DemoQuizz | Éléments de base

page d'accueil

La page d'accueil va comporter 3 éléments :

- le nom de l'application dans un style de titre
- un message explicatif de bienvenue
- un bouton permettant le démarrage du quizz

➔ Ajoutez à l'aide de l'éditeur graphique du fichier les deux composants dans la page : un widget `TextView` de type `Large`, un de type `Medium` et un widget de type `Button`

➔ Ajoutez les textes à votre convenance

Un premier lancement de l'application donne ceci :

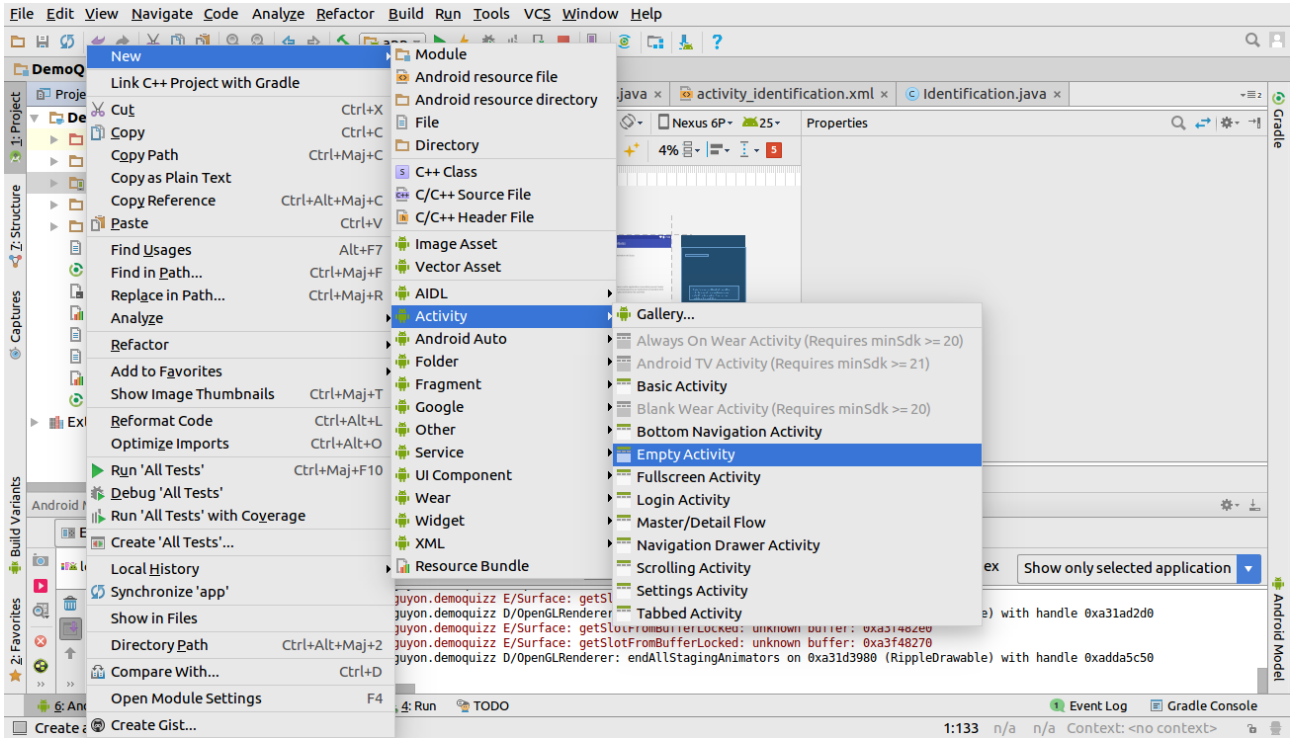
2. Construction de la page de saisie de l'identification

