Android: Basics on development

Desenvolvimento de Software e Sistemas Móveis (DSSMV)
Licenciatura em Engenharia de Telecomunicações e Informática
LETI/ISEP

2025/26

Paulo Baltarejo Sousa and Carlos Filipe Freitas
{pbs,caf}@isep.ipp.pt

Disclaimer

Material and Slides

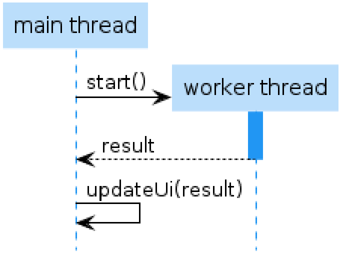
Some of the material/slides are adapted from various:

- Presentations found on the internet;
- Books;
- Web sites;
- ...

Outline

Activities

UI Thread interacting with Worker Threads



- `runOnUIthread`
- `View.post`
- `Handler`

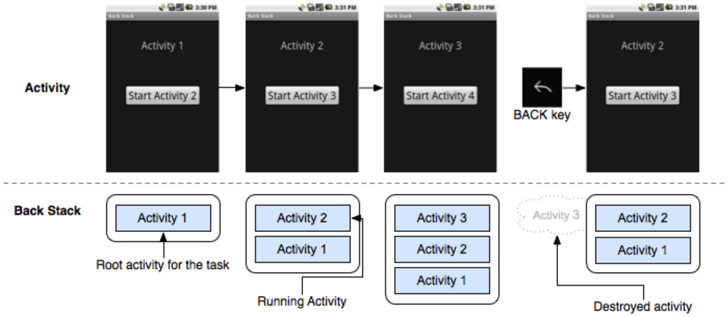
Check: `TP5_05.zip`

Check: `TP5_06.zip`

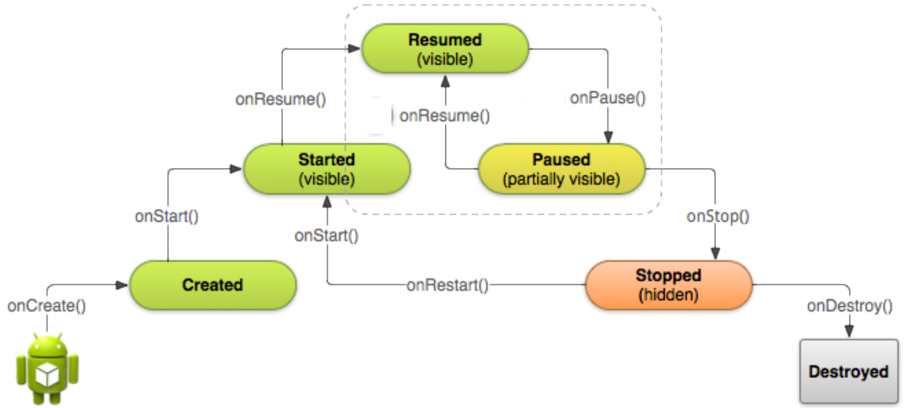
Check: `TP5_07.zip`

Back Stack

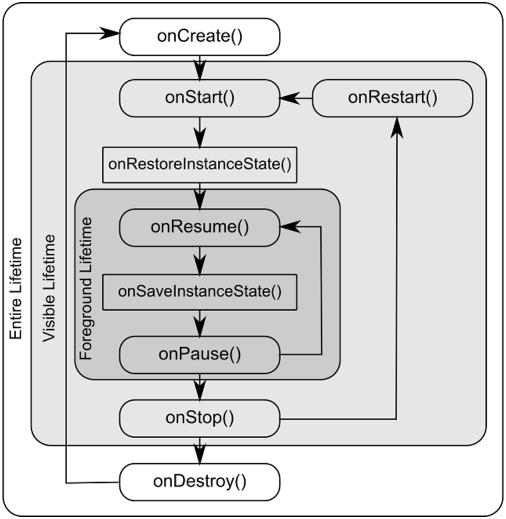
- **Activities are arranged in a stack** (Back Stack), in the order in which each activity is opened.



