Développement des applications Android Avancées

PRÉPARÉ PAR: YASINE LAKHDARI

CONSULTANT DÉVELOPPEUR

Agenda

- Présentation de la plateforme de développement mobile « Android »
- Design d'une interface utilisateur pour une application Android
- Persistances des données: Manipulation des fichiers
- Persistances des données: Manipulation des bases de données
- Communication Réseaux

Présentation Android

- Android est un système d'exploitation basé sur une version modifiée de Linux.
- Lancé par une startup appelée « Android Inc » pour acquérir le monde du web.
- En 2005 le projet ainsi que l'équipe des développeurs sont rachetés par Google.
- Plateforme de développement mobile free & Open Source.

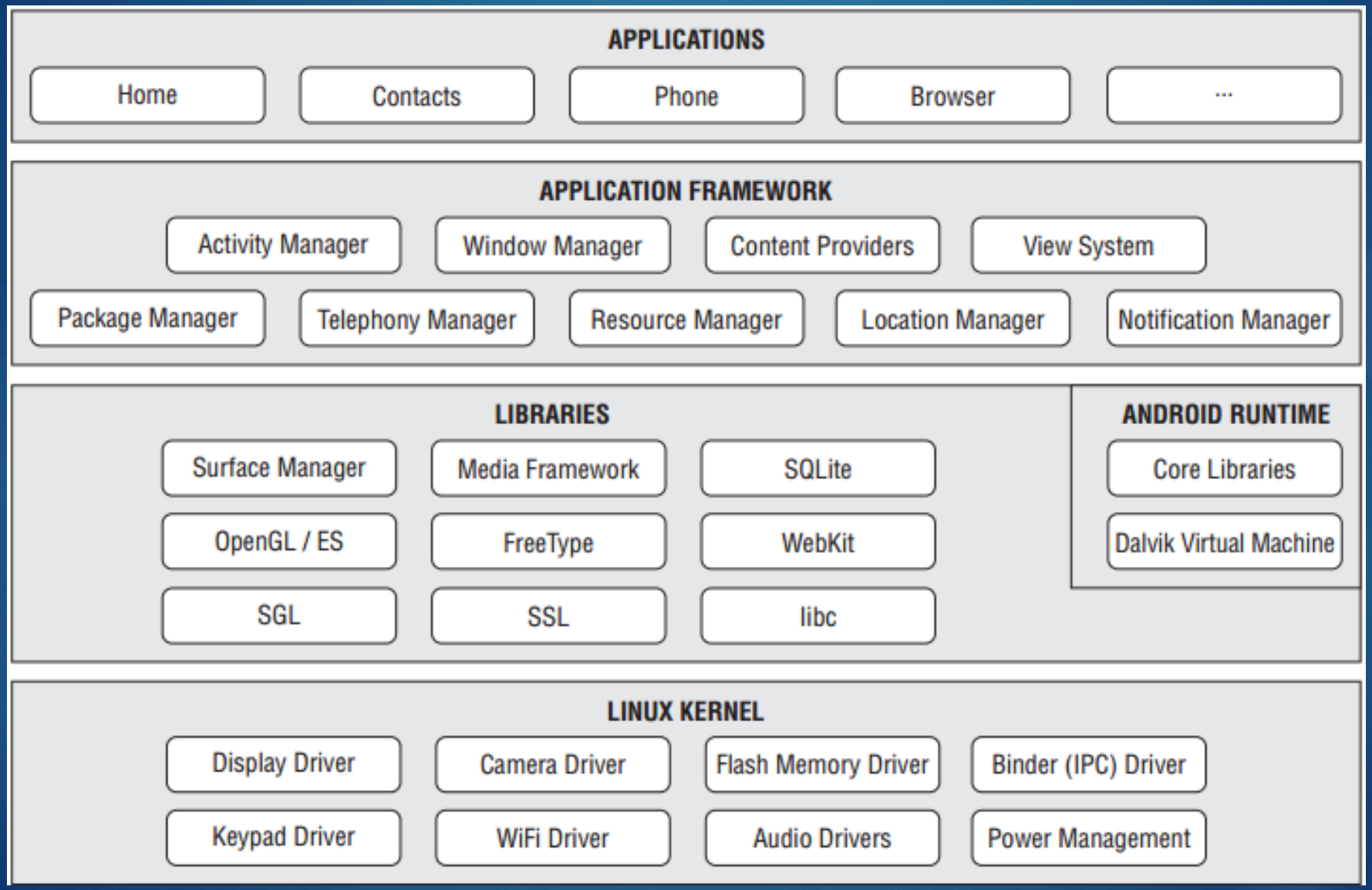
Versions d'Android

Code name	Version	API level
(no code name)	1.0	API level 1
(no code name)	1.1	API level 2
Cupcake	1.5	API level 3, NDK 1
Donut	1.6	API level 4, NDK 2
Eclair	2.0	API level 5
Eclair	2.0.1	API level 6
Eclair	2.1	API level 7, NDK 3
Froyo	2.2.x	API level 8, NDK 4
Gingerbread	2.3 - 2.3.2	API level 9, NDK 5
Gingerbread	2.3.3 - 2.3.7	API level 10
Honeycomb	3.0	API level 11
Honeycomb	3.1	API level 12, NDK 6
Honeycomb	3.2.x	API level 13
Ice Cream Sandwich	4.0.1 - 4.0.2	API level 14, NDK 7
Ice Cream Sandwich	4.0.3 - 4.0.4	API level 15, NDK 8
Jelly Bean	4.1.x	API level 16
Jelly Bean	4.2.x	API level 17
Jelly Bean	4.3.x	API level 18
KitKat	4.4 - 4.4.4	API level 19
Lollipop	5.0	API level 21

