Programação para Dispositivos Móveis

Leopoldo Teixeira
lmt@cin.ufpe.br | @leopoldomt

Vamos criar um app mais interessante...

Leitor RSS

- Capturar o conteúdo de um feed RSS público
- Extrair os dados importantes (título, link, etc)
- Apresentar uma lista de itens RSS
- Ao clicar em um item, devemos abrir o navegador no link associado
- Deve ser possível trocar o feed durante a execução do aplicativo

RSS

- Formato de dados que permite que produtores de conteúdo publiquem informações, áudio/vídeo, etc.
- A publicação se dá geralmente por meio de um feed, que é regularmente atualizado
- É possível ‘assinar’ feeds, para receber novidades automaticamente
- O formato é baseado em XML, mais detalhes em http://www.rssboard.org/rss-specification

Qual o primeiro passo?

Leitor RSS

- Criar um projeto com uma blank activity e colocar um TextView só para testarmos inicialmente
  - outra opção é usar fragmentos…
- No método onStart(), atualizar o conteúdo com uma string qualquer em um text view
  - *(no futuro serão utilizados os dados provenientes do feed)*

Leitor RSS

- Agora vamos capturar o conteúdo de um feed RSS
- Atualize o conteúdo do fragmento com um feed RSS disponível online
- Sugestão: http://rss.cnn.com/rss/edition.rss

Iniciando conexão HTTP

```
URL url = new URL("http://www.android.com/");
HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
  try {
    InputStream in = new BufferedInputStream(urlConnection.getInputStream());
    readStream(in);
   finally {
    urlConnection.disconnect();
  }
}
```

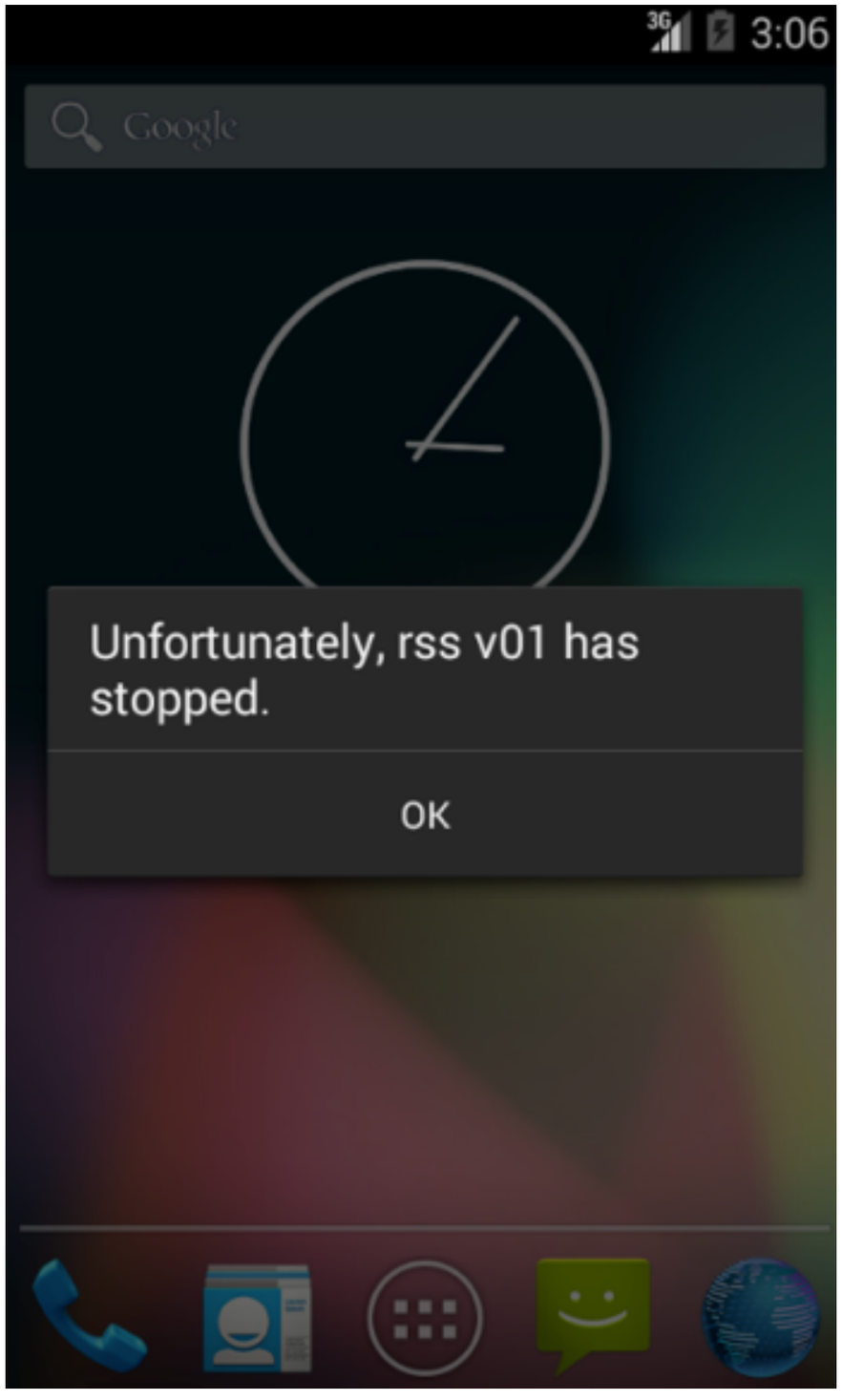
http://developer.android.com/reference/java/net/HttpURLConnection.html

