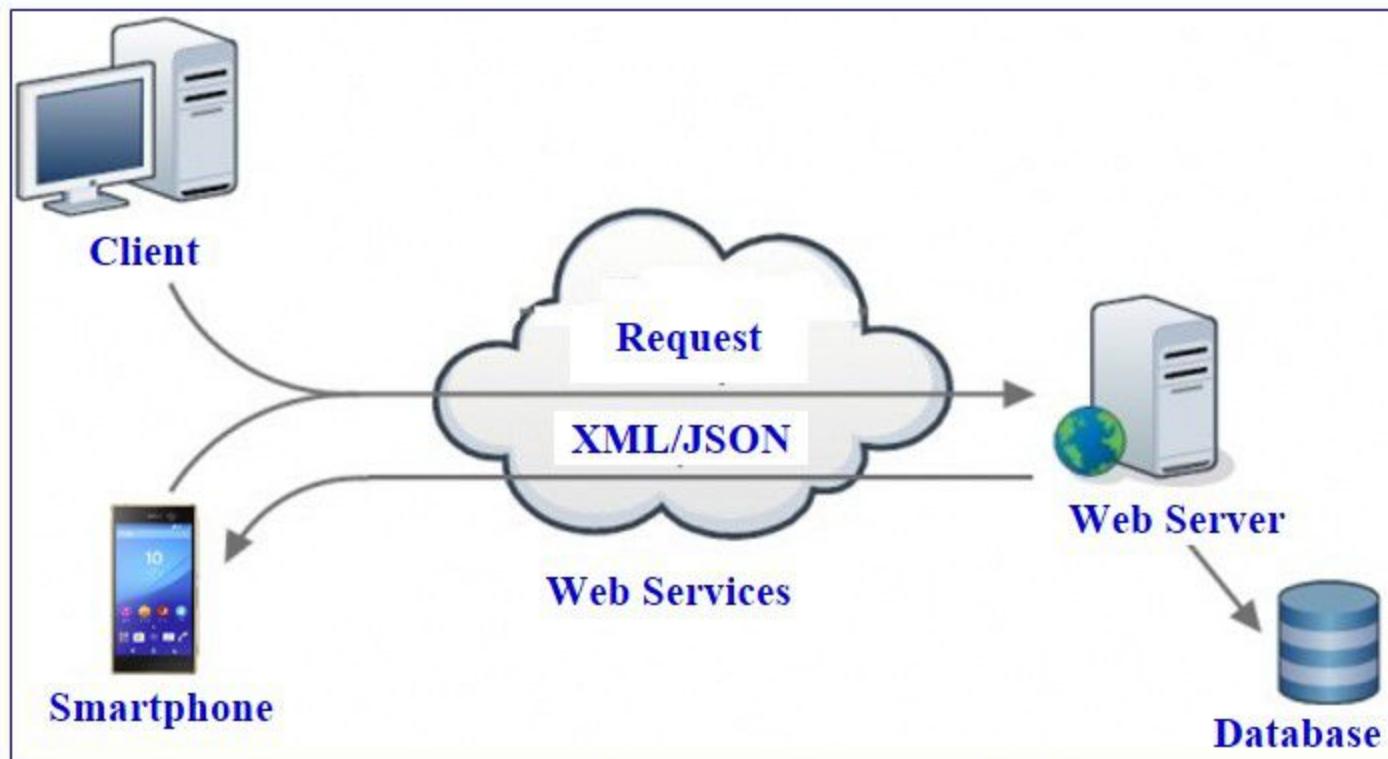
- VD: thực thi lớp NhanKhachHang với khu vực là số nguyên xác định trước

```
int khuvuc = ...;
```

```
NhanKhachHang nhanKhachHang = new NhanKhachHang();  
nhanKhachHang.execute(khuvuc);
```

Trao đổi dữ liệu với Web Server (1)

- Ứng dụng khác nhau trao đổi dữ liệu qua mạng Internet
- Web Services được coi là công nghệ hữu ích đang được sử dụng phổ biến nhất hiện nay