Fonctionnalités Android

- Storage: Avec usage du SQLite
- Connectivité: Support de GSM/GPRS,EDGE, UMTS 3G, 4G LTE, WiFi
- Messagerie: Support du MMS,SMS.
- Navigateurs Web: Navigateurs par défaut installé avec support d'autres navigateur comme: Chrome, Opera..
- Support Media: Codec par défaut installé pour supporter: H.263, H.264 (sur 3GP or MP4), MPEG-4 SP, AMR, AMR-WB (sur 3GP), AAC, HE-AAC (sur MP4 3GP), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF et BMP
- Support Matériel: GPS, Senseur Accelorometre, Boussole et Camera integrés.
- Multi-touch.
- Multi-Tâche
- Support Flash: Android 2.3 supports Flash 10.1.
- Tethering: Pour le partage sans/avec fil d'internet.

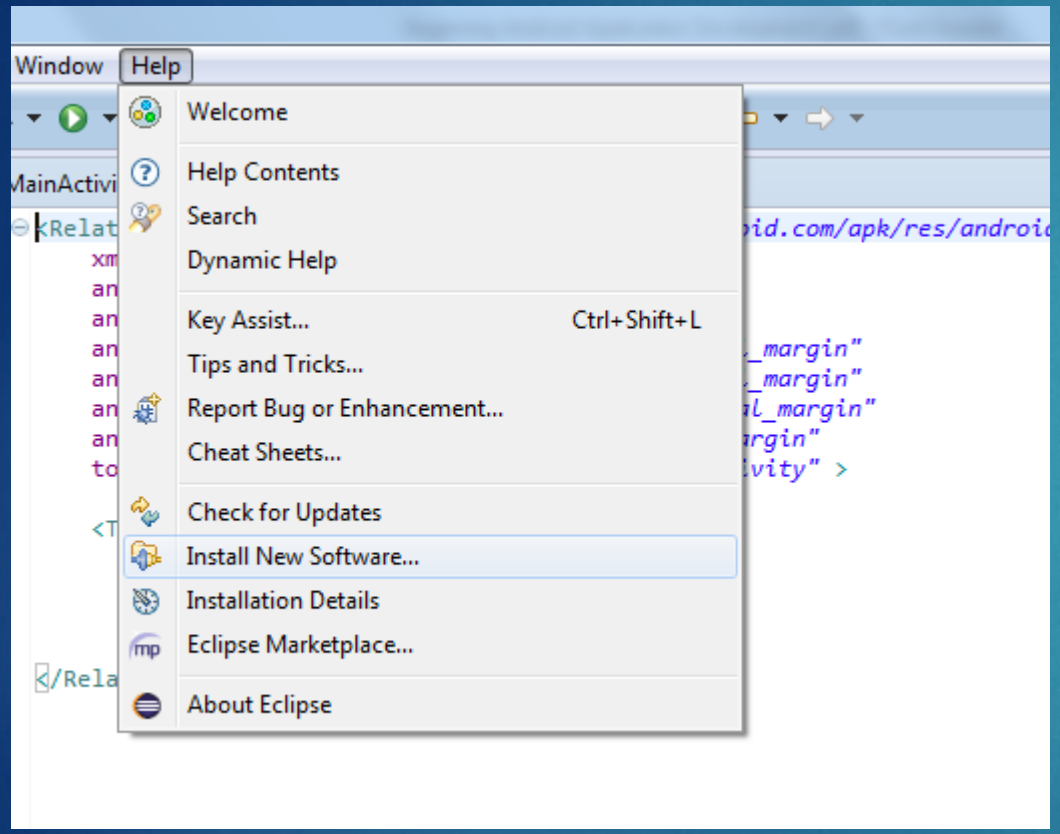
Architecture Android

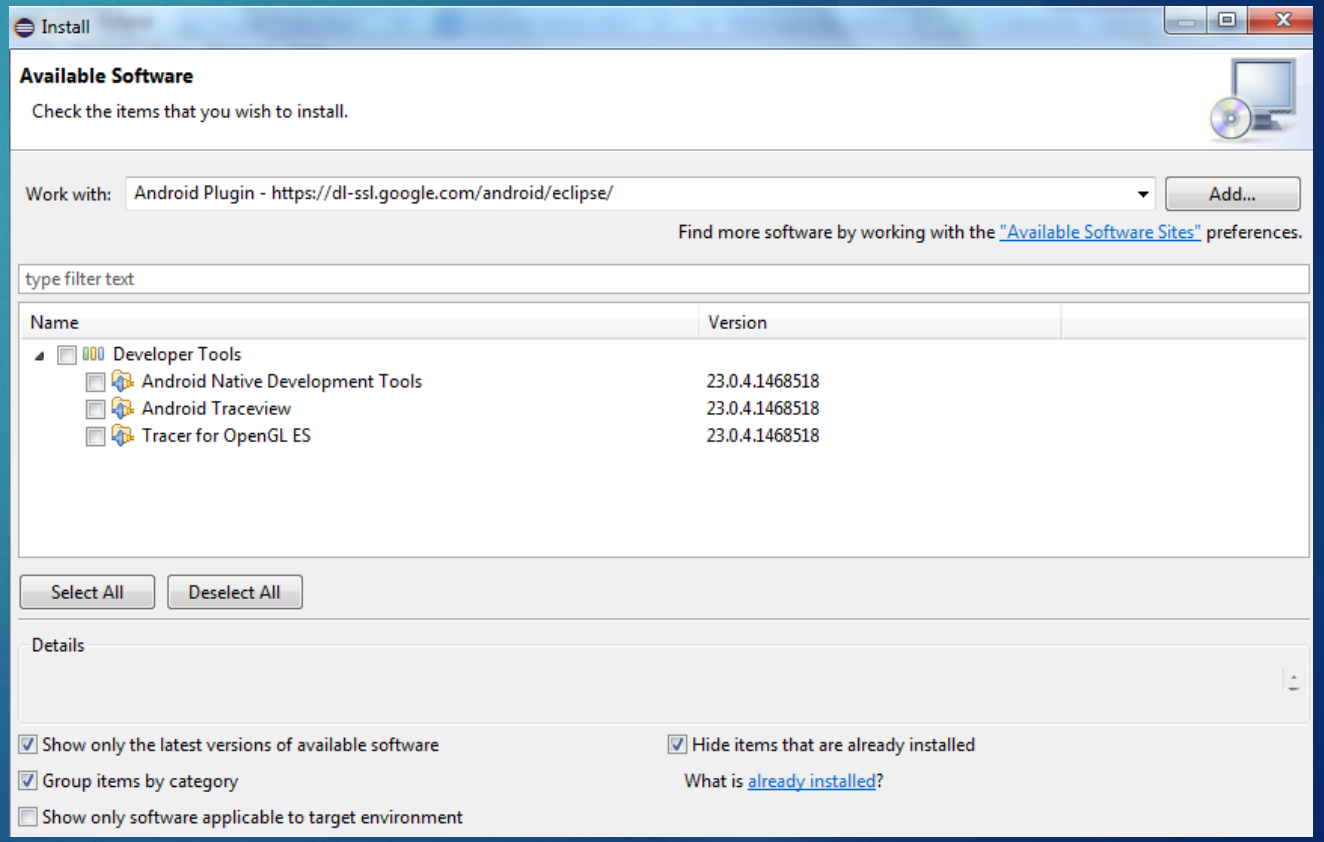


Outils de développement Android

- Java 2 Standard Edition (JDK)
- Android SDK (Software Development Kit)
- Eclipse for Java
- Android Development Tools (ADT) plugin Android pour Eclipse

