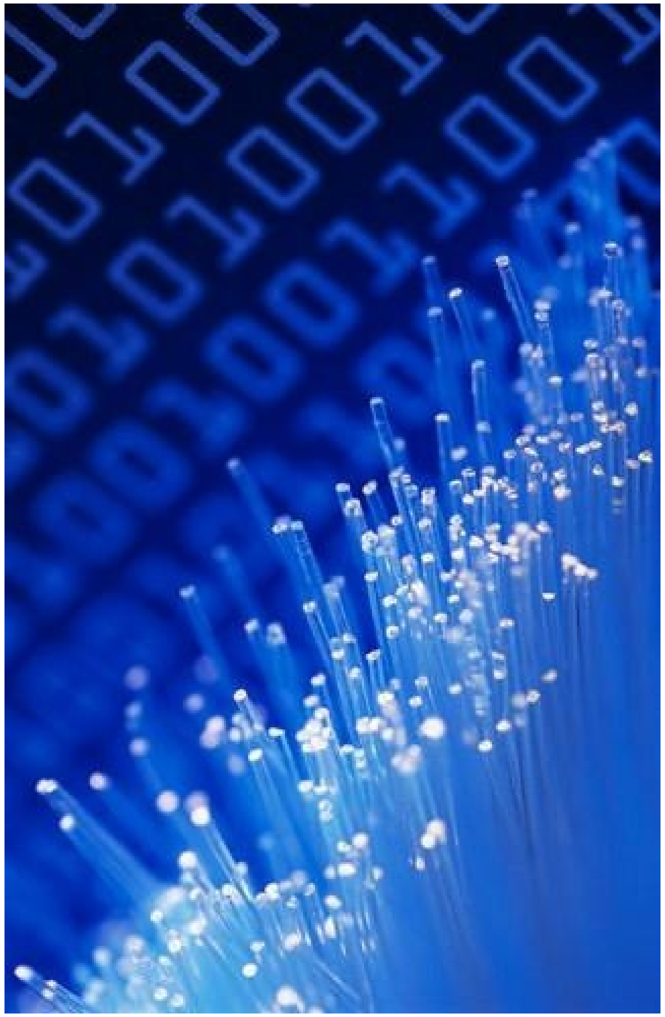
Intent

Communication between components.

Preview

This is the content we'll see

- Presentation
- Launch an Activity
- Include Extra Data
- Implicit Intents
- Native Actions



Presentation

- An intent is an abstract description of an operation to be performed
- We can use it to :
  - Launch an Activity
  - Communicate with components like :
    - Background Services
    - Broadcast Receivers

- The first one is the most common usage

Launch an Activity

- To simply launch an activity :

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

  One of the intent constructors takes only these two parameters
  - The context of the intent, here the activity instance creating it
  - The component class used for the intent

- **startActivity(Intent) :**
  - An instance method of Activity class to start a new activity with an intent

Note

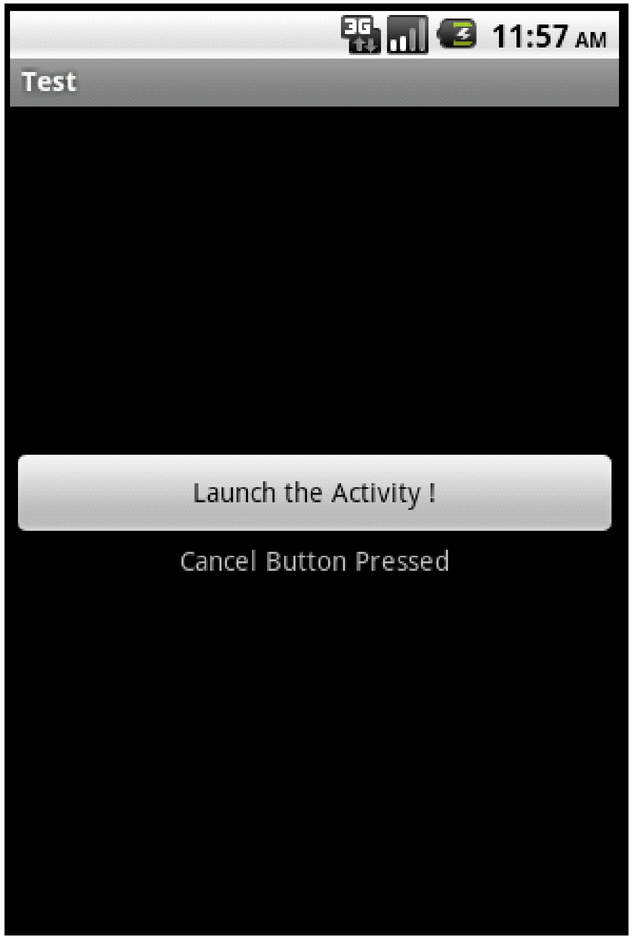
Remember :

An Activity has to be declared inside Android Manifest file to be launched.