getRssFeed(…)

```
private String getRssFeed(String feed) throws IOException {
    InputStream in = null;
    String rssFeed = null;
    try {
        URL url = new URL(feed);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        in = conn.getInputStream();
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        byte[] buffer = new byte[1024];
        for (int count; (count = in.read(buffer)) != -1; ) {
            out.write(buffer, 0, count);
        }
        byte[] response = out.toByteArray();
        rssFeed = new String(response, "UTF-8");
    } finally {
        if (in != null) {
            in.close();
        }
    }
    return rssFeed;
}
```

http://developer.android.com/training/basics/network-ops/connecting.html

O que aconteceu?



Operações de rede devem ser realizadas fora da thread principal!

http://android-developers.blogspot.com.br/2010/07/multithreading-for-performance.html

O que acontece ao chamarmos **`label.setText(…)`**?

Main Thread

ANR

Jank

Ou seja…

Mas o que são processos mesmo?

Processos

- Ambientes de execução auto-contidos
- Gerenciam recursos de forma separada de outros processos, como
  - memória
  - arquivos abertos
  - conexões de rede

https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

O que é uma thread?

Threads

- Visão Conceitual
  - É uma de muitas computações paralelas rodando em um processo do SO
- Visão de implementação
  - Um *program counter* e uma *stack*
  - Compartilha a heap e áreas de memória estáica com outras threads rodando em um processo

