Android Development

Chapter 6 – Data Persistence

katholieke hogeschool
associatie KU Leuven
vives

Android Development
Chapter 6 – Data Persistence

Data Persistence

katholieke hogeschool
associatie KU Leuven
vives

Data Persistence

- Persisting data is an important topic in application development
- Users typically expect to reuse data in the future
- Example
  - User information
  - Configuration settings (preferences)
  - Application data
  - Logs
  - ...

Data Persistence

- 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.

- Your data storage options are
  - **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

Android SQLite

- Saving data to a database is ideal for repeating or structured data
  - Ex.: contact information.

- Using a database is much more efficient for storing this type of data.
- Easy to use and allows selection based on database queries
- Moreover, using databases enables you to enforce data integrity by specifying the relationships between different sets of data.
- Android uses the SQLite database system.
- The database that you create for an application is only accessible to itself; other applications will not be able to access it.

Android Development
Chapter 6 – Data Persistence

User Preferences

katholieke hogeschool
associatie KU Leuven
vives

Saving and Loading User Preferences

- Android provides the SharedPreferences object to help you save simple application data.
- For example
  - Your application may have an option that enables users to specify the font size of the text displayed in your application.
    - In this case, your application needs to remember the size set by the user so that the next time he or she uses the application again, it can set the size appropriately.

- In order to do so, you have several options.
  - You can save the data to a file, but you have to perform some file management routines, such as writing the data to the file, indicating how many characters to read from it, and so on.
    - Also, if you have several pieces of information to save, such as text size, font name, preferred background color, and so on, then the task of writing to a file becomes more onerous.

Saving and Loading User Preferences

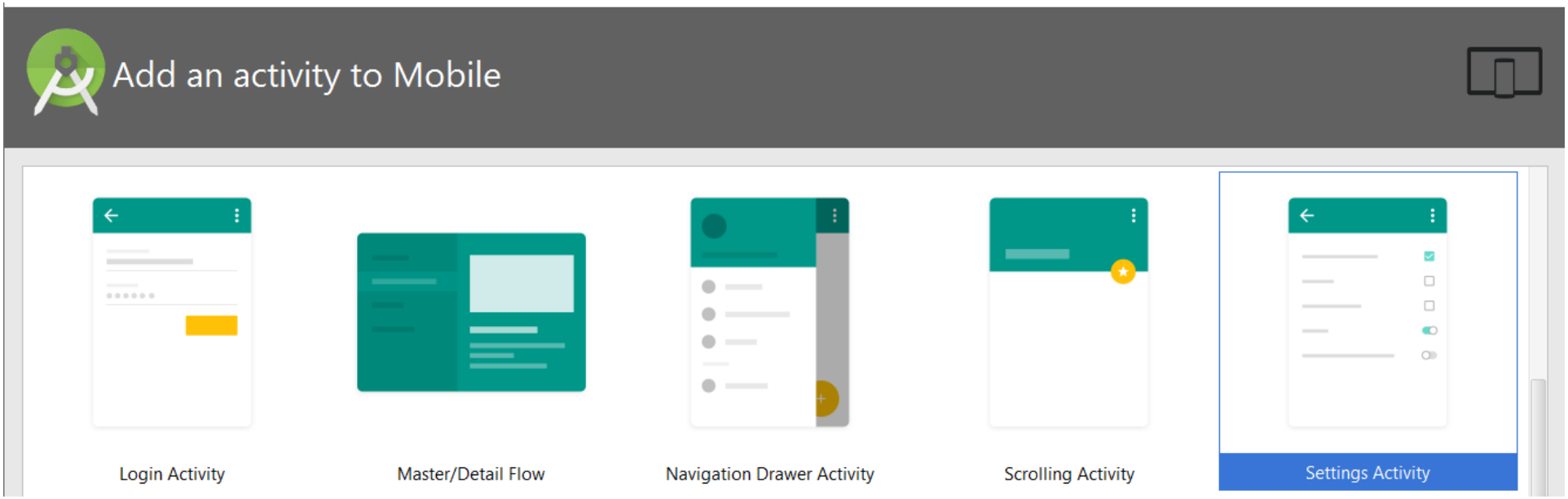
- An alternative to writing to a text file is to use a database
  - However saving simple data to a database is a bit overkill
    - Both from a developer's point of view and in terms of the application's run-time performance.

