Instrumentation

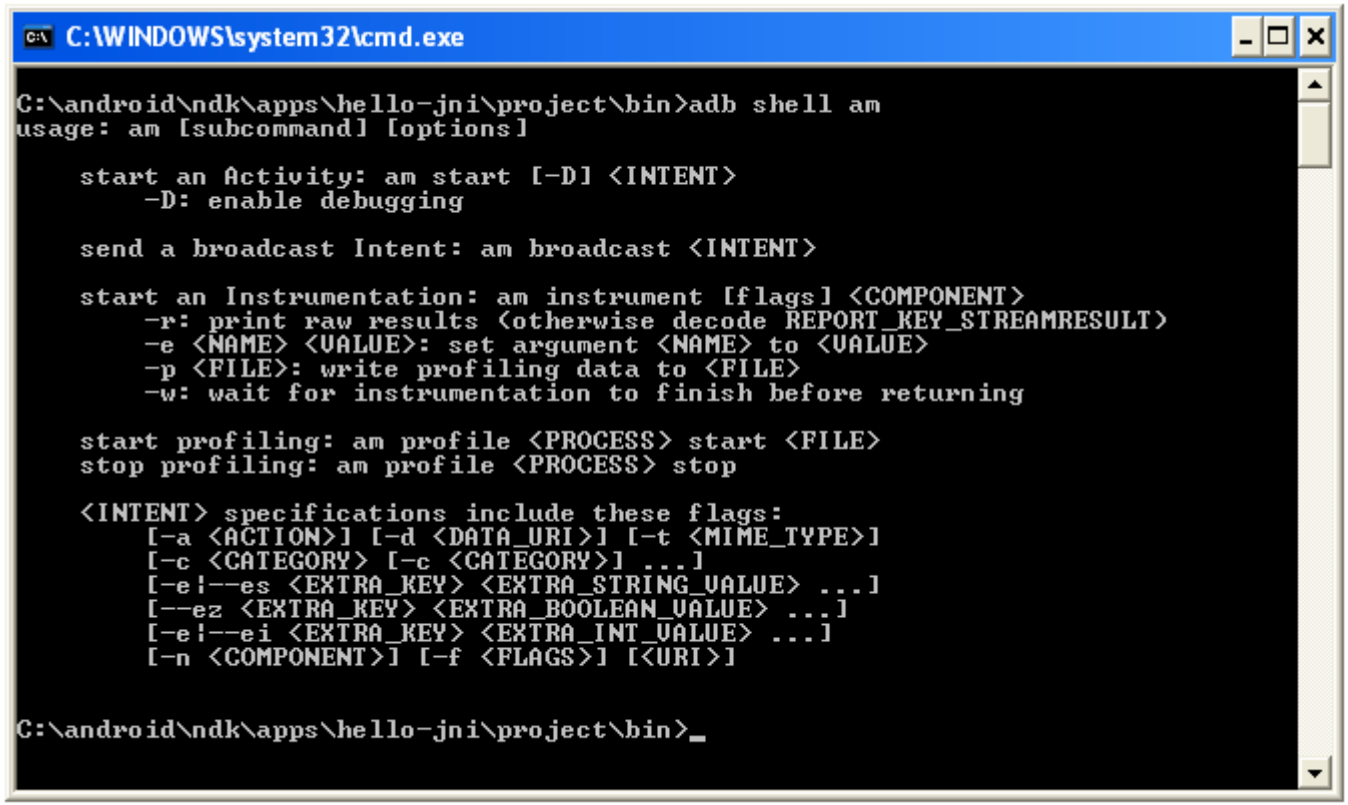
Introduction

- The Instrumentation framework, part of the Android SDK, allows us to automate UI testing by sending events to the application under test, precisely control the start of the activity, and monitor the state of the activity during its life cycle.

The `am` Tool

- This tool allows us to start and instrument activities on the android platform while using the shell command prompt.

The am Tool



The am Tool

this code sample includes a simple activity we install on the emulator and then create an implicit intent object using the instrumentation framework in order to start this activity.

