

Mobilitéé: Programmation Android

1

Erick STATNER

Maître de Conférences en Informatique

Université des Antilles

erick.stattner@univ-antilles.fr

www.erickstattner.com

Description de l'enseignement

Objectifs pédagogiques:

- Se familiariser à la Programmation d'applications pour mobile
- Maîtriser les principes autour des applications Android
- Concevoir des applications graphiques sous Android
- Mettre en place la persistance des données

Organisation:

- 30h
- 1 CC + 1 CT

Sommaire

1. Android: Présentation, configuration et principes
2. Premières applications Android
3. Les interfaces
- 4. Evènements et échanges**
5. Persistance des données

Chapitre IV.

Evènements et échanges

1. Gestion du clic
2. Naviguer entre les activités
3. Echange de données simples
4. Echange d'objets

IV. Évènements et échanges

Gestion du clic

Gestion du clic

- ▶ Indispensable dans la mise en place d'une IHM
- ▶ Repose sur la notion d'évènements
- ▶ Modèle émetteur/récepteur
 1. Un élément déclenche un évènement
 2. Un, ou plusieurs objets, détectent l'évènement et mettent en place les actions appropriées
- ▶ L'écouteur doit être au préalable être enregistré auprès de l'élément

IV. Evènements et échanges

Gestion du clic

Au niveau de l'écouteur

- ▶ Doit implémenter l'interface ***onClickListener***
- ▶ Cette interface ne possède qu'une seule méthode abstraite ***public void onClick(View v)***
 - ▶ Appelée lors du déclenchement de l'évènement
 - ▶ La vue ***v*** en paramètre est l'élément qui a déclenché l'évènement
- ▶ L'écouteur peut être
 - ▶ L'activité elle-même
 - ▶ Une classe dédiée
 - ▶ Une classe anonyme

IV. Evènements et échanges

Gestion du clic

Au niveau de l'élément

- ▶ Associer l'écouteur à l'élément susceptible de déclencher l'évènement
- ▶ Utiliser sur l'objet **view** la méthode ***setOnClickListener***
 - ▶ Prend en paramètre l'écouteur

Comment récupérer une référence vers un élément créé dans le layout ?

IV. Evènements et échanges

Gestion du clic

Au niveau de l'élément

- Associer l'écouteur à l'élément susceptible de déclencher l'évènement
- Utiliser sur l'objet **view** la méthode ***setOnClickListener***
 - Prend en paramètre l'écouteur

Comment récupérer une référence vers un élément créé dans le layout ?

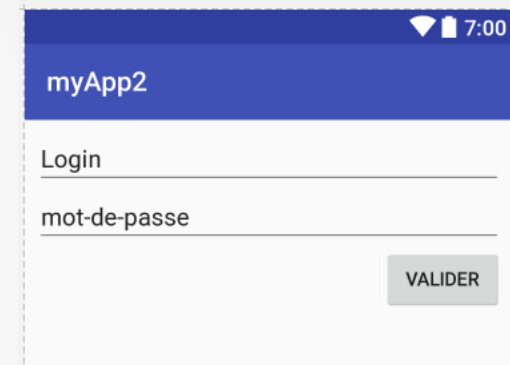
- methode ***findViewById***
Permet de récupérer une référence vers un élément crée dans un layout

IV. Evènements et échanges

Gestion du clic

Exemple de gestion avec 1 bouton

- L'activité gère elle-même le clic



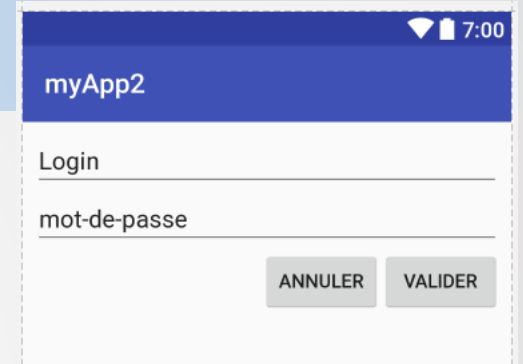
```
public class MainActivity extends AppCompatActivity implements OnClickListener {  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        Button btVal = (Button) findViewById(R.id.btValider);  
        btVal.setOnClickListener(this);  
    }  
  
    public void onClick(View v) {  
        Log.v("TEST", "clic sur le bouton valider");  
    }  
}
```

