Intents

Introduction

- Intent means as it sounds. An intent represents an intention to do something.

  An intent can represent an intention to invoke another application or a component An intent can represent an event you want others to respond.

  An intent can represent an action you ask the android platform to invoke.

- We can consider an intent as a message that specifies an action you want the android to invoke.

- Using an Intent object we can easily start any other specific activity, service or a broadcast receiver.

Introduction

- When we register an activity in our manifest file we can set an intent filter that specifies it as the application entry point.

```
<activity android:name=".PuneHelloActivity"
    android:label="@string/app_name">
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
```

This sample intent filter is the intent that will cause the registered activity to start when trying to execute the application by tapping its icon on the android desktop.

We use this intent filter to mark the entry point of our application.

Using Intents to Start Activities

- The following code shows how to use intent in order to start running another activity.

```
...

Intent intent =
    new Intent(this,com.abelski.intent.action.ShowBasicView.class);

startActivity(intent);

...
```

Late Run-Time Binding

- The intent messaging facility allows late run-time binding between components in the same or in different applications (processes).
- The intent object itself holds an abstract description of the operation we want to be called (in the case of calling another activity) or a description of something we want to announce through a broadcast.

Activities

- There are different separated mechanisms that allow us to deliver an intent.
- When using intent in order to start another activity we pass it over to the `startActivity` or the `startActivityForResult` method. Both were defined in Context.

Activities

- When using intent in order to return a reply back to the previous activity we should pass over the intent to the `setResult` method we invoke on the activity from which the reply is returned.

Services

- We can start another service by passing over an intent object to the `startService()` method that was defined in `Context`. We can use the intent in order to initiate the other service or in order to deliver new instructions to an ongoing one.

Services

- We can bind between our calling component (e.g. activity) and the target service by calling the `bindService()` method. If the target service is still not running our call will start it.

Broadcast Receivers

- Passing over an intent object to a broadcast method, such as the `sendBroadcast()` method that was defined in `Context`, will deliver that intent to all of the interested broadcast receivers.

The Intent Object Structure

- Each intent holds a bundle of information both for the component that receives the intent (e.g. info about the action that should be taken) and for the android system (e.g. the category of the component the intent is been sent to).
- Each intent includes the following information: Component Name, Action, Data, Category, Extras and Flags.

The Intent Component Name

- Each intent object includes the name of the component that should handle the intent. This field is of the `ComponentName` type.
- The `ComponentName` type is holds the fully qualified class name of the target component, such as `com.abelski.HelloDemo`.

The Intent Component Name

- The component name is optional. When the component name is set the Intent object is delivered to instance of the specified class. When it is not set, the android platform locates the suitable target based on the other information the intent object includes.

The Intent Action

- The action is a string that simply names the action we want to be performed (when calling an activity) or the action that took place (when reporting broadcast receivers).
- The `Intent` class already includes a list of predefined action constants.
- It is possible to define our own action strings for specific components in our application.

The Intent Data

- The data is the URI address and its mime types. Different actions are paired with different kinds of data.
- Different actions are paired with different types of data (e.g. The ACTION_CALL action is paired with a URI that starts with 'tel'... The ACTION_VIEW action might be paired with a URI that with 'http').
- The Intent class provides methods that allow us setting both the URI and the MIME TYPE.

The Intent Category

- The category is a string that contains additional information about the component that should handle the intent. Each intent object can includes more than one category. The Intent class already includes several predefined categories.

The Intent Extras

- The Extras are key-value pairs for additional information that should be delivered to the component handling the intent.
- The Intent class defines a series of `put...()` methods for inserting various types of extra data and a similar set of `get...()` methods for reading the data. These methods parallel those for Bundle objects. In fact, the extras can be installed and read as a Bundle using the `putExtras()` and `getExtras()` methods.

The Intent Flags

- Each intent object can include various flags that configure the way it works.

Intents Resolution

- The intents can be either explicit or implicit. An explicit intent targets a specific component in accordance with its name. The component name value is set. An implicit intent doesn't name a specific target. The component name field is empty.
- Explicit intents are usually been used to activate components that belong to the same application.
- Implicit intents are usually been used to activate components that belong to other applications.

Intents Resolution

- When dealing with implicit intents the android platform needs to find the best component (or components) to handle the intent. It can be a single activity or service to perform the requested action or a set of broadcast receivers to respond to the broadcast information.
- The android platform achieves that by comparing the content of the intent object to the available intent filters.

Intents Resolution

- The intent filters are data structures associated with components that can potentially receive intents.
- The intent filters advertise the capabilities of their component and delimit the intents it can handle.
- The intent filters open the component to the possibility of receiving implicit intents of the advertised type.