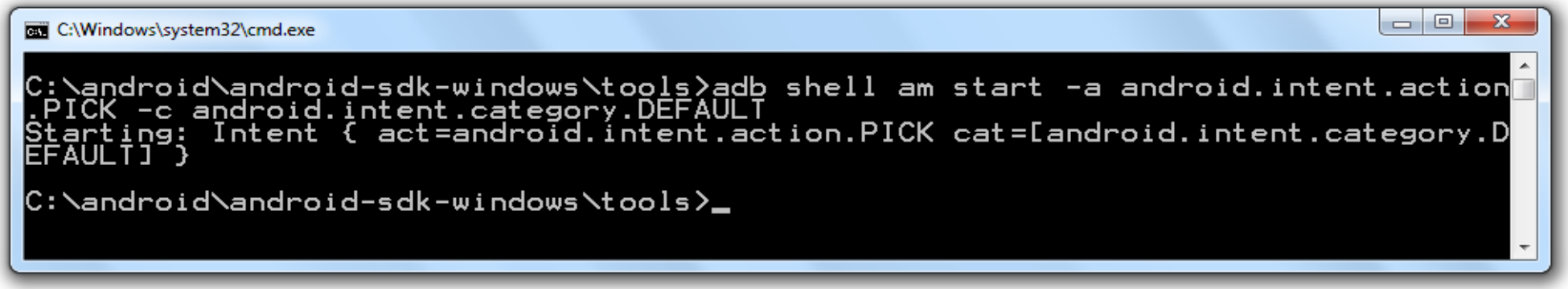
```
package com.abelski.samples;

import android.os.Bundle;
import android.app.Activity;
import android.widget.TextView;

public class HelloActivity extends Activity
{
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("hello canada");
        setContentView(tv);
    }
}
```

The am Tool

```
<?xml version="1.0" encoding="utf-8"?>
<manifest   xmlns:android="http://schemas.android.com/apk/res/android"
            package="com.abelski.samples"
            android:versionCode="1"
            android:versionName="1.0">

  <application android:icon="@drawable/icon"android:label="@string/app_name">
    <activity android:name=".HelloActivity" android:label="@string/app_name">
        <intent-filter>
            <action android:name="android.intent.action.PICK" />
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
    </activity>
  </application>

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

</manifest>
```

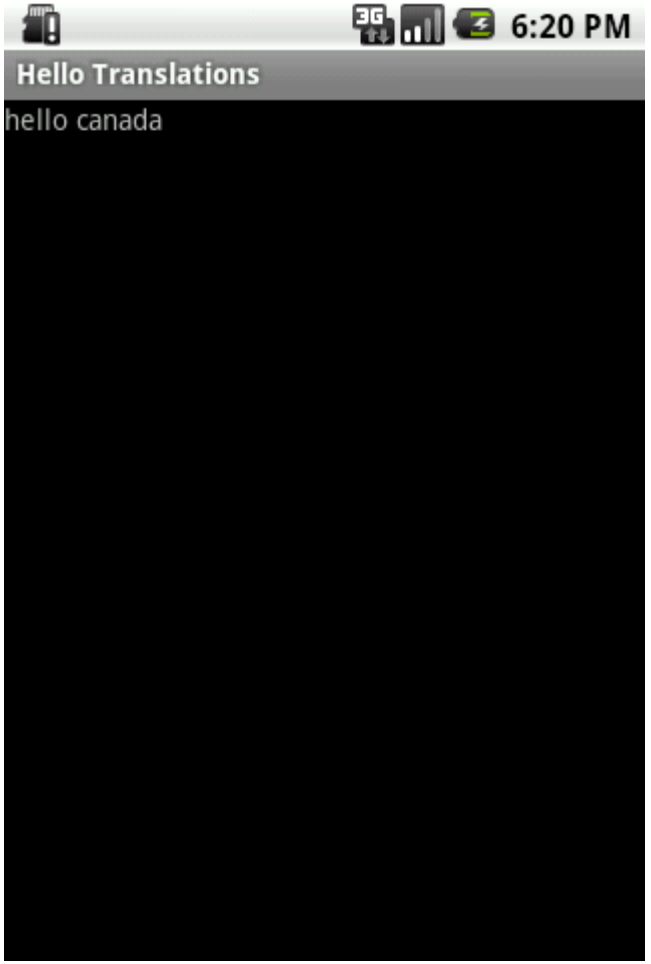
this intent filter is associated with the definition of the activity we define. creating an implicit intent that fits it and pass it over to the android platform will indirectly start our activity.

The am Tool



using the am tool we can indirectly start the activity we developed... running this command from the command line will create an implicit intent and then use it to indirectly create the activity we developed... the intent we created should fit the intent filter that was defined in the manifest file.

The `am` Tool



this is the activity we developed

Unit Tests

- The instrumentation framework is especially useful when creating unit tests.
- The eclipse development environment allows us to create unit tests that utilize the instrumentation framework capabilities.