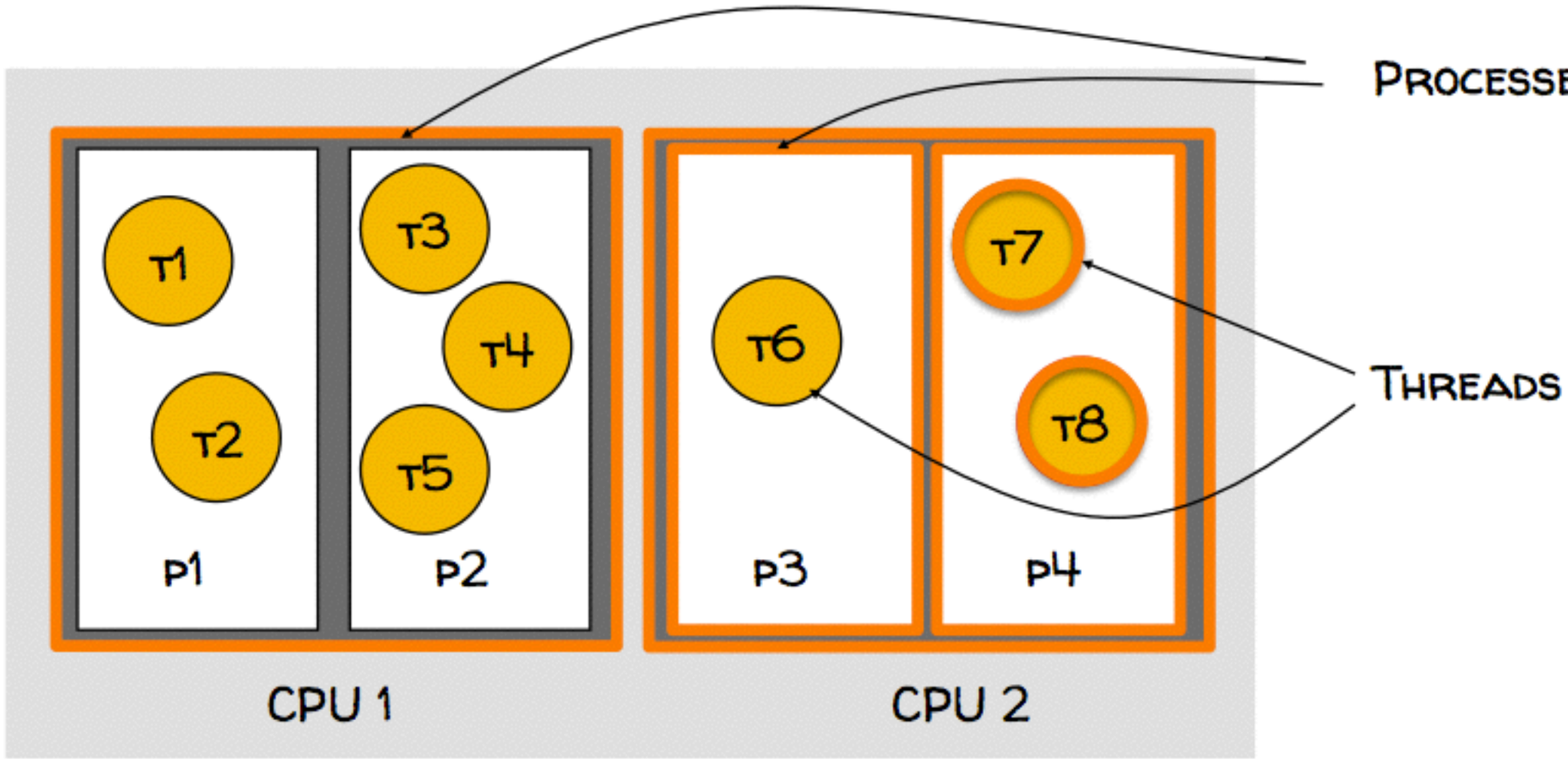


Image from Adam Porter (University of Maryland)

Threads em Java

- Representadas por um objeto do tipo **`java.lang.Thread`**
- Threads devem implementar a interface **`Runnable`**
  - **`void run()`**

Alguns métodos

- **`void start()`**
  - inicia a thread
- **`void sleep(long time)`**
  - coloca a thread ‘para dormir’ durante dado período
- **`void wait()`**
  - thread aguarda até que outra thread invoque o método **`notify()`** neste objeto
- **`void notify()`**
  - acorda uma thread que está esperando

Fluxo Geral

- Instanciar um objeto do tipo Thread
- Chamar o método **`start()`**
  - que por sua vez chama **`run()`**
- A thread termina quando o método **`run()`** retorna

