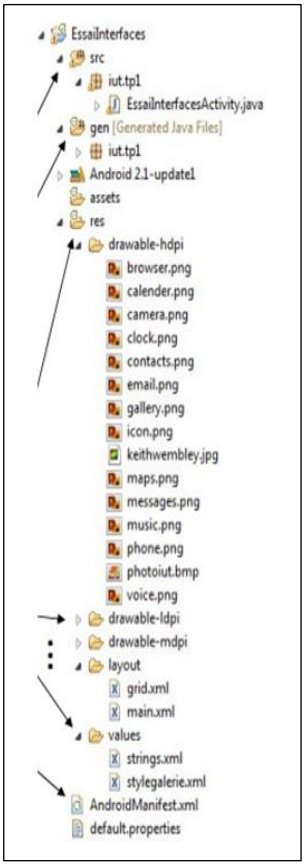
Résumé du cours et Tps : Applications mobile

Contenu de la racine

plusieurs éléments peuvent construire le répertoire racine du projet :

- ***AndroidManifest.xml*** est un fichier XML qui décrit l'application à construire et les composants – activités, services, etc. – fournis par celle-ci.
- ***bin/*** contient l'application compilée.
- ***gen/*** contient le code source produit par les outils de compilation d'Android.
- ***libs/*** contient les fichiers JAR extérieurs nécessaires à l'application.
- ***src/*** contient le code source Java de l'application.
- ***res/*** contient les ressources – icônes, descriptions des éléments de l'interface graphique (*layouts*), etc. – empaquetées avec le code Java compilé.



La première fois que vous compilerez le projet, la chaîne de production d'Android créera le fichier R.java dans le paquetage de l'activité "principale". Ce fichier contient un certain nombre de définitions de constantes liées aux différentes ressources de l'arborescence **res/.**

l'arborescence **res/** contient les ressources, c'est-à-dire des fichiers statiques fournis avec l'application, soit sous leur forme initiale soit, parfois, sous une forme prétraitée.

Parmi les sous-répertoires de **res/,** citons :

- ***res/drawable/*** pour les images (PNG, JPEG, etc.) ;
- ***res/layout/*** pour les descriptions XML de la composition de l'interface graphique ;
- ***res/menu/*** pour les descriptions XML des menus ;
- ***res/raw/*** pour les fichiers généraux (un fichier CSV contenant les informations d'un compte, par exemple) ;
- ***res/values/*** pour les messages, les dimensions, etc. ;
- ***res/xml/*** pour les autres fichiers XML généraux que vous souhaitez fournir.

Utilisation des widgets de base

Labels

Le label (TextView pour Android) est le widget le plus simple, En Java, un label est une instance de la classe TextView. Un élément TextView possède de nombreux autres attributs,
notamment :

- **android:typeface** pour définir le type de la police du label (monospace, par exemple) ;
- **android:textStyle** pour indiquer si le texte doit être en gras (bold), en italique (italic) ou les deux (bold_italic) ;
- **android:textColor** pour définir la couleur du texte du label, au format RGB hexadécimal (#FF0000 pour un texte rouge, par exemple).

Voici le contenu du fichier de positionnement du projet Basic/Label :

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Vous vous attendiez à quelque chose de plus profond ?"
/>
```

Boutons

Button étant une sous-classe de TextView, tout ce qui a été dit dans la section précédente concernant le formatage du texte s'applique également au texte d'un bouton.

Champs de saisie

Outre les boutons et les labels, les champs de saisie forment le troisième pilier de la plupart des outils de développement graphiques. Avec Android, ils sont représentés par le widget **EditText**, qui est une sous-classe de **TextView**, déjà vue pour les labels.

En plus des propriétés standard de **TextView** (android:textStyle, par exemple), EditText possède de nombreuses autres propriétés dédiées à la construction des champs. Parmi elles, citons :

- ***android:autoText*** pour indiquer si le champ doit fournir une correction automatique de l'orthographe.
- ***android:capitalize*** pour demander que le champ mette automatiquement en majuscule la première lettre de son contenu.
- ***android:digits*** pour indiquer que le champ n'acceptera que certains chiffres.
- ***android:singleLine*** pour indiquer si la saisie ne s'effectue que sur une seule ou plusieurs lignes (autrement dit, <Enter> vous place-t-il sur le widget suivant ou ajoute-t-il une nouvelle ligne ?).

Outre ces propriétés, vous pouvez configurer les champs pour qu'ils utilisent des méthodes de saisie spécialisées, avec les attributs ***android:numeric*** pour imposer une saisie uniquement numérique, ***android:password*** pour masquer la saisie d'un mot de passe et ***android:phoneNumber*** pour la saisie des numéros de téléphone. Pour créer une méthode de saisie particulière (afin,

par exemple, de saisir des codes postaux ou des numéros de sécurité sociale), il faut implémenter l'interface **InputMethod** puis configurer le champ pour qu'il utilise cette méthode, à l'aide de l'attribut ***android:inputMethod***.

Méthodes utiles

- **findViewById()** permet de retrouver un widget fils d'après son identifiant.
- setContentView() on définit le View que contiendra (affichera) notre activité. Ici c'est le layout activite_principale (identifié par R.layout.activite_principale).

```
public class MainActivity extends Activity {
TextView tv;
EditText mat, nom, prenom, groupe, moyenne;
Button btn1, btn2;
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);
		tv= (TextView) findViewById(R.id.textView6);
		mat = (EditText) findViewById(R.id.editText1);
		btn1 = (Button) findViewById(R.id.button1);

		btn1.setOnTouchListener(new OnTouchListener() {

			@Override
			public boolean onTouch(View v, MotionEvent event) {
				// TODO Auto-generated method stub
				Toast.makeText(getApplicationContext(), "onTouch",
Toast.LENGTH_LONG).show();
				aff();
				afficher(v),
				return false;
			}
		});
btn1.setOnClickListener(new OnClickListener() {

	@Override
	public void onClick(View v) {

		Toast.makeText(getApplicationContext(), "onclick",
Toast.LENGTH_LONG).show();
	}
});
		}
	private void aff(){
		tv.setText("la méthode private");
	}

	public void afficher(View v){
			tv.setText("la méthode public");
		}
}
```

Partie de déclaration des objets

Partie de instanciation

Appel de la méthode aff() à l'intérieure de la classe

Appel de la méthode afficher (v) à l'extérieure de la classe (dans XML)

Le cadre de la méthode onClick (View v), appelée dans XML

Le cadre de la méthode onClick (View v), appelée dans la classe

Une partie du fichier XML

```
<LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content" >

                <Button
                    android:id="@+id/button1"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:textColor="#ffaaaaaa"
                    android:text="Button" />

                <Button
                    android:id="@+id/button2"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"

                    android:onClick="afficher" />

            </LinearLayout>
```

TP1 (ci-joint)

TP2(ci-joint)

TP3(ci-joint)