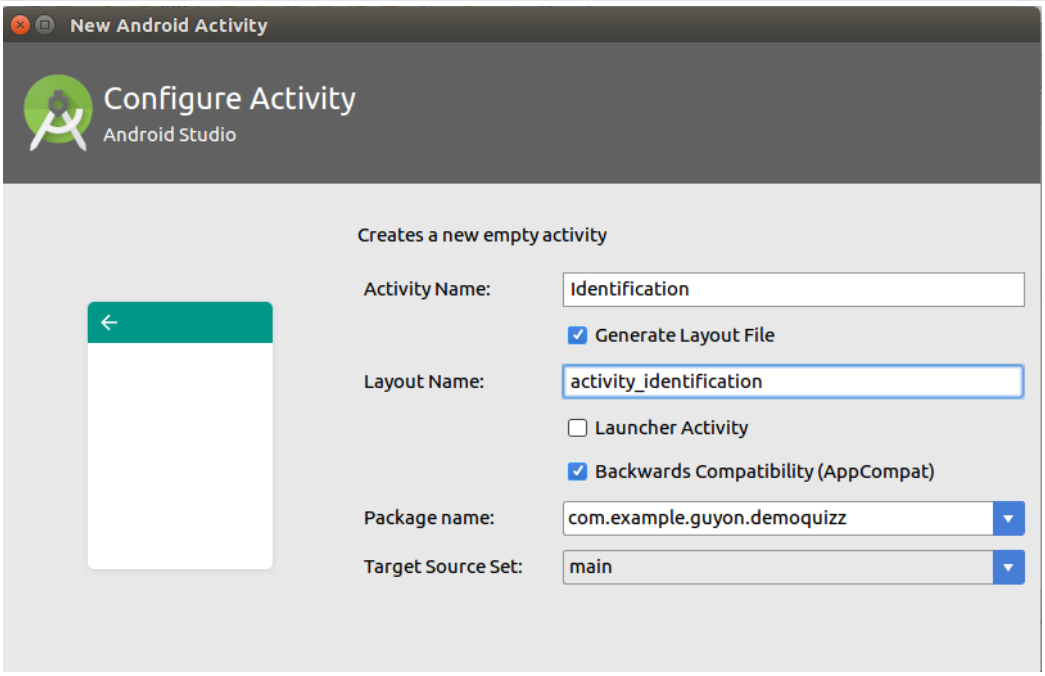
- ***L'appel d'une activité par un bouton***

Jusqu'à présent, les manipulations et la construction de l'application ont porté uniquement sur des ressources XML définies dans un fichier, ce qui pour le programmeur pouvait sembler particulier voire frustrant... On entre cette fois-ci dans le code Java et le SDK Android avec le principe des **intentions**.

Voir le tuto2, pour implémenter l'interface `View.OnClickListener` et ajouter un listener au bouton.

➔ Créez une nouvelle activité (Empty Activity)





➔ Modifier à l'aide de l'éditeur graphique le fichier `activit_identication` `afin d obtenir l'écran suivant :`

- ***L'appel d'une activité par un bouton***

Jusqu'à présent, les manipulations et la construction de l'application ont porté uniquement sur des ressources XML définies dans un fichier, ce qui pour le programmeur pouvait sembler particulier voire frustrant... On entre cette fois-ci dans le code Java et le SDK Android avec le principe des **intentions**.

Littéralement, une intention (`intent`) lie une demande à un récepteur qui peut être une nouvelle activité ou une opération dans l'activité, un service ou un composant (comme par exemple une interface de type `BroadcastReceiver`, concept vu plus tard).

Au niveau du SDK Android, un objet `intent` se déclare classiquement avec l'opérateur `new` :

```
Intent nom = new Intent(this,classe_à_appeler);
```

➔ Ajoutez le code suivant dans la méthode **`onClick`**`()` dans votre fichier **MainActivity.java** qui réagit dès que le bouton go a été cliqué :

```
Intent intentIdentification = new Intent(this,Identification.class);
this.startActivityForResult(intentIdentification,10);
```