- Using the SharedPreferences object, however, you save the data you want through the use of name/value pairs
  - Can also be used to save small chunks of general data
  - Specify a name for the data you want to save, and then both it and its value will be saved automatically to an XML file for you.

key	value
firstName	Bugs
lastName	Bunny
location	Earth

Saving and Loading User Preferences

- Android Studio makes it extremely easy for us to create a nice looking and user friendly Preferences activity
- Just create a new activity based on the Settings template

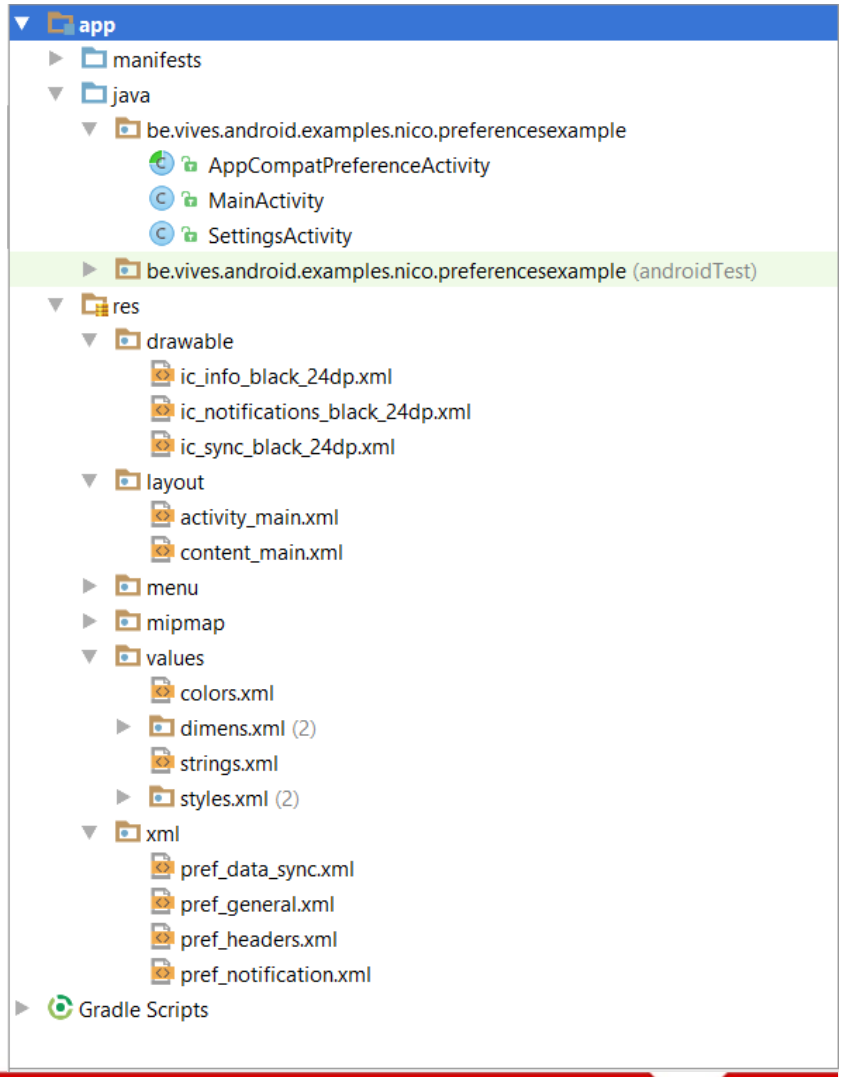


The Settings Activity Template

- This template generates a fair amount of files for you
  - Can be a bit overwhelming
    - Take a deep breath and dive in

- SettingsActivity is the Java file that contains the code to setup the actions bar, render the view of the settings and also fix some compatibility issues for older versions
  - May need some changes when adding or changing preferences !

