TUTORIAL 1: App que recibe notificaciones periodicamente

7 JULIO, 2016 / ALVARIITO55

Buscar … BUSCAR

ENTRADAS RECIENTES

Tutorial 3: Notificaciones en Android Wear

Este es un tutorial que permite a tu aplicación crear un servicio y que haga ciertas comprobaciones, ya sea con la app funcionando o sin funcionar, como por ejemplo comprobar si hay mensajes nuevos y a raíz de que los haya mandar una notificación.

Se deben seguir todos los pasos que voy a describir a continuación.

Se deberán añadir las siguientes 4 clases sin excepción:

SunshineSyncAdapter

```
public class SunshineSyncAdapter extends AbstractThrea
    public static final String LOG_TAG = SunshineSyncA

    // Interval at which to sync with the weather, in
    // 60 seconds (1 minute)  180 = 3 hours
    public static final int SYNC_INTERVAL = 60 * 180;
    public static final int SYNC_FLEXTIME = SYNC_INTER
    private static final long DAY_IN_MILLIS = 1000 * 6


    private NotificationManager notifyMgr;
    private Context context = null;
    private int id = 0;
```

Tutorial 2:
Obtener Localización

TUTORIAL 1: App que recibe
notificaciones periodicame

COMENTARIOS RECIENTES

ARCHIVOS

julio 2016

CATEGORÍAS

Sin categoría

```
    private int numeroActualMensajes = 0;

    public SunshineSyncAdapter(Context context, boolea
        super(context, autoInitialize);
        this.context = context;
        SharedPreferences pref = context.getSharedPref
        id = pref.getInt("id", 0);
        numeroActualMensajes = pref.getInt("numeroMens
    }


    //ESTE ES EL METODO QUE SE EJECUTA CADA VEZ QUE SE
    @Override
    public void onPerformSync(Account account, Bundle
        Log.d(LOG_TAG, "onPerformSync Called.");
        Log.d(LOG_TAG, "Starting sync");

        SharedPreferences pref = context.getSharedPref
        id = pref.getInt("id", 0);
        numeroActualMensajes = pref.getInt("numeroMens


        JSONObject datos = new JSONObject();

        String numeroMensajes = "0";

        try {
            datos.put("motivobtenerNumeroMensajes");
            datos.put("idththis.id);
```

```
            numeroMensajes =  new ConexionMensaje(cont

            //System.out.println("Elro de mensajes ini

        } catch (JSONException | InterruptedException
            e.printStackTrace();
        }

        int mensajesInt = Integer.parseInt(numeroMensa

        System.out.println("NumeroActualMensajes numer

        if (numeroActualMensajes < mensajesInt){
            int numeroMensajesNuevos = mensajesInt - n
            notifyMensaje(
                    1,
                    R.mipmap.fondo,
                    "PETPAD",
                    "¡Tiene " + numeroMensajesNuevos +
            );
        }

        return;
    }

    private void notifyMensaje(int id, int iconId, Str

        Context context = getContext();
```

```
        notifyMgr = (NotificationManager) context.getS

        NotificationCompat.Builder builder =
                (NotificationCompat.Builder) new Notif
                        .setSmallIcon(iconId)
                        .setLargeIcon(BitmapFactory.de
                                context.getResources()
                                R.mipmap.ic_launcher
                                )
                        )
                        .setContentTitle(titulo)
                        .setContentText(contenido);


        // Construir la notificación y emitirla
        notifyMgr.notify(id, builder.build());

    }

    /**
     * Helper method to schedule the sync adapter peri
     */
    public static void configurePeriodicSync(Context c
        Account account = getSyncAccount(context);
        String authority = context.getString(R.string.
        if (Build.VERSION.SDK_INT >= Build.VERSION_COD
            // we can enable inexact timers in our per
            SyncRequest request = new SyncRequest.Buil
```

```
                    syncPeriodic(syncInterval, flexTim
                    setSyncAdapter(account, authority)
                    setExtras(new Bundle()).build();
            ContentResolver.requestSync(request);
        } else {
            ContentResolver.addPeriodicSync(account,
                    authority, new Bundle(), syncInter
        }
    }

    /**
     * Helper method to have the sync adapter sync imm
     * @param context The context used to access the a
     */
    public static void syncImmediately(Context context
        Bundle bundle = new Bundle();
        bundle.putBoolean(ContentResolver.SYNC_EXTRAS_
        bundle.putBoolean(ContentResolver.SYNC_EXTRAS_
        ContentResolver.requestSync(getSyncAccount(con
                context.getString(R.string.content_aut
    }

    /**
     * Helper method to get the fake account to be use
     * if the fake account doesn't exist yet.  If we m
     * onAccountCreated method so we can initialize th
     *
     * @param context The context used to access the a
     * @return a fake account.
```

```
     */
    public static Account getSyncAccount(Context conte
        // Get an instance of the Android account mana
        AccountManager accountManager = (AccountManage

        // Create the account type and default account
        Account newAccount = new Account(
                context.getString(R.string.app_name),



        // If the password doesn't exist, the account
        if ( null == accountManager.getPassword(newAcc

        /*
         * Add the account and account type, no passwo
         * If successful, return the Account object, c
         */

            if (!accountManager.addAccountExplicitly(n
                return null;
            }
            /*
             * If you don't set android:syncable="true
             * in your <provider> element in the manif
             * then call ContentResolver.setIsSyncable
             * here.
             */
            onAccountCreated(newAccount, context);
        }
```

```
        return newAccount;
    }


    private static void onAccountCreated(Account newAc
        /*
         * Since we've created an account
         */
        SunshineSyncAdapter.configurePeriodicSync(cont

        /*
         * Without calling setSyncAutomatically, our p
         */
        ContentResolver.setSyncAutomatically(newAccoun
       /*
        * Finally, let's do a sync to get things start
        */
        syncImmediately(context);
    }

    public static void initializeSyncAdapter(Context c
        getSyncAccount(context);
    }
}
```

