Design Patterns

MSc in Computer Science

Produced by

Eamonn de Leastar (edeleastar@wit.ie)

Department of Computing, Maths & Physics
Waterford Institute of Technology

http://www.wit.ie

http://elearning.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit

Yamba

Xtend version Encapsulated as 3 Labs

- A - Enable Simple Tweet + timeline update on background thread
- B - Move background thread to an Android Service + restructure application to use Lambdas + Command pattern
- C - replace custom event mechanism with generic Broadcast Receivers

Lab

START

Objectives

- Develop an application in Android using the Xtend language
- Base the application on a twitter-like service hosted here:
  - http://yamba.marakana.com
- The structure of the application is derived from Learning Android

Lab

START

Objectives

- Introduce Services into the Xtend YambaX application
- Implement a background service to periodically update the twitter timeline

Lab

START

Objectives

- Incorporate Broadcast Event Senders/Receivers into the application
- Use BootReciever to recieve boot event to start application service on launch
- Use NetworkReveicer to stop/start service when network is starting/stopping
- Use Broadcast Events for status updates from background service to TimelineActivity

Yamba X

Status Update

hello sync!

hello sync!

Update

Preferences

Username

Please enter your username

Password

Please enter your password

API Root

URL of Root API for your service

Preferences

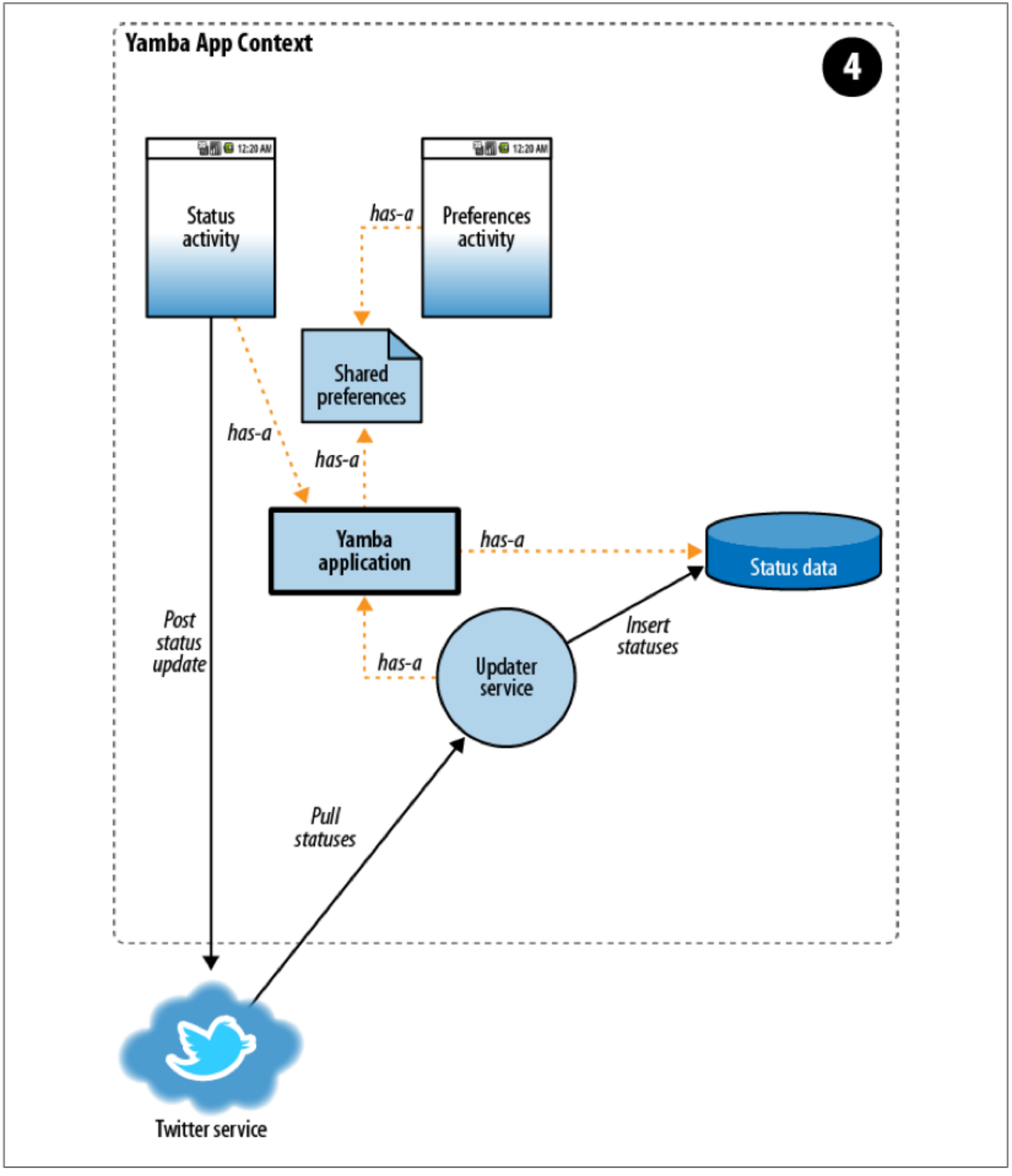


Figure 9-1. Yamba completion

Android Services

- A Service is an application component that can perform long-running operations in the background and does not provide a user interface.