Intent

Launch an Activity

- You can also start an Activity and wait for a result code



Cancel Button Pressed

Launch an Activity

- To do that, just use the method **startActivityForResult(…)** instead of **startActivity(…)** :

```
...

private static final int MY_ACTIVITY_CODE = 1;

...

Intent intent = new Intent(this, ActivityToLaunch.class);

// MY_ACTIVITY_CODE constant is the request id
// that will be used later to identify the activity
// returning the result.
startActivityForResult(intent, MY_ACTIVITY_CODE);
```

Launch an Activity

- In the launched activity, use the **setResult(...)** method to return a result code to the launching activity :

```
Button submitButton = (Button) findViewById(R.id.submit);
submitButton.setOnClickListener(new View.OnClickListener() {

    public void onClick(View view) {
        setResult(RESULT_OK);
        finish();
    }

});
```

Launch an Activity

- In the launching activity, override the **onActivityResult(…)** method :

```
protected void onActivityResult(int requestCode,
                                int resultCode, Intent data) {
    switch (requestCode) {
    case MY_ACTIVITY_CODE:
        TextView textView = ...

        switch (resultCode) {
        case RESULT_CANCELED :
            textView.setText("Cancel Button Pressed");
            break;
        case RESULT_OK :
            textView.setText("Submit Button Pressed");
            break;
        }
    ...
    }
}
```

Include Extra Data

- When you launch another activity, you often need to communicate some information
- You can use the intent methods below :
  - **void putExtra(…)**
  - **Bundle getExtras(…)**
- Supported types are :
  - Primitives : byte, short, int, long, float, double, …
  - Primitive Arrays : int[], long[], …
  - Strings
  - Serializable objects

Include Extra Data

- To put an extra data :

```
Intent intent = new Intent(this, MyActivity.class);
intent.putExtra("smthg", "Hi Activity.");
startActivity(intent, MY_ACTIVITY_CODE);
```

- Intent getIntent() :

```
Bundle extras = getIntent().getExtras();

if(extras != null) {
    String message = extras.getString("smthg");
}
```

Intent

Include Extra Data

- If an Activity has been launched by **startActivityForResult(…)** method :
  - It can send information to the launching Activity
    - By sending Extras through the intent in addition to the result code

- You can retrieve it in the launching Activity in the **onActivityResult(…)** method

Include Extra Data

- Launched Activity :

```
...
setResult(RESULT_OK);
getIntent()
    .putExtra(“message”, “Thank you for calling me”);
finish();
...
```

```
protected void onActivityResult(int requestCode,
                                int resultCode, Intent data)
{
    ...

    Bundle extras = data.getExtras();
    if(extras != null) {
        String message = extras.getString(”message");
    }
}
```

Implicit Intents

- Two primary forms of intents :
  - **Explicit Intents** :
    - Provide the exact class to run
  - **Implicit Intents** :
    - Component to run determined by the system

- We just saw the first one
- We're going to see the second one

Implicit Intents

- Implicit Intents are based on Actions
- Android provide many native Actions
  - But you can create your own.

- You have mainly two constructors to create an implicit Intent :
  - **Intent (String action)**
  - **Intent (String action, Uri uri)**

Intent

Native Actions

Action	Definition
ACTION_ANSWER	Handle an incoming phone call.
ACTION_CALL	Perform a call to someone specified by the data.
ACTION_DELETE	Start an Activity to delete the given data from its container.
ACTION_DIAL	Shows an UI with the number being dialed, allowing the user to explicitly initiate the call.
ACTION_EDIT	Provide explicit editable access to the given data.
ACTION_SEARCH	Perform a search.
ACTION_SEND	Deliver some data to someone else by SMS or e-mail.
ACTION_SENDTO	Send a message to someone specified by the data.
ACTION_VIEW	Starting the default activity associated with the data to view it.
ACTION_WEB_SEARCH	Perform a web search.

Native Actions

- Example :
  - Launch the Android Market :

```
Uri uri = Uri.parse("market://search?"
          + "q=pname:com.google.android.stardroid");
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);
```

```
Uri uri = Uri.parse("http://www.android.com");
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);
```

Native Actions

- Example :
  - Call a number :

```
Uri uri = Uri.parse("tel:0607080910");
Intent intent = new Intent(Intent.ACTION_CALL, uri);
startActivity(intent);
```

    - You have to specify that the application has the permission to call.
    - Just add a **<use-permission>** element in your Android Manifest.

```
<uses-permission
   android:name="android.permission.CALL_PHONE" />
```