IV. Evènements et échanges

Gestion du clic

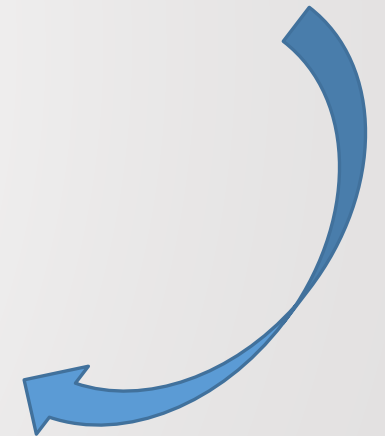
Exemple de gestion avec 2 boutons

- L'activité gère elle-même le clic



```
public class MainActivity extends AppCompatActivity implements OnClickListener
{
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button btVal = (Button) findViewById(R.id.btValider);
        btVal.setOnClickListener(this);
        Button btAnn = (Button) findViewById(R.id.btAnnuler);
        btAnn.setOnClickListener(this);
    }
    public void onClick(View v){
        if(v.getId() == R.id.btAnnuler){
            Log.v("TEST", "clic sur le bouton ANNULER");
        }
        else if(v.getId() == R.id.btValider) {
            Log.v("TEST", "clic sur le bouton VALIDER");
        }
    }
}
```

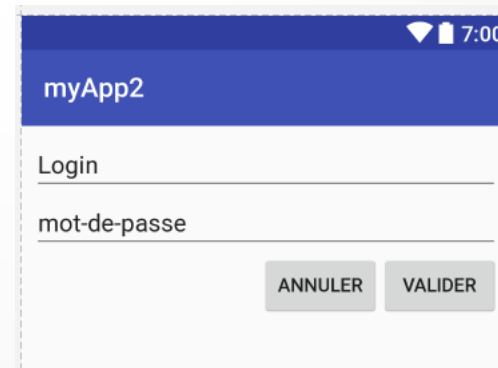


IV. Evènements et échanges

Gestion du clic

Exercice

- Compléter l'écouteur pour qu'il affiche également dans les log le login et le mot de passe si l'utilisateur clique sur valider



The screenshot shows a mobile application window titled "myApp2". The status bar at the top right indicates a Wi-Fi signal, a battery icon, and the time 7:00. The app's main content area contains a login form with two text input fields: "Login" and "mot-de-passe". Below the input fields are two buttons: "ANNULER" and "VALIDER".

IV. Evènements et échanges

Naviguer entre les activités

Navigation entre activités

- Brique essentielle d'une IHM
- Possible grâce aux intentions (**intent**)
- Chaque activité est accessible à l'aide d'un **intent**
- **Objectif**
 - Mettre en place la logique de navigation dans l'application

IV. Evènements et échanges

Naviguer entre les activités

Navigation entre activités

- ▶ Pour exécuter une activité, utiliser la méthode **public void startActivity(Intent intent)**
 - ▶ `intent` correspond à l'activité à exécuter
- ▶ Pour créer un intent, utiliser le constructeur de la classe **Intent**
public Intent(Context context, Class activiteALancer)
- ▶ **Exemple:** passer de l'activité courante à l'activité B
Intent intent = new Intent(this, B.class)
startActivity(intent)

Rappel: Toutes les activités doivent être déclarées dans le manifeste de l'application !