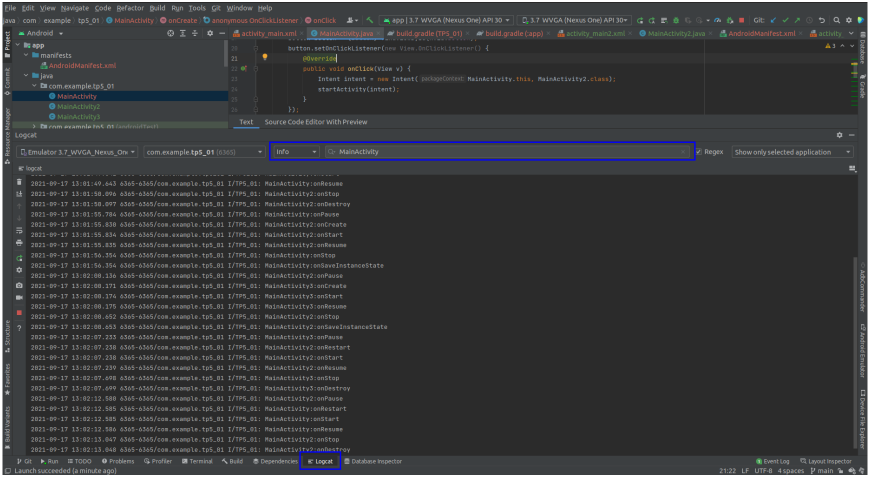
Activity States



Activity Lifecycle (I)



Activity Lifecycle (II)



```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent( packageContext: MainActivity.this, MainActivity2.class);
        startActivity(intent);
    }
});
```

Logcat: Emulator 3.7_WVGA_Nexus_One · com.example.tp5_01 (6365) · Info · MainActivity · Regex · Show only selected application

```
2021-09-17 13:01:49.643 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity:onResume
2021-09-17 13:01:50.096 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity2:onStop
2021-09-17 13:01:50.097 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity2:onDestroy
2021-09-17 13:01:55.784 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity:onPause
2021-09-17 13:01:55.830 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity2:onCreate
2021-09-17 13:01:55.834 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity2:onStart
2021-09-17 13:01:55.835 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity2:onResume
2021-09-17 13:01:56.354 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity:onStop
2021-09-17 13:01:56.354 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity:onSaveInstanceState
2021-09-17 13:02:00.136 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity2:onPause
2021-09-17 13:02:00.171 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity3:onCreate
2021-09-17 13:02:00.174 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity3:onStart
2021-09-17 13:02:00.175 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity3:onResume
2021-09-17 13:02:00.652 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity2:onStop
2021-09-17 13:02:00.653 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity2:onSaveInstanceState
2021-09-17 13:02:07.233 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity3:onPause
2021-09-17 13:02:07.238 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity2:onRestart
2021-09-17 13:02:07.238 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity2:onStart
2021-09-17 13:02:07.239 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity2:onResume
2021-09-17 13:02:07.698 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity3:onStop
2021-09-17 13:02:07.699 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity3:onDestroy
2021-09-17 13:02:12.580 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity2:onPause
2021-09-17 13:02:12.585 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity:onRestart
2021-09-17 13:02:12.585 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity:onStart
2021-09-17 13:02:12.586 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity:onResume
2021-09-17 13:02:13.047 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity2:onStop
2021-09-17 13:02:13.048 6365-6365/com.example.tp5_01 I/TP5_01: MainActivity2:onDestroy
```

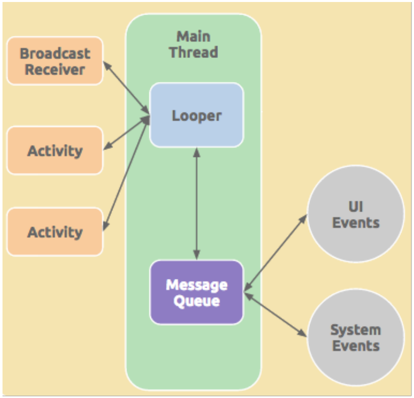
Launch succeeded (a minute ago)

Check: `TP5_01.zip`

UI Events

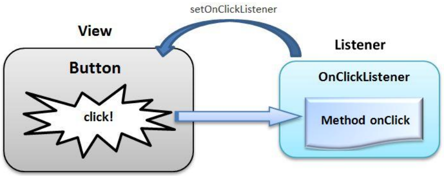
Handling events and messages

- The Android framework maintains an event/message queue into which events are placed as they occur.
- Events/messages are removed from the queue on a first-in, first-out (FIFO) basis.



