**Almacenamiento de Datos.**

**Vista rápida de almacenamiento**

* Utilice las preferencias compartidas para datos primitivos
* Utilice un dispositivo de almacenamiento interno para los datos privados
* El uso de almacenamiento externo para grandes conjuntos de datos que no son privadas
* Usar bases de datos SQLite para el almacenamiento estructurado

**En este documento**

1. [Uso de las preferencias compartidas](http://developer.android.com/guide/topics/data/data-storage.html#pref)
2. [Utilizando el almacenamiento interno](http://developer.android.com/guide/topics/data/data-storage.html#filesInternal)
	1. [Almacenamiento de archivos de caché](http://developer.android.com/guide/topics/data/data-storage.html#InternalCache)
	2. [Otros métodos útiles](http://developer.android.com/guide/topics/data/data-storage.html#InternalMethods)
3. [Utilizando el almacenamiento externo](http://developer.android.com/guide/topics/data/data-storage.html#filesExternal)
	1. [Comprobación de disponibilidad de los medios de comunicación](http://developer.android.com/guide/topics/data/data-storage.html#MediaAvail)
	2. [Acceso a los archivos de almacenamiento externo](http://developer.android.com/guide/topics/data/data-storage.html#AccessingExtFiles)
	3. [Almacenamiento de archivos que deben ser compartidos](http://developer.android.com/guide/topics/data/data-storage.html#SavingSharedFiles)
	4. [Almacenamiento de archivos de caché](http://developer.android.com/guide/topics/data/data-storage.html#ExternalCache)
4. [Uso de bases de datos](http://developer.android.com/guide/topics/data/data-storage.html#db)
	1. [Base de datos de depuración](http://developer.android.com/guide/topics/data/data-storage.html#dbDebugging)
5. [Utilizando una conexión de red](http://developer.android.com/guide/topics/data/data-storage.html#netw)

**Véase también**

1. [Proveedores de contenido y resolución de Contenido](http://developer.android.com/guide/topics/data/data-storage.html#pref)

Android ofrece varias opciones para guardar datos persistentes de aplicaciones. La solución que elija dependerá de sus necesidades específicas, como por ejemplo si los datos deberían ser privados de su aplicación o acceder a otras aplicaciones (y el usuario) y la cantidad de espacio de los datos requiere.

Sus opciones de almacenamiento de datos son los siguientes:

[Preferencias compartidas](http://developer.android.com/guide/topics/data/data-storage.html#pref)

Guarde los datos privados primitivos en pares clave-valor.

[almacenamiento interno](http://developer.android.com/guide/topics/data/data-storage.html#filesInternal)

Guarde los datos privados en la memoria del dispositivo.

[Almacenamiento externo](http://developer.android.com/guide/topics/data/data-storage.html#filesExternal)

Guarde los datos públicos sobre el almacenamiento externo compartido.

[Bases de datos SQLite](http://developer.android.com/guide/topics/data/data-storage.html#db)

Guarde los datos estructurados en una base de datos privada.

[conectividad de red](http://developer.android.com/guide/topics/data/data-storage.html#netw)

Almacenar datos en la web con el servidor de red propia.

Android proporciona una manera para que usted pueda exponer hasta a sus datos privados a otras aplicaciones - con un [proveedor de contenido](http://developer.android.com/guide/topics/providers/content-providers.html) . Un proveedor de contenido es un componente opcional que expone de lectura / escritura a los datos de aplicación, con sujeción a las restricciones que quiere imponer. Para obtener más información sobre el uso de los proveedores de contenido, vea el [contenido de los proveedores de](http://developer.android.com/guide/topics/providers/content-providers.html) la documentación.

**Uso de las preferencias compartidas**

El [SharedPreferences](http://developer.android.com/reference/android/content/SharedPreferences.html) clase proporciona un marco general que le permite guardar y recuperar la persistencia de pares clave-valor de los tipos de datos primitivos. Usted puede utilizar [SharedPreferences](http://developer.android.com/reference/android/content/SharedPreferences.html) de guardar los datos primitivos: booleanos, flotadores, ints, largos, y cuerdas. Estos datos se conserven de una sesión de usuario (incluso si la aplicación está muerto).

**Preferencias de usuario**

Preferencias compartidas no son estrictamente para salvar "las preferencias del usuario", como lo tono de llamada de un usuario ha elegido. Si usted está interesado en la creación de las preferencias del usuario para su aplicación, consulte [PreferenceActivity](http://developer.android.com/reference/android/preference/PreferenceActivity.html) , que proporciona un marco de actividades para crear las preferencias del usuario, que será automáticamente persistido (con las preferencias compartidas).

Para obtener una [SharedPreferences](http://developer.android.com/reference/android/content/SharedPreferences.html) objeto para su aplicación, use uno de dos métodos:

* [getSharedPreferences()](http://developer.android.com/reference/android/content/Context.html#getSharedPreferences%28java.lang.String,%20int%29) : utilice esta opción si necesita los archivos de preferencias múltiples identificados por su nombre, que se especifica con el primer parámetro.
* [getPreferences()](http://developer.android.com/reference/android/app/Activity.html#getPreferences%28int%29) - Utilice esta opción si sólo tiene un archivo de preferencias para su actividad. Debido a que este será el archivo de preferencias sólo para su actividad, no se proporciona un nombre.

Para escribir valores:

1. Llame a [edit()](http://developer.android.com/reference/android/content/SharedPreferences.html#edit%28%29) para obtener un [SharedPreferences.](http://developer.android.com/reference/android/content/SharedPreferences.Editor.html)Editor
2. Añadir valores con métodos tales como [putBoolean()](http://developer.android.com/reference/android/content/SharedPreferences.Editor.html#putBoolean%28java.lang.String,%20boolean%29) y [putString()](http://developer.android.com/reference/android/content/SharedPreferences.Editor.html#putString%28java.lang.String,%20java.lang.String%29) .
3. Confirme los nuevos valores con [commit()](http://developer.android.com/reference/android/content/SharedPreferences.Editor.html#commit%28%29)

Para leer los valores, utilice [SharedPreferences](http://developer.android.com/reference/android/content/SharedPreferences.html) métodos como [getBoolean()](http://developer.android.com/reference/android/content/SharedPreferences.html#getBoolean%28java.lang.String,%20boolean%29) y [getString()](http://developer.android.com/reference/android/content/SharedPreferences.html#getString%28java.lang.String,%20java.lang.String%29) .

Aquí hay un ejemplo que guarda una preferencia por el modo de pulsación de tecla en silencio en una calculadora:

pública Calc clase extiende la actividad {

 public String static final PREFS\_NAME = "MyPrefsFile";

 @Override

 protected void onCreate (estado de paquete) {

 super.onCreate (estado);

 . . .

 / / Restaurar las preferencias

 Ajustes SharedPreferences = getSharedPreferences (PREFS\_NAME, 0);

 operador booleano = settings.getBoolean ("silentmode", false);

 setSilent (silencio);

 }

 @Override

 protected void OnStop () {

 super.onStop ();

 / / Necesitamos un objeto Editor de hacer cambios en la preferencia.

 / / Todos los objetos son de android.context.Contexto

 Ajustes SharedPreferences = getSharedPreferences (PREFS\_NAME, 0);

 SharedPreferences.Editor editor = settings.edit ();

 editor.putBoolean ("silentmode", mSilentMode);

 / / Confirmar las modificaciones!

 editor.commit ();

 }

}

**Utilizando el almacenamiento interno**

Puede guardar los archivos directamente en la memoria interna del dispositivo. De forma predeterminada, los archivos guardados en la memoria interna son privados de su aplicación y otras aplicaciones no pueden acceder a ellos (ni el usuario). Cuando el usuario desinstala la aplicación, estos archivos se eliminan.

Para crear y escribir un archivo privado para el almacenamiento interno:

1. Llame [openFileOutput()](http://developer.android.com/reference/android/content/Context.html#openFileOutput%28java.lang.String,%20int%29) con el nombre del archivo y el modo de funcionamiento. Esto devuelve un [FileOutputStream](http://developer.android.com/reference/java/io/FileOutputStream.html) .
2. Escribir en el fichero con [write()](http://developer.android.com/reference/java/io/OutputStream.html#write%28byte%5B%5D%29) .
3. Cerca del arroyo con [close()](http://developer.android.com/reference/java/io/FileOutputStream.html#close%28%29) .

Por ejemplo:

Cadena de archivo = "hello\_file";

String String = "Hello World!";

FileOutputStream fos = openFileOutput (filename, Contexto.MODE\_PRIVATE);

fos.write (string.getBytes ());

fos.close ();

[MODE\_PRIVATE](http://developer.android.com/reference/android/content/Context.html#MODE_PRIVATE) creará el archivo (o reemplazar un archivo del mismo nombre) y que sea privada de su aplicación. Otros modos disponibles son: [MODE\_APPEND](http://developer.android.com/reference/android/content/Context.html#MODE_APPEND) , [MODE\_WORLD\_READABLE](http://developer.android.com/reference/android/content/Context.html#MODE_WORLD_READABLE) , y [MODE\_WORLD\_WRITEABLE](http://developer.android.com/reference/android/content/Context.html#MODE_WORLD_WRITEABLE) .

Para leer un archivo de almacenamiento interno:

1. Llame [openFileInput()](http://developer.android.com/reference/android/content/Context.html#openFileInput%28java.lang.String%29) y pasarle el nombre del archivo a leer. Esto devuelve un [FileInputStream](http://developer.android.com/reference/java/io/FileInputStream.html) .
2. Lee bytes desde el archivo con [read()](http://developer.android.com/reference/java/io/FileInputStream.html#read%28byte%5B%5D,%20int,%20int%29) .
3. A continuación, cierre la corriente con [close()](http://developer.android.com/reference/java/io/FileInputStream.html#close%28%29) .

