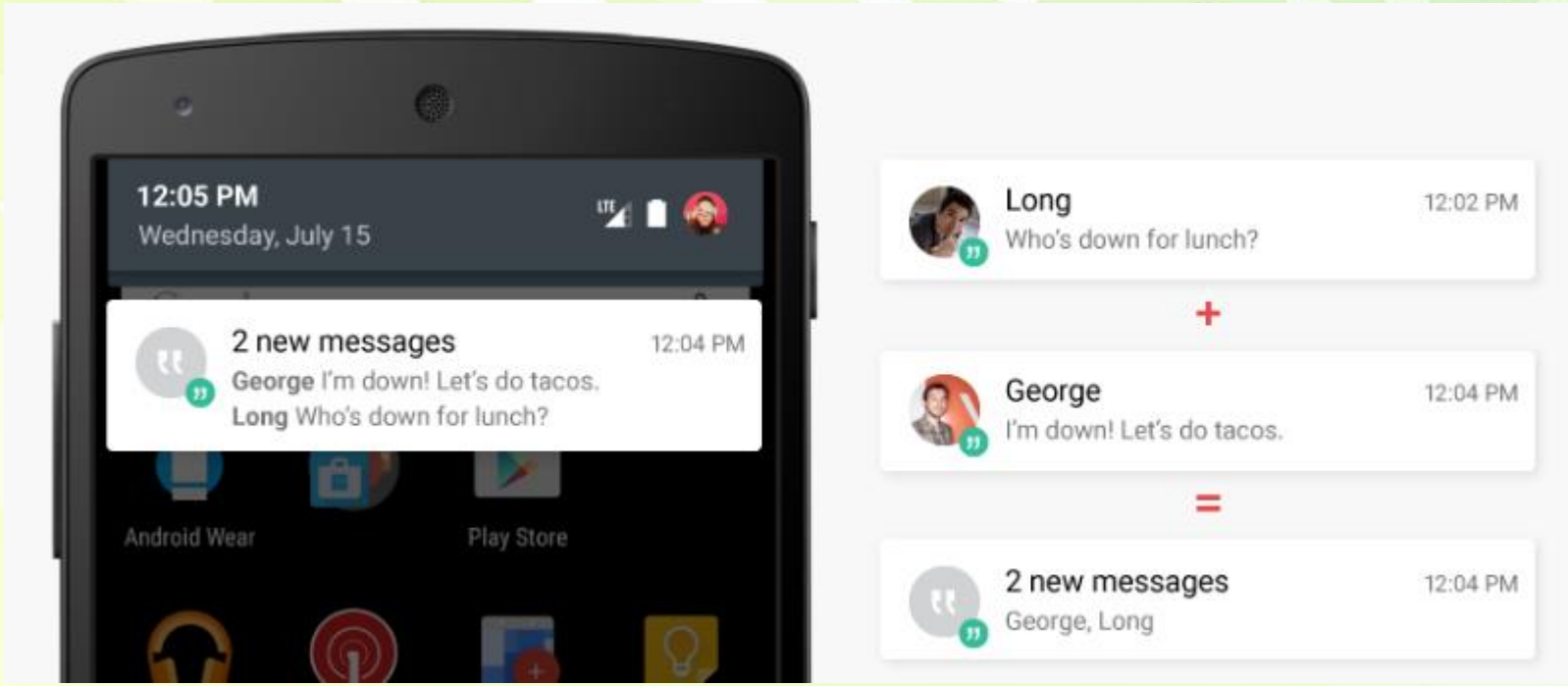
DESENVOLVIMENTO PARA DISPOSITIVOS MÓVEIS
Parte VI

Prof. Msc. Felipe Diniz Dallilo

UNIARA ONLINE

Notification

- Notificações podem ser criadas para sinalizar o usuário sobre determinadas informações.



Notification

- Para começar, você precisa configurar o conteúdo e o canal da notificação usando um objeto NotificationCompat.Builder.

- É possível criar um ícone pequeno, definido pelo método setSmallIcon(). Esse é o único conteúdo visível ao usuário necessário.

