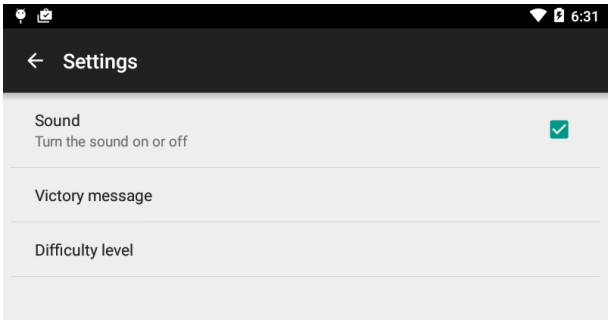
Android Application Programming

Tutorial 6: App Settings

Introduction

Most Android applications have a Settings menu option which allows users to set the application's settings/preferences. This is so common that a specialized Activity class was developed just for changing application settings (`android.preference.PreferenceFragment`). In this tutorial you will learn how to create applications settings using the `PreferenceFragment`, and you will learn how to start another activity using an `Intent`.



Create the Preference XML and PreferenceFragment

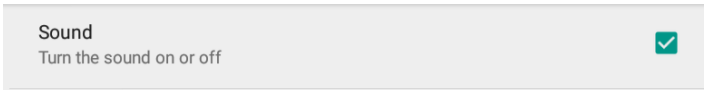
The `PreferenceFragment` simplifies the creation of a settings screen for your application. It uses an XML file (preferences.xml) to list the various application settings. The preferences are saved using the `SharedPreferences` class, just as they were in the previous tutorial. In some cases, very little to no code needs to be written to change application settings.

The first thing you need to do is create an XML file that lists the three application settings for our tic-tac-toe game:

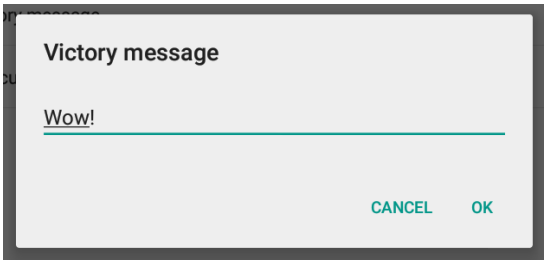
1. Create a `res/xml` directory.

2. Create a `res/xml/preferences.xml` file. We'll be creating three preferences:

   1) a `CheckBoxPreference` for turning the sound on and off,



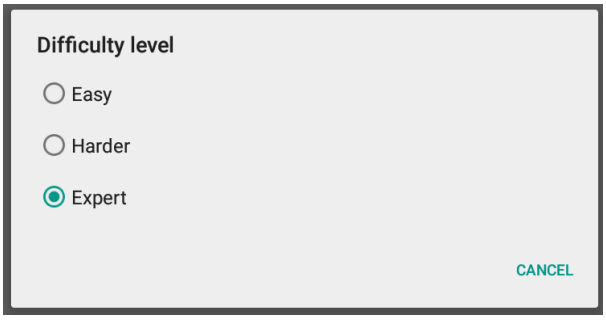
   2) an `EditTextPreference` for setting the text that is displayed when the user wins, and



 Frank McCown at Harding University

Modified by Mike Efram at Healdsburg High School to suit needs of Advanced Programming Projects class.

3) a `ListPreference` for setting the AI's difficulty level.



These three are not the only types of preferences, but they are probably the most popular types of preferences.

3. Type the XML below into `preferences.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">

    <CheckBoxPreference
           android:key="sound"
           android:title="Sound"
           android:defaultValue="true"
           android:summary="Turn the sound on or off" />

    <EditTextPreference
           android:key="victory_message"
           android:summary=""
           android:defaultValue="@string/human_wins"
           android:title="@string/victory_message" />

     <ListPreference
           android:key="difficulty_level"
           android:title="Difficulty level"
           android:summary=""
           android:defaultValue="@string/difficulty_expert"
           android:entries="@array/list_difficulty_level"
           android:entryValues="@array/list_difficulty_level" />