**Sugerencia:** Si desea guardar un archivo estático en su aplicación en tiempo de compilación, guarde el archivo en su proyecto de res/raw/ directorio. Usted puede abrir con [openRawResource()](http://developer.android.com/reference/android/content/res/Resources.html#openRawResource%28int%29) , pasando por el R.raw. *<filename>* ID de recurso. Este método devuelve un [InputStream](http://developer.android.com/reference/java/io/InputStream.html) que se puede utilizar para leer el archivo (pero no se puede escribir en el archivo original).

**Almacenamiento de archivos de caché**

Si desea almacenar en caché algunos datos, en lugar de almacenarla persistente, se debe utilizar [getCacheDir()](http://developer.android.com/reference/android/content/Context.html#getCacheDir%28%29) para abrir un [File](http://developer.android.com/reference/java/io/File.html) que representa el directorio interno, donde su aplicación debe guardar los archivos temporales de caché.

Cuando el dispositivo tiene poco espacio de almacenamiento interno, Android puede eliminar estos archivos de caché para recuperar espacio. Sin embargo, usted no debe confiar en el sistema para limpiar estos archivos para usted. Usted siempre debe mantener los archivos de caché de usted mismo y permanecer dentro de un límite razonable de consumido el espacio, tales como 1 MB. Cuando el usuario desinstala la aplicación, estos archivos se eliminan.

**Otros métodos útiles**

[getFilesDir()](http://developer.android.com/reference/android/content/Context.html#getFilesDir%28%29)

Obtiene la ruta completa del directorio de sistema de archivos donde los archivos se guardan internos.

[getDir()](http://developer.android.com/reference/android/content/Context.html#getDir%28java.lang.String,%20int%29)

Crea (o abre uno ya existente) de directorios dentro de su espacio de almacenamiento interno.

[deleteFile()](http://developer.android.com/reference/android/content/Context.html#deleteFile%28java.lang.String%29)

Elimina un archivo guardado en la memoria interna.

[fileList()](http://developer.android.com/reference/android/content/Context.html#fileList%28%29)

Devuelve una matriz de archivos guardados en la actualidad por su aplicación.

**Utilizando el almacenamiento externo**

Cada dispositivo compatible con Android soporta una responsabilidad compartida "de almacenamiento externo" que se puede utilizar para guardar archivos. Esto puede ser un medio de almacenamiento extraíble (como una tarjeta SD) o una memoria interna (no extraíble). Los archivos guardados en la memoria externa que son legibles y se pueden modificar por el usuario cuando permiten el almacenamiento masivo USB para transferir archivos en una computadora.

**Precaución:** Los archivos externos pueden desaparecer si el usuario se monta el almacenamiento externo en una computadora o elimina los medios de comunicación, y no hay seguridad de obligado cumplimiento para los archivos que guarde en el almacenamiento externo. Todas las aplicaciones pueden leer y escribir ficheros ubicados en el almacenamiento externo y el usuario puede eliminarlos.

**Comprobación de disponibilidad de los medios de comunicación**

Antes de realizar cualquier trabajo con el almacenamiento externo, siempre debe llamar [getExternalStorageState()](http://developer.android.com/reference/android/os/Environment.html#getExternalStorageState%28%29) para comprobar si el medio está disponible. Los medios de comunicación podría ser montado en un ordenador, falta de sólo lectura, o en algún otro estado. Por ejemplo, así es como se puede comprobar la disponibilidad:

boolean mExternalStorageAvailable = false;

boolean mExternalStorageWriteable = false;

Cadena de estado = Environment.getExternalStorageState ();

if (Environment.MEDIA\_MOUNTED.equals (estado)) {

 / / Se puede leer y escribir a los medios

 mExternalStorageAvailable = mExternalStorageWriteable = true;

} Else if (Environment.MEDIA\_MOUNTED\_READ\_ONLY.equals (estado)) {

 / / Sólo podemos leer los medios de comunicación

 mExternalStorageAvailable = true;

 mExternalStorageWriteable = false;

} else {

 / / Otra cosa que está mal. Puede ser uno de los muchos otros estados, pero todo lo que necesitamos

 / / Para que sé es que no saben leer ni escribir

 mExternalStorageAvailable = mExternalStorageWriteable = false;

}

En este ejemplo se comprueba si el almacenamiento externo está disponible para leer y escribir. El [getExternalStorageState()](http://developer.android.com/reference/android/os/Environment.html#getExternalStorageState%28%29) método devuelve otros estados que es posible que desee ver, por ejemplo, si los medios de comunicación está siendo compartida (conectada a un ordenador), falta por completo, se ha eliminado mal, etc Usted puede utilizar estas para notificar al usuario con más información cuando la aplicación necesita para acceder a los medios de comunicación.

**Acceso a los archivos de almacenamiento externo**

Si está utilizando la API de nivel 8 o superior, utilice la [getExternalFilesDir()](http://developer.android.com/reference/android/content/Context.html#getExternalFilesDir%28java.lang.String%29) para abrir un [File](http://developer.android.com/reference/java/io/File.html) que representa el directorio de almacenamiento externo en el que debe guardar sus archivos. Este método toma un type parámetro que especifica el tipo de subdirectorio que desee, como [DIRECTORY\_MUSIC](http://developer.android.com/reference/android/os/Environment.html#DIRECTORY_MUSIC) y [DIRECTORY\_RINGTONES](http://developer.android.com/reference/android/os/Environment.html#DIRECTORY_RINGTONES) (pasa null para recibir la raíz del directorio de archivos de la aplicación). Este método se creará el directorio adecuado si es necesario. Al especificar el tipo de directorio, se garantiza que el escáner de Android medios de categorizar correctamente los archivos en el sistema (por ejemplo, tonos de llamada se identificó como tonos de llamada y la música no). Si el usuario desinstala la aplicación, este directorio y todos sus contenidos serán eliminados.

Si está utilizando la API de nivel 7 o menos, utilice [getExternalStorageDirectory()](http://developer.android.com/reference/android/os/Environment.html#getExternalStorageDirectory%28%29) , para abrir un [File](http://developer.android.com/reference/java/io/File.html) que representa la raíz del almacenamiento externo. A continuación, debe escribir sus datos en el siguiente directorio:

/ Android / datos o *<nombre\_de\_paquete>* / files /

El *<package\_name>* es su estilo de Java, el nombre del paquete, tales como " com.example.android.app ". Si el dispositivo del usuario se está ejecutando el nivel API 8 o superior y desinstalar la aplicación, este directorio y todos sus contenidos serán eliminados.

**Ocultar los archivos desde el explorador de Medios**

Incluye un archivo vacío con el nombre .nomedia en el directorio de archivos externo (tenga en cuenta el prefijo punto en el nombre del archivo). Esto evitará que los medios de comunicación del escáner de Android de la lectura de sus archivos de medios y su inclusión en aplicaciones como la Galería o Música.

**Almacenamiento de archivos que deben ser compartidos**

Si desea guardar los archivos que no son específicas de su aplicación y que *no* debe suprimirse cuando la aplicación se desinstala, guardarlos en uno de los directorios públicos en el almacenamiento externo. Estos directorios se encontraba en el raíz del almacenamiento externo, como por ejemplo Music/ , Pictures/ , Ringtones/ , y otros.

En el nivel API de 8 o superior, utilice la [getExternalStoragePublicDirectory()](http://developer.android.com/reference/android/os/Environment.html#getExternalStoragePublicDirectory%28java.lang.String%29) , pasándole el tipo de directorio público que desee, como por ejemplo [DIRECTORY\_MUSIC](http://developer.android.com/reference/android/os/Environment.html#DIRECTORY_MUSIC) , [DIRECTORY\_PICTURES](http://developer.android.com/reference/android/os/Environment.html#DIRECTORY_PICTURES) , [DIRECTORY\_RINGTONES](http://developer.android.com/reference/android/os/Environment.html#DIRECTORY_RINGTONES) , u otros. Este método se creará el directorio adecuado si es necesario.

Si está utilizando la API de nivel 7 o menos, utilice [getExternalStorageDirectory()](http://developer.android.com/reference/android/os/Environment.html#getExternalStorageDirectory%28%29) para abrir un [File](http://developer.android.com/reference/java/io/File.html) que representa la raíz de la memoria externa, a continuación, guardar los archivos compartidos en uno de los siguientes directorios:

* Music/ - Escáner de medios de comunicación clasifica a todos los medios que se encuentran aquí como la música del usuario.
* Podcasts/ lector de medios de comunicación - clasifica a todos los medios que se encuentran aquí como un podcast.
* Ringtones/ lector de medios de comunicación - clasifica a todos los medios que se encuentran aquí como tono de llamada.
* Alarms/ medios de comunicación - escáner clasifica todos los medios que se encuentran aquí, como un sonido de alarma.
* Notifications/ lector de medios de comunicación - clasifica a todos los medios que se encuentran aquí, como un sonido de notificación.
* Pictures/ - Todas las fotos (excepto las tomadas con la cámara).
* Movies/ - Todas las películas (con exclusión de las tomadas con la cámara).
* Download/ - Varios descargas.