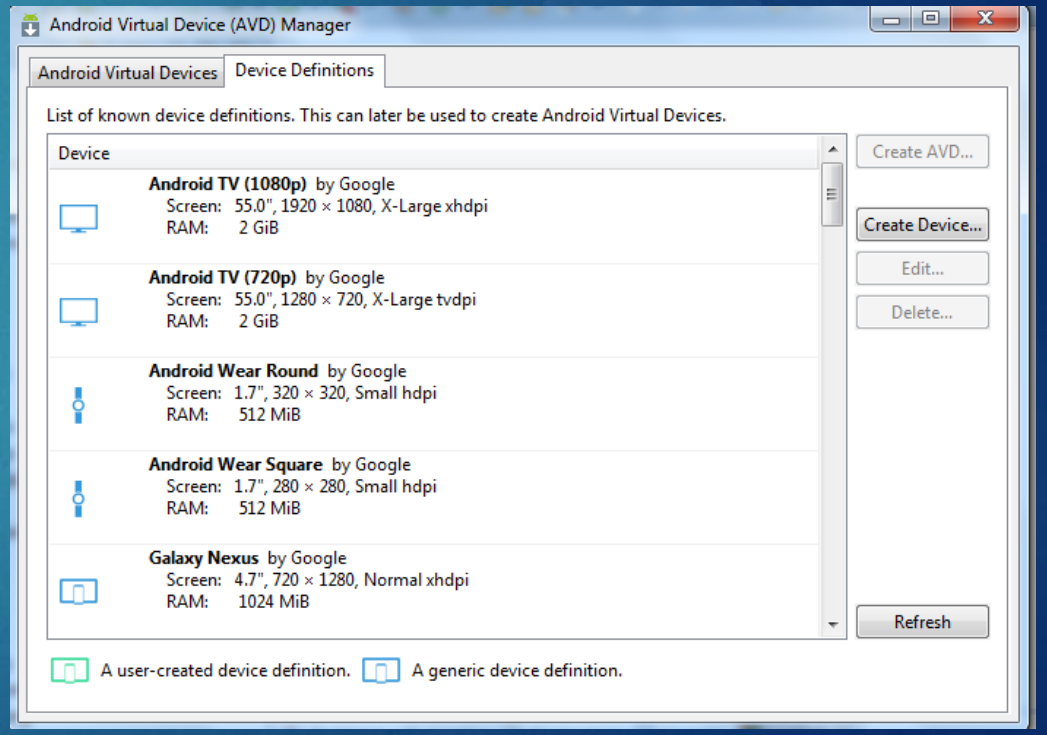
Installation du ADT



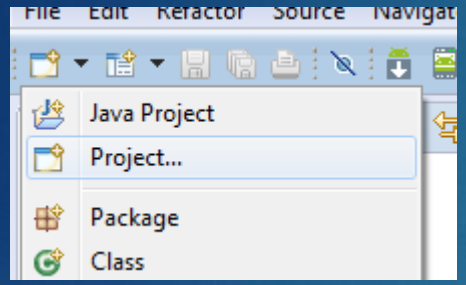


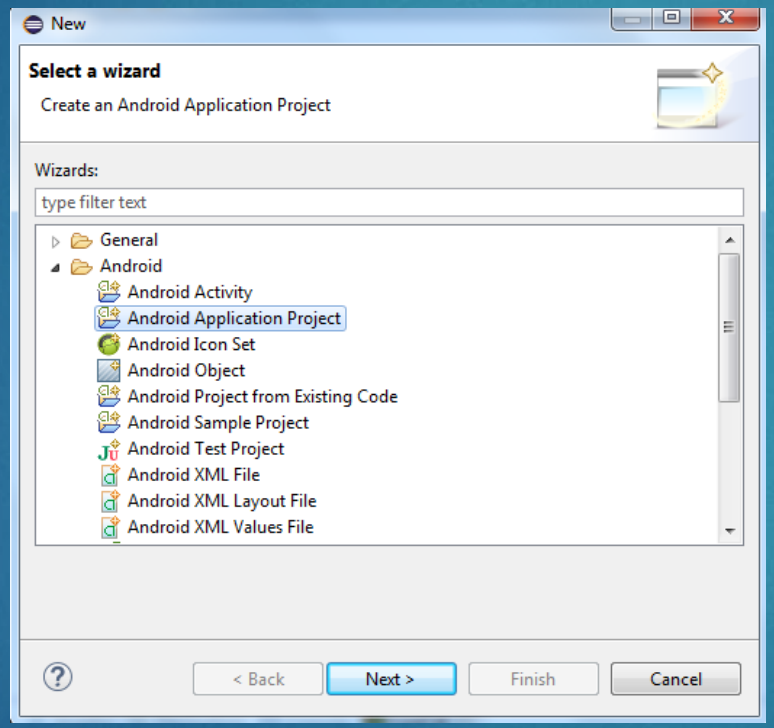
Emulateur « Android Virtual Device »

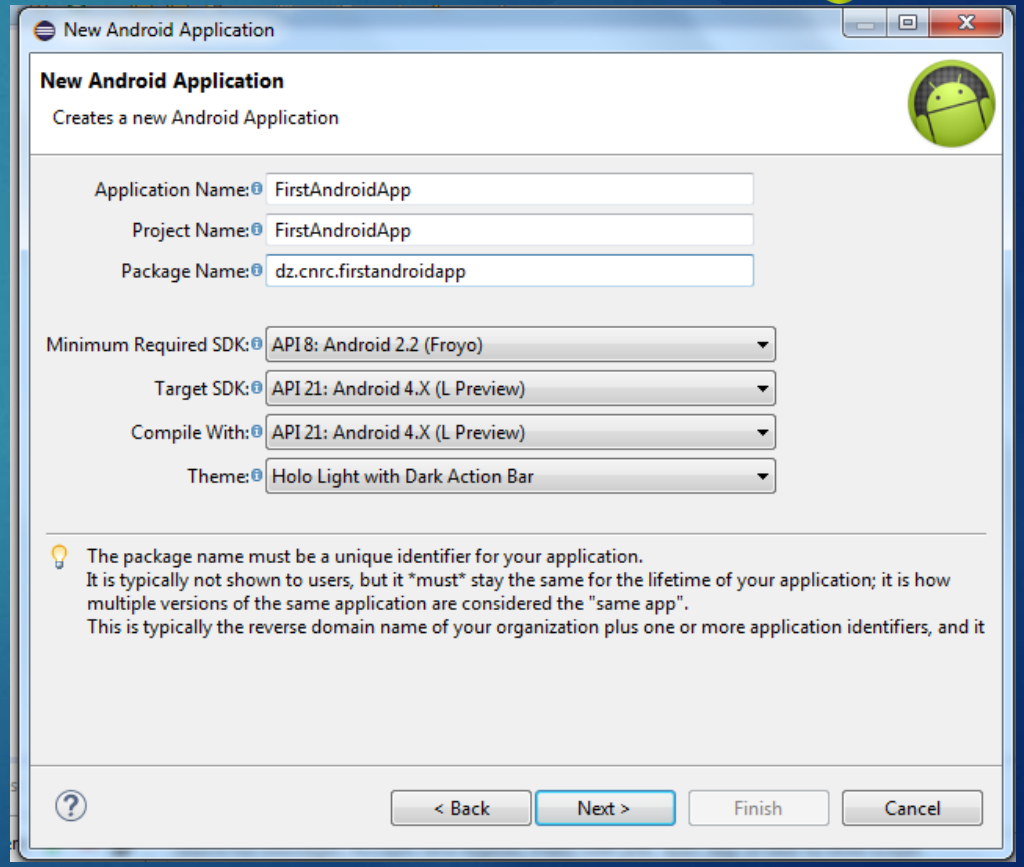
- Outil permet de simuler un smartphone, Tablette…etc, qui fonctionne avec le système d'exploitation Android, il permet aussi de paramétrer:
  - La taille de mémoire allouée.
  - Taille de l'écran
  - Type de processeur
  - Version de l'OS Android.
  - Taille d'espace de stockage interne
  - et carte mémoire miniSD.



