snatik / **android-storage**

Create, read, delete, append, encrypt files and more, on internal or external disk spaces with a really simple API

#android #library #storage

118 commits | **3** branches | **0** releases | **5** contributors | Apache-2.0

Branch: **master** | New pull request | Find file | Clone or download

@s **sromku** Open file in default app | Latest commit `422a2b9` on Jul 20

app	Open file in default app	5 months ago
assets	update preview	6 months ago
gradle/wrapper	updated gradle version	7 months ago
storage	bump to 2.1.0	6 months ago
.gitignore	fixed gitignore	7 months ago
CHANGELOG.md	Added changlog	6 months ago
LICENSE	Initial commit	4 years ago
README.md	Updated readme	6 months ago
build.gradle	fixed gradle files	7 months ago
gradle.properties	Update structure	2 years ago
gradlew	Update structure	2 years ago
gradlew.bat	Update structure	2 years ago
settings.gradle	update lib name	6 months ago

README.md

android-storage

Library to create, read, delete, append, encrypt files and more, on internal or external disk spaces with a really simple API.

Latest Release

Download

```
dependencies {
    compile 'com.snatik:storage:2.1.0'
}
```

Don't forget to update `AndroidManifest.xml` and add next line:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Usage

```
// init
Storage storage = new Storage(getApplicationContext());

// get external storage
String path = storage.getExternalStorageDirectory();

// new dir
String newDir = path + File.separator + "My Sample Directory";
storage.createDirectory(newDir);
```

Check all options, scroll down ;)

Sample app

The app has some simple UI screens and uses the storage library. This is just an example of what can be done with this lib.

Options

- Easy define Internal or External storage
- Create directory
- Create file
- Read file content
- Append content to file
- Copy
- Move
- Delete directory
- Delete file
- Get files
- More options
- Encrypt the file content

Initialize

```
Storage storage = new Storage(getApplicationContext());
```

Work on **External Storage**.

- Check if external writable

  ```
  boolean isWritable = storage.isExternalWritable();
  ```

- Root external storage path

  ```
  String path = storage.getExternalStorageDirectory();
  ```

- If you want to use a particular public directory

  ```
  Storage storage = SimpleStorage.getExternalStorage(Environment.DIRECTORY_PICTURES);
  ```

Work on **Internal Storage**.

- Directory for storing app internal files (documentation):

  ```
  String path = SimpleStorage.getInternalFilesDirectory();
  ```

- Cache dir

  ```
  String path = SimpleStorage.getInternalCacheDirectory();
  ```

- Root internal storage dir

  ```
  String path = SimpleStorage.getInternalRootDirectory();
  ```

Create directory

- Create directory

  ```
  storage.createDirectory(path);
  ```

- Create directory and **override** the existing one.

  ```
  storage.createDirectory(path, true);
  ```

Create file

Create a new file with the content in it.

```
storage.createFile(path, "some content of the file");
```

The `content` of the file can be one of the next types:

- `String`
- `byte[]`
- `Bitmap`
- `Storable`

Read file

Read the content of any file to byte array.

```
byte[] bytes = storage.readFile(path);
```

Read the content of the file to String.

```
String content = storage.readTextFile(path);
```

Append content to file

```
storage.appendFile(path, "more new data");
```

You can append:

- `String`
- `byte[]`

Copy

```
storage.copy(fromPath, toPath);
```

Move

```
storage.move(fromPath, toPath);
```

Delete directory

```
storage.deleteDirectory(path);
```

Delete file

```
storage.deleteFile(path);
```

Get files

- Get files in ordered way by: `name` , `date` , `size`

  ```
  List<File> files = storage.getFiles(path, OrderType.DATE);
  ```

- Get files and filter by regular expression:

  ```
  String regex = ...;
  List<File> files = storage.getFiles(path, regex);
  ```

- Get all nested files (without the directories)

  ```
  List<File> files = storage.getNestedFiles(path);
  ```

More...

- Is directory exists

  ```
  boolean dirExists = storage.isDirectoryExists(path);
  ```

- Is file exists

  ```
  boolean fileExists = storage.isFileExist(path);
  ```

Security configuration

You can write and read files while the content is **encrypted**. It means, that no one can read the data of your files from external or internal storage.

You will continue using the same api as before. The only thing you need to do is to configure the Simple Storage library before the you want to create/read encrypted data.

```
// set encryption
String IVX = "abcdefghijklmnop"; // 16 lenght - not secret
String SECRET_KEY = "secret1234567890"; // 16 lenght - secret
byte[] SALT = "0000111100001111".getBytes(); // random 16 bytes array

// build configuratio
EncryptConfiguration configuration = new EncryptConfiguration.Builder()
        .setEncryptContent(IVX, SECRET_KEY, SALT)
        .build();

// configure the simple storage
storage.setEncryptConfiguration(configuration);
```

Now, you can create a new file with content and the content will be automatically encrypted.
You can read the file and the content will be decrypted.

Example

Create file with next content `"this is the secret data"` :

```
storage.setEncryptConfiguration(configuration);
storage.createFile(path, "this is the secret data");
```

If we open the file to see it's content then it we will something like this: `„f°α�TG†_i�ỉp` . It looks good :)

And now, read the file data with the same api:

```
storage.setEncryptConfiguration(configuration);
String content = storage.readTextFile(path);
```

You will see that the content will be: `"this is the secret data"` .

Tests

Just play and check the sample app ;)

Follow us

Twitter URL Twitter Follow