</PreferenceScreen>
```

The `PreferenceScreen` can have any number of preferences inside it. If there are a large number of preferences which can be organized by type, it's a good idea to organize these into `PreferenceCategory`'s. Since we only have three preferences, a `PreferenceCategory` is not necessary.

Note that some of the default values for the preferences are coming strings that we defined earlier in the `strings.xml` file (`human_wins` and `difficulty_expert`). The `ListPreference` will get its values from an `arrays.xml` file we'll create in the next step. Also note that each preference has a "key". We will use that key to retrieve the preferences as we did with `SharedPreferences` in the last tutorial.

4. Create a `res/values/arrays.xml` file with the following XML:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string-array name="list_difficulty_level">
        <item>@string/difficulty_easy</item>
        <item>@string/difficulty_harder</item>
        <item>@string/difficulty_expert</item>
    </string-array>

</resources>
```

   This file is creating a list that will be displayed when selecting the difficulty level, and it uses strings from the `strings.xml` file rather than hard-coding text that may need to be changed sometime in the future.

5. Now add a Java class to your project called `Settings` which extends the `AppCompatActivity` class. This class should specify in the `onCreate()` method that the `SettingsFragment` file will be used to display the list of preferences:

```
public class Settings extends AppCompatActivity {

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

      // Display the fragment as the main content.
     getFragmentManager().beginTransaction()
        .replace(android.R.id.content, new SettingsFragment())
        .commit();

   }
}
```

6. Now add a Java class to your project called `SettingsFragment` which extends the `PreferenceFragment` class. This class should specify in the `onCreate()` method that the `preferences.xml` file will be used to display the list of preferences:

```
public class SettingsFragment extends PreferenceFragment {

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

      // Load the preferences from an XML resource
      addPreferencesFromResource(R.xml.preferences);

   }
}
```

Now we need to display the settings which we'll do in the next step.

Display the Settings

Since we are using the Preferences to set the difficulty level, we don't need to do this using the application menu anymore. We'll remove the Difficulty menu button with one that displays the settings.

1. In `res/menu/menu_tic_tac_toe.xml`, add a Settings menu option (if you do not already have one).

2. Remove all the code for creating the difficulty level dialog box. This will now be done in Settings!!

3. Add code for handing the Settings menu option in `onOptionsItemSelected()` with code that will launch the `Settings` activity. It will use an `Intent` to start the activity:

```
case R.id.action_settings:
    startActivity(new Intent(this, Settings.class));
    return true;
```

   The `startActivity()` function starts our Settings activity. We will use `onResume()` to update the settings when the `Settings` activity is exited. This is important since we'll need to change some class-level variables if the sound is turned on or off or if the difficulty level is changed.
   There's also a `startActivityForResult()` function which could be used if we need to be notified when the activity is closed but it is not called when using the Action Bar return arrow (which we will be utilizing).

4. Now modify the `onResume()` method which will be called when the Settings activity is exited. It will be invoked when the user clicks the Back button on their device or the back arrow on the action bar.

```
@Override
protected void onResume() {

       (snip)

       // Apply potentially new settings
      SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);

       mSoundOn = prefs.getBoolean("sound", true);

       mVictoryMessage = // to do (hint…you are getting a String!)

       String diffLevel = prefs.getString("difficulty_level", getString(R.string.difficulty_harder));

       // To Do Use the String diffLevel to determine and set the difficulty level of the game in
       // TicTacToeGame
}
```

   The `mSoundOn` and `mVictoryMessage` variables from above are not yet defined. **It is left to you** to declare this class-level variables and use it to determine if the `mHumanMediaPlayer` or `mComputerMediaPlayer` should be played and to change the message when the Human Player wins.

5. In order for the Intent to work, you must also modify the app's `AndroidManifest.xml` file, adding the following inside the `<application>` tags which indicate the `Settings` activity should be included as part of this app:

```
<!-- Allow the Settings activity to be launched -->
<activity android:name=".Settings"
          android:label="Settings"
          android:parentActivityName=".TicTacToe" >
</activity>
```

Run your app and launch the Settings screen by selecting the Menu and pressing **Settings**. Verify that you can now turn the sound on and off, set the victory message, and set the difficulty level. We've gained a lot of functionality with little code.