Lab 14
Android Data Storage and File Upload

NetDB
CS, NTHU,
Fall, 2013

Outline

- Data Storage
- File Upload
- Smail v2

Outline

- Data Storage
- File Upload
- Smail v2

Data Storage in Android

- Android provides several options for you to save persistent application data
- The solution you choose depends on your specific needs, such as whether the data should be private to your application or accessible to other applications (and the user) and how much space your data requires

Storage Options

- Shared Preferences
  - Store private primitive data in key-value pairs
- Internal Storage
  - Store private data on the device memory
- External Storage
  - Store public data on the shared external storage
- SQLite Databases
  - Store structured data in a private database
- Network Connection
  - Store data on the web with your own network server

Outline

- Data Storage
  - Shared Preferences
  - Internal Storage
  - External Storage
- File Upload
- Smail v2

Shared Preferences

- The SharedPreferences class provides a general framework that allows you to save and retrieve persistent **key-value pairs** of primitive data types such as booleans, floats, ints, longs, and strings
- This data will persist across user sessions (even if your application is killed)

Get a SharedPreferences Object

- getSharedPreferences()
    - Use this if you need multiple preferences files identified by name, which you specify with the first parameter
- getPreferences()
    - Use this if you need only one preferences file for your Activity. Because this will be the only preferences file for your Activity, you don't supply a name
- Only one instance of the SharedPreferences object is returned to any callers for the same name, meaning they will see each other's edits as soon as they are made

Read Values

- To read values, call methods such as getBoolean(), getInt(), etc. of the instance of SharedPreferences

```
SharedPreferences settings = getSharedPreferences(“MyPrefsFile”, 0);
boolean silent = settings.getBoolean("silentMode", false);
```

Write Values

1. Call edit() to get a SharedPreferences.Editor
2. Add values with methods such as putBoolean() and putString()
3. Commit the new values with commit()

```
SharedPreferences settings = getSharedPreferences(“MyPrefsFile”, 0);
SharedPreferences.Editor editor = settings.edit();
editor.putBoolean("silentMode", mSilentMode);

editor.commit();
```

Outline

- Data Storage
    - Shared Preferences
    - Internal Storage
    - External Storage
- File Upload
- Smail v2

Internal Storage

- You can save files directly on the device's internal storage

- By default, files saved to the internal storage are private to your application and other applications cannot access them (nor can the user)

- When the user uninstalls your application, these files are removed

Write a File to the Internal Storage

1. Call openFileOutput() with the name of the file and the operating mode. This returns a FileOutputStream

2. Write to the file with write()

3. Close the stream with close()

Write a File to the Internal Storage

```
String FILENAME = "hello_file";
String string = "hello world!";

FileOutputStream fos = openFileOutput(FILENAME,
Context.MODE_PRIVATE);
fos.write(string.getBytes());
fos.close();
```

Read a File from the Internal Storage

1. Call openFileInput() and pass it the name of the file to read. This returns a FileInputStream

2. Read bytes from the file with read()

3. Then close the stream with close()

Outline

- Data Storage
    - Shared Preferences
    - Internal Storage
    - External Storage
- File Upload
- Smail v2

External Storage

- This can be a removable storage media (such as an SD card) or an internal (non-removable) storage
- Files saved to the external storage are **world-readable** and can be modified by the user when they enable USB mass storage to transfer files on a computer
- When the user uninstalls your application, this directory and all its contents are deleted

Getting Access to External Storage

- In order to read or write files on the external storage, your app must acquire the READ_EXTERNAL_STORAGE or WRITE_EXTERNAL_STORAGE system permissions. For example:

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

- Beginning with Android 4.4, these permissions are not required if you're reading or writing only files that are private to your app

Checking Media Availability

- Before you do any work with the external storage, you should always call getExternalStorageState() to check whether the media is available

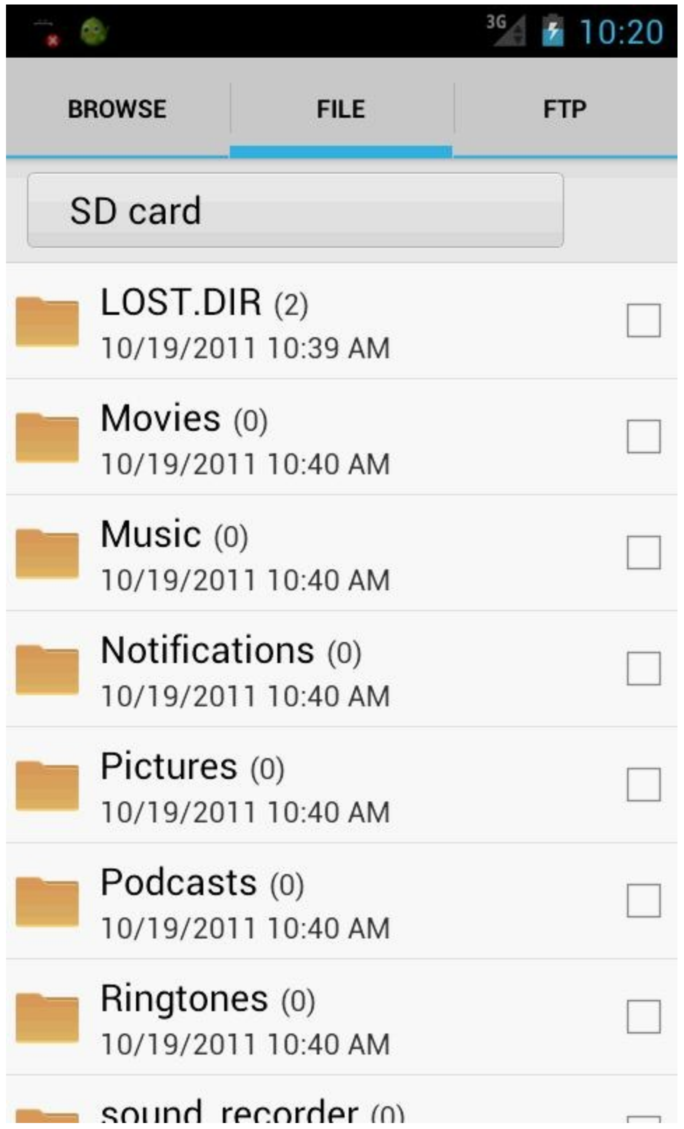
Saving Files That Can Be Shared with Other Apps

