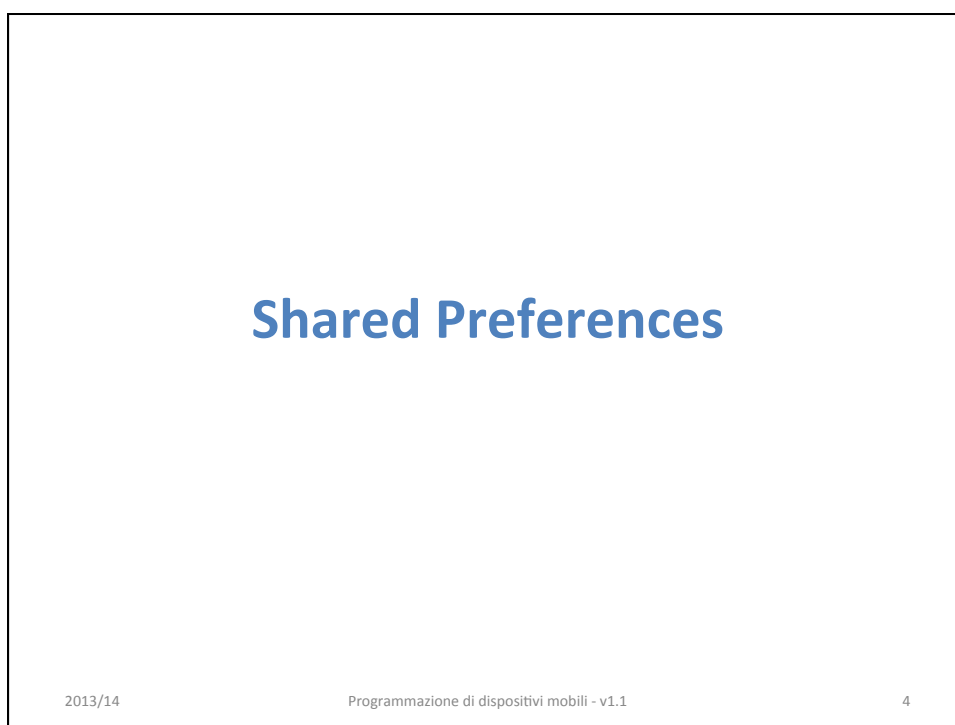


## Gestione dati

## Data storage

- Shared Preferences
  - Memorizza dati primitivi in coppie chiave-valore
- Internal Storage
  - Memorizza dati privati nella memoria del dispositivo
- External Storage
  - Memorizza dati pubblici nella memoria esterna condivisa
- SQLite Databases
  - Memorizza dati strutturati in database privati
- Network Connection
  - Memorizza dati su web server.



## Preferenze

- Framework per la gestione delle preferenze delle applicazioni
- `public SharedPreferences`  
`getSharedPreferences` (String name, int mode)
  - metodo di Context
  - `name`: identificatore delle preferenze
  - `mode`: modalità di accesso
    - Context.MODE\_PRIVATE
    - Context.MODE\_WORLD\_READABLE
    - Context.MODE\_WORLD\_WRITEABLE
- `public SharedPreferences` `getPreferences` (int mode)
  - dati locali all'Activity

2013/14

Programmazione di dispositivi mobili - v1.1

5

## SharedPreferences

- boolean `getBoolean`(String key, boolean defValue)
- float `getFloat`(String key, float defValue)
- int `getInt`(String key, int defValue)
- long `getLong`(String key, long defValue)
- String `getString`(String key, String defValue)
- Map<String, ?> `getAll`()
- SharedPreferences.Editor `edit`()
  - ritorna un editor per modificare le preferenze

2013/14

Programmazione di dispositivi mobili - v1.1

6

## Esempio

```
SharedPreferences prefs =  
    getSharedPreferences(MY_PREFERENCES,  
        Context.MODE_PRIVATE);  
String textData = prefs.getString("", "No Preferences!");
```

## Editor

- `SharedPreferences.Editor` **putBoolean**(String key, boolean value)
- `SharedPreferences.Editor` **putFloat**(String key, float value)
- `SharedPreferences.Editor` **putInt**(String key, int value)
- `SharedPreferences.Editor` **putLong**(String key, long value)
- `SharedPreferences.Editor` **putString**(String key, String value)
  - preparano i valori per la scrittura
- `SharedPreferences.Editor` **clear**()
  - prepara la cancellazione di tutti i valori
- `SharedPreferences.Editor` **remove**(String key)
  - Prepara la rimozione di un valore
- void **apply**()
- boolean **commit**()
  - Eseguono delle modifiche

## Esempio

```
SharedPreferences prefs =  
    getSharedPreferences(MY_PREFERENCES,  
                        Context.MODE_PRIVATE);  
  
SharedPreferences.Editor editor = prefs.edit();  
  
...  
    editor.putString(TEXT_DATA_KEY, textData.toString());  
    editor.commit();  
  
...
```