- Drawable contains some cool looking icons that will be drawn next to the categories of settings
  - To bad that breaks our program ☺

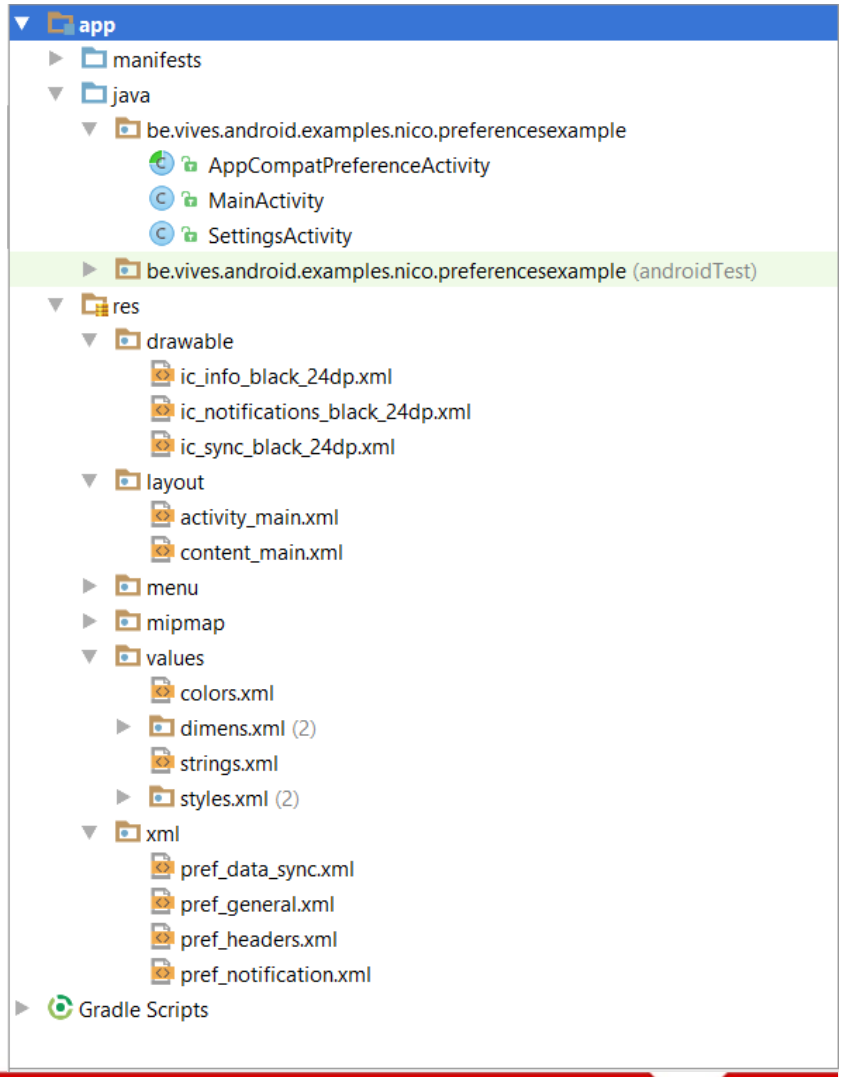


The Settings Activity Template

- To fix the drawable "ic_sync_black_24dp.xml" go to https://github.com/google/material-design-icons/blob/master/notification/drawable-anydpi-v21/ic_sync_black_24dp.xml

