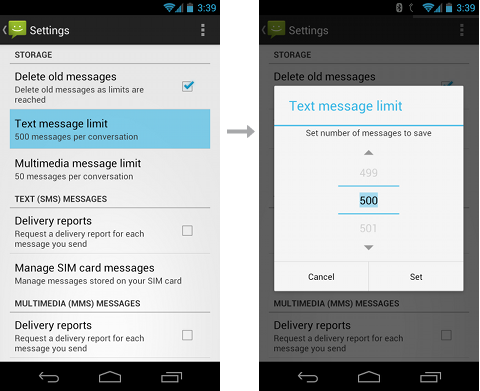
Настройки ([Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru))

В приложениях часто содержатся настройки, которые позволяют пользователю изменять возможности и поведение приложения. Например, некоторые приложения позволяют пользователям включать и выключать уведомления или указывать частоту синхронизации данных приложения с облаком.

Если вы хотите предоставить настройки для вашего приложения, вы должны использовать API-интерфейсы [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) системы Android для построения интерфейса, согласованного с привычным для пользователей других приложений Android (включая системные настройки). В этом документе показано, как построить настройки вашего приложения посредством API-интерфейсов [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru).

**Дизайн настроек**

Подробную информацию о дизайне настроек см. в руководстве по дизайну [настроек](https://developer.android.com/design/patterns/settings.html?hl=ru).



**Рисунок 1.** Снимки экранов настроек приложения Android для обмена сообщениями. Выбор элемента, заданного посредством [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru), открывает интерфейс для изменения значения.

Обзор

Вместо использования отображаемых объектов [View](https://developer.android.com/reference/android/view/View.html?hl=ru) для построения пользовательского интерфейса, настройки создаются с помощью различных подклассов класса [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru), который вы объявляете в XML-файле.

Объект [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) является строительным блоком для отдельной настройки. Каждый объект [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru)отображается в виде элемента в списке и предоставляет соответствующий пользовательский интерфейс для изменения настройки пользователями. Например, [CheckBoxPreference](https://developer.android.com/reference/android/preference/CheckBoxPreference.html?hl=ru) создает элемент списка, который показывает флажок, а [ListPreference](https://developer.android.com/reference/android/preference/ListPreference.html?hl=ru) создает элемент, который открывает диалоговое окно со списком вариантов для выбора.

Каждый добавляемый вами объект [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) имеет соответствующую пару «ключ-значение», которую система использует для сохранения настройки в файле [SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html?hl=ru) значений настроек вашего приложения по умолчанию. Когда пользователь изменяет настройку, система обновляет соответствующее значение в файле [SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html?hl=ru). Вам потребуется напрямую взаимодействовать с файлом, связанным с [SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html?hl=ru), только в случае, когда нужно прочитать значение для определения поведения вашего приложения на основе пользовательских настроек.

Значение, сохраненное в [SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html?hl=ru) для каждой настройки, может относиться к одному из следующих типов данных:

* Логическое значение
* Число с плавающей точкой
* Целое число
* Длинное целое число
* Строка
* Строка [Set](https://developer.android.com/reference/java/util/Set.html?hl=ru)

Поскольку пользовательский интерфейс настроек вашего приложения создается посредством объектов [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru), а не объектов [View](https://developer.android.com/reference/android/view/View.html?hl=ru), вам потребуется использовать специализированные подклассы [Activity](https://developer.android.com/reference/android/app/Activity.html?hl=ru)или [Fragment](https://developer.android.com/reference/android/app/Fragment.html?hl=ru) для отображения настроек из списка:

* Если ваше приложение поддерживает версии Android старше 3.0 (API уровня 10 и ниже), для построения операции необходимо наследовать класс [PreferenceActivity](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru).
* В операционных системах Android 3.0 и более поздних версиях вы должны вместо этого использовать традиционный класс [Activity](https://developer.android.com/reference/android/app/Activity.html?hl=ru), который содержит объект [PreferenceFragment](https://developer.android.com/reference/android/preference/PreferenceFragment.html?hl=ru) для отображения настроек вашего приложения. Однако, когда у вас есть несколько групп настроек, вы можете также использовать [PreferenceActivity](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru) для создания макета с двумя панелями для больших экранов.

Настройка объекта [PreferenceActivity](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru) и экземпляров [PreferenceFragment](https://developer.android.com/reference/android/preference/PreferenceFragment.html?hl=ru) описана в разделах [Создание операции предпочтения](https://developer.android.com/guide/topics/ui/settings?hl=ru#Activity) и [Использование фрагментов предпочтений](https://developer.android.com/guide/topics/ui/settings?hl=ru#Fragment).

