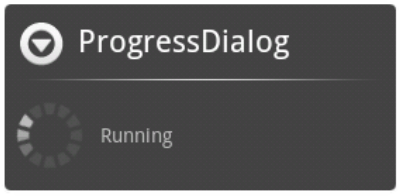
12 мая 2011 в 09:37

Заметки о ProgressDialog или как правильно показать прогресс выполнения

Разработка под Android*



Здравствуйте!

В данном посте я хочу поговорить о таком элементе UI Android как **ProgressDialog** и вообще о теме отображения прогресса в приложении, возможных реализациях и проблемах.

Под катом вас ждет некоторые мысли по теме + совсем немного кода. Наверно, тема, о которой я буду говорить, многим может показаться очевидной, но глядя на одни и те же «решения» в приложениях из Меркета, все видимо не так однозначно. Плюс, мне будет интересно услышать ваши соображения или best practice по теме.

Проблема или Решение №0

ProgressDialog — базовый элемент Android, является крайне популярным средством при необходимости отображения прогресса выполнения задачи. Кошерный пример реализации ProgressDialog + AsyncTask можно взять у хабрапользователя mike114 вот здесь.

Все было бы хорошо, но имеется один существенный недостаток: ProgressDialog **модальный**. Т.е. во время его работы пользователь не в состоянии взаимодействовать с UI. Да, иногда это обоснованно, особенно когда попросту нет альтернативных вариантов действия, например, при первоначальном логоне, но во многих ситуациях это не оправдано, т.к. заставляет пользователя ждать окончания операции, хотя он мог бы заняться чем-то более полезным.

Примеров множество, вот здесь хабрапользователь bugrimov использует ProgressDialog для отображения прогресса пока грузится контент с сервера. При этом, если по каким-то причинам, это происходит не так быстро, пользователь будет обязан ждать, пока ему что-то не вернется, хотя вместо этого он мог бы запустить несколько параллельных задач и пока грузятся одни читать уже загруженные.

Примечание: ProgressDialog коварный элемент, т.к. он настолько привычен и, как следствие, понятен пользователям Android, что они, скорее всего, простят вам его необоснованное использование.

Решение №1. Вообще ничего не показывать

Плюсы:

1. В результате такого решения пользователь никак не блокируется и волен делать что ему вздумается. Здесь хабрапользователь rude подобным образом загружает в фоне картинки для списка.

Минусы:

1. Пользователь все время находится в неведении, он не знает выполняется еще что-то или нет, а это может раздражать в достаточной степени.
2. Для этого и всех последующих решений без ProgressDialog'а, заключается в усложняющейся логике работы, т.к. теперь во время выполнения задачи пользователь может сделать куда больше действий чем при блокирующем ProgressDialog, то нужно будет это учитывать дополнительно, например, обрабатывать повторный запуск задачи до завершения выполнения такой же предыдущей.

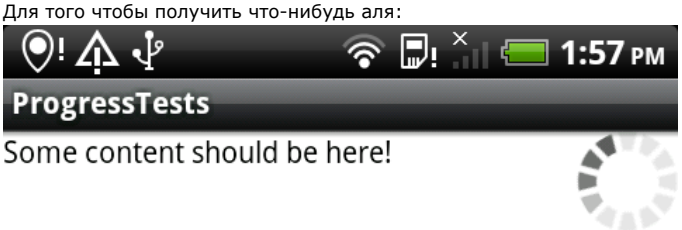
В некоторых ситуациях это может быть нормальным решением, например, если вы пишите свой ICQ-подобный клиент, то нет смысла показывать прогресс для загрузки статуса собеседника, это операция может быть прозрачной для пользователя.

Т.е. вывод, который можно сделать: не стоит доходить до идиотизма при отображении прогрессов, если информации о процессе выполнения операции пользователю не нужна, отображение прогресса можно опустить.

Решение №2. Использовать ProgressBar

ProgressBar — базовый компонент Android, позволяющий отображать прогресс выполнения прямо на View.

Для того чтобы получить что-нибудь аля:



Достаточно создать обычный xml ресурс и определить в нем ProgressBar:

```
<ProgressBar
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    style="?android:attr/progressBarStyleInverse"
    android:layout_alignParentRight="true"/>
```

Далее этот прогресс бар можно использовать, как все прочие компоненты внутри layout'ов:

```
...
progressBar = (ProgressBar) findViewById(R.id.progress_bar);
progressBar.setVisibility(View.VISIBLE);
...
```

Такое решение используется в Android Market, если во время установки/обновления перейти в список установленных приложений, то можно увидеть прогрессы выполнения текущих тасков.