**Almacenamiento de archivos de caché**

Si está utilizando la API de nivel 8 o superior, utilice la [getExternalCacheDir()](http://developer.android.com/reference/android/content/Context.html#getExternalCacheDir%28%29) para abrir un [File](http://developer.android.com/reference/java/io/File.html) que representa el directorio de almacenamiento externo en el que debe guardar los archivos de caché. Si el usuario desinstala la aplicación, estos archivos se eliminan automáticamente. Sin embargo, durante la vida de la aplicación, se deben manejar estos archivos de caché y eliminar aquellas que no son necesarios con el fin de preservar el espacio de archivo.

Si está utilizando la API de nivel 7 o menos, utilice [getExternalStorageDirectory()](http://developer.android.com/reference/android/os/Environment.html#getExternalStorageDirectory%28%29) para abrir un [File](http://developer.android.com/reference/java/io/File.html) que representa la raíz de la memoria externa, a continuación, escriba sus datos de la caché en el siguiente directorio:

/ Android / data / *<nombre\_de\_paquete>* / cache /

El *<package\_name>* es su estilo de Java, el nombre del paquete, tales como " com.example.android.app ".

**Uso de bases de datos**

Android proporciona soporte completo para [SQLite](http://www.google.com/url?q=http%3A%2F%2Fwww.sqlite.org%2F&sa=D&sntz=1&usg=AFQjCNF6eR-3bZbqngxsQcIgZgPduzeppg) bases de datos. Las bases de datos que cree se podrá acceder por su nombre a una clase en la aplicación, pero no fuera de la aplicación.

El método recomendado para crear una nueva base de datos SQLite es crear una subclase de [SQLiteOpenHelper](http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html) y anular la [onCreate()](http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html#onCreate%28android.database.sqlite.SQLiteDatabase%29) método, en el que se puede ejecutar un comando de SQLite para crear tablas en la base de datos. Por ejemplo:

public class extends DictionaryOpenHelper SQLiteOpenHelper {

 privada DATABASE\_VERSION static final int = 2;

 privada DICTIONARY\_TABLE\_NAME static final String = "diccionario";

 private String static final DICTIONARY\_TABLE\_CREATE =

 "CREATE TABLE" + DICTIONARY\_TABLE\_NAME + "(" +

 KEY\_WORD + "texto", +

 KEY\_DEFINITION + "TEXT);"

 DictionaryOpenHelper (context) {

 super (contexto, DATABASE\_NAME, nulo, DATABASE\_VERSION);

 }

 @Override

 onCreate public void (SQLiteDatabase db) {

 db.execSQL (DICTIONARY\_TABLE\_CREATE);

 }

}

A continuación, puede obtener una instancia de su [SQLiteOpenHelper](http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html) la aplicación utilizando el constructor que ha definido. Para escribir y leer desde la base de datos, llamada [getWritableDatabase()](http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html#getWritableDatabase%28%29) y [getReadableDatabase()](http://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html#getReadableDatabase%28%29) , respectivamente. Estos dos devolver un [SQLiteDatabase](http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html) objeto que representa la base de datos y proporciona métodos para operaciones de datos SQLite.

Android no impone ninguna limitación más allá de los conceptos estándar de datos SQLite. Nosotros le recomendamos que incluya un campo autoincrement clave de valor que puede ser utilizado como un identificador único para encontrar rápidamente un registro. Esto no es necesario para los datos privados, pero si se implementa un [proveedor de contenidos](http://developer.android.com/guide/topics/providers/content-providers.html) , debe incluir un identificador único utilizando los [BaseColumns.\_ID](http://developer.android.com/reference/android/provider/BaseColumns.html#_ID) constante.

Puede ejecutar consultas de SQLite utilizando el [SQLiteDatabase](http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html) [query()](http://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html#query%28boolean,%20java.lang.String,%20java.lang.String%5B%5D,%20java.lang.String,%20java.lang.String%5B%5D,%20java.lang.String,%20java.lang.String,%20java.lang.String,%20java.lang.String%29) , que aceptan los métodos de varios parámetros de consulta, como la mesa de consulta, la proyección, la selección, las columnas, la agrupación, y otros. Para consultas complejas, tales como aquellos que requieren de alias de columna, se debe utilizar [SQLiteQueryBuilder](http://developer.android.com/reference/android/database/sqlite/SQLiteQueryBuilder.html) , que proporciona varios métodos para la construcción de convienent consultas.

Cada consulta de SQLite devolverá un [Cursor](http://developer.android.com/reference/android/database/Cursor.html) que apunta a todos los registros encontrados por la consulta. El [Cursor](http://developer.android.com/reference/android/database/Cursor.html) siempre es el mecanismo con el cual se puede navegar los resultados de una consulta de base de datos y leer las filas y columnas.

Para aplicaciones de ejemplo que muestran cómo utilizar bases de datos SQLite en Android, consulte el [Bloc de Notas](http://developer.android.com/resources/samples/NotePad/index.html) y [investigables Diccionario](http://developer.android.com/resources/samples/SearchableDictionary/index.html) aplicaciones.

**Base de datos de depuración**

El SDK de Android incluye un sqlite3 herramienta de base de datos que le permite navegar por contenidos de la tabla, ejecutar comandos SQL, y realizar otras funciones útiles en bases de datos SQLite. Ver [Examen de sqlite3 bases de datos de un shell remoto](http://developer.android.com/guide/developing/tools/adb.html#sqlite) para aprender a ejecutar esta herramienta.

**Utilizando una conexión de red**

Puede utilizar la red (cuando esté disponible) para almacenar y recuperar datos de sus propios servicios basados ​​en web. Para hacer las operaciones de red, utilizar las clases en los siguientes paquetes:

* [java.net.\*](http://developer.android.com/reference/java/net/package-summary.html)
* [android.net.\*](http://developer.android.com/reference/android/net/package-summary.html)

**Los datos de copia de seguridad**

**Vista Rapida**

* Copia de seguridad de los datos del usuario a la nube en el caso de que el usuario pierde
* Si el usuario actualiza a un nuevo dispositivo Android, su aplicación puede restaurar los datos del usuario en el nuevo dispositivo
* SharedPreferences fácilmente copias de seguridad y archivos privados con BackupAgentHelper
* Necesitas ser de nivel 8 de la API

**En este documento**

1. [Los puntos básicos.](http://developer.android.com/guide/topics/data/backup.html#Basics)
2. [La declaración del agente de copia de seguridad en el manifiesto](http://developer.android.com/guide/topics/data/backup.html#BackupManifest)
3. [El registro de servicio de copia de seguridad para Android](http://developer.android.com/guide/topics/data/backup.html#BackupKey)
4. [Extender BackupAgent](http://developer.android.com/guide/topics/data/backup.html#BackupAgent)
	1. [Métodos necesarios](http://developer.android.com/guide/topics/data/backup.html#RequiredMethods)
	2. [Ejecutando Backup](http://developer.android.com/guide/topics/data/backup.html#PerformingBackup)
	3. [Ejecutando Restauración](http://developer.android.com/guide/topics/data/backup.html#PerformingRestore)
5. [Extender BackupAgentHelper](http://developer.android.com/guide/topics/data/backup.html#BackupAgentHelper)
	1. [Copia de seguridad de SharedPreferences](http://developer.android.com/guide/topics/data/backup.html#SharedPreferences)
	2. [Copia de seguridad de archivos privados](http://developer.android.com/guide/topics/data/backup.html#Files)
6. [Comprobación de la versión de restauración de datos](http://developer.android.com/guide/topics/data/backup.html#RestoreVersion)
7. [Solicitud de copia de seguridad](http://developer.android.com/guide/topics/data/backup.html#RequestingBackup)
8. [Solicitud de restauración](http://developer.android.com/guide/topics/data/backup.html#RequestingRestore)
9. [Prueba de su agente de copia de seguridad](http://developer.android.com/guide/topics/data/backup.html#Testing)

**Clases de clave**

1. [BackupManager](http://developer.android.com/reference/android/app/backup/BackupManager.html)
2. [BackupAgent](http://developer.android.com/reference/android/app/backup/BackupAgent.html)
3. [BackupAgentHelper](http://developer.android.com/reference/android/app/backup/BackupAgentHelper.html)

**Véase también**

1. [bmgr herramienta de](http://developer.android.com/guide/developing/tools/bmgr.html)

Android [backup](http://developer.android.com/reference/android/app/backup/package-summary.html) servicio le permite copiar los datos de las aplicaciones existentes a la distancia "nube" de almacenamiento, a fin de proporcionar un punto de restauración de los datos de las aplicaciones y configuraciones. Si un usuario realiza una restauración de fábrica o se convierte en un nuevo dispositivo Android, el sistema restaura automáticamente los datos de copia de seguridad cuando la aplicación se vuelve a instalar. De esta manera, los usuarios no necesitan para reproducir sus datos anteriores o configuración de la aplicación. Este proceso es completamente transparente para el usuario y no afecta a la funcionalidad o la experiencia del usuario en su aplicación.

Durante una operación de copia de seguridad (que la aplicación puede solicitar), encargado de reserva de Android ( [BackupManager](http://developer.android.com/reference/android/app/backup/BackupManager.html) ) consulta a su solicitud de copia de seguridad de datos, y luego se lo pasa a una copia de seguridad de transporte, que luego entrega los datos al almacenamiento en la nube. Durante una operación de restauración, el Backup Manager recupera los datos de copia de seguridad de la copia de seguridad del transporte y la devuelve a su aplicación por lo que su aplicación puede restaurar los datos en el dispositivo. Es posible que su solicitud para pedir una restauración, pero eso no debería ser necesario-para Android de forma automática realiza una operación de restauración cuando su aplicación está instalada y existe copias de seguridad relacionadas con el usuario. El principal escenario en el que los datos de copia de seguridad se restablece es cuando un usuario restablece su dispositivo, o mejoras a un nuevo dispositivo y sus aplicaciones previamente instaladas se vuelven a instalar.