Предпочтения

Каждая настройка для вашего приложения представлена конкретным подклассом класса [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru). Каждый подкласс содержит набор основных свойств, которые позволяют вам указывать, например, заголовок для настройки и ее значение по умолчанию. Каждый подкласс также содержит собственные специализированные свойства и пользовательский интерфейс. В качестве примера на рисунке 1 показан снимок экрана настроек приложения Android для обмена сообщениями. Каждый элемент списка на экране настроек возвращается отдельным объектом [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru).

Ниже приведены самые распространенные предпочтения:

[CheckBoxPreference](https://developer.android.com/reference/android/preference/CheckBoxPreference.html?hl=ru)

Отображает элемент с флажком для настройки, которая может быть включена или выключена. Сохраненное значение является логическим (true, если флажок установлен).

[ListPreference](https://developer.android.com/reference/android/preference/ListPreference.html?hl=ru)

Открывает диалоговое окно со списком переключателей. Сохраненное значение может относиться к одному из поддерживаемых типов значений (перечисленных выше).

[EditTextPreference](https://developer.android.com/reference/android/preference/EditTextPreference.html?hl=ru)

Открывает диалоговое окно с виджетом [EditText](https://developer.android.com/reference/android/widget/EditText.html?hl=ru). Сохраненное значение — [String](https://developer.android.com/reference/java/lang/String.html?hl=ru).

См. класс [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru), который содержит список всех остальных подклассов и их соответствующих свойств.

Конечно, встроенные классы не обеспечивают всех потребностей, и вашему приложению может понадобиться что-либо более специализированное. Например, в настоящее время система не предоставляет класс [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) для выбора числа или даты. Поэтому вам может потребоваться определить свой собственный подкласс [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru). См. раздел [Построение пользовательского предпочтения](https://developer.android.com/guide/topics/ui/settings?hl=ru#Custom).

Определение предпочтений в XML

Хотя вы можете создавать новые экземпляры объектов [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) в режиме выполнения, вы должны определить список настроек в файле XML с иерархией объектов [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru). Использование файла XML для определения вашей коллекции настроек предпочтительней, поскольку файл обладает удобочитаемой структурой, которую легко обновлять. Кроме того, настройки вашего приложения обычно определены заранее, хотя у вас сохраняется возможность изменять коллекцию в режиме выполнения.

Каждый подкласс класса [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) может быть объявлен посредством элемента XML, который соответствует имени класса, например, &lt;CheckBoxPreference&gt;.

Вы должны сохранить файл XML в каталоге res/xml/. Хотя вы можете назвать файл любым именем, традиционно его называют preferences.xml. Обычно вам требуется лишь один файл, поскольку ветви иерархии (которые открывают собственный список настроек) объявлены с помощью вложенных экземпляров [PreferenceScreen](https://developer.android.com/reference/android/preference/PreferenceScreen.html?hl=ru).

**Примечание.** Если вы хотите создать макет с несколькими панелями для ваших настроек, вам потребуются отдельные файлы XML для каждого фрагмента.

Корневой узел XML-файла должен быть элементом [<PreferenceScreen>](https://developer.android.com/reference/android/preference/PreferenceScreen.html?hl=ru). Внутри этого элемента вы добавляете каждый элемент [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru). Каждый дочерний элемент, который вы добавляете внутри элемента [<PreferenceScreen>](https://developer.android.com/reference/android/preference/PreferenceScreen.html?hl=ru), отображается в виде одного пункта в списке настроек.

Например:

<?xml version="1.0" encoding="utf-8"?>  
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">  
    <CheckBoxPreference  
        android:key="pref\_sync"  
        android:title="@string/pref\_sync"  
        android:summary="@string/pref\_sync\_summ"  
        android:defaultValue="true" />  
    <ListPreference  
        android:dependency="pref\_sync"  
        android:key="pref\_syncConnectionType"  
        android:title="@string/pref\_syncConnectionType"  
        android:dialogTitle="@string/pref\_syncConnectionType"  
        android:entries="@array/pref\_syncConnectionTypes\_entries"  
        android:entryValues="@array/pref\_syncConnectionTypes\_values"  
        android:defaultValue="@string/pref\_syncConnectionTypes\_default" />  
</PreferenceScreen>

В этом примере есть [CheckBoxPreference](https://developer.android.com/reference/android/preference/CheckBoxPreference.html?hl=ru) и [ListPreference](https://developer.android.com/reference/android/preference/ListPreference.html?hl=ru). Оба содержат следующие три атрибута:

android:key

Этот атрибут необходим для предпочтений, которые сохраняют значение данных. Он задает уникальный ключ (строку), который использует система при сохранении значения этой настройки в [SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html?hl=ru).