Event listeners

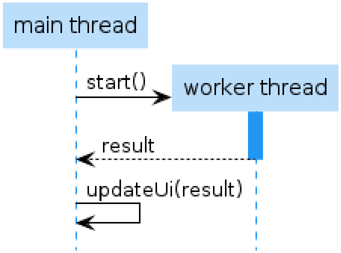
- **Event Listeners**
  - It is an interface in the `View` class that contains a callback method.
  - Callback method is invoked when the `View` is triggered by user interaction.
- **Event Listeners Registration**
  - It is the process by which an Event Handler gets registered with an Event Listener so that the handler is called when the Event Listener fires the event.
- **Event Handlers**
  - It is the callback method that actually handles the event.



Event Listeners (I)

- `onClick()`
  - From `View.OnClickListener`. This is called when the user either touches the item (when in touch mode), or focuses upon the item with the navigation-keys or trackball and presses the suitable "enter" key or presses down on the trackball.
- `onLongClick()`
  - From `View.OnLongClickListener`. This is called when the user either touches and holds the item (when in touch mode), or focuses upon the item with the navigation-keys or trackball and presses and holds the suitable "enter" key or presses and holds down on the trackball (for one second).
- `onFocusChange()`
  - `View.OnFocusChangeListener`. This is called when the user navigates onto or away from the item, using the navigation-keys or trackball.

UI Thread interacting with Worker Threads



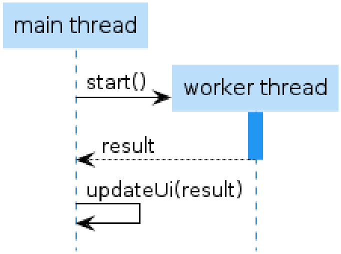
- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

Check: `TP5_06.zip`

Check: `TP5_07.zip`

UI Thread interacting with Worker Threads



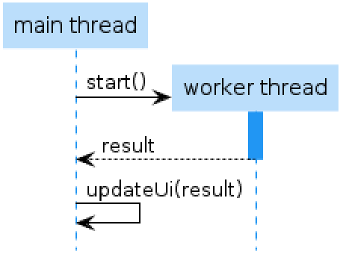
- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

Check: `TP5_06.zip`

Check: `TP5_07.zip`

UI Thread interacting with Worker Threads



- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

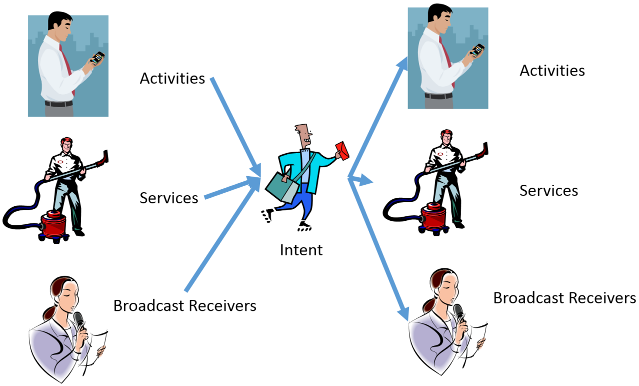
Check: `TP5_06.zip`

Check: `TP5_07.zip`

Intents

What is an Intent

- An Intent is a messaging object you can use to request an action from another app component.
- Intents facilitate communication between components in several ways.



Intents are used for

- To start an activity:
  - The Intent describes the activity to start and carries any necessary data and is passed to `startActivity` method.
  - If you want to receive a result from the activity when it finishes, call `startActivityForResult`. Your activity receives the result as a separate Intent object in your activity's `onActivityResult` callback.
- To start a service:
  - A Service is a component that performs operations in the background without a user interface. You can start a service by passing an Intent to `startService`. The Intent describes the service to start and carries any necessary data.
- To deliver a broadcast:
  - A broadcast is a message that any app can receive. The system delivers various broadcasts for system events, such as when the system boots up or the device starts charging. You can deliver a broadcast to other apps by passing an Intent to `sendBroadcast`.