- Generally, new files that the user may acquire through your app should be saved to a "public" location on the device where other apps can access them and the user can easily copy them from the device

- When doing so, you should use to one of the shared public directories, such as Music/, Pictures/, and Ringtones/

Saving Files That Can Be Shared with Other Apps

- To get a File representing the appropriate public directory, call getExternalStoragePublicDirectory(), passing it the type of directory you want
  - such as DIRECTORY_MUSIC, DIRECTORY_PICTURES, DIRECTORY_RINGTONES, or others



LOST.DIR (2)
10/19/2011 10:39 AM
Movies (0)
10/19/2011 10:40 AM
Music (0)
10/19/2011 10:40 AM
Notifications (0)
10/19/2011 10:40 AM
Pictures (0)
10/19/2011 10:40 AM
Podcasts (0)
10/19/2011 10:40 AM
Ringtones (0)
10/19/2011 10:40 AM
sound_recorder (0)

Saving Files That Are App-Private

- If you are handling files that are not intended for other apps to use (such as graphic textures or sound effects used by only your app), you should use a private storage directory on the external storage by calling getExternalFilesDir()

Saving Files That Are App-Private

- Although the directories provided by getExternalFilesDir() are not accessible by the MediaStore content provider, other apps with the READ_EXTERNAL_STORAGE permission can access all files on the external storage, including these

- If you need to completely restrict access for your files, you should instead write your files to the internal storage

Outline

- Data Storage
- File Upload
- Smail v2

File Storage

- Two ways to store your static media files
  - File system (ex. Amazon S3)
  - Database blob field

Binary Large Object (BLOB)

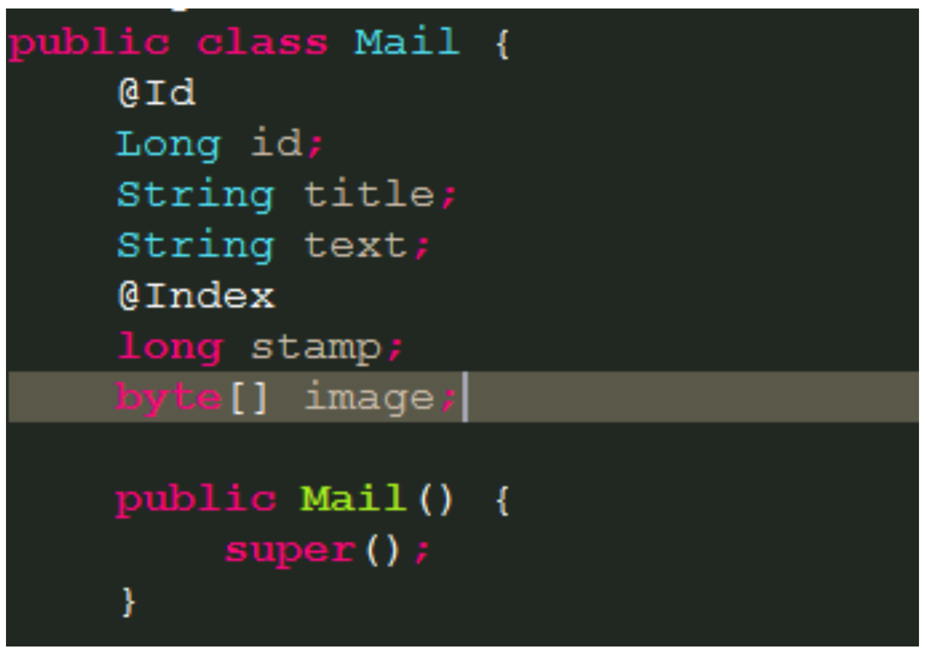
- A data type that can store binary data
- Often used to store image, video or audio files

Store Files to GAE

- Google Cloud Storage
    - Storing and retrieval of any amount of data and at any time
- Blobstore
    - Blobstore creates a blob from the file's contents and returns an blob key that you can later use to serve the blob
- Datastore
    - Directly store the data into gae datastore
    - The size of your data is limited to 1MB

Storing Blob with Objectify

- In your domain object class, the byte[] variables will be automatically stored as a blob by Objectify



```
public class Mail {
    @Id
    Long id;
    String title;
    String text;
    @Index
    long stamp;
    byte[] image;

    public Mail() {
        super();
    }
```

Outline

- Data Storage
- File Upload
- Smail v2