2013/14

Programmazione di dispositivi mobili - v1.1

9

- Progetto
  - SimplePreferencesTest

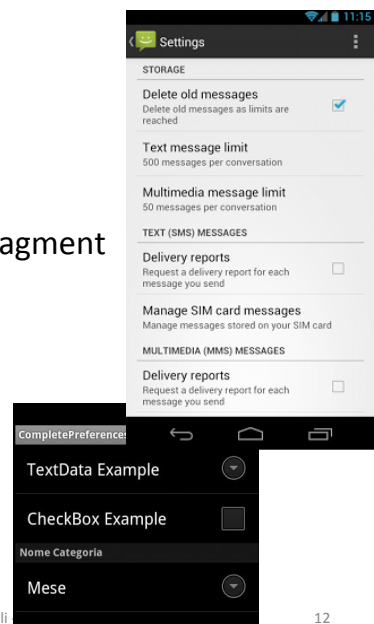
2013/14

Programmazione di dispositivi mobili - v1.1

10

## Gestione preferenze

- Classi in android.preference
  - PreferenceActivity
  - PreferenceFragment
  - Preference
- La PreferenceActivity e PreferenceFragment servono a visualizzare preferenze
- Layout per PreferenceActivity e PreferenceFragment
  - il root tag è <PreferenceScreen>
  - contiene elementi specializzazioni di “Preference”



## Particolari Preference

Specializzazione Preference	Descrizione
CheckBoxPreference	Permette di scegliere una o più opzioni tra quelle disponibili.
DialogPreference	Implementazione di base per quelle Preference che utilizzano una finestra di dialogo.
EditTextPreference	Permette di inserire un'informazione testuale.
ListPreference	Permette di visualizzare un insieme di opzioni all'interno di una finestra di dialogo.
RingtonePreference	Permette di scegliere una suoneria tra quelle disponibili nel dispositivo.

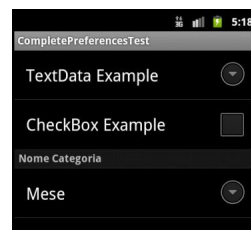
2013/14

Programmazione di dispositivi mobili - v1.1

13

## Esempio

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android"
    android:orderingFromXml="true">
    <EditTextPreference android:order="1" android:key="textData"
        android:title="TextData Example"></EditTextPreference>
    <CheckBoxPreference android:key="enabledData"
        android:title="CheckBox Example" android:order="2">
    </CheckBoxPreference>
    <PreferenceCategory android:title="@string/category_label"
        android:order="3">
        <ListPreference android:entryValues="@array/months_value"
            android:entries="@array/months_name" android:title="@string/month_label"
            android:order="3" android:key="month"></ListPreference>
    </PreferenceCategory>
</PreferenceScreen>
```



2013/14

Programmazione di dispositivi mobili - v1.1

14

## Attributi

Attributo	Descrizione
android:key	La chiave della preferenza in SharedPreferences.
android:defaultValue	Valore di default per la particolare preferenza.
android:title	Titolo da visualizzare associato alla preferenza.
android:enabled	Indica se la preferenza è abilitata o disabilitata.
android:summary	Descrizione della particolare Preferenza nella PreferenceActivity.
android:order	L'ordine di visualizzazione della preferenza. I valori più bassi vengono visualizzati per primi.
android:persistent	Indica se il corrispondente valore debba essere salvato nello SharedPreferences o meno.
android:selectable	Indica se la corrispondente preferenza è selezionabile o meno.
android:layout	L'eventuale layout personalizzato della preferenza nella PreferenceActivity.
android:widgetLayout	Permette di specificare il layout della sola parte controllabile della preferenza.

2013/14

Programmazione di dispositivi mobili - v1.1

15

## Esempio

```

public class MyPreferenceActivity extends PreferenceActivity {

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource (R.xml.my_preferences);
    }

}

```

### Accesso alle preferenze

```

SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);
// Otteniamo i valori delle preferenze
String textValue = prefs.getString("textData", "No Text Value");
boolean checkedValue = prefs.getBoolean("enabledData", false);
String listValue = prefs.getString("month", "No List value");