Этот атрибут *не является обязательным* только когда предпочтение представляет собой[PreferenceCategory](https://developer.android.com/reference/android/preference/PreferenceCategory.html?hl=ru) или [PreferenceScreen](https://developer.android.com/reference/android/preference/PreferenceScreen.html?hl=ru), либо предпочтение указывает намерение [Intent](https://developer.android.com/reference/android/content/Intent.html?hl=ru) для вызова (посредством элемента [&lt;intent&gt;](https://developer.android.com/guide/topics/ui/settings?hl=ru#Intents)) или фрагмент [Fragment](https://developer.android.com/reference/android/app/Fragment.html?hl=ru) для отображения (с помощью атрибута [android:fragment](https://developer.android.com/reference/android/preference/Preference.html?hl=ru" \l "attr_android:fragment)).

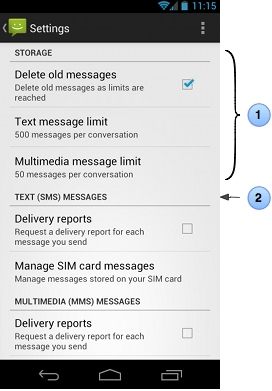
android:title

Этот атрибут предоставляет имя настройки, отображаемое для пользователя.

android:defaultValue

Этот атрибут указывает исходное значение, которое система должна установить в файле [SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html?hl=ru). Вы должны указать значения по умолчанию для всех настроек.

Для получения информации обо всех других поддерживаемых атрибутов см. документацию [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) (и соответствующий подкласс).



**Рисунок 2.** Создание категорий с заголовками.   
**1.** Категория задана элементом [<PreferenceCategory>](https://developer.android.com/reference/android/preference/PreferenceCategory.html?hl=ru).   
**2.** Заголовок задан посредством атрибута android:title.

Когда список ваших настроек содержит более 10 элементов, вы, вероятно, захотите добавить заголовки для определения групп настроек или отобразить эти группы на отдельном экране. Эти возможности описаны в следующих разделах.

Создание групп настроек

Если вы представляете список из 10 или более настроек, пользователям может быть трудно их просматривать, воспринимать и обрабатывать. Это можно исправить, разделив некоторые или все настройки на группы, что эффективно преобразует один длинный список в несколько более коротких списков. Группа связанных настроек может быть представлена одним из двух способов:

* [Использование заголовков](https://developer.android.com/guide/topics/ui/settings?hl=ru#Titles)
* [Использование подэкранов](https://developer.android.com/guide/topics/ui/settings?hl=ru#Subscreens)

Вы можете пользоваться одним или обоими из этих методов группировки для организации настроек в вашем приложении. Принимая решение об используемом варианте и о разделении настроек на группы, вы должны следовать инструкциям в разделе[Настройки](https://developer.android.com/design/patterns/settings.html?hl=ru) руководства «Дизайн для Android».

Использование заголовков

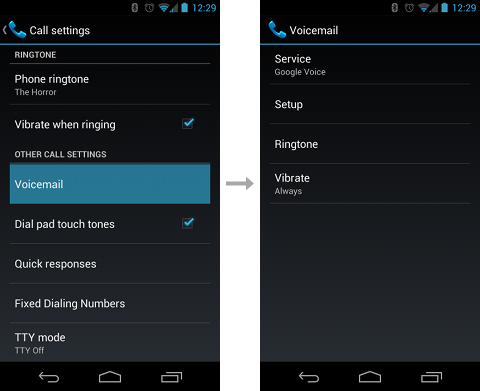
Если вы хотите создать разделители с заголовками между группами настроек (как показано на рисунке 2), поместите каждую группу объектов [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) внутри [PreferenceCategory](https://developer.android.com/reference/android/preference/PreferenceCategory.html?hl=ru).

Например:

<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">  
    <PreferenceCategory  
        android:title="@string/pref\_sms\_storage\_title"  
        android:key="pref\_key\_storage\_settings">  
        <CheckBoxPreference  
            android:key="pref\_key\_auto\_delete"  
            android:summary="@string/pref\_summary\_auto\_delete"  
            android:title="@string/pref\_title\_auto\_delete"  
            android:defaultValue="false"... />  
        <Preference  
            android:key="pref\_key\_sms\_delete\_limit"  
            android:dependency="pref\_key\_auto\_delete"  
            android:summary="@string/pref\_summary\_delete\_limit"  
            android:title="@string/pref\_title\_sms\_delete"... />  
        <Preference  
            android:key="pref\_key\_mms\_delete\_limit"  
            android:dependency="pref\_key\_auto\_delete"  
            android:summary="@string/pref\_summary\_delete\_limit"  
            android:title="@string/pref\_title\_mms\_delete" ... />  
    </PreferenceCategory>  
    ...  
</PreferenceScreen>

Использование подэкранов

Если вы хотите поместить группу настроек на подэкран (как показано на рисунке 3), поместите каждую группу объектов [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) внутри [PreferenceScreen](https://developer.android.com/reference/android/preference/PreferenceScreen.html?hl=ru).