- An application can start a service and it will continue to run in the background even if the user switches to another application.

- Additionally, a component can bind to a service to interact with it and even perform interprocess communication (IPC).

- For example, a service might handle network transactions, play music, perform file I/O, or interact with a content provider, all from the background.

Service Types

- ***Started*** · A service is "started" when an application component (such as an activity) starts it by calling startService().

- ***Bound*** · A service is "bound" when an application component binds to it by calling bindService().

‘Started’ Service

- Once started, a service can run in the background indefinitely, even if the component that started it is destroyed.

- Usually, a started service performs a single operation and does not return a result to the caller.

- For example, it might download or upload a file over the network. When the operation is done, the service should stop itself.

‘Bound’ Service

- A bound service offers a client-server interface that allows components to interact with the service, send requests, get results, and even do so across processes with interprocess communication (IPC).

- A bound service runs only as long as another application component is bound to it.

- Multiple components can bind to the service at once, but when all of them unbind, the service is destroyed.

Caution!

- A service runs in the main thread of its hosting process—the service does not create its own thread and does not run in a separate process (unless you specify otherwise).

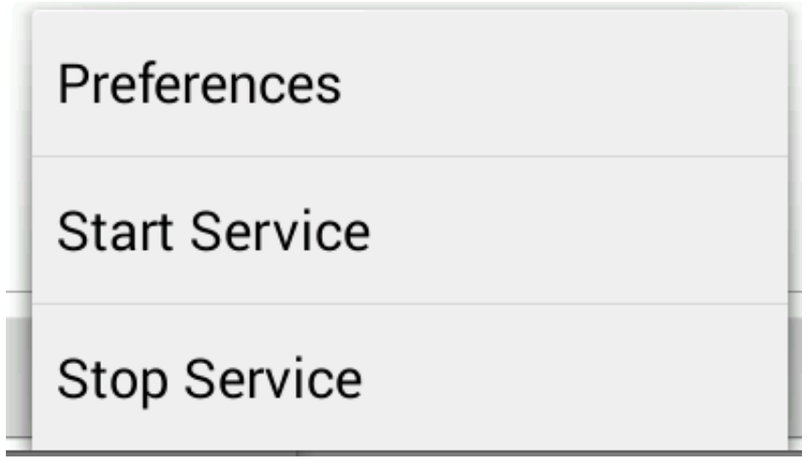
- This means that, if your service is going to do any CPU intensive work or blocking operations (such as MP3 playback or networking), you should create a new thread within the service to do that work.

- By using a separate thread, you will reduce the risk of Application Not Responding (ANR) errors and the application's main thread can remain dedicated to user interaction with your activities.

Initial UpdaterService, including menu for start/stop

edeleastar authored 2 days ago

a4bf40e023




```
class UpdaterService extends Service
{
  override onBind(Intent intent)
  {
  }

  override onCreate()
  {
    super.onCreate
  }

  override onStartCommand(Intent intent, int flags, int startId)
  {
    super.onStartCommand(intent, flags, startId)
    START_STICKY;
  }

  override onDestroy()
  {
    super.onDestroy
  }
}
```

- Start/Stop from Activities Menu

```
class StatusActivity extends Activity
{
  //...
  override onOptionsItemSelected(MenuItem item)
  {
    switch (item.getItemId())
    {
      case R.id.itemServiceStart: startService (new Intent(this, typeof(UpdaterService)))
      case R.id.itemServiceStop:  stopService  (new Intent(this, typeof(UpdaterService)))
      case R.id.itemPrefs:        startActivity(new Intent(this, typeof(PrefsActivity)))
    }
   true
  }
```

Refactor Thread as a lambda

edeleastar authored 2 days ago

- Threads can be modelled as lambda in Xtend using ‘as Runnable’ type specified

- This enables considerable brevity/flexibility

```
class UpdaterService extends Service
{
  val DELAY   = 10000
  var running = false
  var Thread updateThread

  var updater = [ |
                  while (!Thread.currentThread().isInterrupted() && running)
                  {
                    try
                    {
                      Thread.sleep(DELAY);
                    }
                    catch (InterruptedException e)
                    {}
                  } ] as Runnable

  override onCreate()
  {
    updateThread = new Thread(updater)
    super.onCreate
  }

  override onStartCommand(Intent intent, int flags, int startId)
  {
    super.onStartCommand(intent, flags, startId)
    running = true
    updateThread.start
    START_STICKY;
  }

  override onDestroy()
  {
    super.onDestroy
    running = false
    updateThread.interrupt
  }
}
```