What is in an Intent

- An Intent object carries information that the Android system uses to determine which component to start, plus information that the recipient component uses in order to properly perform the action.
- The primary information contained in an Intent is the following:
  - **Component name**: The name of the component to start.
  - **Action**: A string that specifies the generic action to perform (such as view or pick).
  - **Data**: The URI (a Uri object) that references the data to be acted on and/or the MIME type of that data.
  - **Category**: A string containing additional information about the kind of component that should handle the intent.
  - **Extras**: Key-value pairs that carry additional information required to accomplish the requested action.
  - **Flags**: Flags defined in the Intent class that function as metadata for the intent.

Intent Types

- **Implicit intents**
  - Do not name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it.
  - For example, if you want to show the user a location on a map, you can use an implicit intent to request that another capable app show a specified location on a map.
- **Explicit intents**
  - Specify the component to start by name (the fully-qualified class name).
  - Used to start a component in your own app, because you know the class name of the activity or service you want to start.

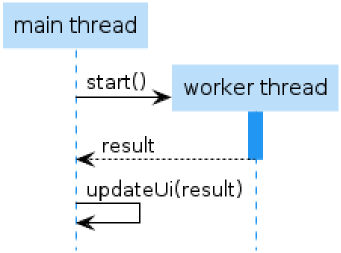
Implicit Intent

- An **implicit intent** specifies an action that can invoke any app on the device able to perform the action.
- Using an implicit intent is useful when your app cannot perform the action, but other apps probably can and you'd like the user to pick which app to use.
    - For example, if you have content you want the user to share with other people, create an intent with the `ACTION_SEND` action and add extras that specify the content to share. When you call `startActivity` with that intent, the user can pick an app through which to share the content.

```
// Create the text message with a string
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");

// Verify that the intent will resolve to an activity
if (sendIntent.resolveActivity(getPackageManager()) != null) {
   startActivity(sendIntent);
}
```

UI Thread interacting with Worker Threads



- `runOnUIthread`
- `View.post`
- `Handler`

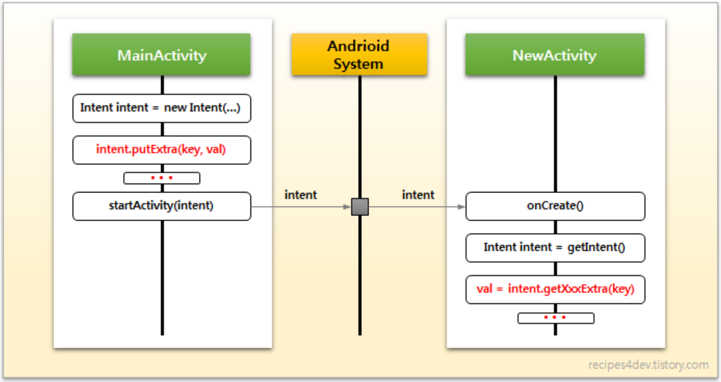
Check: `TP5_05.zip`

Check: `TP5_06.zip`

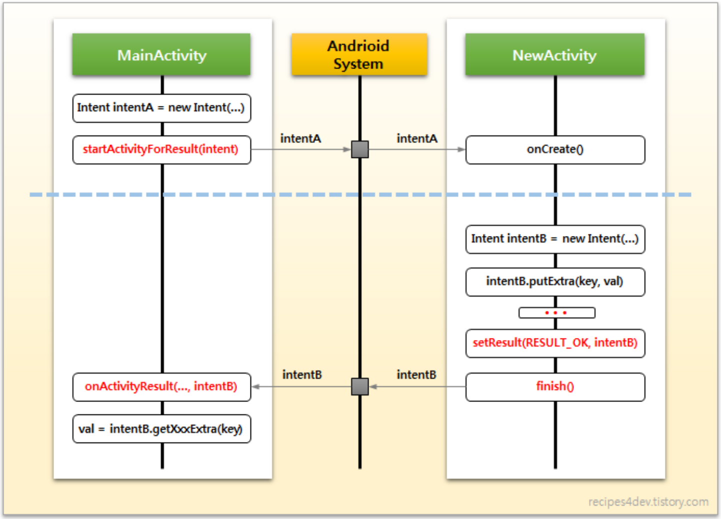
Check: `TP5_07.zip`

Intent: data transfer

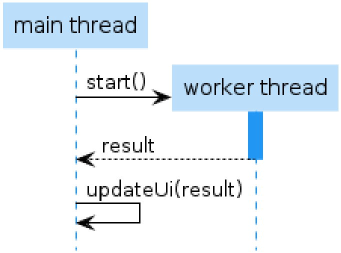
- `putExtra (Key, Value)`
  - Add extended data to the intent.
- `GetXxxxExtra (Key)`
  - Retrieve data from the intent.