Créer votre première application « Android »







Créer votre première application « Android »

Activités & Intent

- Activité:
  - Une fenêtre qui contient une interface utilisateur, avec un ou plusieurs composants UI
  - Classe Java qui hérite de la classe « Activity »

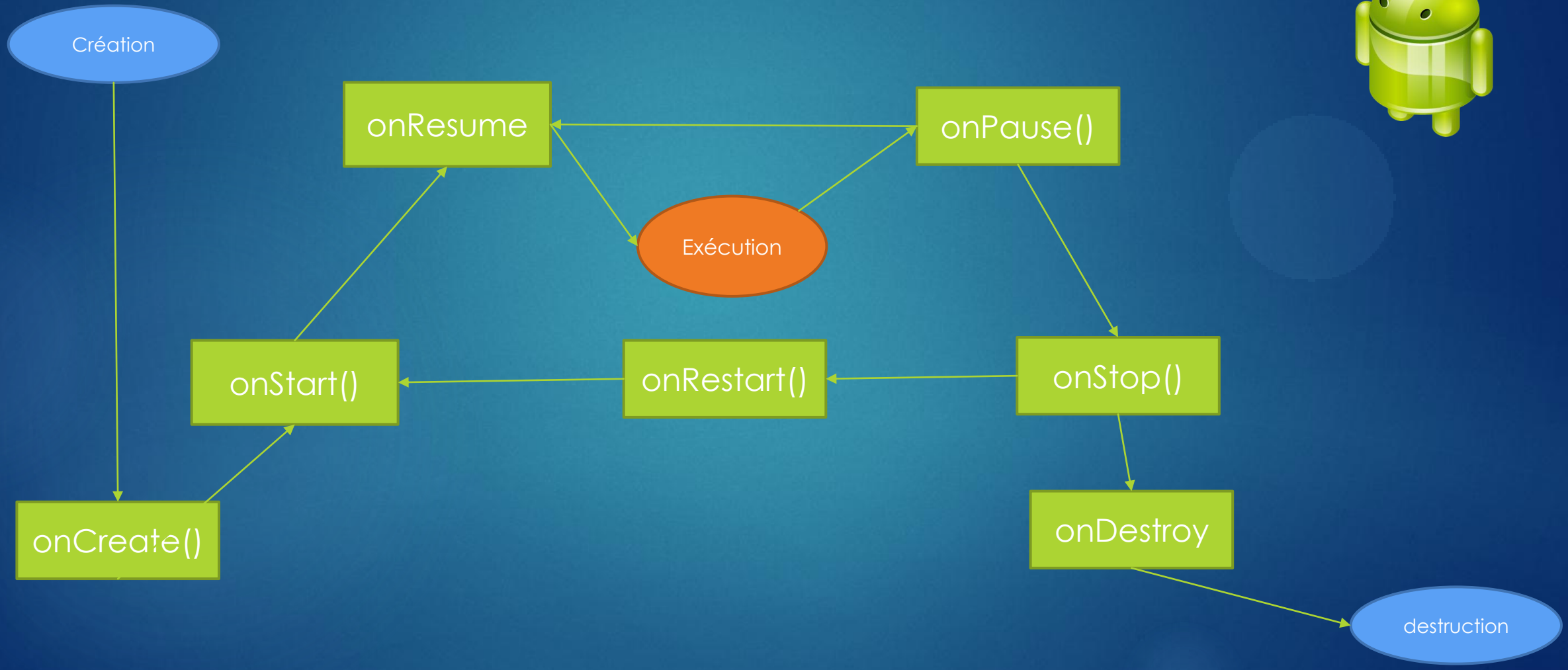
```
import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

  - Chaque activity charge les Composants UI via le fichier XML dénifie dans AndroidManifest.xml

Méthodes de base ”Activity”

- onCreate(): appellee quand l’instance une fois crée.
- onStart(): appellée quand l’activité est visible à l’utilisateur.
- onResume(): appellée quand l’activité commence à interéagire avec l’utilisateur.
- onPause(): appelé quand l’activité en cours est suspendu.
- onStop(): appelé quand l’activité en cours est arrêtée.
- onDestroy(): appelée quand l’activité est détruite par le système, nétoyée de la mémoire.
- onRestart():appellée quand l’activité est redémarrée après un arrêt.

Diagramme d'état « Activité »



Intents

- Ils sont des méchanismes de liaison entre les activités.
- Une Application Android contient de 0 au N activité.
- « intent » est considéré comme un moyen de navigation entre les différentes Activités.
- Syntexe:

```
Intent in = new Intent(MainActivity.this, SecondActivity.class);
```

```
ok.setOnClickListener(new View.OnClickListener()
    {

        @Override
        public void onClick(View v)
        {

            Intent in = new Intent(MainActivity.this, SecondActivity.class);
            startActivity(in);
        }
    });
}
```

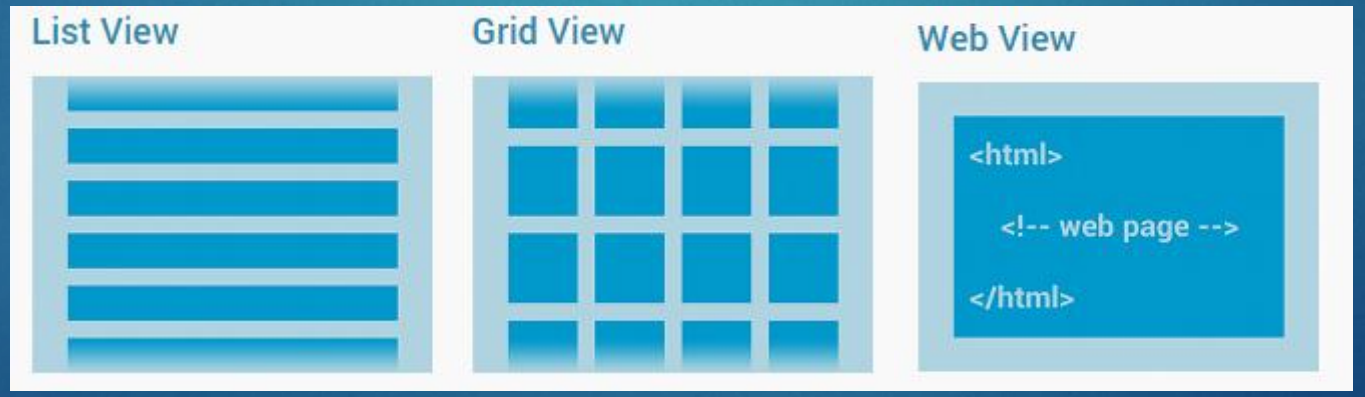
Layouts

- C'est des composant UI
- Définie une structure visuelle pour les activités.
- Peut être définie en mode:
  - Déclaratif XML
  - Impératif sur Java: en créant une instance du Layout dans le fichier classe java
- RelativeLayout
- LinearLayout
- FrameLayout
- GridLayout
- TableLayout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

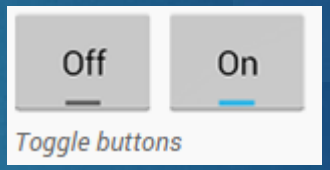
Views

- Représente un formalise de donnée sur l'interface utilisateur
- **ListView**: pour un affichage séquentiel d'éléments
- **GridView**: pour un affichage organisé sous forme de grille.
- **WebView**: pour un affichage des pages via des url web ou bien un directement un flux HTML.
- **ImageView**: pour affichage des média images.



Contrôles UI communs

- Button: Définie un bouton cliquable pour l'utilisateur peut executer une action
- EditText, AutoCompleteTextView: un champs texte editable.
- Checkbox: on/off switch permet au choix booléen.
- RadioGroup, RadioButton: permet la sélection dans un choix multiple.
- ToggleButton: c'est un bouton indiquant l'état on/off
- Spinner: liste déroulante de valeur à sélectionner.

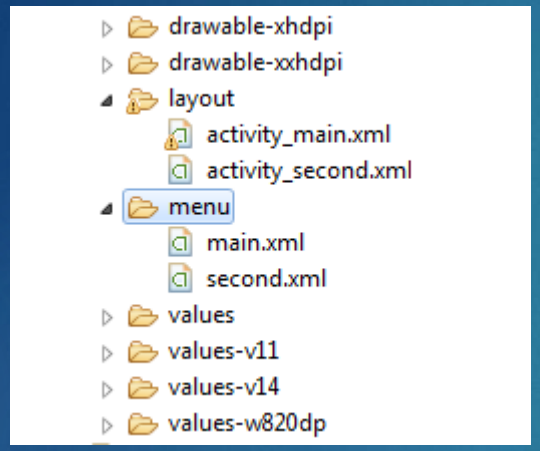


Toggle buttons

- DatePicker, TimePicker: contrôles pour sélection de date et heure.

Menus

- Eléments d'expérience utilisateur.




```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/new_game"
          android:icon="@drawable/ic_new_game"
          android:title="@string/new_game"
          android:showAsAction="ifRoom"/>
    <item android:id="@+id/help"
          android:icon="@drawable/ic_help"
          android:title="@string/help" />
</menu>
```

Persistance des données

Préférence de utilisateurs

- Mécanisme de sauvegarde des donnée de configuration des utilisateurs.
- Exemple: Taille de la police.
- Facilite la gestion des données hétérogènes de petites tailles.
- L'objet « SharedPreferences » permet de saugevarder les données de l'utilisateurs dans un fichier XML.
- L'objet « SharedPreferences » permet aussi de retrouver les donnée avec facilité.
- Le stockage est sous forme de paires « key,Value »

```
btn.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        prefs = getSharedPreferences(prefName, MODE_PRIVATE);
        SharedPreferences.Editor editor= prefs.edit();
        editor.putString(TEXT_VALUE_KEY, editText.getText().toString());
        editor.commit();
        Toast.makeText(getBaseContext(),"Texte est sauvegarder avec success !",Toast.LENGT
    }
});
```

```
prefs = getSharedPreferences(prefName, MODE_PRIVATE);
editText.setText(prefs.getString(TEXT_VALUE_KEY, ""));
```

Stockage sur le fichiers

- Il existe deux type de stockage de fichier:
  - Internal Storage (mémoire interne)
  - SD Storage (carte mémoire)
- Recommandé pour les données non structurés.

- Ecriture:
  - Classe: **FileOutputStream**

```
FileOutputStream str=openFileOutput("test.txt", MODE_WORLD_READABLE);
OutputStreamWriter writer =new OutputStreamWriter(str);
writer.write("C'est un contenu dans un fichier");
writer.flush();
writer.close();
```

- Lecture:
  - Classe: **FileInputStream**

```
FileInputStream str=openFileInput("test.txt");
InputStreamReader reader=new InputStreamReader(str);
char[] inputBuffer = new char[READ_BLOCK_SIZE];
String s = "";
 int charRead;
 while ((charRead = reader.read(inputBuffer))>0)
 {
    String readString =String.copyValueOf(inputBuffer, 0,charRead);
    s += readString;
    inputBuffer = new char[READ_BLOCK_SIZE];
 }
```

Modes d'accès des fichiers

- MODE_WORLD_READABLE: Accessible pour tout le monde.
- MODE_PRIVATE: accès réservé juste à l'application qui l'a créé.
- MODE_APPEND: Ecrit le contenu à la fin du fichier quand il existe déjà.
- MODE_WORLD_WRITABLE: fichier accessible en écriture à tout le monde

Stockage externe (SD Card)

- Ajout de la permission « **android.permission.WRITE_EXTERNAL_STORAGE** »
- Classe « **File** »
- Récupération du chemin du répertoire des la carte mémoire.
- Création du répertoire dans la carte mémoire.
- Création du flux d'écriture « **FileOutputStream** »
- Ecriture du contenu en utilisant la classe: «**OutputStreamWriter**»