Log timeline perioducally to console

edeleastar authored 2 days ago

e509c23f52

Browse code

```
class UpdaterService extends Service
{
  val DELAY   = 10000
  var running = false

  var Thread                updateThread
  var TwitterAPI            twitter
  var List<Twitter.Status> timeline;

  var updater = [ |
                  while (!Thread.currentThread().isInterrupted() && running)
                  {
                    try
                    {
                      timeline = twitter.getFriendsTimeline()
                      timeline.forEach[ Log.d("YAMBA", String.format("%s: %s", it.user.name, it.text)); ]
                      Thread.sleep(DELAY);
                    }
                    catch (TwitterException e)
                    {
                      Log.e("YAMBA", "Failed to connect to twitter service", e);
                    }
                    catch (InterruptedException e)
                    {}
                  } ] as Runnable

...
}
```

Problems | Javadoc | Declaration | **Console** | LogCat | History

Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.

Lev	Time	PID	TID	Application	Tag	Text
D	05-04 06:46:20.880	1660	1687	com.marakana.yambax	dalvikvm	GC_FOR_ALLOC freed 523K, 13% free 4096K/4700K, paused 26ms, total 27ms
D	05-04 06:46:21.020	1660	1687	com.marakana.yambax	YAMBA	Marakana Student: dfgdfg
D	05-04 06:46:21.030	1660	1687	com.marakana.yambax	YAMBA	Marakana Student: ddh
D	05-04 06:46:21.030	1660	1687	com.marakana.yambax	YAMBA	Marakana Student: dasdcaasdcasdc
D	05-04 06:46:21.030	1660	1687	com.marakana.yambax	YAMBA	Marakana Student: dasdca
D	05-04 06:46:21.030	1660	1687	com.marakana.yambax	YAMBA	Marakana Student: erferf
D	05-04 06:46:21.030	1660	1687	com.marakana.yambax	YAMBA	Marakana Student: erferf
D	05-04 06:46:21.030	1660	1687	com.marakana.yambax	YAMBA	Marakana Student: hello vbjjvnnbbvv
D	05-04 06:46:21.030	1660	1687	com.marakana.yambax	YAMBA	Marakana Student: hello
D	05-04 06:46:21.030	1660	1687	com.marakana.yambax	YAMBA	Marakana Student: zsdasdvsd
D	05-04 06:46:21.030	1660	1687	com.marakana.yambax	YAMBA	Marakana Student: Vamsi
D	05-04 06:46:21.030	1660	1687	com.marakana.yambax	YAMBA	Marakana Student: rfrfr
D	05-04 06:46:21.030	1660	1687	com.marakana.yambax	YAMBA	Marakana Student: cvasdvasdv
D	05-04 06:46:21.030	1660	1687	com.marakana.yambax	YAMBA	Marakana Student: asdfasdf
D	05-04 06:46:21.030	1660	1687	com.marakana.yambax	YAMBA	Marakana Student: yyyy
D	05-04 06:46:21.030	1660	1687	com.marakana.yambax	YAMBA	Marakana Student: yy
D	05-04 06:46:21.030	1660	1687	com.marakana.yambax	YAMBA	Marakana Student: uuu

Facror out BackgroundService util class. Simplify UpdaterService

edeleastar authored 2 days ago

b636669097

- General purpose BackgroundService Class
- Can be used by any application
- Encapsulate simple delay/wakeup cycle

```
abstract class BackgroundService extends Service
{
  val DELAY   = 10000
  var running = false
  var Thread updateThread

  var updater = [ |
                  while (!Thread.currentThread().isInterrupted() && running)
                  {
                    try
                    {
                      this.doBackgroundTask()
                      Thread.sleep(DELAY);
                    }
                    catch (InterruptedException e)
                    {}
                  } ] as Runnable

  override onBind(Intent intent)
  {
    null
  }

  def abstract void doBackgroundTask()

  def startBackgroundTask()
  {
    running = true
    updateThread.start
  }
  def stopBackgroundTask()
  {
    running = false
    updateThread.interrupt
  }
  override onCreate()
  {
    updateThread = new Thread(updater)
    super.onCreate
  }
}
```