**Рисунок 3.** Создание подэкранов. Элемент &lt;PreferenceScreen&gt; создает пункт, при выборе которого открывается отдельный список вложенных настроек.

Например:

<PreferenceScreen  xmlns:android="http://schemas.android.com/apk/res/android">  
    <!-- opens a subscreen of settings -->  
    <PreferenceScreen  
        android:key="button\_voicemail\_category\_key"  
        android:title="@string/voicemail"  
        android:persistent="false">  
        <ListPreference  
            android:key="button\_voicemail\_provider\_key"  
            android:title="@string/voicemail\_provider" ... />  
        <!-- opens another nested subscreen -->  
        <PreferenceScreen  
            android:key="button\_voicemail\_setting\_key"  
            android:title="@string/voicemail\_settings"  
            android:persistent="false">  
            ...  
        </PreferenceScreen>  
        <RingtonePreference  
            android:key="button\_voicemail\_ringtone\_key"  
            android:title="@string/voicemail\_ringtone\_title"  
            android:ringtoneType="notification" ... />  
        ...  
    </PreferenceScreen>  
    ...  
</PreferenceScreen>

Использование намерений

В некоторых случаях может потребоваться, чтобы элемент предпочтений открывал другую операцию, а не экран настроек, например, веб-браузер для просмотра веб-страницы. Чтобы вызвать [Intent](https://developer.android.com/reference/android/content/Intent.html?hl=ru), когда пользователь выбирает элемент предпочтений, добавьте элемент &lt;intent&gt; в качестве дочернего элемента соответствующего элемента &lt;Preference&gt;.

Например, здесь показано использование элемента предпочтений для открытия веб-страницы:

<Preference android:title="@string/prefs\_web\_page" >  
    <intent android:action="android.intent.action.VIEW"  
            android:data="http://www.example.com" />  
</Preference>

Вы можете создавать неявные и явные намерения с помощью следующих атрибутов:

android:action

Назначаемое действие, как в методе [setAction()](https://developer.android.com/reference/android/content/Intent.html?hl=ru" \l "setAction(java.lang.String)).

android:data

Назначаемые данные, как в методе [setData()](https://developer.android.com/reference/android/content/Intent.html?hl=ru" \l "setData(android.net.Uri)).

android:mimeType

Назначаемый тип MIME, как в методе [setType()](https://developer.android.com/reference/android/content/Intent.html?hl=ru" \l "setType(java.lang.String)).

android:targetClass

Часть имени компонента, означающая класс, как в методе [setComponent()](https://developer.android.com/reference/android/content/Intent.html?hl=ru" \l "setComponent(android.content.ComponentName)).

android:targetPackage

Пакетная часть имени компонента, как в методе [setComponent()](https://developer.android.com/reference/android/content/Intent.html?hl=ru" \l "setComponent(android.content.ComponentName)).

Создание операции предпочтений

Для отображения ваших настроек в операции наследуйте класс [PreferenceActivity](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru). Это наследование традиционного класса [Activity](https://developer.android.com/reference/android/app/Activity.html?hl=ru), который отображает список настроек на основе иерархии объектов [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru). [PreferenceActivity](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru) автоматически сохраняет настройки, связанные с каждым объектом [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru), когда пользователь вносит изменения.

**Примечание.** При разработке приложения для версии Android 3.0 или выше вместо этого следует использовать **[PreferenceFragment](https://developer.android.com/reference/android/preference/PreferenceFragment.html?hl=ru)**. Прочитайте следующий раздел [Использование фрагментов предпочтений](https://developer.android.com/guide/topics/ui/settings?hl=ru#Fragment).

Запомните самое важное: не загружайте макет отображаемых объектов во время обратного вызова [onCreate()](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru" \l "onCreate(android.os.Bundle)). Вместо этого вызовите [addPreferencesFromResource()](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru" \l "addPreferencesFromResource(int)) для добавления предпочтений, объявленных в XML-файле для операции. Например, здесь приведен минимальный код, необходимый для работы [PreferenceActivity](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru):

public class SettingsActivity extends PreferenceActivity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        addPreferencesFromResource(R.xml.preferences);  
    }  
}

Этого кода действительно достаточно для некоторых приложений, поскольку как только пользователь изменяет предпочтение, система сохраняет изменения в файле [SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html?hl=ru) по умолчанию, который другие компоненты вашего приложения могут читать, когда требуется проверить пользовательские настройки. Однако многим приложениям требуется немного больше кода, чтобы отслеживать изменения, происходящие с предпочтениями. Информацию об отслеживании изменений в файле [SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html?hl=ru) см. в разделе [Чтение предпочтений](https://developer.android.com/guide/topics/ui/settings?hl=ru#ReadingPrefs).

