TP … au menu « UI ANDROID »

Pré-requis & Installation (… du couvert)

- soit installer en natif sur vos postes (!!! ATTENTION !!! FromScratch 1,1 Go à télécharger)
  - JDK http://www.oracle.com/technetwork/java/javase/downloads/index.html
  - Eclipse 3.4 / 3.5 http://www.eclipse.org/downloads/
  - Android SDK : http://developer.android.com/sdk/index.html
  - configuration dans Eclipse
    - dans « *help=>install new soft.*» ajouter l'ADT https://dl-ssl.google.com/android/eclipse/
    - installer les 2 features du plugin, relancer Eclipse
    - dans « *windows => préférences => android* » : ajouter le path du SDK + apply
    - dans « *windows => Android SDK & AVD manager* » (relancer Eclipse avant ?)
      - rubrique « *available package* », installer:
        - ***SDK platform Android 2.3*** (1.6 en +, compatibilité ascendante de vos smartphones)
        - ***Samples for SDK API 8***
        - ***Documentation***
        - « BONUS windaube » (si vous n'avez pas oublié votre android chez vous !) :
          - ***driver USB windows***

- soit copier dossier « dev android » (win 32 bit uniquement) sur le bureaux (refaire raccourcies?)
  - JDK nécessaire tout de même (binaire d'install inclut dans le dossier)

TIPS & TRICKS :
  - documentation hors-ligne PATH_SDK/docs/index.html (tout sauf le forum évidement !)

APERO : Préparation pour le débogage

- création d'un AVD (si vous n'avez pas encore de smarphone Android ;))
  - lancer le SDK manager (depuis Eclipse ou directement depuis le PATH_SDK)
  - rubrique « *virtual devices* » => *new*
  - configurer l'AVD (sélectionner juste la target 1.6 pour ce TP) => *create AVD*
  - lancer le maintenant (la 1er fois prend beaucoup de temps) … ne le fermer pas

- … pour les chanceux qui en ont un (et qui n'ont pas oublié leur câble USB :p) :
  - activer les options de débogage sur votre smartphone (cf slide 'debuggage application')
  - une fois connecté en USB au PC, driver windows dans « PATH_SDK/usb_driver »
  - vérifier la connexion avec le smartphone dans la vue « device » d'Eclipse

NB: rien ne vous empêches pas d'avoir AUSSI un AVD … le choix de la target se fera au « run » de l'application

AMUSE-GUEULE : création d'un nouveau projet

file => new => Android Project

- Champs nécessaire :
  - *project name* nom du projet dans Eclipse
  - *Build target* cochez android 1.6 (ou 2.2)
- Rubrique Properties :
  - *application name* 'tp'
  - *package name* 'fr.cnam.android' (ATTENTION unique d'Android market si vous le publiez … metter un URI)
  - cochez « *Create activity* » : main
  - « *finish* »

=> Dans le projet généré, vous avez ...

- en code source : dossier « src »
  - package + main.java (1er activité spécifiée dans la création de projet)
- en ressource : dossier « res »
  - dossier drawable => icône de l'application (que l'on retrouvera dans « menu d'appli » du tel)
  - dossier layout => layout au même nom que la 1er activité crée (ici « main.xml »)
  - dossier values => chaines de caractères associées au projet (ici 2 entrée : app_name + hello)
- dossier « gen » => R.java fichier auto-généré lors du build (si erreur d'installation le supprimer)
- AndroidManifest.xml
- default.properties

Vérifier si l'application se lance sur l'AVD ou sur votre smartphone

- *run => android application =>* (choix de la target si smartphone + AVD)

… et vous avez bien le « Hello World, main ! »

HORS-D'OEUVRES : DDMS & Logcat … et l'activité

NB : concevoir un filtre dans la vue logcat à partir d'un même champs « tag » pour ces opérations de log. Cela sera plus simple pour différencier les logs systemes de celle de votre application.

- ajouter la methode « *onDestroy* » à l'activité principal
- instancier 'log.i' dans les méthodes « *onCreate* » & « *onDestroy* »
- visualiser (vue 'logcat') les messages précédants lors du lancement et de l'arrêt de l'application
- répéter ces opérations avec les méthodes « *onPause* » & « *onResume* »
- relancez l'application, émuler un appel (vue « emulator control »), retrouver le focus de votre application (en raccrochant !). observer le logcat

=> Bien faire le parallèle entre le logcat avec le cycle de vie d'une activité (perte de focus / retour de focus)

PS : on peut remarquer que le processus de l'application est toujours la (cf vue « Devices » dans DDMS) ... et ceci, quelque soit le « launch mode » de l'activité (cf manifest).

- tuer l'application (le processus donc) dans « *onDestroy* » (librairie java standard)

PS : Tourner l'écran en « landscape » (dans l'emulateur ctrl+F12). Que se passe-t-il ? Pourquoi ?