**Nota:** El servicio de copia de seguridad *no* está diseñado para sincronizar los datos de la aplicación con otros clientes o guardar los datos que le gustaría tener acceso durante el ciclo de vida normal de la aplicación. No se puede leer o escribir datos de copia de seguridad en la demanda y no se puede acceder a ella de ninguna otra manera que a través de las API proporcionadas por el Administrador de copia de seguridad.

El transporte de copia de seguridad es el componente del lado del cliente de un marco de copia de seguridad de Android, que es personalizable por el fabricante del dispositivo y el proveedor de servicios. El transporte de copia de seguridad pueden diferir de un dispositivo a otro y que el transporte de copia de seguridad está disponible en cualquier dispositivo dado es transparente para su aplicación. La API del Administrador de copia de seguridad de aislar la aplicación de copia de seguridad del transporte disponible real en un hecho. Dispositivo de su aplicación se comunica con el Administrador de copia de seguridad a través de un conjunto fijo de las API, independientemente del transporte subyacente

Copia de seguridad de datos *no* se garantiza que esté disponible en todos los dispositivos con Android. Sin embargo, su aplicación no se vea perjudicada en caso de que un dispositivo no proporciona un transporte de copia de seguridad. Si usted cree que los usuarios se beneficiarán de la copia de seguridad de datos en la aplicación, entonces usted puede poner en práctica tal como se describe en este documento, prueba, a continuación, publicar la aplicación sin ningún tipo de preocupación acerca de los dispositivos que realmente realizar copias de seguridad. Cuando la aplicación se ejecuta en un dispositivo que no proporciona una copia de seguridad de transporte, la aplicación funciona con normalidad, pero no va a recibir devoluciones de llamada desde el Administrador de copia de seguridad a los datos de copia de seguridad.

Aunque no se puede saber lo que el transporte actual es, que siempre se aseguró de que sus datos de copia de seguridad no pueden ser leídos por otras aplicaciones en el dispositivo. Sólo el Administrador de copia de copia de seguridad y el transporte tienen acceso a los datos que usted proporcione durante una operación de copia de seguridad.

**Atención:** Debido a que el almacenamiento en la nube y el servicio de transporte puede variar de un dispositivo a otro, Android no ofrece ninguna garantía sobre la seguridad de sus datos durante el uso de copia de seguridad. Usted siempre debe ser cauteloso sobre el uso de copia de seguridad para almacenar datos sensibles, tales como nombres de usuario y contraseñas.

**Los puntos básicos.**

Para copias de seguridad de los datos de aplicación, es necesario implementar un agente de copia de seguridad. Su agente de copia de seguridad es llamado por el Administrador de copia de seguridad para proporcionar los datos que desea hacer copia de seguridad. También se le llama para restaurar los datos de copia de seguridad cuando la aplicación se vuelve a instalar. El Administrador de copia de seguridad se ocupa de todas sus transacciones de datos con el almacenamiento en la nube (con el transporte de copia de seguridad) y su agente de copia de seguridad se ocupa de todas sus transacciones de datos en el dispositivo.

Para llevar a cabo un agente de copia de seguridad, usted debe:

1. Declarar su agente de copia de seguridad en el archivo de manifiesto con el [android:backupAgent](http://developer.android.com/guide/topics/manifest/application-element.html#agent) atributo.
2. Registre su solicitud con un servicio de copia de seguridad. Google ofrece [servicio de copia de seguridad de Android](http://code.google.com/android/backup/index.html) como un servicio de copia de seguridad para la mayoría de dispositivos con Android, lo que requiere que se registre su solicitud a fin de que funcione. Todos los servicios de copia de seguridad disponibles otros también podrían requerir que se registre con el fin de almacenar los datos en sus servidores.
3. Definir un agente de copia de seguridad mediante:
	1. [Extender BackupAgent](http://developer.android.com/guide/topics/data/backup.html#BackupAgent)

El [BackupAgent](http://developer.android.com/reference/android/app/backup/BackupAgent.html) clase proporciona la interfaz central con el que su aplicación se comunica con el Backup Manager. Si se extiende esta clase directamente, se debe reemplazar [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) y [onRestore()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onRestore%28android.app.backup.BackupDataInput,%20int,%20android.os.ParcelFileDescriptor%29) para gestionar la copia de seguridad y restauración de sus datos.

*O*

* 1. [Extender BackupAgentHelper](http://developer.android.com/guide/topics/data/backup.html#BackupAgentHelper)

El [BackupAgentHelper](http://developer.android.com/reference/android/app/backup/BackupAgentHelper.html) clase proporciona una envoltura alrededor de la práctica [BackupAgent](http://developer.android.com/reference/android/app/backup/BackupAgent.html) de clase, lo que minimiza la cantidad de código que necesita para escribir. En su [BackupAgentHelper](http://developer.android.com/reference/android/app/backup/BackupAgentHelper.html) , debe utilizar uno o más objetos "ayudante", que automáticamente copia de seguridad y restauración de ciertos tipos de datos, de modo que no es necesario para poner en práctica [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) y [onRestore()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onRestore%28android.app.backup.BackupDataInput,%20int,%20android.os.ParcelFileDescriptor%29) .

Android actualmente provee a los socorristas de copia de seguridad que se haga copia de seguridad y restaurar archivos completos a partir de [SharedPreferences](http://developer.android.com/reference/android/content/SharedPreferences.html) y [almacenamiento interno](http://developer.android.com/guide/topics/data/data-storage.html#filesInternal) .

**La declaración del agente de copia de seguridad en el manifiesto**

Este es el paso más sencillo, por lo que una vez que usted haya decidido sobre el nombre de la clase de su agente de copia de seguridad, declaran en su manifiesto con el [android:backupAgent](http://developer.android.com/guide/topics/manifest/application-element.html#agent) atributo en el [<application>](http://developer.android.com/guide/topics/manifest/application-element.html) etiqueta.

Por ejemplo:

<manifest ... >

 ...

 <Aplicación para Android: label = "MyApplication"

 **android: BackupAgent = "MyBackupAgent">**

 <Actividad ... >

 ...

 </ Actividad>

 </ Application>

</manifest>

Otro atributo es posible que desee utilizar es [android:restoreAnyVersion](http://developer.android.com/guide/topics/manifest/application-element.html#restoreany) . Este atributo tiene un valor booleano para indicar si desea restaurar los datos de aplicación independientemente de la versión de la aplicación actual en comparación con la versión que produjo los datos de copia de seguridad. El valor por defecto es false. Consulte [Comprobación de la versión de restauración de datos](http://developer.android.com/guide/topics/data/backup.html#RestoreVersion) para obtener más información.

**Nota:** El servicio de copia de seguridad y las API debe utilizar sólo están disponibles en dispositivos que ejecutan la API de nivel 8 (Android 2.2) o superior, por lo que también debe configurar su [android:minSdkVersion](http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#min) atributo en "8". Sin embargo, si se implementa correcta [compatibilidad con versiones anteriores](http://developer.android.com/resources/articles/backward-compatibility.html) en su aplicación, se puede admitir esta función para los dispositivos con la API de nivel 8 o superior, mientras que mantiene la compatibilidad con los dispositivos más antiguos.

**El registro de servicio de copia de seguridad para Android**

Google ofrece un transporte de copia de seguridad con el [servicio de copia de seguridad para Android](http://code.google.com/android/backup/index.html) para la mayoría de dispositivos con Android con Android 2.2 o superior.

Para que la aplicación para realizar copias de seguridad usando el servicio de copia de seguridad de Android, se debe registrar la aplicación con el servicio para recibir una clave de servicio de copia de seguridad, a continuación, declarar la clave de servicio de copia de seguridad en el manifiesto de Android.

Para obtener su clave de Servicio de copia de seguridad, [registrarse para el servicio de copia de seguridad para Android](http://code.google.com/android/backup/signup.html) . Cuando usted se registra, se le proporcionará una clave de servicio de copia de seguridad y la adecuada <meta-data> código XML para el archivo de manifiesto para Android, el cual debe incluir como un niño de la <application> elemento. Por ejemplo:

<Aplicación para Android: label = "MyApplication"

 android: BackupAgent = "MyBackupAgent">

 ...

 <Meta-datos android: name = "com.google.android.backup.api\_key"

 android: value = "AEdPqrEAAAAIDaYEVgU6DJnyJdBmU7KLH3kszDXLv\_4DIsEIyQ" />

</ Application>

El android:name debe ser "com.google.android.backup.api\_key" y el android:value debe ser la clave de servicio de copia de seguridad recibió de la inscripción en el Servicio de copia de seguridad para Android.

Si usted tiene múltiples aplicaciones, debe registrar cada uno de ellos, utilizando el nombre del paquete correspondiente.

**Nota:** El transporte de copia de seguridad proporcionado por el Servicio de copia de seguridad para Android no se garantiza que esté disponible en todos los dispositivos con Android que copia de seguridad de apoyo. Algunos dispositivos podrían apoyar copia de seguridad mediante un transporte diferente, algunos de los dispositivos de copia de seguridad puede no ser compatible en absoluto, y no hay manera de que su aplicación para saber lo que el transporte se utiliza en el dispositivo. Sin embargo, si la aplicación de copia de seguridad para su aplicación, siempre debe incluir una clave de servicio de copia de seguridad para el servicio de copia de seguridad de Android para que su aplicación puede realizar copia de seguridad cuando el dispositivo utiliza el transporte de copia de seguridad Servicio de Android. Si el dispositivo no utilizar el servicio de copia de seguridad de Android, el <meta-data> elemento con la clave de servicio de copia de seguridad se tiene en cuenta.

