**Android Threading**

يحتاج كل مطور Android في وقت أو آخر إلى التعامل مع threads في تطبيقاتهم.

عندما يتم تشغيل تطبيق في Android فإنه يُنشئ thread الأولى والمعروفة باسم thread المحادثات "الرئيسية".

threadالرئيسي مسؤول عن إرسال الأحداث إلى أدوات واجهة المستخدم المناسبة بالإضافة إلى التواصل مع المكونات من مجموعة أدوات واجهة مستخدم Android.

للحفاظ على استجابة تطبيقك من الضروري تجنب استخدام threadالرئيسي لإجراء أي عملية قد تنتهي بإبقائه محظورًا/محجوزاَ.

تعد عمليات الاتصال بالإنترنت واستدعاءات قواعد البيانات بالإضافة إلى تحميل مكونات معينة أمثلة شائعة للعمليات التي يجب تجنبها في threadالرئيسية.

عندما يتم استدعاؤها في threadالمحادثات الرئيسية يتم استدعاؤها بشكل متزامن مما يعني أن واجهة المستخدم ستظل غير مستجيبة تمامًا حتى تكتمل العملية.

لهذا السبب يتم إجراؤها عادةً في سلاسل منفصلة مما يتجنب بالتالي حظر واجهة المستخدم أثناء تنفيذها (على سبيل المثال يتم إجراؤها بشكل غير متزامن من واجهة المستخدم).

يوفر Android العديد من الطرق لإنشاء threadsالرسائل وإدارتها كما توجد العديد من مكتبات الجهات الخارجية التي تجعل إدارة threadsالرسائل أكثر متعة.

ومع ذلك مع وجود العديد من الأساليب المختلفة في متناول اليد قد يكون اختيار الطريقة الصحيحة أمرًا مربكًا للغاية.

**في الاندرويد ، يمكننا تصنيف جميع مكونات الترابط إلى فئتين أساسيتين:**

الخيوط المرفقة بنشاط : هذه الخيوط مرتبطة بدورة حياة النشاط ويتم إنهاؤها بمجرد تدمير النشاط.

الخيوط غير المرفقة بأي نشاط : يمكن أن تستمر سلاسل الرسائل هذه في العمل بعد انتهاء النشاط الذي تم إنتاجها منه.

**الخيوط المرفقة بنشاط**

1)AsyncTask

هو المكون الأساسي للتعامل مع المسارات في نظام التشغيل. إنه سهل الاستخدام ويمكن

أن يكون مفيدًا للسيناريوهات الأساسية.

ومع ذلك فإن عمله يقصر إذا كنت بحاجة إلى مهمتك المؤجلة لتتجاوز عمر النشاط. تجدر الإشارة إلى

أنه حتى شيء بسيط مثل تدوير الشاشة يمكن أن يتسبب في تدمير النشاط.

2)LOADERS

هي الحل للمشكلة ذكرناها. حيث يمكنها ان تتوقف تلقائيًا عند تدمير النشاط ، ويمكنها أيضًا

إعادة تشغيل نفسها بعد إعادة إنشاء النشاط.

هناك نوعان رئيسيان:

AsyncTaskLoader و CursorLoader

**الخيوط غير المرفقة بأي نشاط**

1)SERVICE

عبارة عن مكون مفيد لإجراء عمليات طويلة بدون أي واجهة مستخدم. تعمل في الخيط

الرئيسي لعملية الاستضافة الخاصة بها ؛ ولا تقوم بإنشاء مؤشر ترابط خاص بها ولا يتم

تشغيلها في عملية منفصلة ما لم تحدد خلاف ذلك.

ويتم إيقافها عند اكتمال عملها بشكل يدوي عن طريق استدعاء

stopSelf () - stopService ()

2)INTENTSERVICE

SERVICE تعمل مثل ال

ولكنها توقف نفسا تلقائياً بعد ان تكمل عملها.

ويتم استخدامها عادةً للمهام القصيرة .

**Threading Type**

يوجد سبعة أنواع رئيسية من الthreading فيandroid

حالة الاستخدام رقم 1: إجراء طلب عبر الشبكة دون الحاجة إلى استجابة من الخادم

حالة الاستخدام رقم 2: إجراء مكالمة عبر الشبكة والحصول على الاستجابة

حالة الاستخدام رقم 3: تسلسل مكالمات الشبكة

حالة الاستخدام رقم 4: تواصل معUI thread من threadأخرى مثال(ترغب في تحميل ملف وتحديث واجهة المستخدم بمجرد اكتمالها)

حالة الاستخدام رقم 5: اتصال ثنائي الاتجاه بين thread بناءَ على إجراءات المستخدم

لنفترض أنك تقوم ببناء مشغل وسائط وتريد أن يتمكن من متابعة تشغيل الموسيقى حتى عندما تكون شاشة التطبيق مغلقة في هذا السيناريو ستحتاج إلى أن تكون واجهة المستخدم قادرة على التواصل مع مؤشر ترابط الوسائط (على سبيل المثال التشغيل والإيقاف المؤقت والإجراءات الأخرى) وستريد أيضًا أن يقوم مؤشر ترابط الوسائط بتحديث واجهة المستخدم استنادًا إلى أحداث معينة(مثل الخطأ وحالة التخزين المؤقت ...)

حالة الاستخدام رقم 6: تنفيذ الإجراءات بالتوازي والحصول على النتائج

لنفترض أنك تنشئ تطبيقًا سياحيًا وتريد إظهار المعالم السياحية على خريطة تم جلبها من مصادر متعددة (موفرو بيانات مختلفون) نظرًا لأنه قد لا تكون جميع المصادر موثوقة فقد ترغب في تجاهل المصادر التي فشلت والاستمرار في عرض الخريطة على أي حال

لموازنة العملية يجب أن تتم كل مكالمة API في سلسلة رسائل مختلفة

استخدام الحالة رقم 7: الاستعلام عن قاعدة بيانات SQLite المحلية

**There’s no Silver Bullet Solution to Threading in Android**