Еще часто подобный подход применяют при загрузке множества однотипных элементов, сначала отображают пустые формы с ProgressBar'ом внутри, а далее по ходу загрузки наполняют их содержимым.

Плюсы:

1. Опять же в том, что не блокирует пользователя.
2. А также, в сравнении с первым решением, показывает прогресс, т.е. пользователь в курсе того, что что-то происходит.

Минусы:

1. Занимает место на экране, а на мобильных устройствах каждый «клочок» полезной поверхности ценен на вес золота. Его конечно можно скрывать когда прогресса нет, но тогда будет нарушаться логичность UI, в котором каждый элемент занимает свое место.
2. Еще один минус в позиционности, т.к. данный прогресс привязан к экрану, то не всегда очевидно, что делать если переходишь на другой экран, т.е. показывать что-то пользователю или нет, а главное где.
3. И еще минус в том, что положение прогресс бара подсознательно ассоциируется с элементом рядом с котором он расположен, т.е. нужно продумывать интерфейс еще внимательней, чтобы в итоге не было ложных предположений о том, что сейчас обновляется какой-то конкретный элемент, хотя на самом деле обновляется весь экран.
4. Общий минус всех не ProgressDialog решений: усложнение логики работы (см. Решение №1).

Решение №3. Использовать ProgressBar внутри Notification Area

Такое решение также используется в Android Market, когда пользователь запускает установку приложения, отдельный прогресс добавляется в Notification area, позволяя пользователю перейти туда в независимости от текущего положения.

Чтобы добавить ProgressBar в Notification Area необходимо для начала создать XML ресурс, например, такой:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="5dip"
    android:paddingRight="5dip">
    <TextView
        android:id="@+id/text_progress"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:textColor="@android:color/black"
        android:textStyle="bold" />
    <ProgressBar
        android:id="@+id/status_progress"
        android:layout_width="fill_parent"
        android:layout_height="20dip"
        android:layout_centerVertical="true"
        android:progressDrawable="@android:drawable/progress_horizontal"
        android:indeterminate="false"
        android:indeterminateOnly="false" />
</RelativeLayout>
```

Далее нужно добавить новый layout в Notification Area:

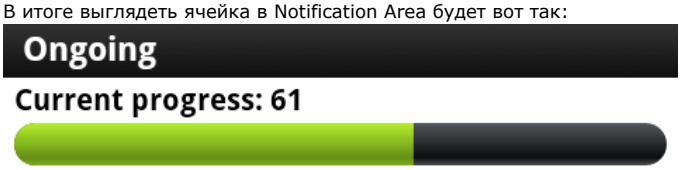
```
final Intent intent = new Intent(this, Main.class);
final PendingIntent contentIntent = PendingIntent.getActivity(getApplicationContext(), 0, intent, 0);
// Текст и иконка в Notification area
final Notification notification = new Notification(R.drawable.icon, "text in notification area", System.currentTimeMillis());
notification.flags = notification.flags | Notification.FLAG_ONGOING_EVENT;
// Связываем Notification View с нашем ресурсом
notification.contentView = new RemoteViews(getApplicationContext().getPackageName(), R.layout.progress_bar);
notification.contentIntent = contentIntent;
// Задаем начальное значение нашего Progress Bar
notification.contentView.setProgressBar(R.id.status_progress, 100, 0, false);

final NotificationManager notificationManager = (NotificationManager) getApplicationContext().getSystemService(Context.NOTIFICATION_SE
RVICE);
// Посылаем уведомление нашему notification'у
```

```
notificationManager.notify(ID, notification);
```

Для работы с созданным прогресс баром можно использовать следующий код:

```
// Изменяем текущую позицию на i
notification.contentView.setProgressBar(R.id.status_progress, 100, i, false);
// Задаем текст
notification.contentView.setTextViewText(R.id.text_progress, "Current progress: " + Integer.toString(i));
// Уведомляем об изменении
notificationManager.notify(ID, notification);
```

В итоге выглядеть ячейка в Notification Area будет вот так:



Плюсы:

1. Опять не блокируем пользователя.
2. Как и в решении №2 прогресс показывается и пользователь в курсе что что-то происходит.
3. Данное решение не привязана к окнам программы, т.е. на него можно перейти из любого окна при этом сами окна изменять не требуется.
4. Не требуется дополнительное место в окне для отображения прогресса.

Минусы:

1. После завершения отображения нотификации в Notification Area (не завершении ее самой, а только сообщения о то, что она появилась) не виден прогресс, т.е. пользователь не знает происходит что-то или нет. Это значит, что ему придется периодически заглядывать туда, что может быть не очень удобно.
2. Данное решение не является интуитивно понятным в достаточной степени.
3. Общий минус всех не ProgressDialog решений: усложнение логики работы (см. Решение №1).

Решение №4. Использовать ProgressBar внутри Title Bar'a приложения

Данное решение применяется во многих продуктах, например, в **gReader**.

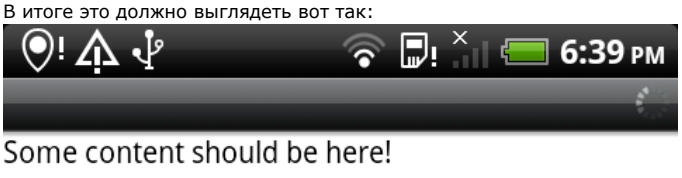
Для реализации необходимо создать layout с содержимым, в моем случае это только ProgressBar:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout     xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <ProgressBar
        android:id="@+id/status_progress"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        style="?android:attr/progressBarStyleSmallTitle"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"
        android:indeterminateOnly="true"
        android:visibility="invisible"/>
</RelativeLayout>
```

И в методе **onCreate** кастомезировать имеющийся title bar нашим layout'ом:

```
Window window = getWindow();
window.requestFeature(Window.FEATURE_CUSTOM_TITLE);
setContentView(R.layout.main);
// Устанавливаем наш layout в качестве основы title bar'a
window.setFeatureInt(Window.FEATURE_CUSTOM_TITLE, R.layout.progress_bar);

// Работаем так же как и с обычным ProgressBar'ом
final ProgressBar progressBar = (ProgressBar) findViewById(R.id.status_progress);
progressBar.setVisibility(View.VISIBLE);
```

В итоге это должно выглядеть вот так:



Данный пример рассчитан на задние произвольного пользовательского layout'a, т.е. в title bar пойдет все содержимое определенное в этом layout'e, т.е. не только ProgressBar, если же вам нужен только он, то для этого существует более простая возможность:

```
requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);
setContentView(R.layout.main);
setProgressBarIndeterminateVisibility(true); // Или false, чтобы скрыть
```

Плюсы:

1. Опять не блокируем пользователя.
2. В отличии от решения №3 прогресс всегда заметен.
3. Как в решениях №2 и №3 прогресс показывается и пользователь в курсе что что-то происходит.
4. Не требуется дополнительное место в окне для отображения прогресса.

Минусы:

1. Прогресс привязан к окну, и если потребуется его показывать между разными окнами это нужно будет предусмотреть отдельно.
2. Title bar уступает по высоте Notification area и поэтому в него можно вместить меньше информации, иначе она станет плохо различимой.
3. Подобное решение подходит только для title'ов со стандартным набором стилей. Т.е. на Activity внутри TabHost его приделать не получится.
4. Общий минус всех не ProgressDialog решений: усложнение логики работы (см. Решение №1).

Решение №5. Использовать все решения

По моему мнению, это самое лучшее решение!

Вы не обязаны использовать только что-то одно, все зависит от решаемой задачи, как банально это и звучит. Иногда лучше всего будет ничего не показывать, иногда ProgressDialog, иногда вариации ProgressBar'a, а иногда еще что-то. Повторюсь, **все зависит от задачи**.

Вместо заключения

Если вы принимаете какое-то решение, то не исходите из соображений простоты разработки и не ограничивайтесь известными вам решениями, учитесь. Подумайте о пользователях, о том как им было бы удобнее и понятнее, а они это обязательно оценят.

Если вы используете решения отличные от приведенных здесь, пожалуйста, расскажите о них, я думаю многим Android-разработчикам будут интересны подобные примеры.

android, development, ProgressDialog, ProgressBar

Комментарии (16)

dzigoro, 12 мая 2011 в 10:40 # +2

Хорошая статья, спасибо.
Единственное, я бы отметил, что код решений номер 2 и 4 надо запускать из Activity, а вот номер 3 — из Service.