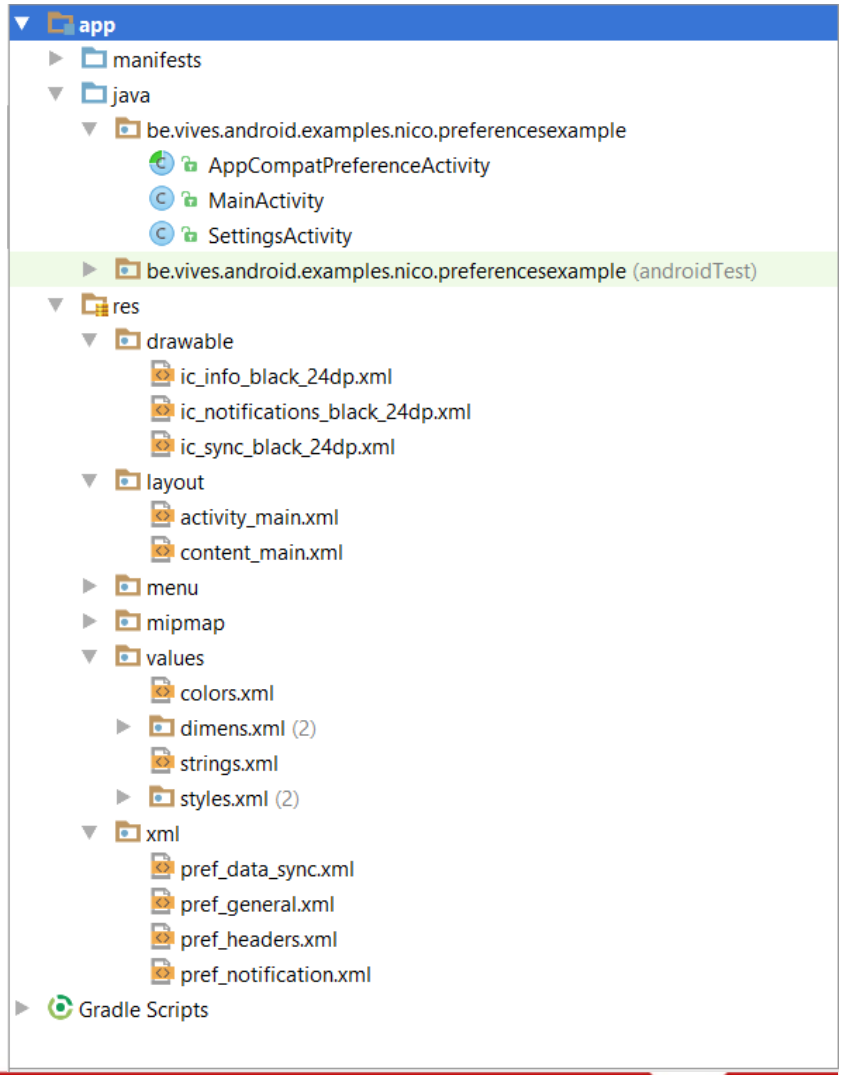
- Take note that not layout files are generated.
  - The full view is build dynamically based on the settings that are defined

- Values/strings.xml contains some new entries such as the category titles and preference labels
  - Good practice, do the same !!



The Settings Activity Template

- Xml
  - pref_data_sync.xml
  - pref_general.xml
  - pref_notification.xml
  - These contain the actual settings for the categories
    - Data & sync
    - General
    - Notifications