Notification

- Um título, pode ser definido pelo método setContentTitle().
- O corpo do texto, é definido pelo método setContentText().
- A prioridade de notificação, definida por setPriority().

Notification

- A prioridade determina se a notificação será intrusiva ou não no Android 7.1 e versões anteriores.

- Para o Android 8.0 e versões mais recentes, você precisa definir a importância do canal.

Notification

```
NotificationCompat.Builder builder = new
NotificationCompat.Builder(this, CHANNEL_ID)
        .setSmallIcon(R.drawable.notification_icon)
        .setContentTitle(textTitle)
        .setContentText(textContent)
        .setPriority(NotificationCompat.PRIORITY_DEFAULT);
```

Notification

- Antes de entregar a notificação no Android 8.0 e versões mais recentes, você precisa registrar o canal de notificação do seu app no sistema, transmitindo uma instância de NotificationChannel para createNotificationChannel(). Portanto, o código a seguir é bloqueado por uma condição na versão SDK_INT:

Notification

```
private void createNotificationChannel() {
    // Criar uma NotificationChannel, mas apenas uma API 26+
    // a classe NotificationChannel é nova e não é suportada na biblioteca
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        CharSequence name = getString(R.string.channel_name);
        String description = getString(R.string.channel_description);
        int importance = NotificationManager.IMPORTANCE_DEFAULT;
        NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name, importance);
        channel.setDescription(description);
        // Registre o canal com o Sistema. Você não pode mudar a importância
        // de outra notificação.
        NotificationManager notificationManager = getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(channel);
    }
}
```

Broadcast Receiver

- **BroadcastReceiver** é basicamente uma entidade configurada para receber informações por mensagens, de outros aplicativos ou do sistema Android e também para enviar informações em modo broadcast.

Broadcast Receiver

- A API BroadcastReceiver é muito similar a uma trigger de banco de dados, isso devido a limitação de 10 segundos que temos de processamento quando a API é acionada e o método **onReceiver()** ser invocado com os dados em mensagem broadcast.
- A mesma, somente deve ser utilizada se certeza que o algoritmo que será executado a partir de **BroadcastReceiver.onReceiver()** não chegará, em hipótese alguma, aos temidos 10 segundos de execução.

Broadcast Receiver

- Apesar de a API BroadcastReceiver não ter sido desenvolvida somente para algoritmos que executam em background, ela é comumente utilizada para aciona-los e esses algoritmos tendem a ter um tempo de processamento não consistente (não conhecido), ou seja, muitas vezes é arriscado deixar o algoritmo principal de execução no **onReceiver()**.

Configuração Dinâmica

```
public class MainActivity extends AppCompatActivity {

  private CustomBroadcastReceiver customBroadcast;

  @Override
  protected void onCreate( Bundle savedInstanceState ) {
    super.onCreate( savedInstanceState );

    customBroadcast = new CustomBroadcastReceiver();
    IntentFilter intentFilter = new IntentFilter( Intent.ACTION_AIRPLANE_MODE_CHANGED );
    registerReceiver( customBroadcast, intentFilter );

    ...
  }

  @Override
  protected void onDestroy() {
      unregisterReceiver( customBroadcast );
      super.onDestroy();
  }
}
```

HTTP URL Connection

Um URLConnection com suporte para recursos específicos do protocolo HTTP.

Os usos desta classe seguem um padrão:

- Obter um novo HttpURLConnection chamando URL # openConnection () e convertendo o resultado em HttpURLConnection.
- Preparar o pedido. A propriedade principal de uma solicitação é seu URI. Os cabeçalhos de solicitação também podem incluir metadados, como credenciais, tipos de conteúdo preferidos e cookies de sessão.
- Opcionalmente, faça upload de um corpo de solicitação.
- As instâncias devem ser configuradas com setDoOutput (true) se incluírem um corpo de solicitação.

HTTP URL Connection

Transmita dados gravando no fluxo retornado por URLConnection.getOutputStream ().