Использование фрагментов предпочтений

При разработке приложений для Android 3.0 (API уровня 11) и более поздних версий необходимо использовать [PreferenceFragment](https://developer.android.com/reference/android/preference/PreferenceFragment.html?hl=ru) для отображения списка объектов [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru). Вы можете добавить [PreferenceFragment](https://developer.android.com/reference/android/preference/PreferenceFragment.html?hl=ru) в любую операцию, при этом необязательно использовать [PreferenceActivity](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru).

[Фрагменты](https://developer.android.com/guide/components/fragments.html?hl=ru) обеспечивают более универсальную архитектуру для вашего приложения по сравнению с использованием отдельных операций, вне зависимости от типа создаваемой операции. Фактически, для управления отображением ваших настроек мы предлагаем вам использовать [PreferenceFragment](https://developer.android.com/reference/android/preference/PreferenceFragment.html?hl=ru) вместо [PreferenceActivity](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru) при каждой возможности.

Ваша реализация [PreferenceFragment](https://developer.android.com/reference/android/preference/PreferenceFragment.html?hl=ru) может содержать просто определение метода [onCreate()](https://developer.android.com/reference/android/preference/PreferenceFragment.html?hl=ru" \l "onCreate(android.os.Bundle)) для загрузки файла предпочтений посредством [addPreferencesFromResource()](https://developer.android.com/reference/android/preference/PreferenceFragment.html?hl=ru" \l "addPreferencesFromResource(int)). Например:

public static class SettingsFragment extends PreferenceFragment {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        // Load the preferences from an XML resource  
        addPreferencesFromResource(R.xml.preferences);  
    }  
    ...  
}

Затем вы можете добавить этот фрагмент в операцию [Activity](https://developer.android.com/reference/android/app/Activity.html?hl=ru), как вы сделали бы это для любого другого фрагмента [Fragment](https://developer.android.com/reference/android/app/Fragment.html?hl=ru). Например:

public class SettingsActivity extends Activity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        // Display the fragment as the main content.  
        getFragmentManager().beginTransaction()  
                .replace(android.R.id.content, new SettingsFragment())  
                .commit();  
    }  
}

**Примечание.** Фрагмент **[PreferenceFragment](https://developer.android.com/reference/android/preference/PreferenceFragment.html?hl=ru)** не содержит собственного объекта **[Context](https://developer.android.com/reference/android/content/Context.html?hl=ru)**. Если вам требуется объект **[Context](https://developer.android.com/reference/android/content/Context.html?hl=ru)**, вы можете вызвать[**getActivity()**](https://developer.android.com/reference/android/app/Fragment.html?hl=ru#getActivity()). Однако разработчик должен быть внимательным и вызывать метод **[getActivity()](https://developer.android.com/reference/android/app/Fragment.html?hl=ru" \l "getActivity())** только в том случае, когда фрагмент прикреплен к операции. Если фрагмент еще не прикреплен или был откреплен в конце его жизненного цикла, метод **[getActivity()](https://developer.android.com/reference/android/app/Fragment.html?hl=ru" \l "getActivity())** вернет null.

Установка значений по умолчанию

Вероятно, создаваемые вами предпочтения определяют важное поведение вашего приложения, поэтому необходимо инициализировать соответствующий файл [SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html?hl=ru), записав в него значения по умолчанию для каждого предпочтения [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) при первом запуске вашего приложения пользователем.

В первую очередь необходимо указать значение по умолчанию для каждого объекта [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) в вашем XML-файле посредством атрибута android:defaultValue. Значение может относиться к любому типу данных, подходящему для соответствующего объекта [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru). Например:

<!-- default value is a boolean -->  
<CheckBoxPreference  
    android:defaultValue="true"  
    ... />  
  
<!-- default value is a string -->  
<ListPreference  
    android:defaultValue="@string/pref\_syncConnectionTypes\_default"  
    ... />

Затем из метода [onCreate()](https://developer.android.com/reference/android/app/Activity.html?hl=ru" \l "onCreate(android.os.Bundle)) основной операции вашего приложения (и из любой другой операции, через которую пользователь может войти в ваше приложение в первый раз) вызовите [setDefaultValues()](https://developer.android.com/reference/android/preference/PreferenceManager.html?hl=ru" \l "setDefaultValues(android.content.Context,%20int,%20boolean)):

PreferenceManager.setDefaultValues(this, R.xml.advanced\_preferences, false);

Вызов этого метода при выполнении [onCreate()](https://developer.android.com/reference/android/app/Activity.html?hl=ru" \l "onCreate(android.os.Bundle)) гарантирует, что ваше приложение правильно инициализируется и получит настройки по умолчанию, которые могут потребоваться вашему приложению для определенного поведения (например, следует ли загружать данные при работе в сотовой сети).