SunshineSyncService

```
public class SunshineSyncService extends Service {
    private static final Object sSyncAdapterLock = new
    private static SunshineSyncAdapter sSunshineSyncAd

    @Override
    public void onCreate() {
        Log.d("SunshineSyncService", "onCreate - Sunsh
        synchronized (sSyncAdapterLock) {
            if (sSunshineSyncAdapter == null) {
                sSunshineSyncAdapter = new SunshineSyn
            }
        }
        System.out.println("SERVICIODO");
    }

    @Override
    public IBinder onBind(Intent intent) {
        return sSunshineSyncAdapter.getSyncAdapterBind
    }
}
```

SunshineAuthenticator

```
public class SunshineAuthenticator extends AbstractAcc

    public SunshineAuthenticator(Context context) {
        super(context);
```

```
    }

    // No properties to edit.
    @Override
    public Bundle editProperties(
            AccountAuthenticatorResponse r, String s)
        throw new UnsupportedOperationException();
    }

    // Because we're not actually adding an account to
    @Override
    public Bundle addAccount(
            AccountAuthenticatorResponse r,
            String s,
            String s2,
            String[] strings,
            Bundle bundle) throws NetworkErrorExceptio
        return null;
    }

    // Ignore attempts to confirm credentials
    @Override
    public Bundle confirmCredentials(
            AccountAuthenticatorResponse r,
            Account account,
            Bundle bundle) throws NetworkErrorExceptio
        return null;
    }
```

```
    // Getting an authentication token is not supporte
    @Override
    public Bundle getAuthToken(
            AccountAuthenticatorResponse r,
            Account account,
            String s,
            Bundle bundle) throws NetworkErrorExceptic
        throw new UnsupportedOperationException();
    }

    // Getting a label for the auth token is not suppo
    @Override
    public String getAuthTokenLabel(String s) {
        throw new UnsupportedOperationException();
    }

    // Updating user credentials is not supported
    @Override
    public Bundle updateCredentials(
            AccountAuthenticatorResponse r,
            Account account,
            String s, Bundle bundle) throws NetworkErr
        throw new UnsupportedOperationException();
    }

    // Checking features for the account is not suppor
    @Override
    public Bundle hasFeatures(
            AccountAuthenticatorResponse r,
```

```
            Account account, String[] strings) throws
        throw new UnsupportedOperationException();
    }
}
```

SunshineAuthenticatorService

```
public class SunshineAuthenticatorService extends Serv
    // Instance field that stores the authenticator ob
    private SunshineAuthenticator mAuthenticator;

    @Override
    public void onCreate() {
        // Create a new authenticator object
        mAuthenticator = new SunshineAuthenticator(thi
    }

    /*
     * When the system binds to this Service to make t
     * return the authenticator's IBinder.
     */
    @Override
    public IBinder onBind(Intent intent) {
        return mAuthenticator.getIBinder();
    }
}
```

Repito que se deben añadir las 4 clases, puesto que si alguna de ellas no se añade no se podrá crear la cuenta virtual que exige google para crear estos servicios.