O corpo da resposta pode ser lido do fluxo retornado por URLConnection.getInputStream (). Se a resposta não tiver corpo, esse método retornará um fluxo vazio.

Para desconectar, depois que o corpo da resposta for lido, o HttpURLConnection deve ser fechado chamando o método disconnect ().

A desconexão libera os recursos mantidos por uma conexão para que possam ser fechados ou reutilizados. Por exemplo, para recuperar a página da web em http://www.android.com/:

HTTP URL Connection

```
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 URL Connection

Para fazer upload de dados para um servidor da web, configure a conexão para saída usando setDoOutput (true). Para obter o melhor desempenho, você deve chamar:

- setFixedLengthStreamingMode (int) quando o comprimento do corpo for conhecido com antecedência
- setChunkedStreamingMode (int) quando não for.

Caso contrário, o HttpURLConnection será forçado a armazenar em buffer o corpo completo da solicitação na memória antes de ser transmitido, desperdiçando (e possivelmente esgotando) o heap e aumentando a latência.

HTTP URL Connection

```
HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
try {
    urlConnection.setDoOutput(true);
    urlConnection.setChunkedStreamingMode(0);

    OutputStream out = new BufferedOutputStream(urlConnection.getOutputStream());
    writeStream(out);

    InputStream in = new BufferedInputStream(urlConnection.getInputStream());
    readStream(in);
} finally {
    urlConnection.disconnect();
}
```

Handler

O Handler é uma classe do pacote android.os, como o próprio nome já diz ele é um “Manipulador”. Quando você cria um novo Handler, ele é destinado à fila de mensagens/threads do thread que o criou. A partir daí toda mensagem que for destinada a ele será entregue para aquela fila de mensagens para serem executadas. Resumidamente ele fica responsável por entregar as mensagens para a fila de sua thread, executando-as em seguida.

Handler

Ele pode ser utilizado de duas maneiras: (1) para agendar mensagens e Threads a serem executadas e (2) para enfileirar uma ação a ser executada em um segmento diferente do qual foi enviada.

Handler

```
//construção e configuração do handler
Handler handler = new Handler() {

    @Override
    public void handleMessage(Message msg) {
        //chamo um método para o tratamento da mensagem
        //e melhor organização do código.
        updateUI(msg);
    }
};
```

Handler

```
Handler  h = new Handler();
// tarefa postergada por 100 milissegundos
h.postDelayed(new Runnable() {
    public void run() {
        fazQualquerCoisa();
    }
    }, 100);
```

Handler

```
// tarefa postergada para tão logo quanto possível
h.post(new Runnable() {
    public void run() {
            fazQualquerCoisa();
    }
});
```

Referências

- SILVA, D. **Desenvolvimento para dispositivos móveis**. São Paulo: Pearson Education do Brasil, 2016.
- LEE, V., SCHNEIDER, H., SCHELL, R. **Aplicações móveis: arquitetura, projeto e desenvolvimento**. São Paulo: Pearson Education do Brasil, 2005.
- MEDNIEKS, Z; et al. **Programando o Android**. 2. ed. São Paulo: Novatec, 2012.
- DEITEL, P; DEITEL, H. Java: como programar. 8. ed. São Paulo: Pearson, 2010.
- DUARTE, W. **Delphi para Android e IOS Desenvolvendo Aplicações Móveis**. Rio de Janeiro: Brasport Livros e Multimídia, 2015.
- ABLESON, W. Frank; et al. **Android em ação**. 3. ed. Rio de Janeiro: Elsevier, 2012.
- LECHETA, R. R. **Google Android para Tablets. Aprenda a desenvolver aplicações para o Android**. São Paulo: Novatec, 2012.
- LECHETA, R. R. **Google Android: aprenda a criar aplicações para dispositivos móveis com o Android SDK**. 2. ed. São Paulo: Novatec, 2010.
- ASCENCIO, A. F. G.; CAMPOS, E. A. V. de. **Fundamentos da programação de computadores: algoritmos, pascal, C/C ++ e Java**. 2. ed. São Paulo: Pearson, 2007