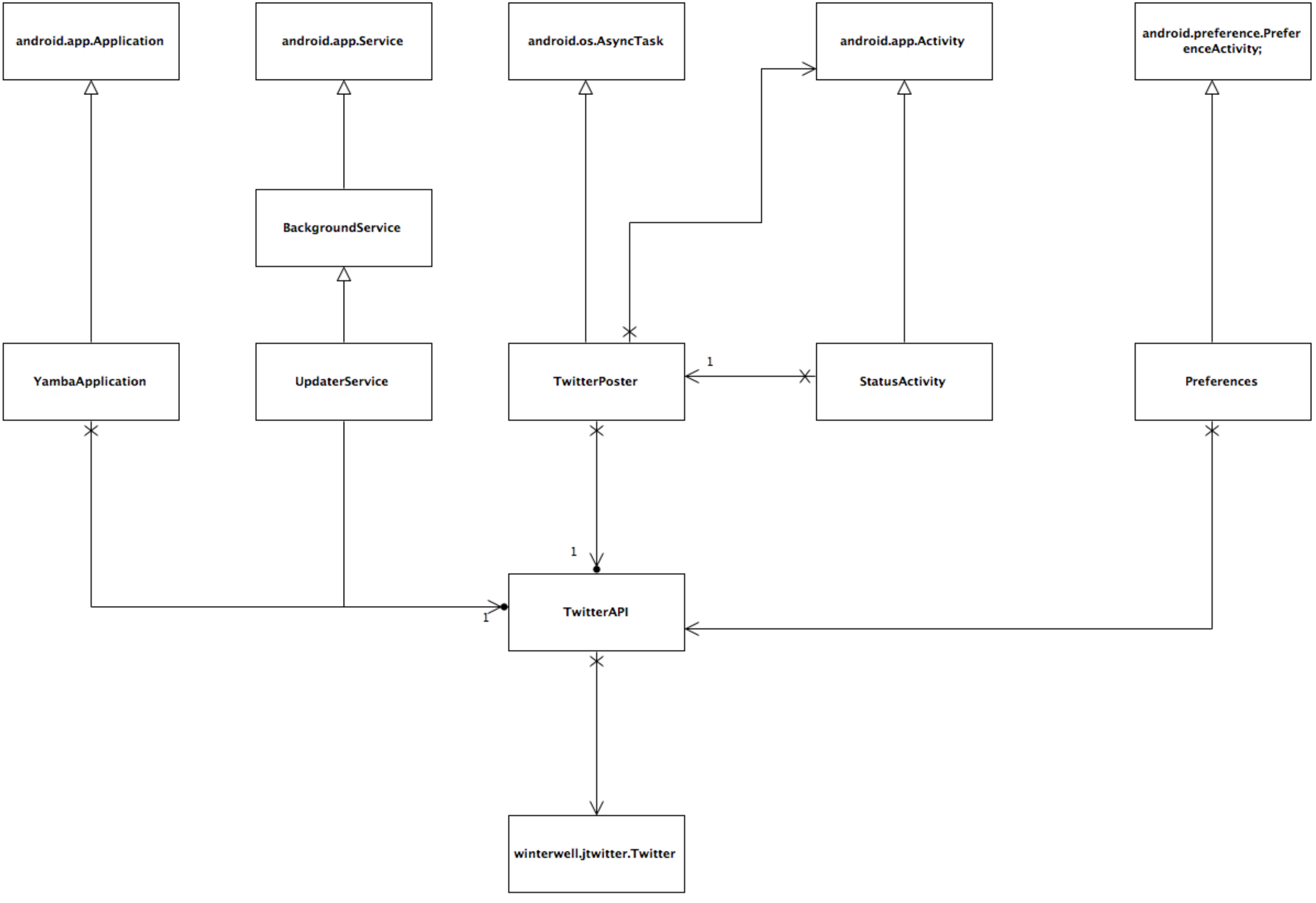
Facror out BackgroundService util class. Simplify UpdaterService

edeleastar authored 2 days ago

b636669097

- UpdaterService simplified with doBackground started on a different thread

```
class UpdaterService extends BackgroundService
{
  var TwitterAPI           twitter
  var List<Twitter.Status> timeline;

  override onBind(Intent intent)
  {
    null
  }

  override def void doBackgroundTask()
  {
    try
    {
      timeline = twitter.getFriendsTimeline()
      timeline.forEach[ Log.d("YAMBA", String.format("%s: %s", it.user.name, it.text))]
    }
    catch (TwitterException e)
    {
      Log.e("YAMBA", "Failed to connect to twitter service", e);
    }
  }

  override onCreate()
  {
    super.onCreate
    var app = getApplication() as YambaApplication
    this.twitter = app.twitter
  }

  override onStartCommand(Intent intent, int flags, int startId)
  {
    super.onStartCommand(intent, flags, startId)
    startBackgroundTask
    START_STICKY;
  }

  override onDestroy()
  {
    super.onDestroy
    stopBackgroundTask
  }
}
```


android.app.Application
android.app.Service
android.os.AsyncTask
android.app.Activity
android.preference.PreferenceActivity;
BackgroundService
YambaApplication
UpdaterService
TwitterPoster
1
StatusActivity
Preferences
1
1
TwitterAPI
winterwell.jtwitter.Twitter