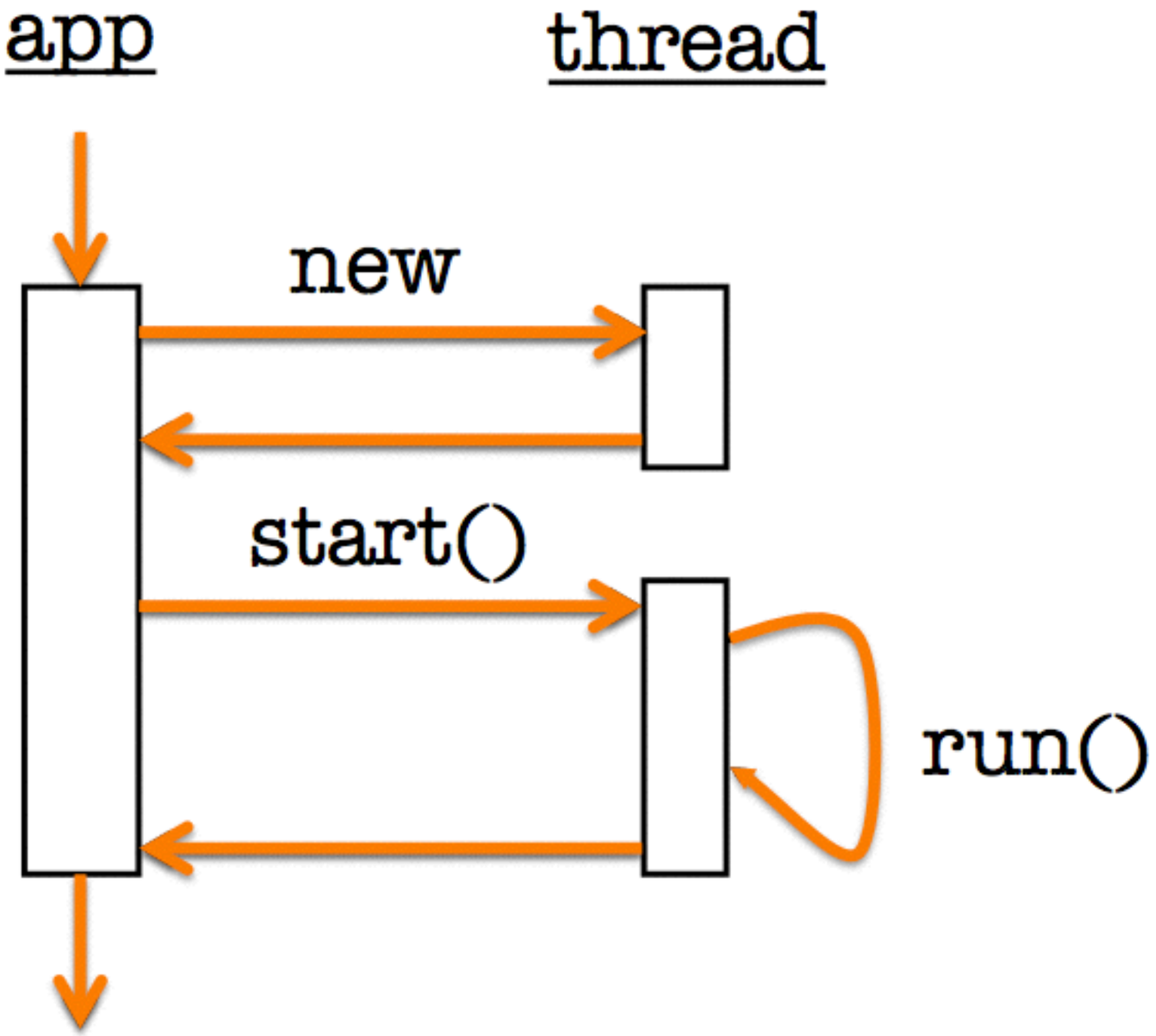


Image from Adam Porter (University of Maryland)

Running example

- Aplicação com dois botões:
  - Carregar Ícone (operação "*demorada*")
    - carrega uma imagem a partir de um arquivo
    - exibe a imagem
- Outro
  - exibe um texto (toast)

Sem Threads

Com Threads (incorreto)

listener do botão dispara uma thread separada para carregar a imagem e exibi-la

O que houve?