Intents Resolution

- If a component does not have any intent filters, it can receive only explicit intents.
- If a component does come with intent filters then it can receive both explicit and implicit intents.
- In the course of finding the most suitable component only three aspects of each and every intent object are checked (action, data and category). The extras and flags play no part in resolving which component will receive the intent.

Intent Filters

- The purpose of the intent filter is to inform the system about the specific implicit intents the component can handle. Each component (e.g. activity) can have one or more intent filters.
- Each intent filter actually defines a set of implicit intents the component is willing to receive. The intent filter actually filters in implicit intents of a desired type and filter out the unwanted ones.

Intent Filters

- Each component can have multiple intents filters. Each intent filter can be for a different job.
- For each intent filter we define in our manifest XML document an instance of the `IntentFilter` class is created.

Intent Filters

- Each intent filter has fields that match the action, the data and the category an Intent object has.
- An implicit intent is tested against the intent filter in all three areas: the action, the data and the category.
- In order to have the implicit intent delivered to the intent filter the implicit intent must pass all three tests. It must match all three areas. The action, the data and the category.

Intent Filters

```
<?xml version="1.0" encoding="utf-8"?>

<manifest   xmlns:android="http://schemas.android.com/apk/res/android"
            package="com.abelski.android"
            android:versionCode="1"
            android:versionName="1.0">

    <application    android:icon="@drawable/icon"
                    android:label="@string/app_name">

        <activity   android:name=".MemoActivity"
                    android:label="@string/app_name">
```

Intent Filter

```
            <intent-filter>
            <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

        </activity>
    </application>

    <uses-sdk android:minSdkVersion="6" />

</manifest>
```

The Action Test

- The actions are defined as sub elements of the intent filter element.
- In order to pass the action test the action must match at least one of the actions the intent filter lists.
- If the intent doesn't specify a specific action it automatically passes the test as long as the intent filter specifies one action at the minimum.

The Action Test

- While an Intent object can name a single action the filter can list more than one.
- If the intent filter doesn't list any action it will block all intents.

The Category Test

- Each Intent object can specify one or more categories. In order to pass the category test each category the intent object includes must have a matched category defined by the intent filter. Both the intent and the intent filter can list more than one category.
- In order to pass the category test the intent filter can list additional categories but it cannot omit the ones the intent object includes.

The Category Test

- The android platform treats all implicit intents passed over to the `startActivity()` method as if they contain at least one category, the '`android.intent.category.DEFAULT`' category.
- Therefore, when defining activities we want to be capable of receiving implicit intents their intent filters must include this category.

The Data Test

- The intent filter XML element can include the data XML child element in order to describe the data it refers.
- The data XML element can include two attributes. The android:mimeType and the android:schema.

```
<intent-filter . . . >
    <data android:mimeType="video/mpeg" android:scheme="http://..." />
    <data android:mimeType="audio/mpeg" android:scheme="http://..." />
    . . .
</intent-filter>
```

The Data Test

- The android:schema specifies a URI. It should be of the following format:

  ```
  scheme://host:port/path
  ```

  The following is a possible example for a URI we can specify:

  ```
  content://samples.abelski.com:1120/folder/subfolder/
  ```

The Data Test

- The android:schema attribute specifies a URI address. This URI address can include wild cards in order to require partial match only.

```
<intent-filter . . . >
    <data android:mimeType="video/mpeg" android:scheme="http://*.3gp" />
    . . .
</intent-filter>
```

The Data Test

- The android:mimeType attribute specifies a MIME type. Using a wild card for specifying the subtype field is feasible.

```
<intent-filter . . . >
    <data android:mimeType="video/*" android:scheme="http://*" />
    . . .
</intent-filter>
```

The Data Test

- In order to pass the data test both the URI and the MIME type are compared.
- The URI the intent includes is compared with the URI included by the intent filter. The comparison includes only those elements mentioned by the intent filter.

  If, for instance, the intent filter includes a URI that includes a schema only then the schema would be the only one that is been compared.
- The MIME type the intent includes is compared with the one specified by the intent filter.

The Data Test

- If an implicit intent doesn't contain neither a URI or a data type then it passes the data test only if the intent filter also doesn't specify any URIs or a data type.

The Data Test

- If an implicit intent contains a URI but doesn't contain any data type and the URI doesn't indicate about the type then the data test will pass only if the URI matches the URI specified by the intent filter and the intent filter doesn't specify any specific type.

The Data Test

- If an intent object does contain a data type but doesn't contain a URI it will pass the test only if the intent filter does indeed list the same data type and doesn't specify any specific URI.