Intent: data returning (I)


MainActivity
Andrioid System
NewActivity
Intent intentA = new Intent(...)
startActivityForResult(intent)
intentA
intentA
onCreate()
Intent intentB = new Intent(...)
intentB.putExtra(key, val)
• • •
setResult(RESULT_OK, intentB)
finish()
intentB
intentB
onActivityResult(..., intentB)
val = intentB.getXxxExtra(key)
recipes4dev.tistory.com

UI Thread interacting with Worker Threads



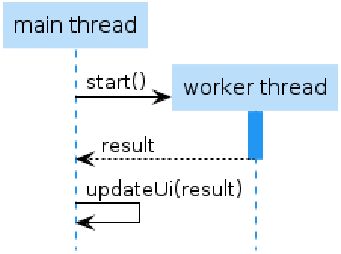
- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

Check: `TP5_06.zip`

Check: `TP5_07.zip`

UI Thread interacting with Worker Threads



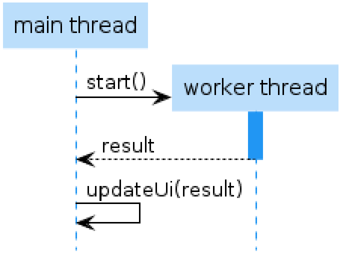
- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

Check: `TP5_06.zip`

Check: `TP5_07.zip`

UI Thread interacting with Worker Threads



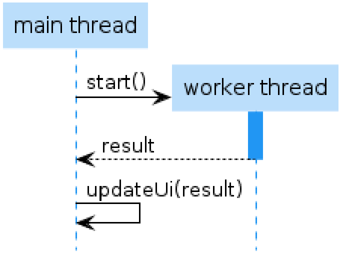
- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

Check: `TP5_06.zip`

Check: `TP5_07.zip`

UI Thread interacting with Worker Threads



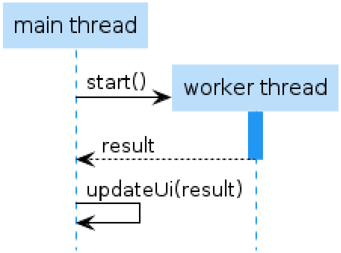
- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

Check: `TP5_06.zip`

Check: `TP5_07.zip`

UI Thread interacting with Worker Threads



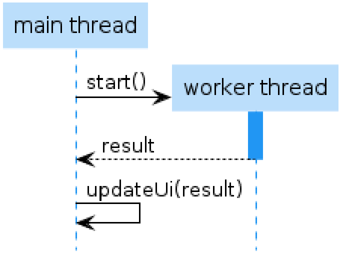
- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

Check: `TP5_06.zip`

Check: `TP5_07.zip`

UI Thread interacting with Worker Threads



- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

Check: `TP5_06.zip`

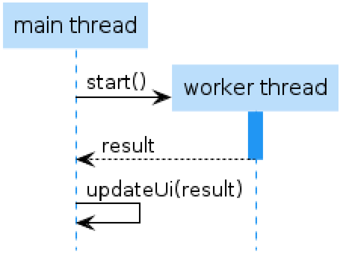
Check: `TP5_07.zip`

Shared Preferences

What are `SharedPreferences`?

- A `SharedPreferences` object points to a file containing `key,value` pairs and provides simple methods to read and write them.
- Each `SharedPreferences` file is managed by the framework and can be private or shared.
- **They are accessible from anywhere within the app to either put data into the file or take data out of the file**.
- For primitive data types: booleans, floats, ints, longs, and strings.

