Handling Persisting Data

Learning Objective

- Learn all the best practices in persisting your data at Android applications with several options using Handling and persisting data.
- learn these several data storage options in android

Introduction

Android gives many options for you to save your application data. The option you choose depends on specific needs, such as storage your data needs, type of data you need to store, and if you want the data to be private to your application or accessible to other any other applications or users.

Storage Options

This different data storage options that are given by Android are:

- Shared Preferences

  It stores the primitive data that is private in key-value pairs.
- Internal Storage

  It stores private data on the device memory (which cannot accessed by other users or applications)
- External Storage

  It stores public data on the shared external storage or any disk (which can be shared by external users and applications)
- SQLite Databases

  It stores the structured data in a private database.
- Network Connection

  It stores data on the web with your own network server.

Internal Storage

It is storage that is not accessible by the any outsider or user, except developer. When app is uninstalled, the system removes all your apps files. It is mostly used when the developer wants no other user to access his/her application. Internal storage is the storage of the private data only on the device memory. These files by default are private and are accessed by only the developer's application and get deleted, when he/she deletes your application.

External Storage

In built shared storage which is "accessible by any user by plugging in a USB cable and mounting it as a drive on a host computer".

Example: Removable storage.

Example: SD Card.

file can be read by bufferreader class which has readline method.

SQLite Database

SQLite is a opensource SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation. It is used to perform database operations on android devices such as storing, manipulating or retrieving persistent data from the database.SQLite supports all the relational database features. In order to access this database, you don't need to establish any kind of connections for it like JDBC,ODBC e.t.c

Method:

Sqlite consists of 2 classes: Manager and helper

The methods of helper are:

public abstract void onCreate(SQLiteDatabase db)	It is called only once when database is created for the first time.

public abstract void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)	It is called when database needs to be when upgraded.

Table-1 Methods of helper class

```
private static final String CREATE_TABLE = "create table " + TABLE_NAME + "(" +
EMP_NAME + " TEXT NOT NULL, "
        + EMP_CITY + " TEXT);";
public Helper(Context context) {
    super(context, DB_NAME, null, DB_VERSION);
}

@Override
public void onCreate(SQLiteDatabase db) {
   db.execSQL(CREATE_TABLE);
}
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS" + TABLE_NAME );
    onCreate(db);
}
```

The methods of Manager are:

void execSQL(String sql)	It is used to execute the sql query not select query.
long insert(String table, String nullColumnHack, ContentValues values)	It is used insert a record on the database. The table specifies the table name, nullColumnHack doesn't allow any null values. If second argument is null, android will store null values if values are empty. The third argument specifies the values to be stored.
int update(String table, ContentValues values, String whereClause, String[] whereArgs)	It is used to update a row.
Cursor query(String table,	It is used to return a cursor over the resultset.

String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy)	

Table-2 Methods of manager class

```
public void insert(String name, String city) {
    ContentValues contentValue = new    ContentValues();
    contentValue.put(Helper.EMP_NAME, name);
    contentValue.put(Helper.EMP_CITY, city);
    database.insert(Helper.TABLE_NAME, null, contentValue);
}

public int update(String name, String city) {
    ContentValues contentValues = new ContentValues();
    contentValues.put(Helper.EMP_NAME, name);
    contentValues.put(Helper.EMP_CITY, city);
    int i = database.update(Helper.TABLE_NAME, contentValues, Helper.EMP_CITY + "
=' " + city + " ' ", null);
    return i;
}

public void delete(String name) {
    database.delete(Helper.TABLE_NAME, Helper.EMP_NAME + "=" + name, null);
}

public Cursor fetch() {
    String[] columns = new String[] { Helper.EMP_NAME, Helper.EMP_CITY};
    Cursor cursor = database.query(Helper.TABLE_NAME, columns, null, null, null, null,
null);
    if (cursor != null) {
        cursor.moveToFirst();
    }
    return cursor;
}
```

Shared Preferences

Shared Preferences gives you the way to save and retrieve data in the form of key,value pair.

In order to use shared preferences, one needs to call a method getSharedPreferences() .

It returns a SharedPreference instance that points to the file containing the values of preferences.

```
SharedPreferences sp = getSharedPreferences(MyPREFERENCES, Context.MODE_PRIVATE);
```

The first parameter is the key and the second parameter is the MODE.Others are:

Mode
MODE_APPEND **It appends the new preferences with the already existing preferences**
MODE_ENABLE_WRITE_AHEAD_LOGGING **When Database open flag is set , it would enable write ahead logging by default**
MODE_MULTI_PROCESS **It will check for modification of preferences even if the sharedpreference instance has already been loaded.**
MODE_PRIVATE **The file can only be accessed using calling application when this mode is been set.**
MODE_WORLD_READABLE **This mode will allow other applications to read the preferences.(Makes reading public)**
MODE_WORLD_WRITEABLE **This mode will allow other applications to write the preferences.(Makes writing public)**

One can save anything sharedpreferences by using SharedPreferences.Editor class.

Methods of editor class:

Mode
apply() **This abstract method will commit your changes back from editor to the sharedPreference object you are calling**
clear() **This method will be removing all values from the editor**
remove(String key) **This method will be removing the value whose key has been passed as a parameter**
putLong(String key, long value) **This method will save a long value in a preference editor**
putInt(String key, int value) **This method will be saving an integer value in a preference editor**
putFloat(String key, float value) **This method will be saving a float value in a preference editor**

Table-3 Methods of editor class

Let us Sum Up

Let's take a quick recap with summary:

- Storage options: Android gives many options for you to save your application data . This options are:
- Internal Storage:This option stores private data on the device memory (which cannot accessed by other users or applications).
- External Storage:This option stores public data on the shared external storage or any disk (which can be shared by external users and applications).
- Shared Preferences: This option stores the primitive data that is private in key-value pairs.
- SQLite Databases:This option stores the structured data in a private database.

Further Reading

Recommended links: http://developer.android.com/

Recommended Books:

1. Teach.Yourself.Android.Application.Development.in.24. Hours. 2nd.Edition.
2. Learning Android-Book by Marko Gargenta (2011)

Acknowledgement: “The content in this module is modifications based on work created and shared by the Android Open-Source Project and used according to terms described in the Creative Commons 2.5 Attribution License.”