The Data Test

- When the intent object contains both a URI and a data type (or the data type can be inferred from the URI) it will pass the data type part of the test only if its type matches the type listed by the intent filter. It will pass the URI part of the test if the URI matches the URI specified by the filter or if the case is of a 'content://' or a 'file://' URI and the filter doesn't specify any URI.

Multiple Matches

- If the intent matches more than one intent filter the user may be asked which component to activate.
- If the intent doesn't match any of the available intent filters an exception is raised.

Android Predefined Intents

- The android platform already has a predefined set of intents that can be used to invoke various activities on our android.

  Browsing at http://developer.android.com/guide/appendix/g-app-intents.html you can find a list of all available predefined applications (e.g. browser, application that can call a telephone number, application that shows the map of the world, application that shows street views etc...) and the intents that can start each one of them.

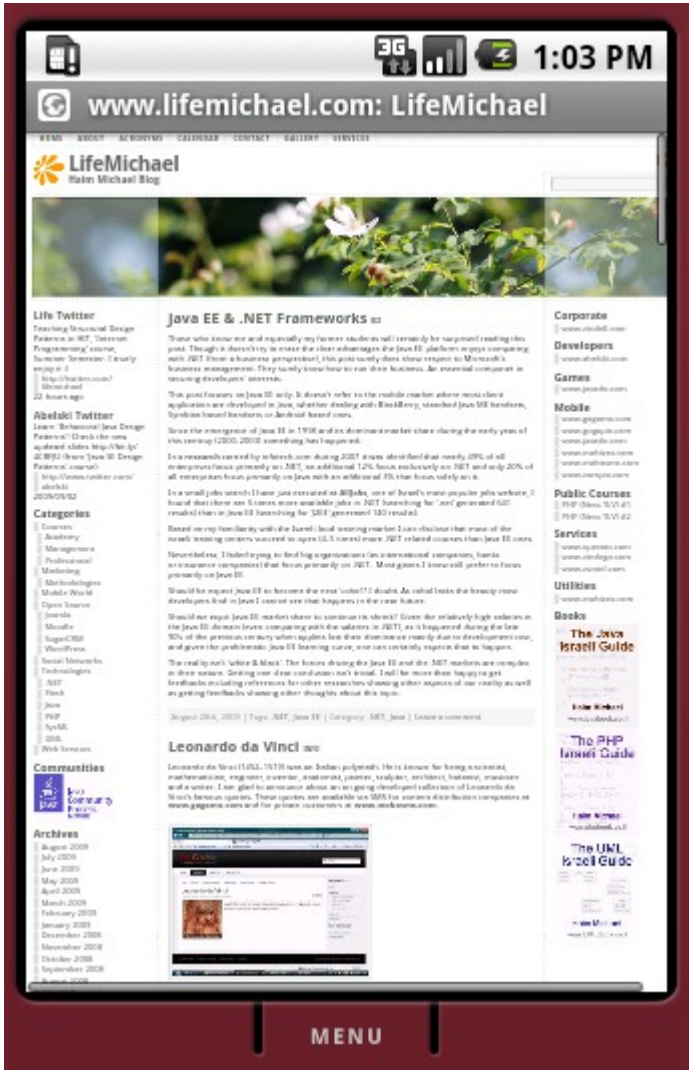
Simple Intent Demo

```
package com.abelski;

import android.app.Activity;
import android.net.Uri;
import android.content.Intent;
import android.os.Bundle;

public class IntentDemoActivity extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        Intent intent = new Intent(Intent.ACTION_VIEW);
        intent.setData(Uri.parse("http://www.lifemichael.com"));
        this.startActivity(intent);
    }
}
```

Simple Intent Demo



The Intent Flags

- Calling the addFlags method on the Intent object we can configure it using various flags.
- Each one of the flags is an integer number that its binary representation includes one bit equals to 1 and all others equal to 0.
- We can take as many flags as we want and create one integer number that represent them all.

