Mobile solutions

Kocsis Gergely

2019.10.07.

Activity save state

Download the app created on the last class named ShoppingList from the page of the class and open it in Android Studio.

Run it and check what happens in the below cases:

- Add a new item to the list and rotate the phone by 90°
- Add a new item to the list, then open the list of items again, but instead of choosing an item go back to the main Activity with the ←
- Exit thee application using the o button and return
- Exit thee application using the □ button and return
- Exit thee application using the ← button and return

Activity save state

Observe the onCreate method of the main Activity and note that it has a parameter to reset a saved instance state.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    ...
```

Activity save state

Observe the onCreate method of the main Activity and note that it has a parameter to reset a saved instance state.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    ...
```

To provide more saved values to the state used in the onCreate method, we need to override the onSaveInstanceState method.

```
@Override
protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    outState.putString(SAVESTATE_LIST_TEXT, itemsList.getText().toString());
    outState.putBoolean(SAVESTATE_LISTEMPTY_BOOL, isListEmpty);
}
```

String constants in order to make the code more readable and to avoid errors by typos

Activity save state

In order to reset a saved state we need to alter the onCreate method.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    itemsList = findViewById(R.id.listTextView);

    if (savedInstanceState!=null){
        itemsList.setText(savedInstanceState.getString(SAVESTATE_LIST_TEXT));
        isListEmpty=savedInstanceState.getBoolean(SAVESTATE_LISTEMPTY_BOOL);
    }
}


@Override
protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    outState.putString(SAVESTATE_LIST_TEXT, itemsList.getText().toString());
    outState.putBoolean(SAVESTATE_LISTEMPTY_BOOL, isListEmpty);
}
```

Activity save state

Download the app created on the last class named ShoppingList from the page of the class and open it in Android Studio.

Run it and check what happens in the below cases:

- ~~Add a new item to the list and rotate the phone by 90°~~
- Add a new item to the list, then open the list of items again, but instead of choosing an item go back to the main Activity with the ←
- Exit thee application using the o button and return
- Exit thee application using the □ button and return
- Exit thee application using the ← button and return

Activity save state

The first issue has been solved but for some reason in the rest of the cases the state is not reset.
The reason is that in theese cases a new Activity instance is ran in whick the state of the other (original) instance is not stored.

There are two solutions for this:

1) Use persistence to save the data
2) Set the Application not to start a new Activity instance when we come back, just return to a previously instantiated instance already present in the calling stack.

Activity save state

The first issue has been solved but for some reason in the rest of the cases the state is not reset.
The reason is that in theese cases a new Activity instance is ran in whick the state of the other (original) instance is not stored.

There are two solutions for this:

1) Use persistence to save the data
2) Set the Application not to start a new Activity instance when we come back, just return to a previously instantiated instance already present in the calling stack.

```
<activity android:name=".MainActivity"
    android:launchMode="singleTop">
```

Activity save state

Download the app created on the last class named ShoppingList from the page of the class and open it in Android Studio.

Run it and check what happens in the below cases:

- ~~Add a new item to the list and rotate the phone by 90°~~
- ~~Add a new item to the list, then open the list of items again, but instead of choosing an item go back to the main Activity with the ←~~
- ~~Exit thee application using the o button and return~~
- ~~Exit thee application using the □ button and return~~
- ~~Exit thee application using the ← button and return~~

:)

Fragments

Create a new dynamic Fragment with which the user get the possibility to select the color of the Apple (red, green) after the Apple button is pushed.

As a first step create a new Fragment (New → Fragment → Fragment (Blank)
When doing this generate the Layout, but do not include the callback and factory methods.

Fragments

Create a new dynamic Fragment with which the user get the possibility to select the color of the Apple (red, green) after the Apple button is pushed.

As a first step create a new Fragment (New → Fragment → Fragment (Blank)
When doing this generate the Layout, but do not include the callback and factory methods.

Add a new TextView and two RadioButtons in a RadioGroup to the Fragmet. It is advised to use ConstraintLayout in order to be able to position the elements easily.

The texts of the RadioButtons shall be “Red” and “Green” The TextView should contain the text “Color: “.

Fragments

Add a so called placeholder to the layout of the ItemsListActivity before the buttons

```
<FrameLayout
    android:id="@+id/frame_containder"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
</FrameLayout>
```

This will be the place of the fragment.

Fragments

Adding the following method to the Java code of the ItemsListActivitywe make it possible to show the fragment in the placeholder.

```
public void displayColorFragment(){
    BlankFragment colorFragment = BlankFragment.newinstance();
    FragmentManager fragmentManager = getSupportFragmentManager();
    FragmentTransaction fragmentTransaction = fragmentManager
            .beginTransaction();
    fragmentTransaction.add(R.id.frame_containder,
            colorFragment).addToBackStack(null).commit();
}
```

In ordr to make the above code work of course we need to add the newinstance method to the Fragment

Fragments

The only thing missing is to add the finctionality to the Radiogroup. Namely after seecting a color if we push the Apple button again, the color is added together with the “Apple” text to the list.

```
(onCreateView)
final View rootView =
        inflater.inflate(R.layout.fragment_blank, container, false);
final RadioGroup radioGroup = rootView.findViewById(R.id.radioGroup);
radioGroup.setOnCheckedChangeListener(new
RadioGroup.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(RadioGroup radioGroup, int i) {
        View radioButton = radioGroup.findViewById(i);
        int index = radioGroup.indexOfChild(radioButton);
        switch (index) {
            case 0: colorName="Red";   break;
            case 1: colorName="Green"; break;
            default: break;
        }
    }
});
return rootView;
```