Ну и да, `Main.class` — имя ни о чем не говорит. Activity это или Service — не понятно)

NevRA, 12 мая 2011 в 11:08 # ↑ +1

В примерах Main везде одинаковый, т.е. всегда запускается из Activity.

Hoorsh, 12 мая 2011 в 11:12 # +1

Расскажите, если знаете как сделать анимированный Progress в виджете :)

w32blaster, 12 мая 2011 в 11:23 # ↑ +2

Присоединяюсь. Я тоже сделал анимированный прелоадер в виджете, а он не анимированный :(Гуглил много, так пришёл к выводу, что в виджете вообще не поддерживается никакая анимация. Кто-то может что-то сказать и посоветовать по этому поводу?

w32blaster, 12 мая 2011 в 11:22 # +1

«то можно увидеть прогрессы выполнения текущих тасок.»

Тасков, наверно?

И да, статья приятная к прочтению. Совсем недавно задумывался на эту тему, тоже пришёл к тем же мыслям.

NevRA, 12 мая 2011 в 11:56 # 0

Да, исправил.

Fury, 12 мая 2011 в 11:28 # +2

Вывод довольно логичный :)
Действительно, каждый вид диалогов подходит для своего круга задач.
Сам использую 0й вариант для блокирующих операций (логин, операции с деньгами) и 4 для остальных. Когда придётся работать с большими объёмами, то, думаю, воспользуюсь 3м вариантом.

Кстати про проблему 4го варианта:

> 3. Подобное решение подходит только для title'ов со стандартным набором стилей. Т.е. на Activity внутри TabHost его приделать не получится.

У меня все активити наследуются от базовой AbstractActivity, в которой все основные методы. А для тех активити, который могут запускаться внутри табов, есть ещё AbstractTabbedActivity, которая наследуется от AbstractActivity и как раз таки получает доступ к прогрессбару (да и вообще всему, что расположено в title) через родительскую активити — получается, что для конечной активити вообще неважно, в табе она запустилась или нет.

NevRA, 12 мая 2011 в 11:59 # 0

Спасибо за информацию, не знал.

Fury, 12 мая 2011 в 14:28 # 0

Да не за что. Я сам потратил довольно много времени, пытаясь решить проблему доступа из активити внутри TabHost к экшенбару, который находится снаружи TabHost'a, прежде, чем узнал про метод getParent(). Даже была мысль в Intent каждой активити в табах положить в экстра сам экшенбар :) Но это невозможно, так как View не Serializable и не Parcelable.

caezar, 12 мая 2011 в 15:53 # 0

4е решение можно реализовать значительно проще, так как это «стандартная возможность»
сначала
```
requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);
```
потом
```
setProgressBarIndeterminateVisibility(true);
setProgressBarIndeterminateVisibility(false);
```

caezar, 12 мая 2011 в 15:58 # +3

Ну или FEATURE_PROGRESS для детерминированного прогресса
P.S. очень полезный документ developer.android.com/resources/faq/commontasks.html

NevRA, 12 мая 2011 в 16:08 # 0

Да, вы правы, но я хотел показать общей случай, т.к. на пользовательском layout'e мог находиться не только ProgressBar, а и любые другие виджеты. Т.е. из приведенного примера можно понять, что в title может находится произвольное содержимое (т.к. указываем пользовательский layout). Но если нужно добавить только Progress тогда ваш пример самое то.

caezar, 12 мая 2011 в 16:09 # +1

Стоит добавить в статью, а то побегут копипастить сложное не зная о существовании простого ;)

NevRA, 12 мая 2011 в 16:20 # 0

:) Добавил

Alexx_ps, 13 мая 2011 в 10:27 # 0

Мне кажется или в любом решении минусов больше, чем плюсов? В 3-м усложнение логики работы можно считать за 2 минуса.

NevRA, 13 мая 2011 в 11:01 # 0

Усложнение логики работы есть во всех решениях кроме нулевого.) Логика имелась в виде программная, про это написано в решении №1.
В зависимости от задачи вес минуса не будет равен весу плюса, т.е. если, например, вы хотите написать крайне дружелюбное приложение вам может быть все равно насколько там, что усложнилось и один плюс перекроет все минусы.