The Intent Flags

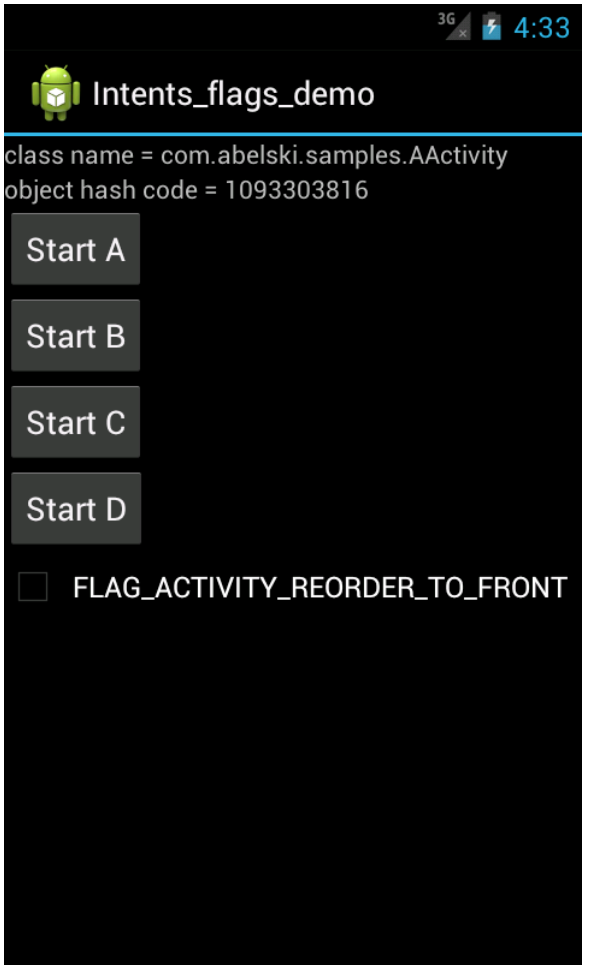
- Each flag has a specific meaning. Combining multiple flags into one new integer number will have the accumulative meaning of all flags.

Sample

- Setting the `FLAG_ACTIVITY_REORDER_TO_FRONT` flag on the Intent object we pass over to the `startActivity` method, if the launched activity is already running it will be brought to the front instead of creating a new object.
- The following code sample includes an application composed of 4 separated activities. Each activity allows the user to start any of the other three activities.

Sample

- Selecting the checkbox will configure the Intent object with this flag.
- The hashcode value indicates about staying with the same object or about the creation of a new one.



Sample

```
public class AActivity extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView text1 = (TextView) findViewById(R.id.textView1);
        TextView text2 = (TextView) findViewById(R.id.textView2);
        text1.setText("class name = " + this.getClass().getName());
        text2.setText("object hash code = " + this.hashCode());
        final Button btA = (Button) findViewById(R.id.button1);
        final Button btB = (Button) findViewById(R.id.button2);
        final Button btC = (Button) findViewById(R.id.button3);
        final Button btD = (Button) findViewById(R.id.button4);
        final CheckBox checkBox = (CheckBox) findViewById(R.id.checkBox1);
```

Sample

```
class ButtonsListener implements View.OnClickListener
{
    Intent intent = null;

    public void onClick(View v)
    {
        if (v == btA)
        {
            intent = new Intent(AActivity.this, AActivity.class);
        }
        else if (v == btB)
        {
            intent = new Intent(AActivity.this, BActivity.class);
        }
        else if (v == btC)
        {
            intent = new Intent(AActivity.this, CActivity.class);
        }
```

Sample

```
                else if (v == btD)
                {
                    intent = new Intent(AActivity.this, DActivity.class);
                }
                if (checkBox.isChecked())
                {
                    intent.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
                    Toast.makeText(AActivity.this, "flag was added",
Toast.LENGTH_LONG).show();
                }
                startActivity(intent);
            }

        }
        View.OnClickListener listener = new ButtonsListener();
        btA.setOnClickListener(listener);
```

Android Tasks

- The applications are usually composed of activities. Each and every activity is designed for a specific action the user perform. Usually, each and every activity has its own separated user interface. Usually, each and every activity is capable of starting other activities.
- The user goes through activities that might belong to different separated applications.

Android Tasks

- Each task is a collection of activities the user goes through when performing a specific operation.
- The activities that form a task are organized in a stack in the same order in which they were opened.
- The home screen activity is the starting point for most tasks. When the user touches an icon of a specific application that application's task comes to the foreground. If there is no task for that specific application then a new task is created.

The Task Root Activity

- When a new task is created then the activity that starts it is considered to be the root activity of that stack.
- When all of the activities that form the task are removed then the task is destroyed.
- Each task is a cohesive unit that can be moved to the background when the user starts a new task or go to the home screen activity.

Managing Tasks

- When calling the `startActivity` function in order to start a new activity we can pass over the `Intent.FLAG_ACTIVITY_NEW_TASK` static variable in order to specify that we want the new activity to start a new task.
- It is also possible to mark those activities we want to start new tasks in the manifest file. We can add the `launchmode` attribute into the `<activity>` element assigned with the "`singleTask`" value.