UI Thread

- Aplicações tem uma thread principal (UI Thread)
- Componentes da aplicação que compartilham o processo usam a mesma UI thread
- Interação com o usuário, callbacks do sistema e métodos do ciclo de vida de componentes são todos tratados na UI thread
- UI thread não é thread-safe

Implicações

- Bloquear UI Thread prejudica uso e capacidade de resposta da aplicação
    - operações que levem tempo devem rodar em threads de background

- Não devemos acessar UI de uma thread que não seja UI

Então como fazer para atualizar a UI a partir de uma background thread?

Melhorando a Solução

- Precisamos realizar alguma tarefa em segundo-plano, mas depois é necessário atualizar a UI

- Android fornece métodos que são garantidos de rodar na UI Thread

- **`boolean View.post(Runnable)`**

- **`void Activity.runOnUiThread(Runnable)`**

- **`boolean View.postDelayed(Runnable, long)`**

Com Threads e `view.post(…)`

Com Threads e `...runOnUi...(...)`

Alternativa:

AsyncTasks

AsyncTask

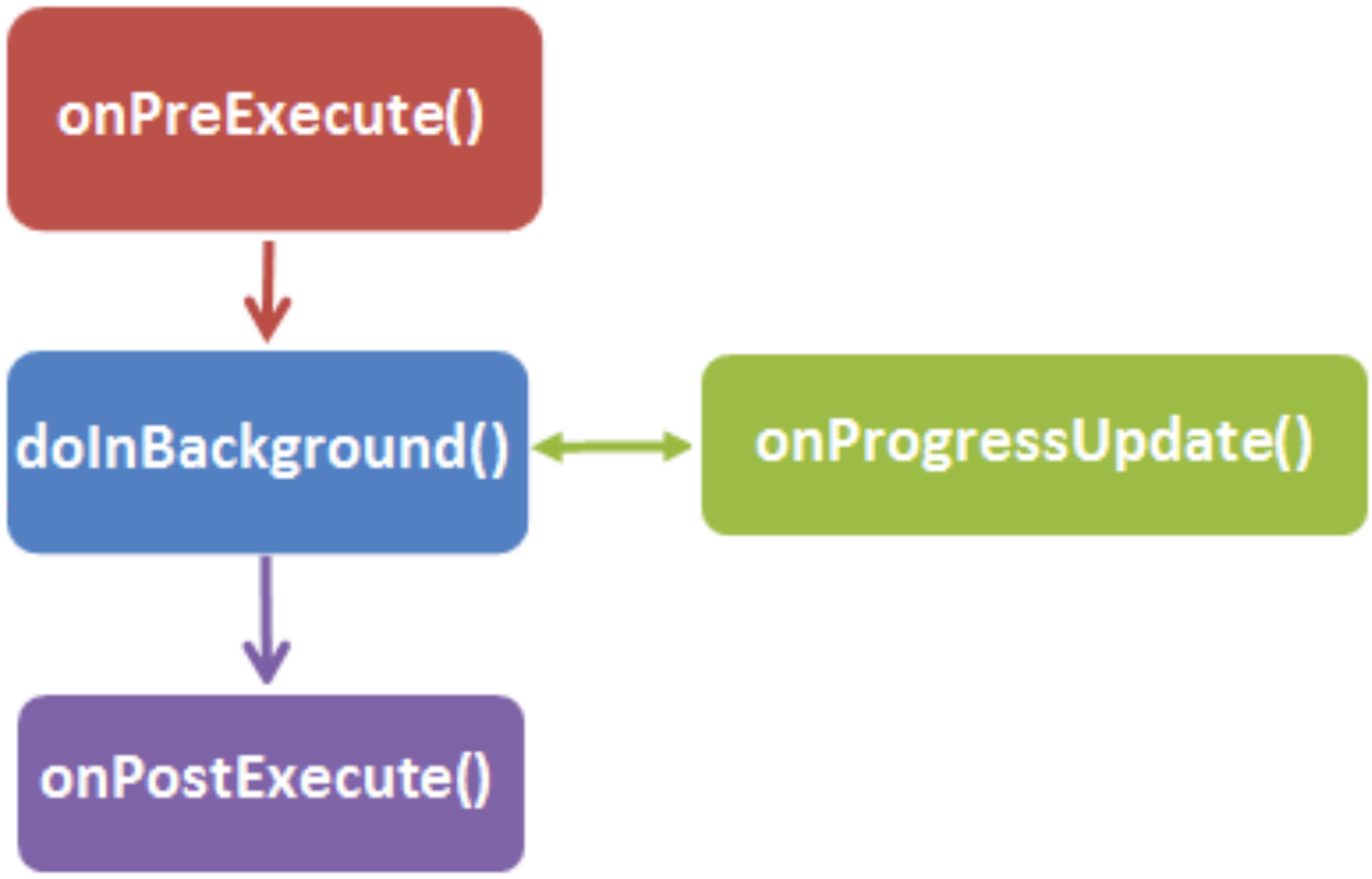
- Divide o trabalho em
  - Background thread
    - realiza a tarefa
    - indica progresso
- UI Thread
  - configuração inicial
  - publica progresso intermediária
  - usa o resultado final

