УДК 004

Создание виджета «генератор случайных чисел» для Android смартфона

Семченко Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассмотрена возможность создания виджета, который генерирует случайное число при действии пользователя, либо обновляясь через определенный промежуток времени. Практическим результатом является рабочий виджет
Ключевые слова: Джава, виджет, андроид, генератор

Creating a random number generator widget for an Android smartphone

Semchenko Regina Viktorovna
Sholom-Aleichem Priamursky State University
Student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article discusses the possibility of creating a widget that generates a random number when a user acts, or updated after a certain period of time. The bottom line is a working widget
Keywords: Java, widget, android, generator

Виджеты приложений можно рассматривать как небольшое окно или контроллер для приложения Android, которое можно встроить в другое приложение (например, на домашний экран). Они могут быть очень полезны, позволяя пользователям просматривать или контролировать приложение, не запуская его. Например, пропуск треков с помощью виджета музыкального проигрывателя или просмотр информации о погоде. Отличительной особенностью виджетов является то, что они могут обновляться автоматически (после определенного периода времени) или в ответ на действия пользователя.

Цель данной статьи создание собственного виджета, который генерирует каждый определенный промежуток времени случайное число, либо на действие пользователя.

Исследованиями в области разработки мобильных приложений занимались многие российские и зарубежные исследователи. Е.Л. Касьянова, П.М. Кикин [1] рассмотрели основные способы разработки приложений для мобильных устройств, провели обзор отличительных особенностей разработки картографических мобильных приложений и выявили проблемы, возникшие в процессе их создания. Н.М.Виштак, В.А.Дорожкин [2] рассмотрели основные требования к средствам разработки, а также определили наиболее подходящую платформу для разработки мобильных приложений дополненной реальности. П.И.Свентицкий, Н.А.Иванова в своей работе [3] провели анализ основных технологий кроссплатформенной разработки приложений для мобильных устройств. В.Ю. Ким [4] рассмотрел ключевые особенности создания дизайна пользовательского интерфейса мобильного приложения. К.В. Аксенов в своей работе [5] рассмотрел наиболее популярные среды разработки мобильных приложений под операционные системы Android, iOS и Windows Phone, а также привел их краткие характеристики, преимущества и недостатки. Так же обзор на поисковую оптимизацию веб-сайтов и фактов, доказывающих ее эффективность для повышения конкурентоспособности бизнеса рассмотрели Овчинников С.А. и Белков С.В.[4]. Якименко А.Н. и Костромицкий А.И. рассмотрели вопросы, связанные с возможностью получения доходов, используя глобальную информационную сеть[5].

Первое, что придется сделать, это создать макет виджета. Хотя размещение виджета приложения аналогично планированию действия или фрагмента, следует отметить очень важный фактор, макеты виджетов приложений основаны на макетах «RemoteViews». Это означает, что не все подклассы «View» могут использоваться в виджете. Фактически единственными поддерживаемыми классами являются «FrameLayout, LinearLayout, RelativeLayout, GridLayout, AnalogClock, Button, Chronometer, ImageButton, ImageView, ProgressBar, TextView, ViewFlipper, ListView, GridView, StackView и AdapterViewFlipper». Подклассы и их потомки даже не поддерживаются.

Помня об этом, создаем макет виджета с именем simple_widget.xml (рис.1).

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="@dimen/widget_margin"
    android:background="#55000000">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:src="@drawable/aa"/>

    <TextView
        android:id="@+id/textView"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:gravity="center"
        android:text="000"
        android:textSize="@dimen/abc_text_size_large_material"
        android:textStyle="bold"/>

    <Button
        android:id="@+id/actionButton"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:text="Refresh"/>

</LinearLayout>
```

Рисунок 1 – шаблон виджета

Теперь расширьте «AppWidgetProvider», создав класс «SimpleWidgetProvider». «AppWidgetProvider» имеет методы, которые вызываются, когда виджет приложения обновляется, удаляется, включается и отключается среди других. Для реализации переопределяем только «onUpdate ()», потому что это метод, вызываемый всякий раз, когда виджет добавляется к хосту (рис.2).

