Android Threading, BroadcastReceiver, Connectivity (Leitold, Leopold)

Threading und Synchronisation

Um das Blockieren des UIs zu verhindern, können sogennannte Threads verwendet werden, die eine bestimmte Routine im Hintergrund durchführen. Ein Beispiel dafür wäre das Ausführen eines HTTP-Requests, oder das Berechnen von bestimmten Daten. Ein einhergehendes Problem von Multithreading ist das Synchronisieren von Threads, denn das Zugreifen auf gemeinsame Daten kann zu Race-Conditions führen. Deswegen sollte das Aufrufen von Methoden aus einem Thread, die direkt mit dem UI arbeiteten, unbedingt verhindert werden. Stattdessen sollten Helper-Methoden wie `runOnUiThread` verwendet werden, um einen Code auf dem UI-Thread auszuführen. Das unten gezeigte Beispiel zeigt, wie eine Methode sicher aus einem Thread aufgerufen werden kann.

```
new Thread(() -> {
    // Do some CPU intensive or I/O operation here
    runOnUiThread(() -> {
        // do something on the UI
        progressDialog.dismiss();
    });
}).start();
```

Netzwerkbezogene Permissons

Sollte die Android-Applikation mit dem Internet kommunizieren, z.b., mittels HTTP, dann muss dieser Zugriff explizit im `AndroidManifest.xml` erlaubt werden.

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Sollte die Android-Applikation auf den aktuellen Netzwerkstatus zugreifen, dann muss auch dieser Zugriff explizit erlaubt werden.

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

BroadcastReceiver

BroadcastReceiver erlauben das Empfangen von speziellen Events, die vom System an die Applikation gesendet werden. Ein Beispiel dafür ist die Statusänderung des Netzwerkes oder eine eingehende SMS. Um solche Events abzufangen, müssen diese zuerst registriert werden. Dafür kann eine Klasse vom `BroadcastReceiver` abgeleitet und eine Instanz davon mittels `registerReceiver` bzw. `unregisterReceiver` innerhalb einer `Activity` registriert oder auch wieder entfernt werden. Das erste

Beispiel zeigt, wie ein `BroadcastReceiver` erstellt wird und im zweiten Beispiel wird eine Instanz davon registriert bzw. wieder entfernt.

```
class NetworkStateReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        if (isConnected(context)) {
            // do something here if we are connected
        }
    }

    private boolean isConnected(Context context) {
        ConnectivityManager cm = (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);

        NetworkInfo networkInfo = cm.getActiveNetworkInfo();

        return (networkInfo != null && networkInfo.isConnected());
    }
}
```

```
@Override
protected void onResume() {
    super.onResume();
    IntentFilter intentFilter = new IntentFilter();
    intentFilter.addAction("android.net.conn.CONNECTIVITY_CHANGE");

    registerReceiver(networkStateReceiver, intentFilter);
}

@Override
protected void onPause() {
    super.onPause();
    unregisterReceiver(networkStateReceiver);
}
```