Como fazer isto?

Usando AsyncTask

- Classe Genérica
  - `class AsyncTask<Params,Progress,Result> {…}`
- Parâmetros
  - **`Params`** - tipo usado para tarefa em background
  - **`Progress`** - tipo usado para indicar progresso
  - **`Result`** - tipo do resultado

Fluxo Geral



http://programmerguru.com/android-tutorial/wp-content/uploads/2014/01/asynctask_steps.png

Fluxo Geral

- **`void onPreExecute()`**
  - roda na UI Thread antes de **`doInBackground()`**
- **`Result doInBackground(Params… params)`**
  - realiza a tarefa na thread separada
  - pode chamar **`void publishProgress(Progress… values)`**
- **`void onProgressUpdate(Progress… values)`**
  - invocado nas chamadas de **`publishProgress(…)`**
- **`void onPostExecute(Result result)`**
  - roda após **`doInBackground()`**

```
private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {
    protected Long doInBackground(URL... urls) {
        int count = urls.length;
        long totalSize = 0;
        for (int i = 0; i < count; i++) {
            totalSize += Downloader.downloadFile(urls[i]);
            publishProgress((int) ((i / (float) count) * 100));
            // Escape early if cancel() is called
            if (isCancelled()) break;
        }
        return totalSize;
    }

    protected void onProgressUpdate(Integer... progress) {
        setProgressPercent(progress[0]);
    }

    protected void onPostExecute(Long result) {
        showDialog("Downloaded " + result + " bytes");
    }
}
```

```
new DownloadFilesTask().execute(url1, url2, url3);
```

http://developer.android.com/reference/android/os/AsyncTask.html

Carregando imagem com AsyncTask

O que você **não**
precisa fazer

Estados

- **`AsyncTask.Status getStatus()`**
  - retorna o estado atual de um objeto **`AsyncTask`**
  - **`PENDING`**
    - ainda não foi executado
- **`RUNNING`**
  - rodando no momento
- **`FINISHED`**
  - indica que **`onPostExecute(Result)`** finalizou

Cancelando…

- **`task.cancel(boolean)`**
  - tenta cancelar execução da tarefa (interrupção depende do booleano passado como argumento)
  - seta uma flag, ao chamar **`isCancelled()`**
  - após **`doInBackground()`** chama **`onCancelled()`** ao invés de **`onPostExecute()`**
  - boa prática: chamar **`isCancelled()`** periodicamente dentro de **`doInBackground()`**

algumas regras…

- A classe deve ser carregada na UI thread (*default*)
- A instância deve ser criada na UI thread
- **`execute(Params…)`** deve ser invocado na UI thread
- Não devemos chamar manualmente **`onPreExecute`**, **`onPostExecute`**, **`doInBackground`**…
- A tarefa pode ser executada apenas uma vez

Programação para Dispositivos Móveis

Leopoldo Teixeira
lmt@cin.ufpe.br | @leopoldomt