Этот метод имеет три аргумента:

* [Context](https://developer.android.com/reference/android/content/Context.html?hl=ru) вашего приложения.
* Идентификатор ресурса для XML-файла предпочтений, для которого вы хотите установить значения по умолчанию.
* Логическое значение, которое указывает, требуется ли значения по умолчанию устанавливать более одного раза.

При значении false система устанавливает значения по умолчанию только в том случае, если этот метод никогда не вызывался ранее (или атрибут [KEY\_HAS\_SET\_DEFAULT\_VALUES](https://developer.android.com/reference/android/preference/PreferenceManager.html?hl=ru#KEY_HAS_SET_DEFAULT_VALUES) в файле общих предпочтений по умолчанию имеет значение false).

Когда для третьего аргумента установлено значение false, вы можете вызывать этот метод при каждом запуске операции, не опасаясь перезаписи сохраненных пользовательских предпочтений из-за их сброса в состояние по умолчанию. Однако, если установить для этого аргумента значение true, вы будете перезаписывать все предыдущие значения значениями по умолчанию.

Использование заголовков предпочтений

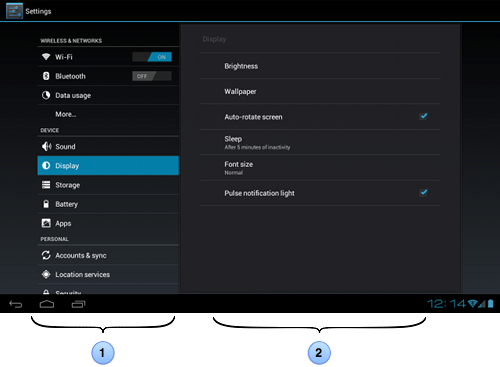
В редких случаях может потребоваться такая структура настроек, при которой на первом экране отображается только список [подэкранов](https://developer.android.com/guide/topics/ui/settings?hl=ru" \l "Subscreens) (например, как в приложении системных настроек, показанных на рисунках 4 и 5). При разработке такого дизайна для Android 3.0 и более поздних версий вы должны использовать новую возможность Android 3.0 — «заголовки», вместо создания подэкранов посредством вложенных элементов [PreferenceScreen](https://developer.android.com/reference/android/preference/PreferenceScreen.html?hl=ru).

Чтобы создать настройки с заголовками, выполните следующие действия:

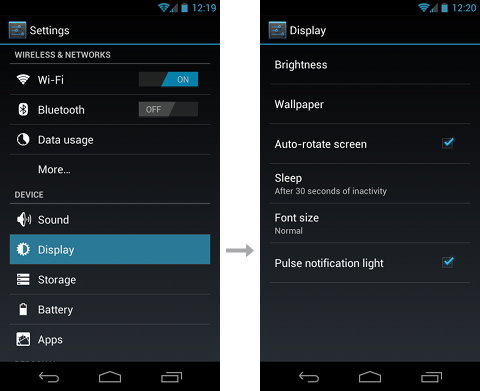
1. Выделите каждую группу настроек в отдельный экземпляр [PreferenceFragment](https://developer.android.com/reference/android/preference/PreferenceFragment.html?hl=ru). Таким образом, каждая группа настроек должна иметь отдельный XML-файл.
2. Создайте XML-файл заголовков, в котором перечислены все группы настроек и объявления, какой фрагмент содержит соответствующий список настроек.
3. Наследуйте класс [PreferenceActivity](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru), который будет содержать ваши настройки.
4. Реализуйте обратный вызов [onBuildHeaders()](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru" \l "onBuildHeaders(java.util.List%3Candroid.preference.PreferenceActivity.Header%3E)), чтобы указать файл заголовков.

Огромное преимущество использования этого дизайна состоит в том, что при запуске на больших экранах [PreferenceActivity](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru) автоматически создает макет с двумя панелями, показанный на рисунке 4.