UI Thread interacting with Worker Threads



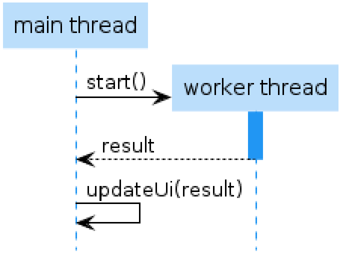
- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

Check: `TP5_06.zip`

Check: `TP5_07.zip`

UI Thread interacting with Worker Threads



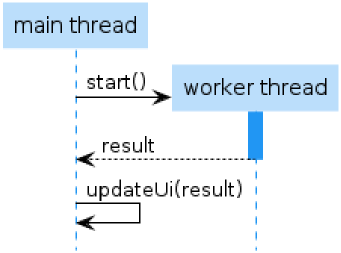
- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

Check: `TP5_06.zip`

Check: `TP5_07.zip`

UI Thread interacting with Worker Threads



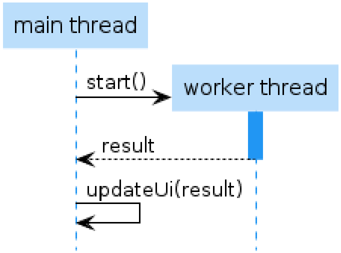
- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

Check: `TP5_06.zip`

Check: `TP5_07.zip`

UI Thread interacting with Worker Threads



- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

Check: `TP5_06.zip`

Check: `TP5_07.zip`

Commit changes

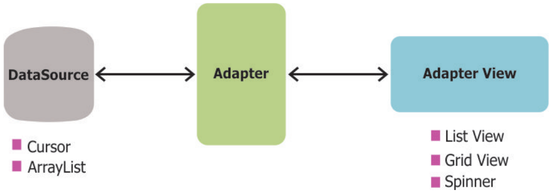
- In order to commit changes can be used two methods:
  - `apply`
    - This saves your data into memory immediately and saves the data to disk on a separate thread. So there is no chance of blocking the main thread (your app won't hang).
    - It is the preferred technique but has only been available since Gingerbread (API 9, Android 2.3).
  - `commit`
    - Calling this will save the data to the file however, the process is carried out in the thread that called it, stopping everything else until the save is complete. It returns true on successful completion, false on failure.
    - Use `commit` if you need confirmation of the success of saving your data or if you are developing for pre-Gingerbread devices.
    - It has been available since API 1

Check: `TP5_03.zip`

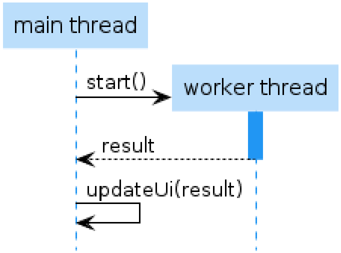
Android Adapter

Adapters (I)

- Android's Adapter is described in the API documentation, as **a bridge** between an `AdapterView` and the underlying data for that `View`.
  - An `AdapterView` is a group of `View` components in Android that include the `ListView`, `Spinner`, and `GridView`.



UI Thread interacting with Worker Threads



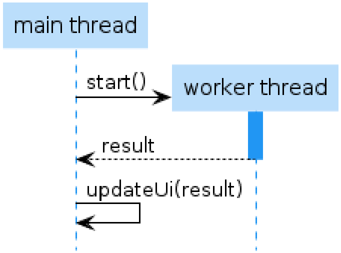
- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

Check: `TP5_06.zip`

Check: `TP5_07.zip`

UI Thread interacting with Worker Threads



- `runOnUIthread`
- `View.post`
- `Handler`

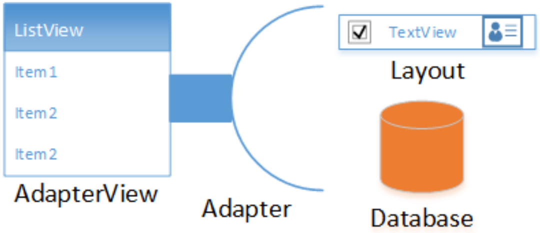
Check: `TP5_05.zip`

Check: `TP5_06.zip`

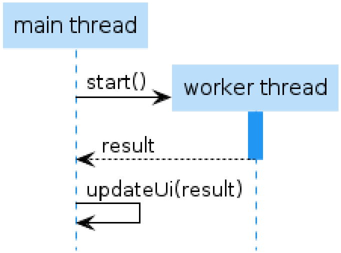
Check: `TP5_07.zip`

Custom Adapter (II)

- In order to display a series of items into a list using a custom representation of the items:
    - Create a layout with a `ListView`.
    - Create a custom XML layout for each row.
    - Create a custom `Adapter` class, which could be a subclass of `BaseAdapter` class.