IV. Evènements et échanges

Naviguer entre les activités

Exercice

- ▶ Compléter l'exercice précédent pour que
 - ▶ Si les données ne sont pas saisies, les champs login et mot de passe sont vidés
 - ▶ Si les données sont saisies, l'utilisateur soit redirigé sur une nouvelle activité qui affiche simplement le message "Bonjour"

IV. Evènements et échanges

Echange de données simples

Passage de données entre les vues

- ▶ Effectuer à l'aide des **Extra** disponibles dans les **intent**
- ▶ Basé sur un couple *clé/valeur* via un *bundle*
- ▶ Utiliser la méthode **putExtra(<clé>, <valeur>)**
 - ▶ *clé*: identifiant de l'élément à passer
 - ▶ *valeur*: valeur de la donnée que l'on souhaite passer
- ▶ Disponibles uniquement pour les types de base en Java (*int, String, float, double, boolean, byte, etc.*)

IV. Evènements et échanges

Echange de données simples

Exemple: dans la méthode `onClick`

```
public void onClick(View v) {  
    EditText nom = (EditText) this.findViewById(R.id.champNom);  
  
    Intent inte = new Intent(this, activite2.class);  
    inte.putExtra("NOM", nom.getText().toString());  
    startActivity(inte);  
}
```


IV. Evènements et échanges

Echange de données simples

Récupérer les données dans une activité

- ▶ Pour récupérer les données au sein d'une activité
- ▶ Trois étapes
 1. Récupérer l'**intent** de l'activité à l'aide de la méthode **getIntent()**
 2. Appeler la méthode pour récupérer la donnée, selon le type
 - String getStringExtra(<clé>)
 - int getIntExtra (<clé>)
 - float getFloatExtra (<clé>)
 - boolean getBooleanExtra (<clé>)
 - etc.
 3. Traiter les données dans l'activité

IV. Evènements et échanges

Echange de données simples

Récupérer les données dans une activité

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_activite2);  
  
    Intent inte = getIntent();  
    String t = inte.getStringExtra("NOM");  
  
    TextView nomUser = (TextView) findViewById(R.id.nomUser);  
    nomUser.setText(t);  
}
```

IV. Evènements et échanges

Echange d'objets

Passer des objets d'une activité à l'autre

- ▶ Le système d'**extra** est limité aux types primitifs
- ▶ Android introduit les **parcelable**
- ▶ Objectif
 - ▶ Transférer des objets d'une activité A à une activité B
 - ▶ Proche du mécanisme de sérialisation

IV. Evènements et échanges

Echange d'objets

Passer des objets d'une activité à l'autre

- ▶ La classe doit implémenter l'interface **Parcelable**
- ▶ Deux fonctions à implémenter
 - ▶ **int describeContents()**
pour décrire le contenu du Parcel, en particulier le nombre d'objets spéciaux
 - ▶ **void writeToParcel(Parcel dest, int flags)**
pour écrire le contenu de l'objet dans un Parcel
- ▶ La classe doit contenir également un objet **Creator** chargé de créer une instance de l'objet depuis un Parcel

IV. Evènements et échanges

Echange d'objets

Passer des objets d'une activité à l'autre

```
public class ID implements Parcelable {
    private String login;
    private String mdp;

    protected ID(Parcel in) {
        login = in.readString();
        mdp = in.readString();
    }

    @Override
    public void writeToParcel(Parcel dest, int f){
        dest.writeString(login);
        dest.writeString(mdp);
    }

    @Override
    public int describeContents() {
        return 0;
    }
}

public static final Creator<ID> CREATOR = new
Creator<ID>(){
    @Override
    public ID createFromParcel(Parcel in) {
        return new ID(in);
    }

    @Override
    public ID[] newArray(int size) {
        return new ID[size];
    }
};
```

IV. Evènements et échanges

Echange d'objets

Passer un objet d'une classe A à une classe B

- ▶ De côté de la classe A

```
Intent inte = new Intent(this, activite2.class);
ID idUser = new ID(log.getText().toString(), mdp.getText().toString());
inte.putExtra("NOM", idUser);
this.startActivity(inte);
```

- ▶ Du côté de la classe B

```
Intent inte = this.getIntent();
ID idUser = inte.getParcelableExtra("NOM");
TextView nomUser = (TextView)findViewById(R.id.nomUser);
nomUser.setText(idUser.getLogin()+" "+idUser.getMdp());
```