Даже если ваше приложение поддерживает версии Android старше 3.0, вы можете создать приложение, использующее [PreferenceFragment](https://developer.android.com/reference/android/preference/PreferenceFragment.html?hl=ru) для двухпанельного представления на новых устройствах и поддерживающее традиционную многоэкранную иерархию на более старых устройствах (см. раздел [Поддержка старых версий посредством заголовков предпочтений](https://developer.android.com/guide/topics/ui/settings?hl=ru#BackCompatHeaders)).



**Рисунок 4.** Двухпанельный макет с заголовками.   
**1.** Заголовки определяются посредством XML-файла заголовков.   
**2.** Каждая группа настроек определяется с помощью фрагмента [PreferenceFragment](https://developer.android.com/reference/android/preference/PreferenceFragment.html?hl=ru), который указывается элементом &lt;header&gt; в файле заголовков.



**Рисунок 5.** Смартфон с заголовками настроек. При выборе пункта вместо заголовков отображается соответствующий фрагмент [PreferenceFragment](https://developer.android.com/reference/android/preference/PreferenceFragment.html?hl=ru).

Создание файла заголовков

Каждая группа настроек в вашем списке заголовков указывается отдельным элементом &lt;header&gt;внутри корневого элемента &lt;preference-headers&gt;. Например:

<?xml version="1.0" encoding="utf-8"?>  
<preference-headers xmlns:android="http://schemas.android.com/apk/res/android">  
    <header  
        android:fragment="com.example.prefs.SettingsActivity$SettingsFragmentOne"  
        android:title="@string/prefs\_category\_one"  
        android:summary="@string/prefs\_summ\_category\_one" />  
    <header  
        android:fragment="com.example.prefs.SettingsActivity$SettingsFragmentTwo"  
        android:title="@string/prefs\_category\_two"  
        android:summary="@string/prefs\_summ\_category\_two" >  
        <!-- key/value pairs can be included as arguments for the fragment. -->  
        <extra android:name="someKey" android:value="someHeaderValue" />  
    </header>  
</preference-headers>

Посредством атрибута android:fragment каждый заголовок объявляет экземпляр фрагмента [PreferenceFragment](https://developer.android.com/reference/android/preference/PreferenceFragment.html?hl=ru), который должен открываться при выборе этого заголовка пользователем.

Элемент &lt;extras&gt; позволяет передавать пары «ключ-значение» фрагменту в объекте [Bundle](https://developer.android.com/reference/android/os/Bundle.html?hl=ru). Фрагмент может извлекать аргументы путем вызова метода [getArguments()](https://developer.android.com/reference/android/app/Fragment.html?hl=ru" \l "getArguments()). Вы можете передавать аргументы фрагменту по различным причинам, но хорошим поводом является повторное использование одного и того же подкласса [PreferenceFragment](https://developer.android.com/reference/android/preference/PreferenceFragment.html?hl=ru) для каждой группы и использование аргументов для указания XML-файла предпочтений, который должен быть загружен фрагментом.

Например, здесь приведен фрагмент, который можно использовать повторно для нескольких групп настроек, когда каждый заголовок определяет аргумент &lt;extra&gt; с ключом "settings":

public static class SettingsFragment extends PreferenceFragment {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        String settings = getArguments().getString("settings");  
        if ("notifications".equals(settings)) {  
            addPreferencesFromResource(R.xml.settings\_wifi);  
        } else if ("sync".equals(settings)) {  
            addPreferencesFromResource(R.xml.settings\_sync);  
        }  
    }  
}

Отображение заголовков

Чтобы отобразить заголовки предпочтений, вы должны реализовать метод обратного вызова [onBuildHeaders()](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru" \l "onBuildHeaders(java.util.List%3Candroid.preference.PreferenceActivity.Header%3E)) и вызвать [loadHeadersFromResource()](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru" \l "loadHeadersFromResource(int,%20java.util.List%3Candroid.preference.PreferenceActivity.Header%3E)). Например:

public class SettingsActivity extends PreferenceActivity {  
    @Override  
    public void onBuildHeaders(List<Header> target) {  
        loadHeadersFromResource(R.xml.preference\_headers, target);  
    }  
}

Когда пользователь выбирает пункт в списке заголовков, система открывает связанный [PreferenceFragment](https://developer.android.com/reference/android/preference/PreferenceFragment.html?hl=ru).

**Примечание.** При использовании заголовков предпочтений ваш подкласс **[PreferenceActivity](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru)** не должен реализовывать метод **[onCreate()](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru" \l "onCreate(android.os.Bundle))**, поскольку единственной обязательной задачей операции является загрузка заголовков.