Trao đổi dữ liệu với Web Server (2)

- Kiểm tra trạng thái Internet và chuyển đổi dữ liệu
 - **Đối tượng và phương thức thông dụng**
 - + **ConnectivityManager**: thuộc lớp android.net.ConnectivityManager, hỗ trợ kiểm tra trạng thái kết nối mạng. Phương thức thông dụng:
getSystemService(Context.CONNECTIVITY_SERVICE): khởi tạo đối tượng, ép kiểu để trả về đối tượng ConnectivityManager.
getActiveNetworkInfo(): trả về đối tượng lưu thông tin trạng thái mạng (NetworkInfo).
 - + **NetworkInfo**: thuộc lớp android.net.NetworkInfo, hỗ trợ mô tả thông tin trạng thái mạng. Phương thức thông dụng là isConnected().
isConnected(): xác định trạng thái mạng có được kết nối hay không.

Trao đổi dữ liệu với Web Server (3)

- Kiểm tra trạng thái Internet và chuyển đổi dữ liệu

```
public class Publics {  
  
    public static String URLGOIDULIEU = "...";  
    public static String URLNHANDULIEU = "...";  
    public static int KHUVUC;  
  
    public static boolean HasInternet(Context context) {  
        ConnectivityManager conn = (ConnectivityManager)  
            context.getSystemService(  
                Context.CONNECTIVITY_SERVICE);  
        NetworkInfo netInfo = conn.getActiveNetworkInfo();  
        return ((netInfo != null) && (netInfo.isConnected()));  
    }  
}
```

Trao đổi dữ liệu với Web Server (4)

- Kiểm tra trạng thái Internet và chuyển đổi dữ liệu

```
public static String StreamToString(InputStream tream) {  
    StringBuilder builder = new StringBuilder();  
    BufferedReader reader = new BufferedReader(  
        new InputStreamReader(tream));  
    String temp;  
    try {  
        while ((temp = reader.readLine()) != null) {  
            builder.append(temp).append('\n');  
        }  
        tream.close();  
    } catch (IOException e) {  
    } finally {  
    }  
    return builder.toString();  
}
```

Trao đổi dữ liệu với Web Server (5)

- Khung mã lệnh kiểm tra và thực thi công việc

```
if (HasInternet(context)) {  
    ... // Thực hiện công việc  
}  
else {  
    Toast.makeText(this, "Lỗi kết nối Internet",  
        Toast.LENGTH_SHORT).show();  
}
```

Trao đổi dữ liệu với Web Server (6)

- Các đối tượng và phương thức hỗ trợ
 - **URL** (Uniform Resource Locator: định vị tài nguyên thống nhất): thuộc lớp `java.net.URL`, là một con trỏ trỏ tới một “tài nguyên” trên Web. Phương thức thông dụng
 - `new URL(urlString)`: khởi tạo URL theo địa chỉ Web
 - `openConnection()`: trả về đối tượng `URLConnection` được kết nối
 - `openStream()`: trả về đối tượng `InputStream` cho phép đọc dữ liệu

Trao đổi dữ liệu với Web Server (7)

- Các đối tượng và phương thức hỗ trợ
 - **HttpURLConnection**: hỗ trợ giao thức truyền dữ liệu siêu văn bản, thuộc lớp `java.net.HttpURLConnection`. Phương thức thông dụng
 - (`HttpURLConnection`) `url.openConnection()`: khởi tạo đối tượng kết nối
 - `setRequestMethod("POST")`: xác định phương thức truyền nhận dữ liệu (POST hoặc GET)
 - `setRequestProperty("Content-Type", "application/json")`: xác định thuộc tính dữ liệu truyền nhận (ví dụ: JSON)

Trao đổi dữ liệu với Web Server (8)

- Các đối tượng và phương thức hỗ trợ
 - **HttpURLConnection**
 - setReadTimeout(15000): xác định thời gian đọc tối đa
 - setConnectTimeout(15000): xác định thời gian kết nối tối đa
 - setDoInput(boolean): xác định chế độ nhận dữ liệu
 - setDoOutput(boolean): xác định chế độ gửi dữ liệu
 - connect(): kết nối tới Web Server
 - getOutputStream(): trả về OutputStream gửi dữ liệu đến kết nối