```
//---SD Card Storage---
File sdCard = Environment.getExternalStorageDirectory();
File directory = new File (sdCard.getAbsolutePath() +
    "/MyFiles");
directory.mkdirs();
File file = new File(directory, "textfile.txt");
FileOutputStream fOut = new FileOutputStream(file);
```

Manipulation des bases de données

- Moyen efficace et simple pour le stockage des données.
- Sélection des donnée performante via les requêtes SQL.
- Assure l'intégrité des données.
- Android utilise le système des bases de donnée SQLite
- SQLite est un système de gestion des bases de donnée basé sur des fichiers
- Les fichiers de base de données sont sauvegardés dans le repertoire:
  - **Data/data/<nom du package>/databases**
- Classe de manipulation des bases de données hérite de la base de base : **SQLiteOpenHelper**

- Bonne pratique est de créer une Classe **DBHelper** qui contient toutes les méthodes de création, insertion, modification et suppression des données.

Manipulation des bases de données

- Initialisation des variables:

```
public class DBAdapter {
    public static final String KEY_ROWID = "_id";
    public static final String KEY_NAME = "name";
    public static final String KEY_EMAIL = "email";
    private static final String TAG = "DBAdapter";

    private static final String DATABASE_NAME = "MyDB";
    private static final String DATABASE_TABLE = "contacts";
    private static final int DATABASE_VERSION = 1;

    private static final String DATABASE_CREATE =
        "create table contacts (_id integer primary key autoincrement, "
        + "name text not null, email text not null);";

    private final Context context;

    private DatabaseHelper DBHelper;
    private SQLiteDatabase db;

    public DBAdapter(Context ctx)
    {
        this.context = ctx;
        DBHelper = new DatabaseHelper(context);
    }
```

Manipulation des bases de données

- Création de la base de donnée si elle n'existe pas.

```
private static class DatabaseHelper extends SQLiteOpenHelper
{
    DatabaseHelper(Context context)
    {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db)
    {
        try {
            db.execSQL(DATABASE_CREATE);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
    {
        Log.w(TAG, "Upgrading database from version " + oldVersion + " to "
                + newVersion + ", which will destroy all old data");
        db.execSQL("DROP TABLE IF EXISTS contacts");
```

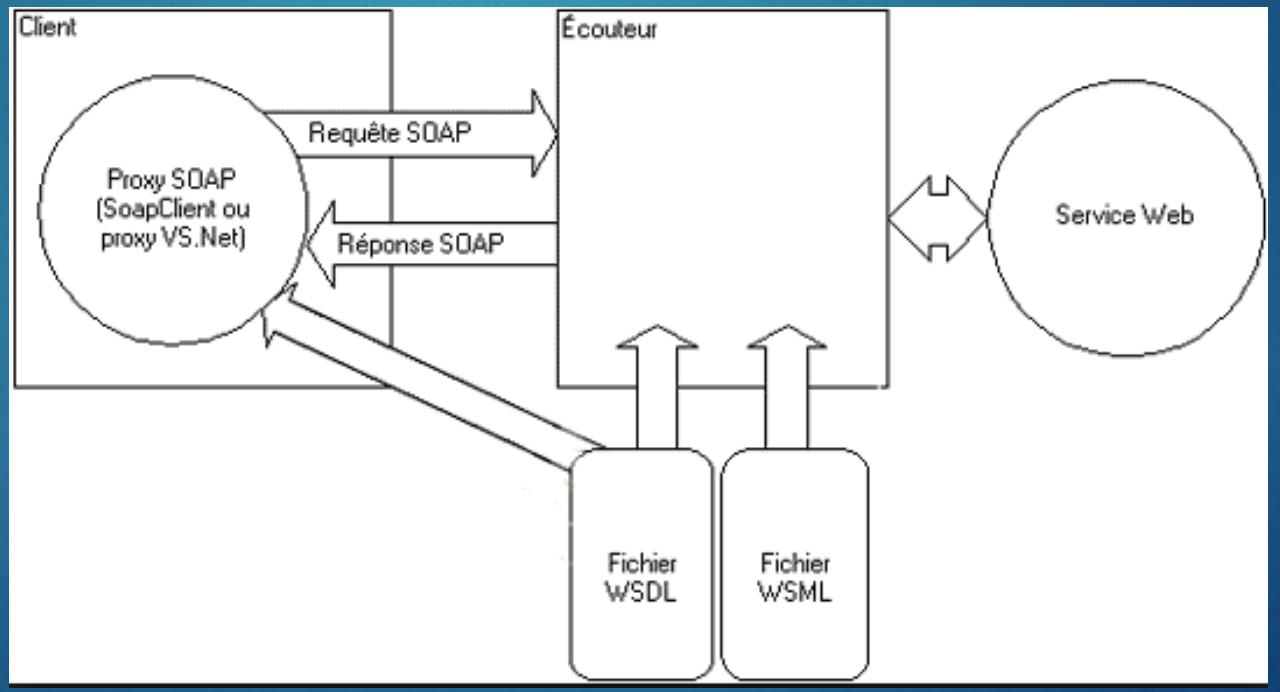
Manipulation des bases de données

```
public long insertContact(String name, String email)
{
    ContentValues initialValues = new ContentValues();
    initialValues.put(KEY_NAME, name);
    initialValues.put(KEY_EMAIL, email);
    return db.insert(DATABASE_TABLE, null, initialValues);
}

