Android Development

Fisher Yu
Outline

• Intent
• Server
Intent
Intent

• Intent messaging is a facility for late run-time binding between components in the same or different applications.

• Three things come first
  • Context
  • Parcel
  • Parcelable
  • Bundle
Context

- Definition:
  - `public abstract class Context extends Object`

- Importance
  - Interface to global information about an application environment.
  - The famous `Activity` is its indirect class
  - Allow access to application-specific resources and classes, up-calls for application-level operations (such as launching activities, broadcasting and receiving intents, etc).
Parcel

• Container for a message (data and object reference) that can be sent.

• It is mainly used by the system

• You can use it in some sophisticated applications.

• The bulk of the Parcel API revolves around reading and writing data of various types.
Parcel

• There are six major classes of such functions available.
  • Primitives (byte, int, double, etc.)
  • Primitive Arrays
  • Parcelables
  • Bundles
  • Active Objects
  • Untyped Containers
Parcelable

• Interface for classes whose instances can be written to and restored from a Parcel.

• Classes implementing the Parcelable interface must also have a static field called CREATOR, which is an object implementing the Parcelable.Creator interface.
public class MyParcelable implements Parcelable {
    private int mData;

    public int describeContents() {
        return 0;
    }

    public void writeToParcel(Parcel out, int flags) {
        out.writeInt(mData);
    }

    public static final Parcelable.Creator<MyParcelable> CREATOR =
        new Parcelable.Creator<MyParcelable>() {
            public MyParcelable createFromParcel(Parcel in) {
                return new MyParcelable(in);
            }
            public MyParcelable[] newArray(int size) {
                return new MyParcelable[size];
            }
        };

    private MyParcelable(Parcel in) {
        mData = in.readInt();
    }
}
Bundle

• A mapping from String values to various Parcelable types.

• Use a series of put* and get* API to set the content.
Intent Object  In Depth

• An Intent object is a bundle of information
• Two kinds of information
  • For component: Action & Data
  • For Android system: Component name & Category
• Intent filter is used to resolve the intent matching
Different Activities have different functions

We use intent to connect activities within our application

Switch between applications doesn’t hurt performance a lot. But choose what to pass wisely.
Only three aspects of an Intent object are consulted when the object is tested against an intent filter:

- action
- data (both URI and data type)
- category

However, if you only want to create a standalone application, you don’t have to deal with the messy aspect.
Intent Object

- Create An Intent

- Most of the time, We want to connect to another component (activity) of the same application

- Information about the context (where to find) and the component name (who to find) is enough
Intent Object

In Practice

• NotePad example (Send intent)

```
// NoteEdit is the target of this intent
Intent i = new Intent(this, NoteEdit.class);
// Extra information are inserted as pairs in extra
i.putExtra(NotesDbAdapter.KEY_ROWID, id);
i.putExtra(NotesDbAdapter.KEY_TITLE, c.getString(
    c.getColumnIndexOrThrow(NotesDbAdapter.KEY_TITLE)));   
i.putExtra(NotesDbAdapter.KEY_BODY, c.getString(    
    c.getColumnIndexOrThrow(NotesDbAdapter.KEY_BODY)));
startActivityForResult(i, ACTIVITY_EDIT);
```
• **NotePad example (Send intent back)**

```java
// When NoteEdit finishes, it pass back the results
// using a bundle to contain the information
// The bundle could be put as extra in intent directly
Bundle bundle = new Bundle();

bundle.putString(NotesDbAdapter.KEY_TITLE, mTitleText.getText().toString());
bundle.putString(NotesDbAdapter.KEY_BODY, mBodyText.getText().toString());
if (mRowId != null) {
    bundle.putLong(NotesDbAdapter.KEY_ROWID, mRowId);
}

Intent mIntent = new Intent();
mIntent.putExtras(bundle);
// When the intent is ready, set the result and
// call finish() to finish the current activity
setResult(RESULT_OK, mIntent);
finish();
```
Intent Object In Practice

- NotePad example (Receive result)

```java
@Override
// When the result is send back, onActivityResult is called
// by the Android system
protected void onActivityResult(int requestCode, int resultCode,
Intent intent) {
    super.onActivityResult(requestCode, resultCode, intent);
    // Get the data out of the intent
    Bundle extras = intent.getExtras();

    ...
}
```
• Camera Example (Launch Camera)

Intent intent = new Intent("android.media.action.IMAGE_CAPTURE");
//mImageUri contains the image uri, which identifies the image
intent.putExtra(MediaStore.EXTRA_OUTPUT, mImageUri);
//DEFAULT_CAMERA is a constant defined by the current class
//It can be used when the activity returns the result.
startActivityForResult(intent, DEFAULT_CAMERA);
Intent Object  In Practice

• Camera Example (Receive Result)

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    ...
}
Demo
Problems

• It is about the lifecycle management.
• The next reading would be here
Server

- Computer Vision is hard
- More computation is needed
- The smartphone client need to interact with a remote powerful server
- Many technology options
  - Servlet, CGI, etc.
Servlet

• Easy to use
• More efficient than script language
• The client also uses Java
  • Easy to send data between Server and Client
HelloServlet

- Create a servlet project in Eclipse
- Switch on the server (Tomcat)
- Deal with the request and response properly
- Access by URL
Demo
Some Others