Trao đổi dữ liệu với Web Server (9)

- Các đối tượng và phương thức hỗ trợ
 - **HttpURLConnection**
 - `getInputStream()`: trả về `InputStream` lưu dữ liệu đọc được từ kết nối
 - `getResponseCode()`: nhận mã kết quả thực hiện, nếu thành công thì `responseCode = HttpsURLConnection.HTTP_OK`
 - `disconnect()`: hủy kết nối với Web Server

Trao đổi dữ liệu với Web Server (10)

• Thực hiện trao đổi dữ liệu với Web Server

- Kiểm tra trạng thái kết nối Internet,
- Khai báo lớp kế thừa AsyncTask hỗ trợ xử lý đa luồng,
 - + Trong phương thức *doInBackground*, thực hiện:
 - Khai báo URL và HttpURLConnection hỗ trợ mở kết nối đến Web Server,
 - Khai báo các thuộc tính cho kết nối,
 - Thực hiện gửi dữ liệu tới Server (nếu có):
 - Chuyển dữ liệu thường thành JSONObject,
 - Gửi dữ liệu thông qua OutputStream.
 - Thực hiện nhận dữ liệu từ Server (nếu có):
 - Nhận dữ liệu thông qua InputStream
 - Nhận và kiểm tra mã kết quả thực hiện (getResponseCode())
 - Chuyển dữ liệu InputStream thành chuỗi (String)
 - Chuyển dữ liệu chuỗi thành JSONObject
 - Nhận dữ liệu đối tượng từ JSONObject
 - Hủy kết nối.
 - + Trong phương thức *onPostExecute*: thông báo kết quả trao đổi dữ liệu đến Web Server và cập nhật giao diện (nếu có).

Trao đổi dữ liệu với Web Server (11)

- Thực hiện trao đổi dữ liệu với Web Server
 - Kiểm tra trạng thái kết nối Internet,
 - Khai báo lớp kế thừa AsyncTask hỗ trợ xử lý đa luồng
 - + Trong phương thức ***doInBackground***, thực hiện:
 - Khai báo URL và HttpURLConnection hỗ trợ mở kết nối đến Web Server,
 - Khai báo các thuộc tính cho kết nối,
 - Thực hiện gửi dữ liệu tới Server (nếu có):
 - Chuyển dữ liệu thường thành JSONObject,
 - Gởi dữ liệu thông qua OutputStream

Trao đổi dữ liệu với Web Server (12)

- Thực hiện trao đổi dữ liệu với Web Server
 - Khai báo lớp kế thừa AsyncTask hỗ trợ xử lý đa luồng
 - + Trong phương thức ***doInBackground***, thực hiện:
 - Thực hiện nhận dữ liệu từ Server (nếu có):
 - Nhận dữ liệu thông qua InputStream
 - Nhận và kiểm tra mã kết quả thực hiện (getResponseCode())
 - Chuyển dữ liệu InputStream thành chuỗi (String)
 - Chuyển dữ liệu chuỗi thành JSONObject
 - Nhận dữ liệu đối tượng từ JSONObject
 - Hủy kết nối

Trao đổi dữ liệu với Web Server (13)

- Thực hiện trao đổi dữ liệu với Web Server
 - + Trong phương thức ***onPostExecute***, thực hiện:
 - Thông báo kết quả trao đổi dữ liệu đến Web Server
 - Cập nhật giao diện (nếu có)

Trao đổi dữ liệu với Web Server (14)