**Extender BackupAgent**

La mayoría de aplicaciones no es necesario extender la [BackupAgent](http://developer.android.com/reference/android/app/backup/BackupAgent.html) clase directamente, sino que debe [extenderse BackupAgentHelper](http://developer.android.com/guide/topics/data/backup.html#BackupAgentHelper) para tomar ventaja de la incorporada en las clases de ayuda que automáticamente copia de seguridad y restaurar sus archivos. Sin embargo, es posible que desee ampliar [BackupAgent](http://developer.android.com/reference/android/app/backup/BackupAgent.html) directamente si es necesario:

* La versión de su formato de datos. Por ejemplo, si se prevé la necesidad de revisar el formato en el que se escriben los datos de la aplicación, puede crear un agente de copia de seguridad para cotejar la versión de su aplicación durante una operación de restauración y realizar cualquier trabajo de la compatibilidad necesaria si la versión del dispositivo es diferente que la de los datos de copia de seguridad. Para obtener más información, consulte [Comprobación de la versión de datos de restauración](http://developer.android.com/guide/topics/data/backup.html#RestoreVersion) .
* En lugar de copias de seguridad de un archivo completo, puede especificar las porciones de los datos de la iniciativa debería ser respaldada y cómo cada parte se restaura en el dispositivo. (Esto también puede ayudar a gestionar las diferentes versiones, ya que leer y escribir sus datos como entidades únicas, en lugar de archivos completos).
* Copia de seguridad de datos en una base de datos. Si usted tiene una base de datos SQLite que desea restaurar cuando el usuario vuelve a instalar la aplicación, usted necesita para construir una costumbre [BackupAgent](http://developer.android.com/reference/android/app/backup/BackupAgent.html) que lee los datos apropiados durante una operación de copia de seguridad, crear la tabla e insertar los datos durante una operación de restauración.

Si no es necesario que realice cualquiera de las tareas anteriores y desea hacer copias de seguridad completas de [SharedPreferences](http://developer.android.com/reference/android/content/SharedPreferences.html) o [almacenamiento interno](http://developer.android.com/guide/topics/data/data-storage.html#filesInternal) , usted debe saltar a la [ampliación BackupAgentHelper](http://developer.android.com/guide/topics/data/backup.html#BackupAgentHelper) .

**Métodos necesarios**

Cuando se crea un agente de copia de seguridad mediante la ampliación de [BackupAgent](http://developer.android.com/reference/android/app/backup/BackupAgent.html) , debe implementar los métodos de devolución de llamada siguientes:

[onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29)

El Administrador de copia de seguridad llama a este método después de [solicitar una copia de seguridad](http://developer.android.com/guide/topics/data/backup.html#RequestingBackup) . En este método, usted lee los datos de aplicación desde el dispositivo y pasar los datos que desea hacer copia de seguridad para el Administrador de copia de seguridad, tal como se describe a continuación en la [realización de copia de seguridad](http://developer.android.com/guide/topics/data/backup.html#PerformingBackup) .

[onRestore()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onRestore%28android.app.backup.BackupDataInput,%20int,%20android.os.ParcelFileDescriptor%29)

El Administrador de copia de seguridad llama a este método durante una operación de restauración (se puede [solicitar una restauración](http://developer.android.com/guide/topics/data/backup.html#RequestingRestore) , pero el sistema realiza automáticamente restaurar cuando el usuario vuelva a instalar la aplicación). Cuando se llama a este método, el Backup Manager proporciona los datos de copia de seguridad, que luego restaurar en el dispositivo, como se describe a continuación en la [Realización de restauración](http://developer.android.com/guide/topics/data/backup.html#PerformingRestore) .

**Ejecutando Backup**

Cuando es tiempo de copia de seguridad de los datos de aplicación, el administrador de copia de seguridad llama a su [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) método. Aquí es donde usted debe proporcionar los datos de aplicación para el Administrador de copia de seguridad por lo que se puede guardar en almacenamiento en la nube.

Sólo el Administrador de copia de seguridad puede llamar a su agente de copia de seguridad de [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) método. Cada vez que los cambios de datos de aplicación y desea realizar una copia de seguridad, usted debe solicitar una operación de copia de seguridad llamando al [dataChanged()](http://developer.android.com/reference/android/app/backup/BackupManager.html#dataChanged%28%29) (ver [copia de seguridad de Solicitud](http://developer.android.com/guide/topics/data/backup.html#RequestingBackup) para más información). Una solicitud de copia de seguridad no se traduce en una llamada inmediata a su [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) método. En cambio, el Administrador de copia de seguridad espera el momento oportuno, a continuación, realiza copias de seguridad para todas las aplicaciones que han solicitado una copia de seguridad desde la última copia de seguridad se llevó a cabo.

**Consejo:** Si bien el desarrollo de su aplicación, se puede iniciar una operación de copia de seguridad inmediata desde el Administrador de copia de seguridad con la [bmgr herramienta](http://developer.android.com/guide/developing/tools/bmgr.html) .

Cuando el Administrador de copia de seguridad llama a su [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) método, se pasa a tres parámetros:

oldState

Un proceso abierto, de sólo lectura [ParcelFileDescriptor](http://developer.android.com/reference/android/os/ParcelFileDescriptor.html) que apunta a la última copia de seguridad del Estado recibida por su aplicación. No se trata de los datos de copia de seguridad de almacenamiento en la nube, pero con una representación local de los datos que fueron respaldados la última vez que [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) se llama (como se define en newState , a continuación, o de [onRestore()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onRestore%28android.app.backup.BackupDataInput,%20int,%20android.os.ParcelFileDescriptor%29) -más sobre esto en la siguiente sección ). Debido a que [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) no le permite leer los datos de copia de seguridad existentes en el almacenamiento en la nube, puede utilizar esta representación local para determinar si sus datos han cambiado desde la última copia de seguridad.

data

Un [BackupDataOutput](http://developer.android.com/reference/android/app/backup/BackupDataOutput.html) objeto, que se utiliza para proporcionar sus datos de copia de seguridad para el Backup Manager.

newState

Un proceso abierto, de lectura / escritura [ParcelFileDescriptor](http://developer.android.com/reference/android/os/ParcelFileDescriptor.html) apunta a un archivo en el cual usted debe escribir una representación de los datos que se entregan a data (una representación no puede ser tan simple como la fecha y hora de última modificación para el archivo). Este objeto se devuelve como oldState la próxima vez que el Administrador de copia de seguridad llama a su [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) método. Si usted no escribe sus datos de copia de seguridad de newState , entonces oldState apuntará a un archivo vacío Gerente de la próxima copia de seguridad de tiempo de llamadas [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) .

Utilizando estos parámetros, debe implementar su [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) método para hacer lo siguiente:

1. Compruebe si sus datos han cambiado desde la última copia de seguridad mediante la comparación de oldState a sus datos actuales. ¿Cómo leer los datos en oldState depende de cómo se escribió originalmente para newState (ver paso 3). La forma más sencilla para registrar el estado de un archivo es con su fecha y hora de la última modificación. Por ejemplo, he aquí cómo usted puede leer y comparar una marca de tiempo de oldState :
2. / / Obtener el flujo de entrada oldstate
3. FileInputStream dentro de la corriente = new FileInputStream (oldState.getFileDescriptor ());
4. DataInputStream in = new DataInputStream (dentro de la corriente);
5. try {
6. / / Obtener la fecha y hora última modificación del archivo de Estado y de archivo de datos
7. a largo stateModified = in.readLong ();
8. a largo fileModified = mDataFile.lastModified ();
9. if (stateModified!=) FileModified {
10. / / El archivo ha sido modificado, por lo que hace una copia de seguridad
11. / / O la vez en el dispositivo de cambio, así que sea seguro y hacer una copia de seguridad
12. } else {
13. / / No hace copia de seguridad porque el archivo no ha cambiado
14. Retorno
15. }
16. } Catch (Exception e) {
17. / / No se puede leer archivo de estado ... estar seguro y hacer una copia de seguridad
18. }

Si nada ha cambiado y no es necesario realizar copias de seguridad, vaya al paso 3.

1. Si sus datos han cambiado, en comparación con oldState , escribir los datos actuales a data que lo respalden en el almacenamiento en la nube.

Usted debe escribir cada fragmento de datos como una "entidad" en el [BackupDataOutput](http://developer.android.com/reference/android/app/backup/BackupDataOutput.html) . Una entidad es un binario aplanado registro de datos que se identifica por una cadena de clave única. De este modo, el conjunto de datos que copia de seguridad es conceptualmente un conjunto de pares clave-valor.

Para añadir una entidad a su grupo de respaldo de datos, usted debe:

* 1. Llame [writeEntityHeader()](http://developer.android.com/reference/android/app/backup/BackupDataOutput.html#writeEntityHeader%28java.lang.String,%20int%29) , pasando por una clave de cadena única para que los datos que está a punto de escribir y el tamaño de los datos.
	2. Llame [writeEntityData()](http://developer.android.com/reference/android/app/backup/BackupDataOutput.html#writeEntityData%28byte%5B%5D,%20int%29) , pasando por un búfer de bytes que contiene los datos y el número de bytes a escribir en la memoria intermedia (que debe coincidir con el tamaño pasa a [writeEntityHeader()](http://developer.android.com/reference/android/app/backup/BackupDataOutput.html#writeEntityHeader%28java.lang.String,%20int%29) ).