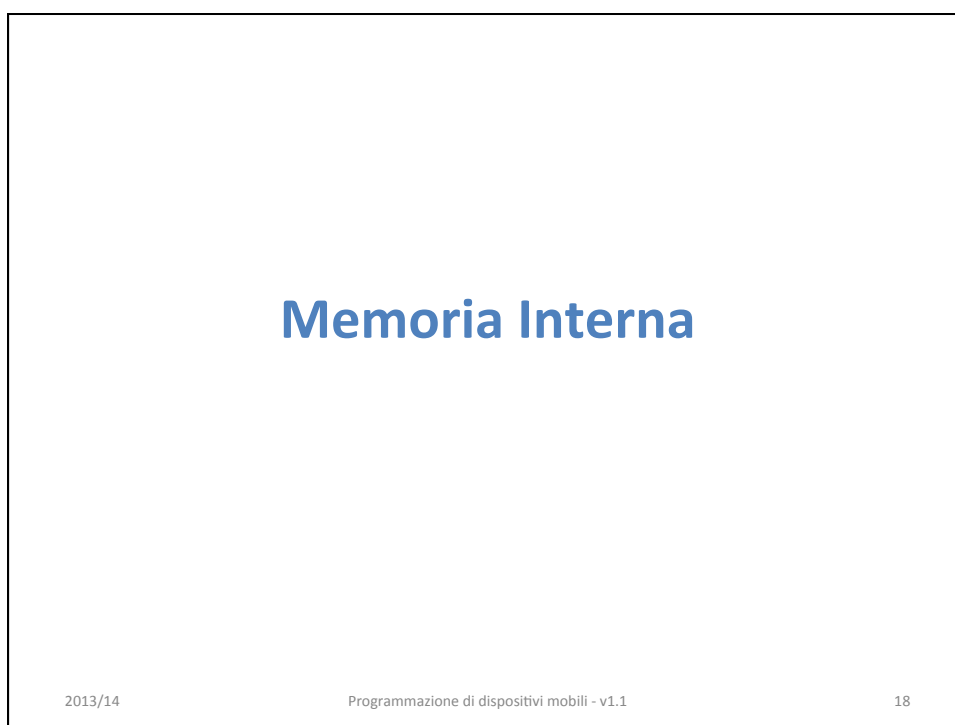
```

2013/14

Programmazione di dispositivi mobili - v1.1

16





## Memoria Interna

## Memoria interna

- Ogni APP dispone di una “sandbox” in cui salvare file
- Per leggere un file:  
`public abstract FileInputStream openFileInput (String name)`
- Per scrivere un file:  
`public FileOutputStream openFileOutput (String name, int mode)`

ed altri metodi di Context...

- **getFilesDir()**
- **getDir()**
- **deleteFile()**
- **fileList()**
- Path di un file nella sandbox
  - `/data/data/<package applicazione>/files/<nome file>`

2013/14

Programmazione di dispositivi mobili - v1.1

19

## Esempio

```
try {
    FileOutputStream fos = openFileOutput(FILE_PATH, mode);
    DataOutputStream dos = new DataOutputStream(fos);
    dos.writeUTF(inputData.toString());
    dos.close();
    output.setTextColor(SUCCESS_COLOR);
    output.setText(R.string.ok_message);
} catch (IOException e) {
    e.printStackTrace();
    output.setTextColor(ERROR_COLOR);
    output.setText(e.getMessage());
}

try {
    FileInputStream fis = new FileInputStream(new File(FILE_PATH));
    DataInputStream dis = new DataInputStream(fis);
    String text = dis.readUTF();
    dis.close();
    output.setTextColor(SUCCESS_COLOR);
    editText.setText(text);
    output.setText(R.string.ok_message);
} catch (IOException e) {
    e.printStackTrace();
    output.setTextColor(ERROR_COLOR);
    output.setText(e.getMessage());
}
```

2013/14

Programmazione di dispositivi mobili - v1.1

20

## Esempi

- FileOutputStream
- FileInputStream

## Memoria esterna

2013/14

Programmazione di dispositivi mobili - v1.1

23

## Memoria esterna

- Serve a memorizzare file
  - in particolare da condividere fra app

- Il path va chiesto ad Android

```
File sdcardDir = Environment.getExternalStorageDirectory();
```

```
File file = new File(sdcardDir, FILE_PATH);
```

- Serve il seguente permesso per scrivere
  - android.permission.WRITE\_EXTERNAL\_STORAGE

2013/14

Programmazione di dispositivi mobili - v1.1

24

## Esempio

```
try {
    // Otteniamo la root della memoria esterna
    File sdcardDir = Environment.getExternalStorageDirectory();
    // Creiamo il riferimento al nostro file
    File file = new File(sdcardDir, FILE_PATH);

    FileOutputStream fos = new FileOutputStream(file);
    DataOutputStream dos = new DataOutputStream(fos);
    dos.writeUTF(inputData.toString());
    dos.close();
    output.setText(R.string.ok_message);
} catch (IOException e) {
    e.printStackTrace();
    output.setText(R.string.error_message);
}
```

## Stato della memoria

```
boolean mExternalStorageAvailable = false;
boolean mExternalStorageWriteable = false;
String state = Environment.getExternalStorageState();

if (Environment.MEDIA_MOUNTED.equals(state)) {
    // We can read and write the media
    mExternalStorageAvailable = mExternalStorageWriteable = true;
} else if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
    // We can only read the media
    mExternalStorageAvailable = true;
    mExternalStorageWriteable = false;
} else {
    // Something else is wrong. It may be one of many other states,
    // to know is we can neither read nor write
    mExternalStorageAvailable = mExternalStorageWriteable = false;
}
```

## Esempio

- SDCardTest

## File nell'apk

- File nella cartella /res/raw
- `public InputStream openRawResource (int id)`
  - della classe `Risorse`
  - in una activity posso chiamare `getResources()`
- File nella cartella /asset