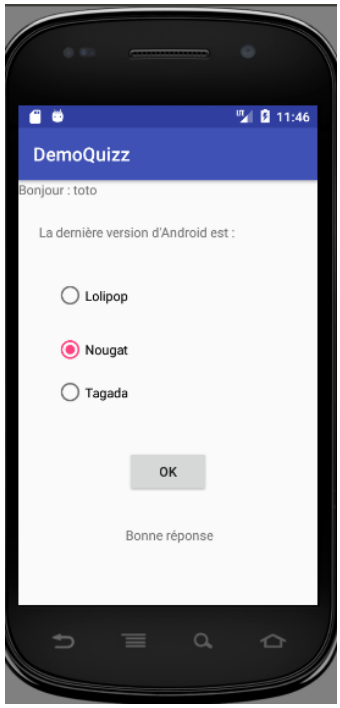
Notez que la création de l'intention demande comme paramètres le contexte (`this`) et la classe (`Identification.class`).

- ***Vérification***

➔ Lancez l'pplication ; un clic sur le bouton go affiche la sous-activité Identification (et le bouton retour revient à la page principale) .

3. Construction de la page de Quizz

Il faut maintenant traiter le passage au Quizz proprement dit. Cela consiste à créer une nouvelle intention qui contiendra les premières questions avec un écran personnalisé au prénom de l'utilisateur. On a donc l'obligation de transmettre la valeur prénom d'une intention à l'autre.



Voir le tuto2 pour présenter les boutons radios et le traitement

➔ Comme précédemment, rajoutez le code du gestionnaire d'événements pour la gestion du deuxième bouton

Le passage d'un ou plusieurs paramètres se réalise grâce à la méthode `putExtra()` d'une intention avec dans ce cas deux paramètres (clé, valeur). La méthode `putExtras()` s'utilise avec l'objet `Bundle` pour le passage d'une série de paramètres comparable à la notion `serializable` en Java mais utilisé par Android sous un type plus adapté et rapide.

➔ Ajoutez alors dans la méthode `onClick()` dans le fichier **Identification.java** le code suivant qui est assez simple à comprendre :

```
String prenom = editText.getText().toString();
Intent intentQuizz = new Intent(this,Quizz1.class);
intentQuizz.putExtra("Utilisateur",prenom);
this.startActivityForResult(intentQuizz,10);
```

- ***Récupération d'une valeur et utilisation du widget Bouton radio***

On se base maintenant dans la classe `Quizz.java`.

➔ Intégrez un `TextView` et ne lui donnez-lui pas de valeur initiale

La valeur de ce widget va être positionnée par le code en récupérant la valeur du widget `EditView` transmise par la précédente intention.

➔ Utilisez le code ci-dessous afin de récupérer la valeur entrée à l'activité précédente et l'affecter au `TextView` (déclarez aussi les variables nécessaires induites par ce code) :

```
TextView leUtilisateur = (TextView)findViewById(R.id.prenom);
String utilisateur = this.getIntent().getExtras().getString("Utilisateur");
leUtilisateur.setText(utilisateur);
```

➔ Intégrez un dernier `TextView` dans le fichier XML représentant la réponse en le nommant `reponse` et en laissant la valeur du texte vide.

Évolution de votre projet :

- ***Validation d'une valeur inexistante avec message***

On doit interdire la validation d'une valeur vide ou inversement, on doit imposer l'entrée d'un prénom ou identifiant quelconque. Sous Android l'apparition de messages est dévolue à l'objet `Toast` qui informe l'utilisateur et qui disparaît sans intervention de l'utilisateur.

Voici le code de déclaration d'un **Toast** (pour plus de détail voir la documentation de l'API (http://developer.android.com/reference/android/widget/Toast.html) :

- ***Faire un quizz de plusieurs questions (une activité par question) , on passera automatiquement à la question suivante lorsque l'on cliquera sur le bouton OK et on affichera le nombre de points par exemple si la personne a répondu convenablement à la première question elle passe automatiquement à la deuxième question et on affiche en bas 1 point ou 0 point dans le cas contraire (avec un cumul des points pour les autres questions).***