The Settings Activity Template

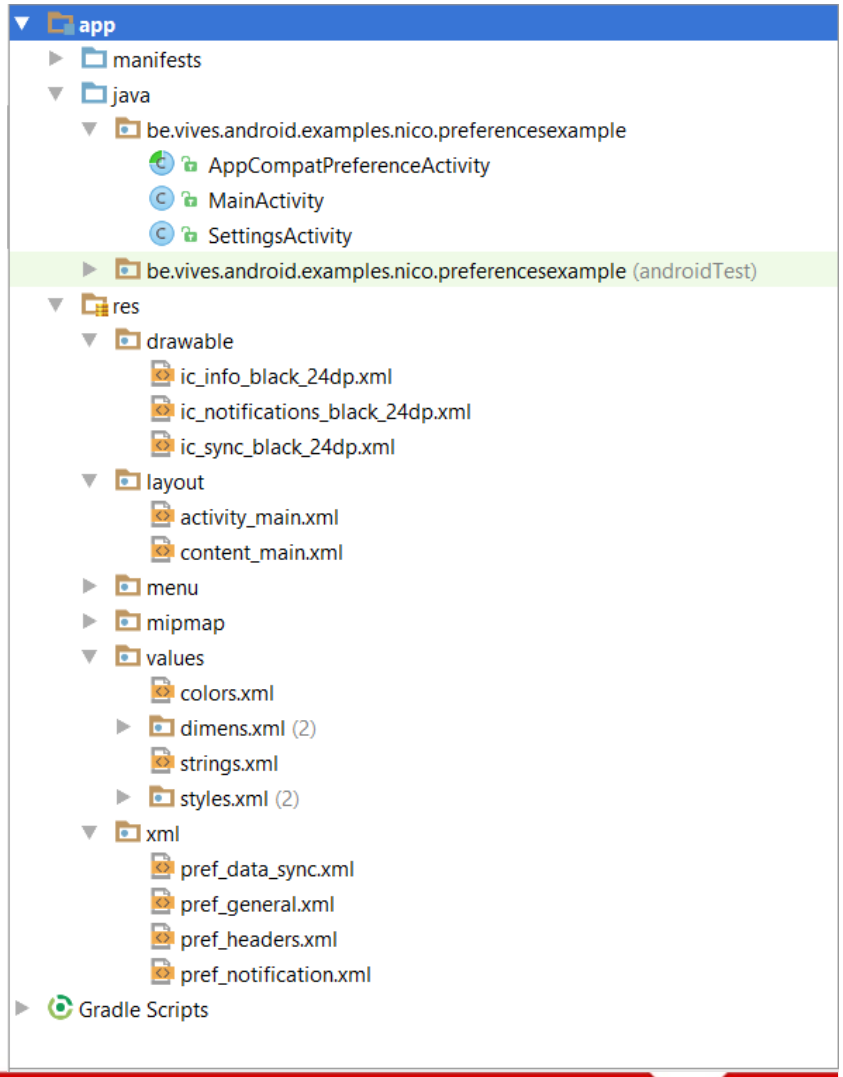
- Xml
  - pref_headers.xml contains the header view when the settings activity is opened.
    - Allows us to easily group our settings in different groups
    - Checkout the icons that are shown on the left

```
<header
    android:fragment="be.vives.android.examples
    android:title="General" android:icon="@draw

<header
    android:fragment="be.vives.android.examples
    android:title="Notifications"
    android:icon="@drawable/ic_notifications_bl

<header
    android:fragment="be.vives.android.examples
    android:title="Data & sync" android:icon="@

</preference-headers>
```



Let's Hack @ it

Accessing the User's Preferences

- Now that we got our settings defined we need to access the values of the preferences

- To make life easier we are first going to declare a static string for each setting key inside the Settings Activity class
  - The actual string name contains the key of the setting defined in the xml file

```
public class SettingsActivity extends AppCompatPreferenceActivity {

    // Setting Keys
    public static final String PREF_DISPLAY_NAME = "display_name";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setupActionBar();
    }
...
```