```
public class SimpleWidgetProvider extends AppWidgetProvider {

    @Override
    public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[] appWidgetIds) {
        final int count = appWidgetIds.length;

        for (int i = 0; i < count; i++) {
            int widgetId = appWidgetIds[i];
            String number = String.format("%03d", (new Random().nextInt(900
new RemoteViews(context.getPackageName(),
                    R.layout.simple_widget);
            remoteViews.setTextViewText(R.id.textView, number);

            Intent intent = new Intent(context, SimpleWidgetProvider.class);
            intent.setAction(AppWidgetManager.ACTION_APPWIDGET_UPDATE);
            intent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS, appWidgetIds);
            PendingIntent pendingIntent = PendingIntent.getBroadcast(context,
                    0, intent, PendingIntent.FLAG_UPDATE_CURRENT);
            remoteViews.setOnClickPendingIntent(R.id.actionButton, pendingIntent);
            appWidgetManager.updateAppWidget(widgetId, remoteViews);
        }
    }
}
```

Рисунок 2 – добавление классов

В методе «onUpdate ()» перебираются все виджеты, в случае, если пользователь разместил несколько виджетов, получаем объект «RemoteViews», обновляем текстовое представление новым случайным числом от 100 до 999, а затем определяем действие, это должно произойти, когда кнопка нажата.

Чтобы запросить обновление вручную при нажатии кнопки обновления, используем «PendingIntent». Действие для «Intent» установлено в «AppWidgetManager.ACTION_APPWIDGET_UPDATE». Это то же самое действие, которое отправляет система, когда виджет должен обновляться автоматически. Также указываем виджеты, которые должны быть обновлены.

Большинство атрибутов имеют довольно понятные имена, «minWidth» и «minHeight» указывают минимальную ширину и высоту, которые может иметь виджет, «updatePeriodMillis» указывает частоту обновления виджета в миллисекундах. обратите внимание, что частые обновления будут существенно влиять на батарею пользователя, а так же на атрибут «widgetCategory», он указывает, может ли виджет быть доступен на экране блокировки, а также на домашнем экране. Все виджеты доступны на главном экране по умолчанию, и если они не указаны. В Android 4.2 включена опция «keyguard», указывающая, что виджет можно добавить на экран блокировки.

Приведенный ниже пример кода проверяет «AppWidgetHost» и отображает разную компоновку для каждого типа хоста (рис.3).

```
AppWidgetManager appWidgetManager;
int widgetId;
Bundle myOptions = appWidgetManager.getAppWidgetOptions (widgetId);


int category = myOptions.getInt(AppWidgetManager.OPTION_APPWIDGET_HOST_CATEGORY, -1);


boolean isKeyguard = category == AppWidgetProviderInfo.WIDGET_CATEGORY_KEYGUARD;

int baseLayout = isKeyguard ? R.layout.keyguard_widget_layout : R.layout.widget_layout;
```

Рисунок 3 – Проверка атрибута

Последний шаг - добавить виджет приложения в манифест приложения. В теги элемента «<application> </ application>» необходимо добавить код представленный ниже (рис.4)

```
<receiver android:name="SimpleWidgetProvider" >
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
    </intent-filter>
    <meta-data android:name="android.appwidget.provider"
        android:resource="@xml/simple_widget_info" />
</receiver>
```

Рисунок 4 – добавление разрешений

Для работоспособности необходимо поменять получатель «android: name» на реализацию «AppWidgetProvider», метаданные «android:» на ресурс для xml-файла «AppWidgetProviderInfo». На этом этапе уже можно запустить приложение и разместить виджет либо на домашнем экране, либо на экране блокировки.

Практическим результатом является рабочий мобильный виджет, генерирующий случайное число, через определенный промежуток времени, либо при нажатии кнопки пользователем.

Библиографический список

1. Касьянова Е.Л., Кикин П.М., Грищенко Д.В. Разработка картографических приложений для мобильных устройств // Интерэкспо гео-сибирь. 2015. №7. С. 75-78.
2. Виштак Н.М., Дорожкин В.А. Средства разработки мобильных приложений дополненной реальности // Инновации в науке. 2015. №6 (43). С. 1-6.
3. Свентицкий П.И., Иванова Н.А. Инструменты кроссплатформенной разработки мобильных приложений // Инновации в науке. 2014. №40. С. 1-5.
4. Ким В.Ю. Особенности разработки дизайна пользовательского интерфейса для мобильного приложения // Новые информационные

технологии в автоматизированных системах. 2015. №18. С. 479-481.
5. Аксенов К.В. Обзор современных средств для разработки мобильных приложений // Новые информационные технологии в автоматизированных системах. 2014. №17. С. 508-513.