Instrumentation

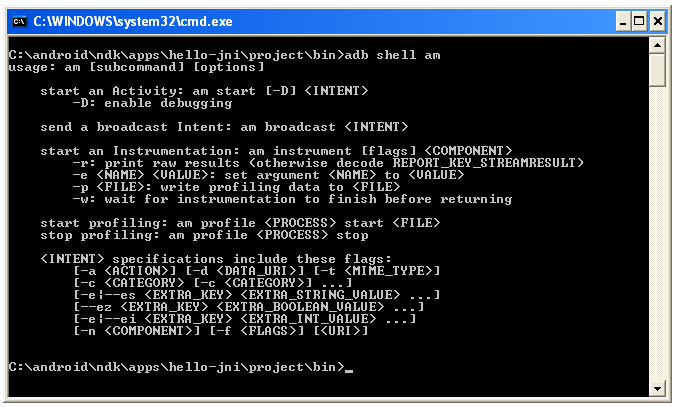
Introduction

- The Instrumentation framework, part of the Android SDK, allows us to automate UI testing by sending events to the application under test, precisely control the start of the activity, and monitor the state of the activity during its life cycle.

The `am` Tool

- This tool allows us to start and instrument activities on the android platform while using the shell command prompt.

The `am` Tool


C:\WINDOWS\system32\cmd.exe
C:\android\ndk\apps\hello-jni\project\bin>adb shell am
usage: am [subcommand] [options]
start an Activity: am start [-D] <INTENT>
-D: enable debugging
send a broadcast Intent: am broadcast <INTENT>
start an Instrumentation: am instrument [flags] <COMPONENT>
-r: print raw results (otherwise decode REPORT_KEY_STREAMRESULT)
-e <NAME> <VALUE>: set argument <NAME> to <VALUE>
-p <FILE>: write profiling data to <FILE>
-w: wait for instrumentation to finish before returning
start profiling: am profile <PROCESS> start <FILE>
stop profiling: am profile <PROCESS> stop
<INTENT> specifications include these flags:
[-a <ACTION>] [-d <DATA_URI>] [-t <MIME_TYPE>]
[-c <CATEGORY> [-c <CATEGORY>] ...]
[-e|--es <EXTRA_KEY> <EXTRA_STRING_VALUE> ...]
[--ez <EXTRA_KEY> <EXTRA_BOOLEAN_VALUE> ...]
[-e|--ei <EXTRA_KEY> <EXTRA_INT_VALUE> ...]
[-n <COMPONENT>] [-f <FLAGS>] [<URI>]
C:\android\ndk\apps\hello-jni\project\bin>

The `am` Tool

this code sample includes a simple activity we install on the emulator and then create an implicit intent object using the instrumentation framework in order to start this activity.

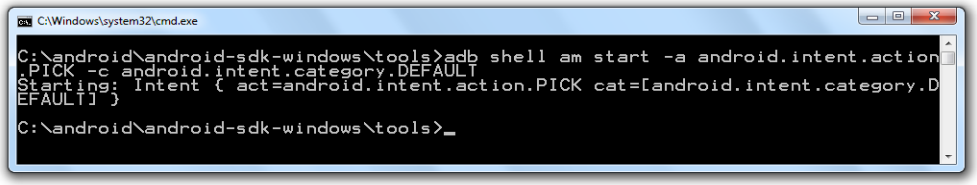
```
package com.abelski.samples;

import android.os.Bundle;
import android.app.Activity;
import android.widget.TextView;

public class HelloActivity extends Activity
{
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("hello canada");
        setContentView(tv);
    }
}
```

The `am` Tool

```
<?xml version="1.0" encoding="utf-8"?>
<manifest   xmlns:android="http://schemas.android.com/apk/res/android"
            package="com.abelski.samples"
            android:versionCode="1"
            android:versionName="1.0">

  <application android:icon="@drawable/icon"android:label="@string/app_name">
    <activity android:name=".HelloActivity" android:label="@string/app_name">
        <intent-filter>
            <action android:name="android.intent.action.PICK" />
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
    </activity>
  </application>

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

</manifest>
```

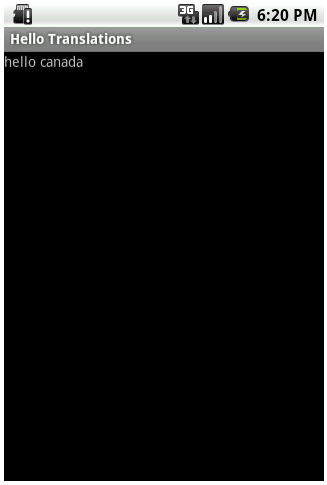
this intent filter is associated with the definition of the activity we define. creating an implicit intent that fits it and pass it over to the android platform will indirectly start our activity.

The `am` Tool



using the am tool we can indirectly start the activity we developed... running this command from the command line will create an implicit intent and then use it to indirectly create the activity we developed... the intent we created should fit the intent filter that was defined in the manifest file.

The `am` Tool



this is the activity we developed

Unit Tests

- The instrumentation framework is especially useful when creating unit tests.
- The eclipse development environment allows us to create unit tests that utilize the instrumentation framework capabilities.