UI Thread interacting with Worker Threads



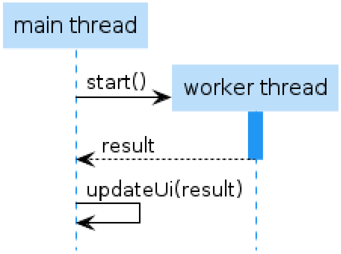
- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

Check: `TP5_06.zip`

Check: `TP5_07.zip`

UI Thread interacting with Worker Threads



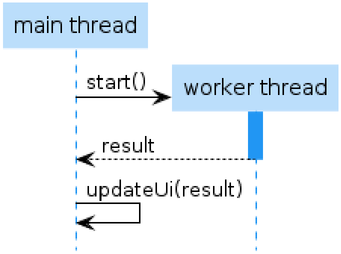
- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

Check: `TP5_06.zip`

Check: `TP5_07.zip`

UI Thread interacting with Worker Threads



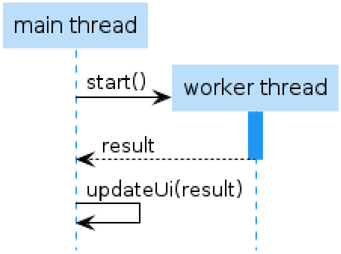
- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

Check: `TP5_06.zip`

Check: `TP5_07.zip`

UI Thread interacting with Worker Threads



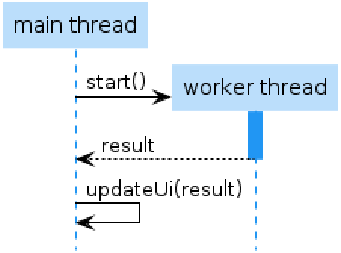
- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

Check: `TP5_06.zip`

Check: `TP5_07.zip`

UI Thread interacting with Worker Threads



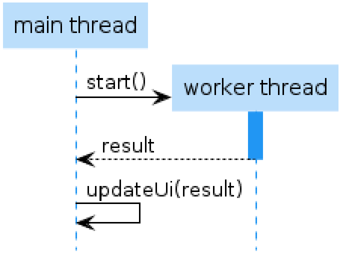
- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

Check: `TP5_06.zip`

Check: `TP5_07.zip`

UI Thread interacting with Worker Threads



- `runOnUIthread`
- `View.post`
- `Handler`

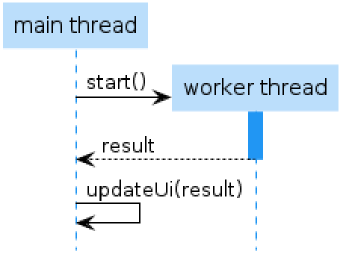
Check: `TP5_05.zip`

Check: `TP5_06.zip`

Check: `TP5_07.zip`

Threads & WorkerThreads

UI Thread interacting with Worker Threads



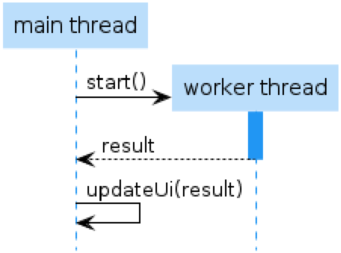
- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

Check: `TP5_06.zip`

Check: `TP5_07.zip`

UI Thread interacting with Worker Threads



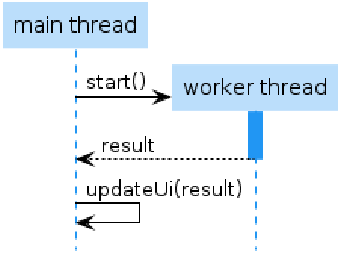
- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

Check: `TP5_06.zip`

Check: `TP5_07.zip`

UI Thread interacting with Worker Threads



- `runOnUIthread`
- `View.post`
- `Handler`

Check: `TP5_05.zip`

Check: `TP5_06.zip`

Check: `TP5_07.zip`

Bibliography

Resources

- "Mastering Android Application Development", by Antonio Pachon Rui, 2015
- `https://developer.android.com/index.html`
- `http://simple.sourceforge.net/home.php`
  `http://simple.sourceforge.net/download/stream/doc/tutorial/tutorial.php`