Mobile solutions

Kocsis Gergely

2019.10.01.

Multi-window applications

In order to make an application build up from multiple windows you just simply have to create multiple activities.

Intents help you navigate between Activities:

“An Intent provides a facility for performing late runtime binding between the code in different applications. Its most significant use is in the launching of activities, where it can be thought of as the glue between activities. It is basically a passive data structure holding an abstract description of an action to be performed.”

Shoppinglist application

In the main window we have a list and an “Add” button. Pressing the button result in the opening of a new window where we can choose from different predefined items to be added to the list.

At first create a simple application the opens a new Activity when we press the button.

Shoppinglist application

In the main window we have a list and an “Add” button. Pressing the button result in the opening of a new window where we can choose from different predefined items to be added to the list.

At first create a simple application the opens a new Activity when we press the button.

To start a new Activity:

```
Intent intent = new Intent(this, ItemsActivity.class);
startActivity(intent);
```

Shoppinglist application

In the main window we have a list and an “Add” button. Pressing the button result in the opening of a new window where we can choose from different predefined items to be added to the list.

At first create a simple application the opens a new Activity when we press the button.

To start a new Activity:

```
Intent intent = new Intent(this, ItemsActivity.class);
startActivity(intent);
```

Set parent-child relation in AndroidManifest.xml:

```
<activity android:name=".ItemsActivity"
    android:parentActivityName=".MainActivity"></activity>
```

Shoppinglist application

Now modify the code so that the texts of the buttons on the second Activity are added to the list in the first Activity.

To do this first we have to mark somehow that we start the new Activiti in order to get some result.

```
public void addButtonClicked(View view) {
    Intent intent = new Intent(this, ItemsActivity.class);
    startActivityForResult(intent, 1);
}
```

Request Code: This helps us to distinguish between cases. So it tells us for what question we get the given result.

Shoppinglist application

Now modify the code so that the texts of the buttons on the second Activity are added to the list in the first Activity.

To do this first we have to mark somehow that we start the new Activiti in order to get some result.

```
public void addButtonClicked(View view) {
    Intent intent = new Intent(this, ItemsActivity.class);
    startActivityForResult(intent, 1);
}
```

Request Code: This helps us to distinguish between cases. So it tells us for what question we get the given result.

The we have to get back an Intent from the started Activity.

```
public void addItem(View view) {
    Intent replyIntent = new Intent();
    setResult(RESULT_OK, replyIntent);
    finish();
}
```

Shoppinglist application

Data can be sent in intents using name-value pairs in “Extra” attributes of the Intent.

E.g. in the code beow we set the text of the button to the name “ITEM”.

```java
public void addItem(View view) {
    Button btn = (Button)view;
    Intent replyIntent = new Intent();
    replyIntent.putExtra("ITEM", btn.getText().toString());
    setResult(RESULT_OK, replyIntent);
    finish();
}
```

Shoppinglist application

To get the data we need to Override the onActivityResult method of the caller Activity:

```java
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode==1){
        if (resultCode==RESULT_OK){
            if (listIsEmpty) {listTextView.setText("");listIsEmpty=false;}
            listTextView.append("\n" +data.getStringExtra("ITEM"));
        }
    }
}
```

Shoppinglist application

To get the data we need to Override the onActivityResult method of the caller Activity:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode==1){
        if (resultCode==RESULT_OK){
            if (listIsEmpty) {listTextView.setText("");listIsEmpty=false;}
            listTextView.append("\n" +data.getStringExtra("ITEM"));
        }
    }
}
```

```
Note the previous use of
startActivityForResult(intent, 1);
```

Shoppinglist application

To get the data we need to Override the onActivityResult method of the caller Activity:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode==1){
        if (resultCode==RESULT_OK){
            if (listIsEmpty) {listTextView.setText("");listIsEmpty=false;}
            listTextView.append("\n" +data.getStringExtra("ITEM"));
        }
    }
}
```

```
Vesd össze:
startActivityForResult(intent, 1);
```

```
Note the previous use of:
replyIntent.putExtra("ITEM", btn.getText().toString());
```

Shoppinglist application

The application is done. However we could have been a bit more professional.

- Use static constants for the names in the name-value pairs of the Intent sso that it cannot be mistyped
- Use string resources instead of String literals (ALT+ENTER → Extract String resource)
- Let the user search for an item in google

```
if (requestCode==2){
    if (resultCode==RESULT_OK){
        Intent intent = new Intent(Intent.ACTION_VIEW,
          Uri.parse("https://www.google.com/search?q=" + data.getStringExtra("ITEM")));
        if (intent.resolveActivity(getPackageManager()) != null) {
            startActivity(intent);
        }
    }
}
```

- Extend the application so that it can handle item numbers.
- Add item groups to the application.