//---deletes a particular contact---
public boolean deleteContact(long rowId)
{
    return db.delete(DATABASE_TABLE, KEY_ROWID + "=" + rowId, null) > 0;
}

//---retrieves all the contacts---
public Cursor getAllContacts()
{
    return db.query(DATABASE_TABLE, new String[] {KEY_ROWID, KEY_NAME,
            KEY_EMAIL}, null, null, null, null, null);
}

//---retrieves a particular contact---
public Cursor getContact(long rowId) throws SQLException
{
    Cursor mCursor =
            db.query(true, DATABASE_TABLE, new String[] {KEY_ROWID,
            KEY_NAME, KEY_EMAIL}, KEY_ROWID + "=" + rowId, null,
            null, null, null, null);
    if (mCursor != null) {
        mCursor.moveToFirst();
    }
```

```
public Cursor getContact(long rowId) throws SQLException
{
    Cursor mCursor =
            db.query(true, DATABASE_TABLE, new String[] {KEY_ROWID,
            KEY_NAME, KEY_EMAIL}, KEY_ROWID + "=" + rowId, null,
            null, null, null, null);
    if (mCursor != null) {
        mCursor.moveToFirst();
    }
    return mCursor;
}

//---updates a contact---
public boolean updateContact(long rowId, String name, String email)
{
    ContentValues args = new ContentValues();
    args.put(KEY_NAME, name);
    args.put(KEY_EMAIL, email);
    return db.update(DATABASE_TABLE, args, KEY_ROWID + "=" + rowId, null) > 0;
}
```

Consommation des Web Services

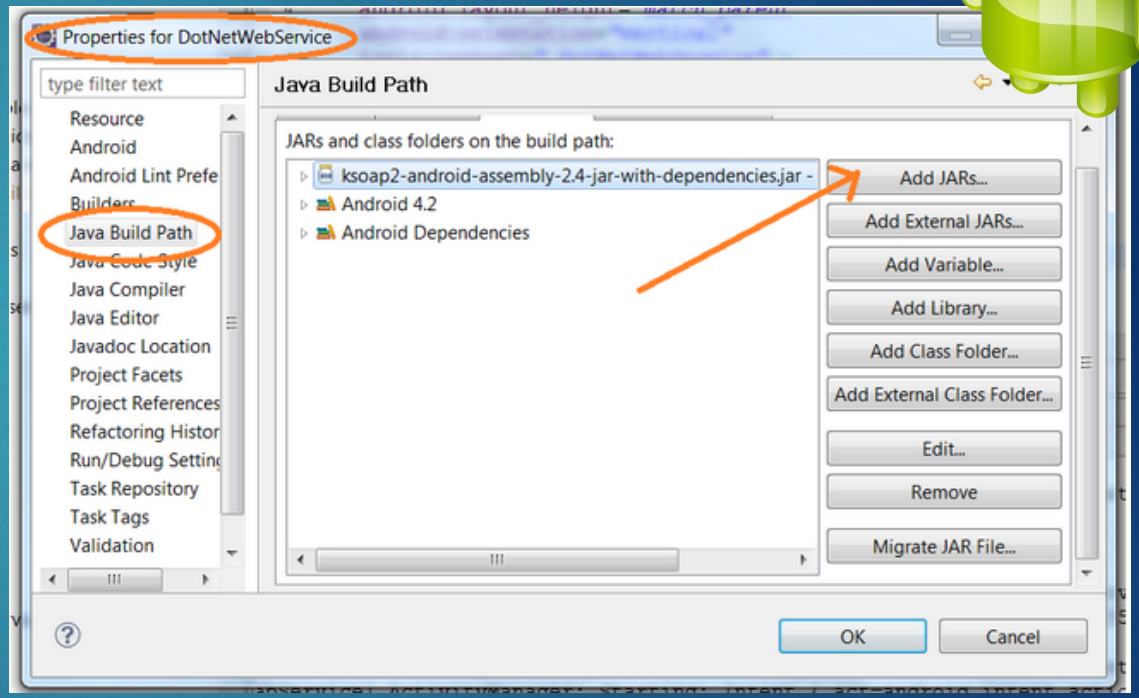
- Web Services sont un moyen de communication de donnée inter-plateformes
- Communication basée sur les protocoles HTTP/HTTPS.
- Format d'échange de donnée est basé sur SOAP (**Simple Object Access Protocol**).



Consommation des Web Services

- Implémentation:
  - Inspection du fichier descriptif du Web Service WSDL.
  - Ajout de la référence kSoap.
  - Ajout du fichier jar ksoap dans

  le répertoire /libs



Consommation des Web Services

- Initialisation des variables:
  - Namespace du Web Service
  - URL: adresse du Service Web
  - Enveloppe Soap avec la version

```
envelope =new SoapSerializationEnvelope(SoapEnvelope.VER12);
envelope.dotNet=true;
envelope.setOutputSoapObject(request);
```

```
String namespace="http://tempuri.org/";
private String url ="http://192.168.1.3/MainService.asmx";
String SOAP_ACTION;
SoapObject request =null, objMessages =null;
SoapSerializationEnvelope envelope;
HttpTransportSE  androidHttpTransport;
```

Consommation des Web Services

- Les appels Web Services doivent être implémenté dans les thread séparés.
- Les appels WS héritent des la classe **AsyncTask<?,?,?>**

```
public class CallWSGetMessage extends AsyncTask<String, String, String>{

    @Override
    protected String doInBackground(String... params) {
        DotNetWebServices ws=new DotNetWebServices();
        String result= ws.GetMessage(params[0]);
        return result;
    }
```