- VD: Trong Activity NhanKhachHangTuServer, tạo lớp NhanKhachHang kế thừa AsyncTask

```
private class NhanKhachHang extends AsyncTask<Integer, Void, Integer> {  
    @Override  
    protected Integer doInBackground(Integer... arg0) {  
        int soKhachHangMoi = 0;  
        try {  
            URL url = new URL(Publics.URLNHANDULIEU);  
            HttpURLConnection conn =  
                (HttpURLConnection) url.openConnection();  
            conn.setRequestMethod("POST");  
            conn.setRequestProperty("Content-Type", "application/json");  
            conn.setDoOutput(true);  
            conn.setDoInput(true);  
            conn.setReadTimeout(10000);  
            conn.setConnectTimeout(15000);  
            conn.connect();  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
        return soKhachHangMoi;  
    }  
}
```

Trao đổi dữ liệu với Web Server (15)

- VD: Lớp NhanKhachHang kế thừa AsyncTask

```
int khuvuc = arg0[0];
JSONObject ttKhuVuc = new JSONObject();
ttKhuVuc.put("KhuVuc", khuvuc);
OutputStream os = conn.getOutputStream();
OutputStreamWriter out =
    new OutputStreamWriter(os, "UTF-8");
BufferedWriter writer = new BufferedWriter(out);
writer.write(ttKhuVuc.toString());
writer.flush();
writer.close();
out.close();
os.close();
InputStream in =
    new BufferedInputStream(conn.getInputStream());
String strKhachHang = Publics.StreamToString(in);
```

Trao đổi dữ liệu với Web Server (16)

- VD: Lớp NhanKhachHang kế thừa AsyncTask

```
int responseCode = conn.getResponseCode();
if (responseCode == HttpsURLConnection.HTTP_OK) {
    JSONObject jsonObj = new JSONObject(strKhachHang);
    JSONArray arrKhachHang = jsonObj.getJSONArray("data");
    KhachHangAdapter khAdapter =
        new KhachHangAdapter(NhanKhachHangTuServer.this);
    for (int i=0; i<arrKhachHang.length(); i++) {
        JSONObject khMoi = arrKhachHang.getJSONObject(i);
        int mskh = khMoi.getInt("MSKH");
        String hoten = khMoi.getString("HoTen");
        String dienthoai = khMoi.getString("DienThoai");
        String doituong = khMoi.getString("DoiTuong");
        String thanhtoan = khMoi.getString("ThanhToan");
```

Trao đổi dữ liệu với Web Server (17)

- VD: Lớp NhanKhachHang kế thừa AsyncTask

```
if (!khAdapter.KiemTraKH(mskh)) {  
    khAdapter.insertKhachHang(new KhachHang(mskh, hoten,  
                                            dienthoai, doituong, thanhtoan, khuvuc));  
    soKhachHangMoi++;  
} else  
    khAdapter.updateKhachHang(mskh, hoten, dienthoai,  
                               doituong, thanhtoan, khuvuc);  
}  
}  
in.close();  
conn.disconnect();  
} catch (Exception e) {  
}  
return soKhachHangMoi;  
}
```

Trao đổi dữ liệu với Web Server (18)

- VD: Lớp NhanKhachHang kế thừa AsyncTask

```
@Override  
protected void onPostExecute(Integer result) {  
    super.onPostExecute(result);  
    if (result > 0) {  
        setResult(RESULT_OK);  
        Toast.makeText(NhanKhachHangTuServer.this, "Nhận được "  
                + result + " khách hàng mới!", Toast.LENGTH_LONG).show();  
    } else {  
        setResult(RESULT_CANCELED);  
        Toast.makeText(getApplicationContext(), "Không nhận được  
khách hàng mới!", Toast.LENGTH_SHORT).show();  
    }  
    finish();  
}
```

Trao đổi dữ liệu với Web Server (19)

- VD: Trong Activity NhanKhachHangTuServer: kiểm tra trạng thái Internet và thực thi lớp NhanKhachHang (với khu vực là số nguyên xác định trước)

```
if (Publics.HasInternet(this)) {  
    NhanKhachHang nhanKhachHang = new NhanKhachHang();  
    nhanKhachHang.execute(Publics.KHUVUC);  
} else {  
    Toast.makeText(this, "Lỗi kết nối Internet!",  
        Toast.LENGTH_LONG).show();  
}
```