Persistent Data

Shared Preferences

Stores primitive key-value pairs somewhere in the directory structure. It is used to save just primitive data (int, float, boolean, …). You can have one or multiple preference files.

```
public class PrefTest extends Activity {
    public static final String PREFNAME1 = "Pref1";
    public static final String PREFNAME2 = "Pref2";

    @Override
    protected void onCreate(Bundle state){
       super.onCreate(state);
       // do whatever you want here

       // Restore preferences
       // can also be done in the onResume() callback
       SharedPreferences settings1 = getSharedPreferences(PREFNAME1, 0);
       SharedPreferences settings1 = getSharedPreferences(PREFNAME2, 0);

       boolean v1 = settings1.getBoolean("key1", false);
       int v2 = settings2.getInt("key2", 55);

       // use the retrieved values somehow

    }

    // you can do this in the onPause() method
    @Override
    protected void onStop(){
       super.onStop();

      // Save values at this point ; handled as a transaction
      // needs an Editor object to handle the transaction

      SharedPreferences settings1 = getSharedPreferences(PREFNAME1, 0);
      SharedPreferences.Editor editor1 = settings1.edit();
      editor.putBoolean("key1", true) ;
      editor1.commit();


      SharedPreferences settings2 = getSharedPreferences(PREFNAME2, 0);
      SharedPreferences.Editor editor2 = settings2.edit();
      editor.putInt("key2", 15) ;
      editor1.commit();


    }
}
```

You can even listen to shared preferences changes using the following method.

```
void registerOnSharedPreferenceChangeListener
(SharedPreferences.OnSharedPreferenceChangeListener listener)
```

To unregister, use the following method :

```
void unregisterOnSharedPreferenceChangeListener
(SharedPreferences.OnSharedPreferenceChangeListener listener)
```

The method to implement in order to handle changes is :

```
void onSharedPreferenceChanged (SharedPreferences sharedPreferences,
                String key)
```

The key of the preference that was changed. Obviously, if you have multiple shared preference files, then the first reference points to the corresponding file that has been changed.

Files

You can use normal files in order to store information.

```
String FILENAME = "myFile";

// you can use Context.MODE_APPEND to append data to the file
FileOutputStream fos = openFileOutput(FILENAME, Context.MODE_PRIVATE);
fos.write(string.getBytes());
fos.close();

byte[] b = new byte[100] ;
FileInputStream fis = openFileInput(FILENAME, Context.MODE_PRIVATE) ;
fis.read(b, 0, 100) ;
fis.close() ;
```

This uses the **internal storage** in order to store the data. Obviously, usual I/O classes (BufferedOutputStream, ObjectOutputStream, DataOutputStream, …) can and should be used in order to format data correctly when storing it into a file.

Interesting methods that can be used (belong to the Context class – Activity is a Context): getFilesDir(), getDir(), deleteFile(). Look them up in the documentation !

External Storage

1. Add the right permissions to your application

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

2. Check whether external storage is available for writing or reading, depending on your use

   ```
   Environment.getExternalStorageState();
   Environment.MEDIA_MOUNTED.equals(state)
   ```

3. Get access to the external storage and create a file

   ```
   File file = new File(Environment.getExternalStorageDirectory(),
   albumName);
   ```

   You can even use «getExternalStoragePublicDirectory » (look it up in the Environment class)

4. Use the I/O java classes to read or write data.

Light Database

Explanations and code snippets are given in the slides !