También se necesitará crear un proveedor de contenidos que se hará creando la siguiente clase, yo la tengo creada aunque no le doy ninguna utilidad:

PetPadProvider

```
public class PetPadProvider extends ContentProvider {

    /*
        Students: We've coded this for you.  We just c
        here.
     */
    @Override
    public boolean onCreate() {
        return true;
    }

    @Nullable
    @Override
```

```
    public Cursor query(Uri uri, String[] projection,
        return null;
    }

    @Nullable
    @Override
    public String getType(Uri uri) {
        return null;
    }

    @Nullable
    @Override
    public Uri insert(Uri uri, ContentValues values) {
        return null;
    }

    @Override
    public int delete(Uri uri, String selection, Strin
        return 0;
    }

    @Override
    public int update(Uri uri, ContentValues values, S
        return 0;
    }

    // You do not need to call this method. This is a
    // framework in running smoothly. You can read mor
    // http://developer.android.com/reference/android/
```

```
    @Override
    @TargetApi(11)
    public void shutdown() {

        super.shutdown();
    }
}
```

Una vez que estén añadidas las 4 clases y el contentProvider, se debe ir al archivo AndroidManifest y añadir los siguiente:

En la parte superior:

```
<uses-permission android:name="android.permission.READ
<uses-permission android:name="android.permission.WRIT
<uses-permission android:name="android.permission.AUTH
```

y en la parte posterior antes de salir de la etiqueta de </aplicacion>

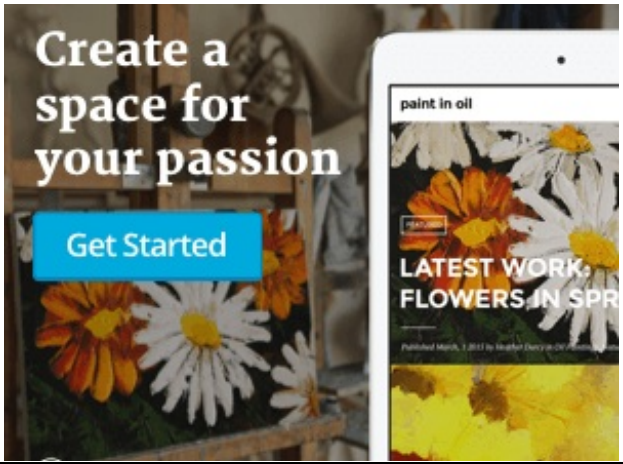
```
<provider
    android:authorities="@string/content_authority"
```

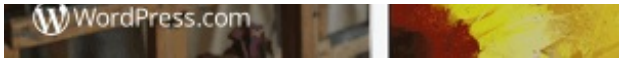
```
    android:name=".PetPadProvider"
    android:enabled="true"
    android:exported="true"
    android:syncable="true"
    android:permission="com.example.android.sunshine.a

<!-- SyncAdapter's dummy authentication service -->
<service android:name=".servicios.SunshineAuthenticato
    <intent-filter>
        <action android:name="android.accounts.Account
    </intent-filter>
    <meta-data
        android:name="android.accounts.AccountAuthenti
        android:resource="@xml/authenticator" />
</service>

<!-- SyncAdapter service -->
<service
    android:name=".servicios.SunshineSyncService"
    android:exported="true">
    <intent-filter>
        <action android:name="android.content.SyncAdap
    </intent-filter>
    <meta-data
        android:name="android.content.SyncAdapter"
        android:resource="@xml/syncadapter" />
</service>
```

Por último habrá que dirigirse al mainActivity de nuestra app y añadir la siguiente línea en el onCreate para iniciar el servicio y que este haga las comprobaciones cada 3 horas en nuestro caso, que fue el tiempo que le pusimos en la primera clase creada del tutorial:

```
SunshineSyncAdapter.initializeSyncAdapter(this);
```

Con esto ya debería de empezar nuestro servicio a funcionar y permitiría hacer comprobaciones sobre algo cada el tiempo que queramos dependiendo de la importancia.





Comparte:

Twitter Facebook Google

★ Me gusta

Sé el primero en decir que te gusta.

Sin categoría

Tutorial 2:
Obtener Localización →

Deja una respuesta

Introduce aquí tu comentario...

Buscar ... BUSCAR

ENTRADAS RECIENTES

Tutorial 3: Notificaciones en Android Wear

Tutorial 2: Obtener Localización

TUTORIAL 1: App que recibe notificaciones periodicamente

COMENTARIOS RECIENTES

ARCHIVOS

julio 2016

CATEGORÍAS

Sin categoría