Por ejemplo, el siguiente código se aplana algunos datos en un flujo de bytes y lo escribe en una sola entidad:

/ / Crear una secuencia de buffer y flujo de salida de datos para nuestros datos

ByteArrayOutputStream bufStream = new ByteArrayOutputStream ();

DataOutputStream outWriter = new DataOutputStream (bufStream);

/ / Escribir los datos estructurados

outWriter.writeUTF (mPlayerName);

outWriter.writeInt (mPlayerScore);

/ / Enviar los datos al Administrador de copia de seguridad a través de la BackupDataOutput

byte [] buffer = bufStream.toByteArray ();

int len ​​= buffer.length;

data.writeEntityHeader (TOPSCORE\_BACKUP\_KEY, len);

data.writeEntityData (buffer, len);

Realice esta operación para cada pieza de datos que desea hacer copia de seguridad. ¿Cómo se dividen los datos en las entidades depende de usted (y usted podría usar tan solo una entidad).

1. Sea o no de realizar una copia de seguridad (en el paso 2), escribe una representación de los datos actuales a la newState [ParcelFileDescriptor](http://developer.android.com/reference/android/os/ParcelFileDescriptor.html) . El Administrador de copia de seguridad mantiene este objeto a nivel local como una representación de los datos que se encuentra actualmente una copia de seguridad. Pasa esto a usted como oldState la próxima vez que llame [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) para que pueda determinar si otra copia de seguridad es necesario (como controlado en el paso 1). Si usted no escribe el estado actual de datos de este archivo, a continuación, oldState estará vacía durante la siguiente respuesta.

El ejemplo siguiente guarda una representación de los datos actuales en newState utilizando el archivo de la última modificación Fecha y hora:

FileOutputStream flujoout = new FileOutputStream (newState.getFileDescriptor ());

DataOutputStream salida = new DataOutputStream (flujoout);

a largo modificada = mDataFile.lastModified ();

out.writeLong (modificada);

**Precaución:** Si los datos de la aplicación se guarda en un archivo, asegúrese de que use los estados sincronizados en su acceso al archivo para que su agente de copia de seguridad no lee el archivo, mientras que una actividad en la aplicación también está escribiendo el archivo.

**Ejecutando Restauración**

Cuando sea el momento para restaurar los datos de la aplicación, el administrador de copia de seguridad llama a su agente de copia de seguridad de [onRestore()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onRestore%28android.app.backup.BackupDataInput,%20int,%20android.os.ParcelFileDescriptor%29) método. Cuando se llama a este método, el Backup Manager proporciona los datos de copia de seguridad para que pueda recuperar en el dispositivo.

Sólo el Administrador de copia de seguridad puede llamar [onRestore()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onRestore%28android.app.backup.BackupDataInput,%20int,%20android.os.ParcelFileDescriptor%29) , que se produce automáticamente cuando el sistema se instala la aplicación y encuentra los datos existentes de copia de seguridad. Sin embargo, usted puede solicitar una operación de restauración para su aplicación llamando al [requestRestore()](http://developer.android.com/reference/android/app/backup/BackupManager.html#requestRestore%28android.app.backup.RestoreObserver%29) (véase la [Solicitud de restauración](http://developer.android.com/guide/topics/data/backup.html#RequestingRestore) para más información).

**Nota:** Durante el desarrollo de su aplicación, también puede solicitar una operación de restauración con la [bmgr herramienta](http://developer.android.com/guide/developing/tools/bmgr.html) .

Cuando el Administrador de copia de seguridad llama a su [onRestore()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onRestore%28android.app.backup.BackupDataInput,%20int,%20android.os.ParcelFileDescriptor%29) método, se pasa a tres parámetros:

data

Un [BackupDataInput](http://developer.android.com/reference/android/app/backup/BackupDataInput.html) , que le permite leer los datos de copia de seguridad.

appVersionCode

Un entero que representa el valor de su aplicación [android:versionCode](http://developer.android.com/guide/topics/manifest/manifest-element.html#vcode) atributo de manifiesto, como lo fue cuando estos datos se copia de seguridad. Usted puede usar esto para cotejar la versión de la aplicación actual y determinar si el formato de datos es compatible. Para obtener más información sobre el uso de esta para manipular las diferentes versiones de los datos de restauración, vea la sección debajo de [Comprobación de la versión de datos de restauración](http://developer.android.com/guide/topics/data/backup.html#RestoreVersion) .

newState

Un proceso abierto, de lectura / escritura [ParcelFileDescriptor](http://developer.android.com/reference/android/os/ParcelFileDescriptor.html) apunta a un archivo en el cual usted debe escribir el estado de copia de seguridad final que se le proporcionó data . Este objeto se devuelve como oldState la próxima vez que [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) se llama. Recuerde que también debe escribir el mismo newState objeto en el [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) de devolución de llamada, también hacerlo aquí asegura que el oldState objeto especificado en [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) es válido, incluso la primera vez que [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) se llama después de que el dispositivo se restablece.

En la implementación de [onRestore()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onRestore%28android.app.backup.BackupDataInput,%20int,%20android.os.ParcelFileDescriptor%29) , usted debe llamar a [readNextHeader()](http://developer.android.com/reference/android/app/backup/BackupDataInput.html#readNextHeader%28%29) sobre los data para recorrer todas las entidades en el conjunto de datos. Para cada entidad que se encuentre, haga lo siguiente:

1. Obtener la clave de entidad con [getKey()](http://developer.android.com/reference/android/app/backup/BackupDataInput.html#getKey%28%29) .
2. Comparar la clave de entidad a una lista de los más conocidos los valores clave que usted debe haber declarados como estáticos cadenas finales dentro de su [BackupAgent](http://developer.android.com/reference/android/app/backup/BackupAgent.html) clase. Cuando la clave coincide con una de sus cadenas principales conocidos, entrar en una declaración para extraer los datos de la entidad y guardarla en el dispositivo:
	1. Obtener el tamaño de los datos entidad con [getDataSize()](http://developer.android.com/reference/android/app/backup/BackupDataInput.html#getDataSize%28%29) y crear una matriz de bytes de ese tamaño.
	2. Llame [readEntityData()](http://developer.android.com/reference/android/app/backup/BackupDataInput.html#readEntityData%28byte%5B%5D,%20int,%20int%29) y pasarle la matriz de bytes, que es donde los datos se van, y especificar el desplazamiento de inicio y el tamaño de leer.
	3. Su matriz de bytes está lleno y se puede leer y escribir los datos al dispositivo como usted quiera.
3. Después de leer y escribir los datos al dispositivo, anote el estado de sus datos a la newState parámetro de la misma que tiene que hacer durante [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) .

Por ejemplo, aquí es cómo puede restaurar los datos respaldados por el ejemplo en la sección anterior:

@Override

public void OnRestore (BackupDataInput datos, int appVersionCode,

 ParcelFileDescriptor newstate) throws IOException {

 / / Debe haber una sola entidad, pero el más seguro

 / / Forma de consumir es usando un bucle while

 mientras que (data.readNextHeader ()) {

 Cadena clave = data.getKey ();

 int DataSize data.getDataSize = ();

 / / Si la clave es la nuestra (para guardar puntaje máximo). Nota Esta clave se utiliza cuando

 / / Escribimos el encabezado de copia de seguridad de la entidad

 if (TOPSCORE\_BACKUP\_KEY.equals (clave)) {

 / / Crear un flujo de entrada para el BackupDataInput

 byte [] DATABUF = new byte [DataSize];

 data.readEntityData (DATABUF, 0, DataSize);

 ByteArrayInputStream baStream = new ByteArrayInputStream (DATABUF);

 DataInputStream in = new DataInputStream (baStream);

 / / Lee el nombre del jugador y la puntuación de los datos de copia de seguridad

 mPlayerName in.readUTF = ();

 mPlayerScore in.readInt = ();

 / / Graba la puntuación en el dispositivo (en un archivo o algo así)

 recordScore (mPlayerName, mPlayerScore);

 } else {

 / / No se sabe esta clave de la entidad. Saltar la misma. (No debería ocurrir.)

 data.skipEntityData ();

 }

 }

 / / Por último, escriba a la burbuja del estado (newstate) que describe los datos restaurados

 FileOutputStream flujoout = new FileOutputStream (newState.getFileDescriptor ());

 DataOutputStream salida = new DataOutputStream (flujoout);

 out.writeUTF (mPlayerName);

 out.writeInt (mPlayerScore);

}

En este ejemplo, el appVersionCode parámetro pasado a [onRestore()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onRestore%28android.app.backup.BackupDataInput,%20int,%20android.os.ParcelFileDescriptor%29) no se utiliza. Sin embargo, es posible que desee usarlo si usted ha elegido para realizar copia de seguridad cuando la versión del usuario de la aplicación de hecho ha movido hacia atrás (por ejemplo, el usuario pasó de la versión 1.5 de su aplicación a 1,0). Para obtener más información, consulte la sección acerca de [Comprobación de la versión de datos de restauración](http://developer.android.com/guide/topics/data/backup.html#RestoreVersion) .

Para un ejemplo de implementación [BackupAgent](http://developer.android.com/reference/android/app/backup/BackupAgent.html) , consulte la [ExampleAgent](http://developer.android.com/resources/samples/BackupRestore/src/com/example/android/backuprestore/ExampleAgent.html) clase en la [copia de seguridad y restauración de](http://developer.android.com/resources/samples/BackupRestore/index.html) aplicación de la muestra.

