DAA – Lab 1

Auteurs :
Thibault Seem
Pascal Perrenoud

Date : 30.10.2022

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD

www.heig-vd.ch

1. ACTIVITÉS 2

Mise en œuvre 2
Bouton « back » 2
Rotation d'écran 2

2. FRAGMENTS 3

Mise en œuvre 3
Sauvegarde de ColorFragment 3
Rotation de l'écran 3

3. FRAGMENTMANAGER 4

Première activité 4
Rotation de l'écran, maintien de la pile 4
Fragment : « add » et « replace » 4

1. Activités

Mise en œuvre

La disposition des deux activités sont gérées à travers leur layout.xml respectif. Ces layouts décrivent les boutons présents, leur identifiant, leur texte ainsi que leur disposition à l'écran. Un « Contrat de résultat d'activité » lie les deux activités. Ce contrat est appelé par la première activité pour appeler la seconde et recevoir en retour un statut et une valeur.

Bouton « back »

Que se passe-t-il si l'utilisateur appuie sur « back » lorsqu'il se trouve sur la seconde Activité ?

Le contrat retourné n'aura pas été changé, il va retourner un « null ». Le texte ne sera donc pas changé.

Rotation d'écran

Que faut-il mettre en place pour que vos Activités supportent la rotation de l'écran ? Est-ce nécessaire de le réaliser pour les deux Activités, quelle est la différence ?

La rotation d'écran est gérée dans le Manifest ou dans les layouts, qui peuvent ou non forcer une orientation.

La rotation d'écran provoque la « réinitialisation » de l'état de l'activité. Il faut donc être attentif à sauvegarder l'état des différents champs modifiés à l'aide de la fonction « onSaveInstanceState » qui permet de sauvegarder un état. Il faut également récupérer les valeurs dans le «onRestoreInstanceState » afin de remettre les valeurs en place, comme celle du texte, ainsi que celle du champ de saisie.

2. Fragments

Mise en œuvre

Les deux fragments sont utilisés dans un layout à l'aide d'un « RelativeLayout » et des « FragmentContainerView » qui permettent de charger les fragments dans des views et des containers au lieu d'utiliser l'écran entier. L'application principale ne gère donc que le chargement du layout. Les fragments chargent leur layout non plus dans le OnCreate avec le « setContentView » mais dans « onCreateView » en donnant leur layout à un « LayoutInflater ».

Sauvegarde de ColorFragment

Les deux Fragments fournis implémentent la restauration de leur état. Si on enlève la sauvegarde de l'état sur le ColorFragment sa couleur sera tout de même restaurée, comment pouvons-nous expliquer cela ?

Le « ARG_HEX_COLOR » est également sauvegardé dans un companion marqué JvmStatic. Sa valeur est gérée à travers cet objet et donc sauvegardé et restauré grâce à ça.

Rotation de l'écran : 2 counters

Si nous plaçons deux fois le CounterFragment dans l'Activité, nous aurons deux instances indépendantes de celui-ci. Comment est-ce que la restauration de l'état se passe en cas de rotation de l'écran ?

Les deux compteurs gardent bien leur propre valeur puisqu'elles sont sauvegardées par la fonction « onSaveInstanceState » dans un « outState » différent pour les 2 compteurs. Ces « instances » sont des instances de la classe « CounterFragment » et sont donc indépendantes. C'est le principe de la POO.

3. FragmentManager

Première activité

A l'initialisation de l'Activité, comment peut-on faire en sorte que la première étape s'affiche automatiquement ?

La première activité est chargée en appelant le layout du premier fragment dans le layout de l'activité hôte. Nous nous servons des valeurs par défaut de ce premier fragment comme initialisation. Il serait également possible d'ajouter un chargement dans l'appel à la fonction « onCreate » : en testant si une state est donnée, on peut savoir si c'est la première création de l'activité et initialiser à ce moment la première étape.

Rotation de l'écran, maintien de la pile

Comment pouvez-vous faire en sorte que votre implémentation supporte la rotation de l'écran ? Nous nous intéressons en particulier au maintien de l'état de la pile de Fragments et de l'étape en cours lors de la rotation.

La pile de Fragments est gérée automatiquement lors d'une rotation de l'écran. Elle est détruite et reconstruite à l'identique, sans intervention de notre part, par le FragmentManager. Cependant, les états interne aux différents fragments ne sont pas reproduits. Ayant réutilisé un fragment de la partie 2, son état est géré, sauvegardé et restauré.

Pour gérer ces dernier, l'utilisation de la méthode onSaveInstanceState permet de sauvegarder les informations dans un Bundle lors de la destruction du Fragment par le FragmentManager au retournement d'écran. Lors de la recréation de ce Fragment pour l'écran retourné, le Bundle lui est attaché. Une fois ceci fait, lors de la prochaine utilisation du Fragment, la méthode onViewCreated est appelée, et dans celle-ci, on vérifie si le Bundle est vide ou non, et en fonction, on peut récupérer les informations présentent auparavant.

Fragment : « add » et « replace »

Dans une transaction sur le Fragment, quelle est la différence entre les méthodes add et replace ?

La fonction « add » permet d'ajouter un fragment à la view. La fonction « replace » permet de retirer les précédents fragments et de remplacer la view par le nouveau fragment.