Emerging Technologies: Mobile Development for Android Devices

Multithreading

Griffith College

Introduction

- The android applications that you have seen thus far you have mainly dealt with single threaded execution.
- However, when you start to build more complex applications you may add so much functionality and delays that your application may suddenly crash for no reason.
- This is because you will have run into a series of restrictions related to responsiveness that are defined by the android system so that it can maintain overall control of the device and maintain user responsiveness.

Rules Imposed by Android

- There are two rules that android uses to determine if an app is unresponsive.
- An application must completely process a user input event in less than 5 seconds.
- A Broadcast Receiver must completely finish processing an intent in less than 10 seconds regardless of source.
- For every input event and every time a broadcast receiver is activated, a timer is started to monitor the response time of that event.
- If it exceeds the limit, then android considers it unresponsive and will kill the application
  immediately.
- An unresponsive app prevents the user from using the device.

Keeping the Application Responsive

- Long running tasks should not be run on the main UI thread.
- Run such tasks on a background thread.
- Leaves the main UI thread responsive to user actions.
- Contributes to a better user experience.

Three Approaches

- Three forms of multithreading are worth exploring:

- Asynchronous tasks.
- Background threads.
- Services

Asynchronous Task

- An asynchronous task provides a simple framework around the use of a single background thread.
- This background thread is expected to have well defined starting and ending points of execution.
- It will create the thread start it executing and upon completion will return a result and kill the thread.
- To use an asynchronous task you are required to extend the AsyncTask class and provide at least one method definition.
- As an asynchronous task denotes a single execution it is generally used for tasks that are expected to run only once and finish.
- For example a file download where a connection is setup and then torn down after the download.
- Or simple request reply behaviour on a short network connection that does not happen very often.

Threads

- A thread is the unit of execution in Android.
- An application is permitted to have as many threads as necessary.
- A general thread should be used for any application processing that can potentially last forever (e.g. game threads).
- Another example of this is if you require an open network connection for the lifetime of your application.
- Because there is no definite end to tasks like these we need a general thread.

Services

- Unlike asynchronous tasks or threads, services can be detached from an application or activity and still run in the background.
- They will usually run in their own separate process.
- Communication will either happen through IPC or RPC.

Services

- Generally a service is used when you need to keep something happening in the background while the application is not running.
- One of the most common examples of this is polling for notifications for a user.
- How the Facebook app (amongst others) are capable of sending you notifications even though the application is not active.
- Or used for monitoring something without the need for a foreground application.
- Examples would include battery monitors.

Priorities

- Generally it is a good idea to give a background service a low process priority.
- As processing capability should be given to the foreground activity.
- By assigning a lower priority to a service it will generally take CPU time when there is no user interaction with the device.
- In this way, it does not unduly impinge on user interactions and helps to maintain a responsive system.