**Extender BackupAgentHelper**

Usted debe construir su agente de copia de seguridad mediante [BackupAgentHelper](http://developer.android.com/reference/android/app/backup/BackupAgentHelper.html) si desea realizar copias de seguridad completas (ya sea [SharedPreferences](http://developer.android.com/reference/android/content/SharedPreferences.html) o [almacenamiento interno](http://developer.android.com/guide/topics/data/data-storage.html#filesInternal) ). La construcción de su agente de copia de seguridad con [BackupAgentHelper](http://developer.android.com/reference/android/app/backup/BackupAgentHelper.html) requiere mucho menos código que se extiende [BackupAgent](http://developer.android.com/reference/android/app/backup/BackupAgent.html) , porque usted no tiene que poner en práctica [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) y [onRestore()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onRestore%28android.app.backup.BackupDataInput,%20int,%20android.os.ParcelFileDescriptor%29) .

La implementación de [BackupAgentHelper](http://developer.android.com/reference/android/app/backup/BackupAgentHelper.html) debe utilizar uno o más ayudantes de copia de seguridad. Un ayudante de copia de seguridad es un componente especializado que [BackupAgentHelper](http://developer.android.com/reference/android/app/backup/BackupAgentHelper.html) citación para realizar copias de seguridad y restaurar las operaciones de un tipo particular de datos. El marco de Android en la actualidad dispone de dos ayudantes diferentes:

* [SharedPreferencesBackupHelper](http://developer.android.com/reference/android/app/backup/SharedPreferencesBackupHelper.html) de copia de seguridad [SharedPreferences](http://developer.android.com/reference/android/content/SharedPreferences.html) archivos.
* [FileBackupHelper](http://developer.android.com/reference/android/app/backup/FileBackupHelper.html) copia de seguridad de archivos de [almacenamiento interno](http://developer.android.com/guide/topics/data/data-storage.html#filesInternal) .

Puede incluir varios ayudantes en su [BackupAgentHelper](http://developer.android.com/reference/android/app/backup/BackupAgentHelper.html) , pero sólo una ayuda que se necesita para cada tipo de datos. Es decir, si usted tiene múltiples [SharedPreferences](http://developer.android.com/reference/android/content/SharedPreferences.html) archivos, entonces usted necesita un solo [SharedPreferencesBackupHelper](http://developer.android.com/reference/android/app/backup/SharedPreferencesBackupHelper.html) .

Por cada ayudante que desee agregar a su [BackupAgentHelper](http://developer.android.com/reference/android/app/backup/BackupAgentHelper.html) , usted debe hacer lo siguiente durante su [onCreate()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onCreate%28%29) método:

1. Una instancia en la instancia de la clase de ayuda que desee. En el constructor de la clase, se debe especificar el archivo correspondiente (s) que desea hacer copia de seguridad.
2. Llame [addHelper()](http://developer.android.com/reference/android/app/backup/BackupAgentHelper.html#addHelper%28java.lang.String,%20android.app.backup.BackupHelper%29) para añadir la ayuda a su [BackupAgentHelper](http://developer.android.com/reference/android/app/backup/BackupAgentHelper.html) .

Las secciones siguientes describen cómo crear un agente de copia de seguridad con cada uno de los ayudantes disponibles.

**Copia de seguridad de SharedPreferences**

Cuando una instancia de un [SharedPreferencesBackupHelper](http://developer.android.com/reference/android/app/backup/SharedPreferencesBackupHelper.html) , debe incluir el nombre de uno o más [SharedPreferences](http://developer.android.com/reference/android/content/SharedPreferences.html) archivos.

Por ejemplo, para respaldar un [SharedPreferences](http://developer.android.com/reference/android/content/SharedPreferences.html) archivo llamado "user\_preferences", un agente de copia de seguridad completa utilizando [BackupAgentHelper](http://developer.android.com/reference/android/app/backup/BackupAgentHelper.html) se parece a esto:

public class extends MyPrefsBackupAgent BackupAgentHelper {

 / / El nombre del archivo SharedPreferences

 estáticos Prefs final String = ""; user\_preferences

 / / Una de las claves para identificar el conjunto de datos de copia de seguridad

 PREFS\_BACKUP\_KEY static final String = "Preferencias";

 / / Asignar un ayudante y agregarlo a la agente de copia de seguridad

 @Override

 onCreate public void () {

 SharedPreferencesBackupHelper helper = new SharedPreferencesBackupHelper (esto, PREFS);

 addHelper (PREFS\_BACKUP\_KEY, ayudante);

 }

}

¡Eso es! Esa es su agente de copia de seguridad completa. El [SharedPreferencesBackupHelper](http://developer.android.com/reference/android/app/backup/SharedPreferencesBackupHelper.html) incluye todo el código necesario para respaldar y restaurar un [SharedPreferences](http://developer.android.com/reference/android/content/SharedPreferences.html) archivo.

Cuando el Administrador de copia de seguridad llama [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) y [onRestore()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onRestore%28android.app.backup.BackupDataInput,%20int,%20android.os.ParcelFileDescriptor%29) , [BackupAgentHelper](http://developer.android.com/reference/android/app/backup/BackupAgentHelper.html) llama a sus ayudantes de copia de seguridad para realizar copias de seguridad y restauración de los archivos especificados.

**Nota:** [SharedPreferences](http://developer.android.com/reference/android/content/SharedPreferences.html) son multi-hilo, lo que con seguridad se puede leer y escribir en el archivo de preferencias en la residencia de su agente de copia de seguridad y otras actividades.

**Copia de seguridad de otros archivos**

Cuando una instancia de un [FileBackupHelper](http://developer.android.com/reference/android/app/backup/FileBackupHelper.html) , debe incluir el nombre de uno o más archivos que se guardan en la aplicación de [almacenamiento interno](http://developer.android.com/guide/topics/data/data-storage.html#filesInternal) (según lo especificado por [getFilesDir()](http://developer.android.com/reference/android/content/ContextWrapper.html#getFilesDir%28%29) , que es el mismo lugar donde [openFileOutput()](http://developer.android.com/reference/android/content/Context.html#openFileOutput%28java.lang.String,%20int%29) escribe los archivos).

Por ejemplo, para copias de seguridad de dos archivos llamados "resultados" y "Estadísticas", un agente de copia de seguridad mediante [BackupAgentHelper](http://developer.android.com/reference/android/app/backup/BackupAgentHelper.html) se parece a esto:

public class extends MyFileBackupAgent BackupAgentHelper {

 / / El nombre del archivo SharedPreferences

 final String estática TOP\_SCORES = "resultados";

 estáticos PLAYER\_STATS final String = "Estadísticas";

 / / Una de las claves para identificar el conjunto de datos de copia de seguridad

 FILES\_BACKUP\_KEY static final String = "myfiles";

 / / Asignar un ayudante y agregarlo a la agente de copia de seguridad

 onCreate vacío () {

 FileBackupHelper helper = new FileBackupHelper (esto, TOP\_SCORES, PLAYER\_STATS);

 addHelper (FILES\_BACKUP\_KEY, ayudante);

 }

}

El [FileBackupHelper](http://developer.android.com/reference/android/app/backup/FileBackupHelper.html) incluye todo el código necesario para realizar copias de seguridad y restaurar archivos que se guardan en la aplicación de [almacenamiento interno](http://developer.android.com/guide/topics/data/data-storage.html#filesInternal) ..

Sin embargo, la lectura y escritura a los archivos de almacenamiento interno **no** es **multi-hilo.** Para asegurarse de que su agente de copia de seguridad no leer o escribir archivos en el mismo tiempo que sus actividades, debe utilizar las declaraciones sincronizados cada vez que realiza una lectura o escritura. Por ejemplo, en cualquier actividad en la que leer y escribir en el fichero, se necesita un objeto para usar como la cerradura intrínseca de las normas de sincronización:

**Información de Interés**

Una matriz de longitud cero es más liviana que un objeto normal, por lo que es ideal para un bloqueo intrínseco.

/ / Objeto de bloqueo intrínseco

Objeto static final [] sDataLock = new Object [0];

A continuación, cree una declaración sincronizado con este bloqueo cada vez que leer o escribir los archivos. Por ejemplo, aquí hay una declaración sincronizado para escribir la última puntuación en un juego en un archivo:

try {

 sincronizado (MyActivity.sDataLock) {

 Archivo archivo de datos = new File ( [getFilesDir()](http://developer.android.com/reference/android/content/Context.html%22%20%5Cl%20%22getFilesDir%28%29) , TOP\_SCORES);

 RandomAccessFile raFile = new RandomAccessFile (archivo de datos, "rw");

 raFile.writeInt (puntuación);

 }

} Catch (Exception e) {

 Log.e (TAG, "No se puede escribir en el archivo");

}

Usted debe sincronizar sus estados de leer con el mismo bloqueo.

Entonces, en su [BackupAgentHelper](http://developer.android.com/reference/android/app/backup/BackupAgentHelper.html) , se debe reemplazar [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) y [onRestore()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onRestore%28android.app.backup.BackupDataInput,%20int,%20android.os.ParcelFileDescriptor%29) para sincronizar la copia de seguridad y restaurar las operaciones con el bloqueo intrínseca misma. Por ejemplo, el MyFileBackupAgent ejemplo desde arriba necesita los siguientes métodos:

@Override

pública onBackup void (ParcelFileDescriptor oldstate, BackupDataOutput de datos,

 ParcelFileDescriptor newstate) throws IOException {

 / / Mantener el bloqueo, mientras que el FileBackupHelper realiza copia de seguridad

 sincronizado (MyActivity.sDataLock) {

 super.onBackup (oldstate, datos, newstate);

 }

}

@Override

public void OnRestore (BackupDataInput datos, int appVersionCode,

 ParcelFileDescriptor newstate) throws IOException {

 / / Mantener el bloqueo, mientras que el FileBackupHelper restaura el archivo

 sincronizado (MyActivity.sDataLock) {

 super.onRestore (datos, appVersionCode, newstate);

 }

}

Eso es todo. Todo lo que necesitas hacer es agregar su [FileBackupHelper](http://developer.android.com/reference/android/app/backup/FileBackupHelper.html) en el [onCreate()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onCreate%28%29) y anular el método [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) y [onRestore()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onRestore%28android.app.backup.BackupDataInput,%20int,%20android.os.ParcelFileDescriptor%29) para sincronizar la lectura y escritura.

