Как реализовать загрузку изображений в список в отдельном потоке на Android

Задача:

Реализовать механизм загрузки изображений из Интернета и отображения их в списке. При этом загрузка изображений должна быть реализована в отдельном потоке, во избежания «зависания» UI приложения.

Реализация:

Для реализации поставленной задачи использованы стандартный виджет ListView и адаптер — ArrayAdapter. Для работы с изображениями создан helper-класс ImageManager, который имеет два метода downloadImage() и fetchImage(). Первый загружает изображений из Интернета. Второй — вызывает загрузку изображений в отдельном потоке и устанавливает результат в ImageView.

Пример испольования:

Реализацию поставленной задачи будем расматривать на примере моего проекта. И на его код я буду ссылаться в статье.  
Исходники: [fileshare.in.ua/3053597](http://fileshare.in.ua/3053597)  
APK: [fileshare.in.ua/3053596](http://fileshare.in.ua/3053596)

Описание реализации:

Давайте рассмотрим поподробнее каждый из методов ImageManager'а:

package com.rudenko.android.ListIconFetching;

import java.io.BufferedInputStream;

import java.io.IOException;

import java.net.HttpURLConnection;

import java.net.MalformedURLException;

import java.net.URL;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.os.Handler;

import android.os.Message;

import android.util.Log;

import android.widget.ImageView;

public class ImageManager {

  private final static String TAG = "ImageManager";

  /\*\* Private constructor prevents instantiation from other classes \*/

  private ImageManager () {}

  public static void fetchImage(final String iUrl, final ImageView iView) {

    if ( iUrl == null || iView == null )

      return;

    final Handler handler = new Handler() {

      @Override

      public void handleMessage(Message message) {

        final Bitmap image = (Bitmap) message.obj;

        iView.setImageBitmap(image);

      }

    };

    final Thread thread = new Thread() {

      @Override

      public void run() {

        final Bitmap image = downloadImage(iUrl);

        if ( image != null ) {

          Log.v(TAG, "Got image by URL: " + iUrl);

          final Message message = handler.obtainMessage(1, image);

          handler.sendMessage(message);

        }

      }

    };

    iView.setImageResource(R.drawable.icon);

    thread.setPriority(3);

    thread.start();

  }

  public static Bitmap downloadImage(String iUrl) {

    Bitmap bitmap = null;

    HttpURLConnection conn = null;

    BufferedInputStream buf\_stream = null;

    try {

      Log.v(TAG, "Starting loading image by URL: " + iUrl);

      conn = (HttpURLConnection) new URL(iUrl).openConnection();

      conn.setDoInput(true);

      conn.setRequestProperty("Connection", "Keep-Alive");

      conn.connect();

      buf\_stream = new BufferedInputStream(conn.getInputStream(), 8192);

      bitmap = BitmapFactory.decodeStream(buf\_stream);

      buf\_stream.close();

      conn.disconnect();

      buf\_stream = null;

      conn = null;

    } catch (MalformedURLException ex) {

      Log.e(TAG, "Url parsing was failed: " + iUrl);

    } catch (IOException ex) {

      Log.d(TAG, iUrl + " does not exists");

    } catch (OutOfMemoryError e) {

      Log.w(TAG, "Out of memory!!!");

      return null;

    } finally {

      if ( buf\_stream != null )

        try { buf\_stream.close(); } catch (IOException ex) {}

      if ( conn != null )

        conn.disconnect();

    }

    return bitmap;

  }

}

\* This source code was highlighted with Source Code Highlighter.

Метод fetchImage():

public static void fetchImage(final String iUrl, final ImageView iView);

Входные параметры:

iUrl — URL к изображению для загрузки  
iView — ссылка на виджет ImageView, которому будет назначено изображение после загрузки.  
Оба параметра являются обязательными.

Результат:

Функция не возвращает ничего.

Краткое описание:

Функция создает поток для загрузки изображения. На время загрузки во входной ImageView устанавливается стандартное изображение. После завершения загрузки, изображение входного ImageView обновляется загруженным.  
  
Несколько слов о потоке: в строке 46 происходит понижение приоритета потока. Это сделано для того, что бы данный поток не забирал ресурсы необходимые для корректной работы приложения.

Метод downloadImage():

public static Bitmap downloadImage(String iUrl);

Входные параметры:

iUrl — URL к изображению для загрузки

Результат:

Изображение загруженное из интернета, либо Null — если операция не была выполнена успешно.

Краткое описание:

В функции создается соединение с сервером, где находится изображение. Происходит получение входного потока, который далее передается в BitmapFactory для создания изображения.

Как использовать ImageManager:

Рассмотрим на примере к статье. В методе FetchImageAdapter.getView(), для подгрузки изображений в ImageView строки списка, используется следующая строка:  
ImageManager.fetchImage(android.image, holder.ib\_logo);  
где android.image — URL к изображению, а holder.ib\_logo — ImageView строки списка.

Заключение:

Данный механизм подходит для загрузки изображения из Интернета в параллельном потоке, для любого виджета ImageView Android. Т. е. данный механизм можно использовать не только для конкретно поставленной задачи.