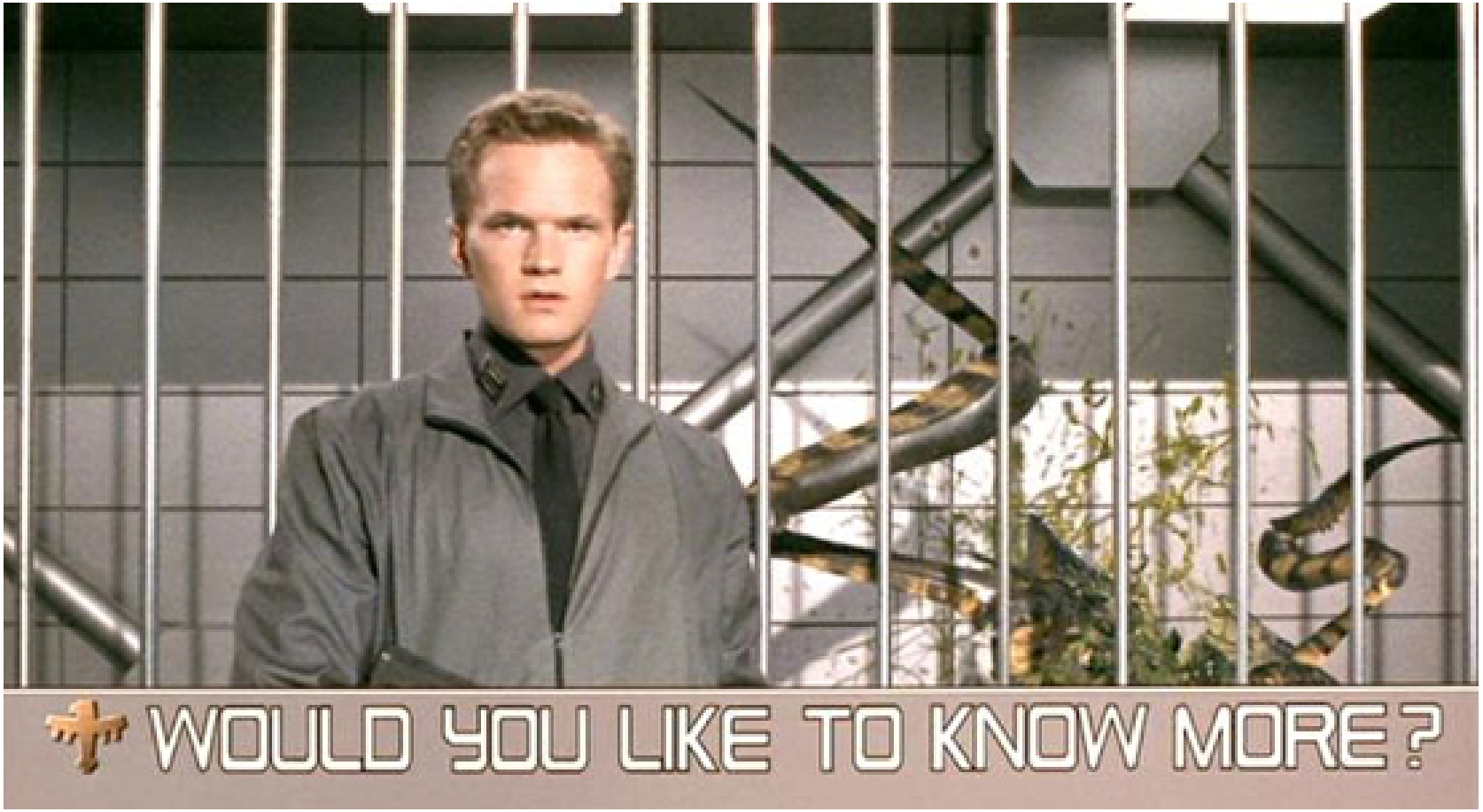
Accessing the User's Preferences

- Now we can access the settings inside any activity as follows

```
SharedPreferences sharedPref = PreferenceManager.getDefaultSharedPreferences(this);
String name = sharedPref.getString(SettingsActivity.PREF_DISPLAY_NAME, "John Doe");
```

Default value if key was empty or not found

Would You Like to Know More ?



- http://developer.android.com/guide/topics/ui/settings.html