Para un ejemplo de implementación [BackupAgentHelper](http://developer.android.com/reference/android/app/backup/BackupAgentHelper.html) con [FileBackupHelper](http://developer.android.com/reference/android/app/backup/FileBackupHelper.html) , consulte la FileHelperExampleAgent clase en la [copia de seguridad y restauración de](http://developer.android.com/resources/samples/BackupRestore/index.html) aplicación de la muestra.

**Comprobación de la versión de restauración de datos**

Cuando el Administrador de copia de seguridad guarda los datos de almacenamiento en la nube, que incluye automáticamente la versión de su aplicación, según se define en el archivo de manifiesto de [android:versionCode](http://developer.android.com/guide/topics/manifest/manifest-element.html#vcode) atributo. Antes de que el Administrador de copia de seguridad llama a su agente de seguridad para restaurar los datos, se ve en el [android:versionCode](http://developer.android.com/guide/topics/manifest/manifest-element.html#vcode) de la aplicación instalada y lo compara con el valor registrado en el conjunto de restauración de datos. Si la versión registrada en el conjunto de restauración de datos es más *reciente* que la versión de la aplicación en el dispositivo, el usuario ha rebajado su aplicación. En este caso, el Administrador de copia de seguridad se abortará la operación de restauración para su aplicación y no llame a su [onRestore()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onRestore%28android.app.backup.BackupDataInput,%20int,%20android.os.ParcelFileDescriptor%29) método, debido a que el conjunto de restauración se considera de sentido a una versión anterior.

Puede reemplazar este comportamiento con el [android:restoreAnyVersion](http://developer.android.com/guide/topics/manifest/application-element.html#restoreany) atributo. Este atributo es " true "o" false "para indicar si desea restaurar la aplicación, independientemente de la versión de conjunto de restauración. El valor por defecto es false. Si se define que se trata de " true ", entonces el Administrador de copia de seguridad de ignorar el [android:versionCode](http://developer.android.com/guide/topics/manifest/manifest-element.html#vcode) y llame a su [onRestore()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onRestore%28android.app.backup.BackupDataInput,%20int,%20android.os.ParcelFileDescriptor%29) método en todos los casos. De este modo, se puede verificar manualmente la versión para la diferencia en su [onRestore()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onRestore%28android.app.backup.BackupDataInput,%20int,%20android.os.ParcelFileDescriptor%29) el método y tomar las medidas necesarias para hacer compatibles los datos si el conflicto versiones.

Para ayudarle a manejar diferentes versiones durante una operación de restauración, el [onRestore()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onRestore%28android.app.backup.BackupDataInput,%20int,%20android.os.ParcelFileDescriptor%29) método que pasa el código de la versión incluida con los datos de restauración establecidos como el appVersionCode parámetro. A continuación, puede consultar el código de la aplicación actual versión con el [PackageInfo.versionCode](http://developer.android.com/reference/android/content/pm/PackageInfo.html#versionCode) campo. Por ejemplo:

Información PackageInfo;

try {

 String nombre = [getPackageName](http://developer.android.com/reference/android/content/ContextWrapper.html#getPackageName%28%29) ();

 info = [getPackageManager](http://developer.android.com/reference/android/content/ContextWrapper.html#getPackageManager%28%29) . () [getPackageInfo](http://developer.android.com/reference/android/content/pm/PackageManager.html#getPackageInfo%28java.lang.String,%20int%29) (nombre, 0);

Captura} (NameNotFoundException nnfe) {

 info = null;

}

int versión;

if (info!nulo

 = versión info.versionCode;

}

Después, simplemente comparar la version adquirida de [PackageInfo](http://developer.android.com/reference/android/content/pm/PackageInfo.html) a la appVersionCode pasó a [onRestore()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onRestore%28android.app.backup.BackupDataInput,%20int,%20android.os.ParcelFileDescriptor%29) .

**Precaución:** Asegúrese de que usted entiende las consecuencias de la creación [android:restoreAnyVersion](http://developer.android.com/guide/topics/manifest/application-element.html#restoreany) de " true "para su aplicación. Si cada versión de la aplicación que permite copias de seguridad no responda adecuadamente en cuenta las variaciones en el formato de los datos durante [onRestore()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onRestore%28android.app.backup.BackupDataInput,%20int,%20android.os.ParcelFileDescriptor%29) , entonces los datos en el dispositivo podría ser guardado en un formato compatible con la versión actualmente instalada en el dispositivo.

**Solicitud de copia de seguridad**

Usted puede solicitar una operación de copia de seguridad en cualquier momento llamando al [dataChanged()](http://developer.android.com/reference/android/app/backup/BackupManager.html#dataChanged%28%29) . Este método indica el Administrador de copia de seguridad que le gustaría hacer copias de seguridad de sus datos con su agente de copia de seguridad. El Administrador de copia de seguridad a continuación, llama a su agente de copia de seguridad de [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) el método en un momento oportuno en el futuro. Por lo general, usted debe solicitar una copia de seguridad cada vez que sus datos cambian (por ejemplo, cuando el usuario cambia la preferencia de aplicación que desea hacer copia de seguridad). Si usted llama [dataChanged()](http://developer.android.com/reference/android/app/backup/BackupManager.html#dataChanged%28%29) varias veces consecutivas, antes de que las peticiones del Administrador de copia de seguridad una copia de seguridad de su agente, su agente sigue recibiendo sólo una llamada a [onBackup()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup%28android.os.ParcelFileDescriptor,%20android.app.backup.BackupDataOutput,%20android.os.ParcelFileDescriptor%29) .

**Nota:** Durante el desarrollo de su aplicación, usted puede solicitar una copia de seguridad y poner en marcha una operación de copia de seguridad inmediata con la [bmgr herramienta](http://developer.android.com/guide/developing/tools/bmgr.html) .

**Solicitud de restauración**

Durante la vida normal de la aplicación, no es necesario solicitar una operación de restauración. Ellos sistema comprueba automáticamente si los datos de copia de seguridad y lleva a cabo una restauración cuando su aplicación está instalada. Sin embargo, usted puede solicitar manualmente una operación de restauración llamando [requestRestore()](http://developer.android.com/reference/android/app/backup/BackupManager.html#requestRestore%28android.app.backup.RestoreObserver%29) , si es necesario. En cuyo caso, el Administrador de copia de seguridad llama a su [onRestore()](http://developer.android.com/reference/android/app/backup/BackupAgent.html#onRestore%28android.app.backup.BackupDataInput,%20int,%20android.os.ParcelFileDescriptor%29) la implementación, pasando por los datos del conjunto actual de datos de copia de seguridad.

**Nota:** Durante el desarrollo de su aplicación, usted puede solicitar una operación de restauración con la [bmgr herramienta](http://developer.android.com/guide/developing/tools/bmgr.html) .

**Prueba de su agente de copia de seguridad**

Una vez que haya aplicado a su agente de copia de seguridad, puede probar la copia de seguridad y restaurar la funcionalidad con el siguiente procedimiento, usando [bmgr](http://developer.android.com/guide/developing/tools/bmgr.html) .

1. Instale la aplicación en una imagen de sistema adecuado para Android
	* Si se utiliza el emulador, crear y utilizar un AVD con Android 2.2 (Nivel API 8).
	* Si se utiliza un dispositivo, el dispositivo debe estar ejecutando Android 2.2 o superior y han construido adentro Google Escuchar
2. Asegúrese de que copia de seguridad está activado
	* Si se utiliza el emulador, se puede habilitar copia de seguridad con el siguiente comando desde el SDK tools/ trazado:

adb shell BMGR enable verdad

* + Si se utiliza un dispositivo, abra la **configuración del** sistema, seleccione **de privacidad,** a continuación, **activar** la función de **seguridad de mis datos** y **restauración automáticas.**
1. Abra la aplicación e inicializar algunos datos

Si usted ha aplicado adecuadamente copia de seguridad en su aplicación, entonces debe solicitar una copia de seguridad cada vez que cambian los datos. Por ejemplo, cada vez que el usuario cambia algunos datos, su aplicación debe llamar a [dataChanged()](http://developer.android.com/reference/android/app/backup/BackupManager.html#dataChanged%28%29) , que añade una petición de copia de seguridad a la cola de Backup Manager. Para propósitos de prueba, también se puede hacer una solicitud con la siguiente bmgr comando:

adb shell BMGR copia de seguridad *your.package.name*

1. Iniciar una operación de copia de seguridad:

adb shell BMGR plazo

Esto obliga a que el Administrador de copia de seguridad para llevar a cabo todas las peticiones de copia de seguridad que están en la cola.

1. Desinstalar la aplicación:

adb desinstalación *your.package.name*

1. Vuelva a instalar la aplicación.

Si su agente de copia de seguridad tiene éxito, todos los datos que inicializados en el paso 4 se restaura.