- trouver le moyen pour la tuer 'proprement' … lorsque l' « activity » sait qu'elle « *estfinie* » ;)

PLAT PRINCIPAL: Création de Widget / UI et Interaction avec le code

Dans cette partie nous allons faire abstraction de la programmation de widget en java en travaillant directement sur la template xml (main.xml ici).

– Modifier la template (vous resterez toujours avec le LinearLayout comme parent) afin de rajouter 2 champs de saisies et un bouton afin d'obtenir ceci :

conseils :

– *Travailler avec le layout et non l'XML brut.*
– *Changer les identifiants (id) des champs de saisies et du boutton afin de les retrouver facilement dans le code.*
– *Changer d'autres propriétés des différents widget afin d'obtenir le résultat*

– Créer le handler (dans la methode « onResume ») du bouton ENVOYER. Prendre l'exemple fourni dans la javaDOC de la classe Button. Afficher cet événement dans le logcat (recrée un nouveau tag + filtre)
– Créer une fonction « envoie_message » dans l'activité permettant de récupérer les champs de saisies et les afficher dans le logcat lors de l'appui du bouton ENVOYER
– Créer un Toast à la fin de la fonction en écrivant que 'le message est prêt à être envoyé'

– Créer un handler de menu pour l'activité (méthode « onCreateOptionsMenu ») … en « *ajoutant* » un « *menu* » intitulé ' envoyer ' ;)
– Créer le sélectionneur d'item de menu (méthode « onOptionsItemSelected ») qui appellera la fonction « envoie_message » crée précédemment (uniquement sur sélection du menu 'envoyer')

DESSERT : Communication avec d'autres applications

– Créer un intent de type ACTION_SEND dans la fonction « envoie_message ». Préparer dans celui-ci les différents champs saisies. Démarrer le en lui demandant de « creerunchoix ». ;)

DIGESTIF : Déploiement sur smartphones

– aller dans le fichier manifest rajouter le use-sdk avec la version utilisée android 1.6 (API 4)
– puis rubrique 'exporting' => ' Use the Export Wizard '
– créer un keystore (à ne pas perdre sinon vous ne pourrez plus signer vos applications avec le même nom dans le market) => créer une clé alias + champs nécessaires, exporter l'APK
– Pour l'installation soit … :
  - … par le 'market publisher' si vous avez un compte (… et 25 € à dépenser via Checkout)
  - … en le téléchargeant sur SD puis en l'installant avec une application (ex : appMonster)
  - … ou encore plus simple (et compatible avec l'émulateur) en console :
PATH_SDK/tools/adb install <path+nom>.apk

CORRECTION

main.java

```
package fr.cnam.android;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class main extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Log.i("cycle_appli", "onCreate");
    }

    @Override
    protected void onPause() {
       super.onPause();
       Log.i("cycle_appli", "onPause");
    }

    @Override
    protected void onResume() {
       super.onResume();
       Log.i("cycle_appli", "onResume");

       final Button button_envoyer = (Button) findViewById(R.id.envoyer);
        button_envoyer.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                 Log.i("envoie","appui de la touche ENVOYER ");
                             envoie_message();
            }
        });
    }

    @Override
    protected void onDestroy() {
       // TODO Auto-generated method stub
       super.onDestroy();
       Log.i("cycle_appli", "onDestroy");
       if (isFinishing())
       {
              java.lang.System.exit(0);
       }
    }

    protected void envoie_message()
    {
       EditText sujet = (EditText) findViewById(R.id.sujet);
       Log.i("envoie","sujet:\t"+sujet.getText().toString());
       EditText message = (EditText) findViewById(R.id.message);
       Log.i("envoie","message:\t"+message.getText().toString());

      Toast.makeText(this, "le message est pret à être envoyé", 2000).show();

       Intent EnvoieMessageIntent = new Intent(Intent.ACTION_SEND);
       EnvoieMessageIntent.putExtra(Intent.EXTRA_SUBJECT,sujet.getText().toString());
       EnvoieMessageIntent.putExtra(Intent.EXTRA_TEXT,message.getText().toString());
       EnvoieMessageIntent.setType("text/plain");
        startActivity(Intent.createChooser(EnvoieMessageIntent, "Envoyer un message"));
    }
```

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        menu.add("envoyer");
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getTitle().toString() == "envoyer")
        {
                envoie_message();
        }
        return super.onOptionsItemSelected(item);
    }
}
```

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_height="fill_parent"
android:layout_width="fill_parent">
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" android:text="Sujet"/>
<EditText android:layout_height="wrap_content" android:layout_width="fill_parent"
android:id="@+id/sujet" android:text="entrer le sujet"></EditText>
<TextView android:layout_height="wrap_content" android:layout_width="fill_parent"
android:text="Message"></TextView>
<EditText android:layout_height="wrap_content" android:layout_width="wrap_content"
android:id="@+id/message" android:text="entrer le message"></EditText>
<Button android:layout_height="wrap_content" android:layout_width="fill_parent"
android:id="@+id/envoyer" android:text="ENVOYER"></Button>
</LinearLayout>
```