
***Atelier : Création d'un Splash Screen Animé avec* ProgressBar**

Etape 1 : Assurez-vous d'avoir ajouté le logo ou l'image dans le dossier `res/drawable`.

Etape 2 : Création de l'interface utilisateur (XML) du Layout SplashScreen

Etape 3 : Ajoutez une ImageView pour afficher l'image animée et une ProgressBar pour indiquer le chargement.

Etape 4 : Personnalisez la disposition et le style de votre Splash Screen selon vos préférences.

Etape 5 : Créez une nouvelle classe Java pour gérer la logique de chargement du Splash Screen.

Etape 6 : Utilisez des méthodes asynchrones comme AsyncTask ou Handler pour simuler un chargement pendant quelques secondes

Etape 7 : Après un délai de chargement, utilisez un Intent pour passer à l'activité suivante de votre application.

Etape 8 : Testez votre Splash Screen sur différents appareils et émulateurs pour vous assurer qu'il fonctionne correctement.

Etape 9 : Expérimentez avec différentes animations, styles et durées de chargement pour créer une expérience utilisateur unique.

NB : N'oubliez pas d'ajouter votre activité Splash Screen dans le fichier `AndroidManifest.xml`

Annexe :

Explication des Méthodes Asynchrones pour Simuler un Chargement : AsyncTask et Handler

Les méthodes asynchrones sont des fonctionnalités essentielles dans le développement d'applications Android pour gérer des tâches qui ne peuvent pas être exécutées sur le thread principal de l'application sans bloquer l'interface utilisateur.

Deux des méthodes les plus couramment utilisées pour cela sont AsyncTask et Handler. Explorons chacune de ces méthodes en détail :

1.AsynkTask :

Fonctionnement : AsyncTask est une classe fournie par le framework Android qui permet d'exécuter des opérations en arrière-plan et de mettre à jour l'interface utilisateur sur le thread principal. Elle prend en charge trois étapes principales :

`onPreExecute(),doInBackground(),onPostExecute()`

Utilisation : Pour simuler un chargement, vous pouvez exécuter la partie de chargement dans la méthode **`doInbackground()`** Dans **`onPreExecute(),`**vous pouvez initialiser votre ProgressBar ou effectuer d'autres opérations de préchargement, et dans **`onPostExecute()`**vous pouvez masquer la ProgressBar ou passer à l'activité suivante après le chargement.

Voici un exemple simple d'utilisation d'AsyncTask pour simuler un chargement pendant quelques secondes :

2.Handler :

Fonctionnement : Handler est une classe utilisée pour communiquer avec le thread associé à un thread particulier. Il permet d'exécuter des tâches sur le thread principal à des moments spécifiques.

Utilisation : Vous pouvez utiliser un Handler pour définir un délai avant d'exécuter une action particulière, comme masquer une ProgressBar ou passer à l'activité suivante.

Voici un exemple simple d'utilisation de Handler pour pour définir un délai avant de passer à l'activité suivante :

```
new Handler().postDelayed(new Runnable() {
    @Override
    public void run() {
        // Passer à l'activité suivante après un délai de 5 secondes
        startActivity(new Intent(MainActivity.this, NextActivity.class));
finish();  }
}, 5000); // Délai de 5 secondes
```

```
public class LoadingTask extends AsyncTask<Void, Void, Void> {
@Override
protected void onPreExecute() {
// Initialisation avant le chargement
progressBar.setVisibility(View.VISIBLE);
}
@Override
protected Void doInBackground(Void... voids) {
// Chargement en arrière-plan
try {
Thread.sleep(5000); // Simuler un chargement pendant 5 secondes
} catch (InterruptedException e) {
e.printStackTrace();
}
return null;
}
@Override
protected void onPostExecute(Void aVoid) {
// Actions après le chargement
progressBar.setVisibility(View.GONE);
// Passer à l'activité suivante
startActivity(new Intent(MainActivity.this, NextActivity.class));
}
}
```