Accelerating Information Technology Innovation

http://aiti.mit.edu

Cali, Colombia
Summer 2013
Lesson 5 – Application Object and Services
Stuff

• Difference of level
  – Interdisciplinary
  – Show case implement a Stock marcket tracker

• Bussiness helping coders.
Agenda

- Application Object (REFACTORING!!)
- Services
- Java Thread and Comparisons
Application Object

- Object shared along all the application.
- Not to confuse with Shared Preferences
- You can use `android.app.Application` or implement your own.
- Steps (create your own):
  1. Create a Java class representing it `MyApplication.java`
  2. Register the new class in the manifest.
Step 1: MyApplication.java

MyApplication.java

```java
public class MyApplication extends Application{
    @Override
    public void onCreate(){
        super.onCreate();
        ...
    }
}
```

From any other activity, to call it:

```java
MyApplication myApplication = (MyApplication) getApplication();
```
Step 2: update manifest

Automatically done by Android Studio if we select “New Application”

AndroidManifest.xml

```xml
<application>
  ...
  android:name = "MyApplication">
```
Services vs Activities

- Activity has a UI, Services not 😞
- Runs in the background (but in the UI thread)
- Example: a service to periodically connect and check for new statuses.
- Like Activities, they have Lifecycle
Started Service

- Steps:
  1. Create Java class
  2. Register the service in the manifest
  3. Start the service
Step 1: Create the Java

```java
public class UpdaterService extends Service {

    @Override
    public void onCreate() {
        // called when the service created for the first time
        super.onCreate();
    }

    @Override
    public void onDestroy() {
        // called when the service is terminated
        super.onDestroy();
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        // called when the service is started
        return super.onStartCommand(intent, flags, startId);
    }

    @Override
    public IBinder onBind(Intent intent) {
        // just used for bound services
        return null;
    }
}
```
Step 2: Update Manifest

```xml
<manifest ...

  <application 
    
    <service android:name="UpdaterService" />

</manifest>
```
Thread

class MyThread extends Thread{
    public void run(){
        Log.v("thread", "Hilo corriendo!");
    }
}

MyThread myThread = new MyThread();
myThread.start();
Comparison

- **Thread:**
  - Simpler way of creating new thread

- **AsyncTask:**
  - UI support
  - For quicker and one shot tasks
  - E.g.: fetching/processing data when a button pressed

- **Service:**
  - Usually combined with thread
  - For persistence and robust to activity *destroyed*, running in the background
  - E.g.: playing music, fetching data