Поддержка старых версий с заголовками предпочтений

Если ваше приложение поддерживает версии Android старше 3.0, вы можете использовать заголовки для предоставления двухпанельного макета при работе на Android 3.0 или более поздней версии. Достаточно создать дополнительный XML-файл настроек, использующий базовые элементы [<Preference>](https://developer.android.com/reference/android/preference/Preference.html?hl=ru), которые ведут себя аналогично пунктам заголовка (для использования в более старых версиях Android).

Вместо открытия новых экранов [PreferenceScreen](https://developer.android.com/reference/android/preference/PreferenceScreen.html?hl=ru) каждый из элементов [<Preference>](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) отправляет намерение [Intent](https://developer.android.com/reference/android/content/Intent.html?hl=ru) в [PreferenceActivity](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru) с указанием XML-файла предпочтений для загрузки.

В качестве примера приведен XML-файл для заголовков предпочтений, который используется в Android версии 3.0 и более поздних версий (res/xml/preference\_headers.xml):

<preference-headers xmlns:android="http://schemas.android.com/apk/res/android">  
    <header  
        android:fragment="com.example.prefs.SettingsFragmentOne"  
        android:title="@string/prefs\_category\_one"  
        android:summary="@string/prefs\_summ\_category\_one" />  
    <header  
        android:fragment="com.example.prefs.SettingsFragmentTwo"  
        android:title="@string/prefs\_category\_two"  
        android:summary="@string/prefs\_summ\_category\_two" />  
</preference-headers>

А здесь представлен файл предпочтений, который содержит те же самые заголовки для версий старше Android 3.0 (res/xml/preference\_headers\_legacy.xml):

<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">  
    <Preference  
        android:title="@string/prefs\_category\_one"  
        android:summary="@string/prefs\_summ\_category\_one"  >  
        <intent  
            android:targetPackage="com.example.prefs"  
            android:targetClass="com.example.prefs.SettingsActivity"  
            android:action="com.example.prefs.PREFS\_ONE" />  
    </Preference>  
    <Preference  
        android:title="@string/prefs\_category\_two"  
        android:summary="@string/prefs\_summ\_category\_two" >  
        <intent  
            android:targetPackage="com.example.prefs"  
            android:targetClass="com.example.prefs.SettingsActivity"  
            android:action="com.example.prefs.PREFS\_TWO" />  
    </Preference>  
</PreferenceScreen>

Так как поддержка &lt;preference-headers&gt; была добавлена в версии Android 3.0, система вызывает[onBuildHeaders()](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru#onBuildHeaders(java.util.List%3Candroid.preference.PreferenceActivity.Header%3E)) в методе [PreferenceActivity](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru) только при работе в Android версии 3.0 или более поздней версии. Чтобы загрузить «старый» файл заголовков (preference\_headers\_legacy.xml), вы должны проверить версию Android и, если версия старше Android 3.0 ([HONEYCOMB](https://developer.android.com/reference/android/os/Build.VERSION_CODES.html?hl=ru#HONEYCOMB)), вызвать [addPreferencesFromResource()](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru" \l "addPreferencesFromResource(int)) для загрузки старого файла заголовков. Например:

@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    ...  
  
    if (Build.VERSION.SDK\_INT < Build.VERSION\_CODES.HONEYCOMB) {  
        // Load the legacy preferences headers  
        addPreferencesFromResource(R.xml.preference\_headers\_legacy);  
    }  
}  
  
// Called only on Honeycomb and later  
@Override  
public void onBuildHeaders(List<Header> target) {  
   loadHeadersFromResource(R.xml.preference\_headers, target);  
}

Остается обработать намерение [Intent](https://developer.android.com/reference/android/content/Intent.html?hl=ru), переданное в операцию, чтобы идентифицировать файл предпочтений для загрузки. Поэтому извлеките операцию намерения и сравните ее с известными строками действия, которые вы использовали в тегах &lt;intent&gt; XML-файла предпочтений:

final static String ACTION\_PREFS\_ONE = "com.example.prefs.PREFS\_ONE";  
...  
  
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
  
    String action = getIntent().getAction();  
    if (action != null && action.equals(ACTION\_PREFS\_ONE)) {  
        addPreferencesFromResource(R.xml.preferences);  
    }  
    ...  
  
    else if (Build.VERSION.SDK\_INT < Build.VERSION\_CODES.HONEYCOMB) {  
        // Load the legacy preferences headers  
        addPreferencesFromResource(R.xml.preference\_headers\_legacy);  
    }  
}

При этом помните, что последующие вызовы [addPreferencesFromResource()](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru" \l "addPreferencesFromResource(int)) будут помещать все предпочтения в один список, поэтому обязательно используйте операторы else-if, чтобы обеспечить только однократный вызов метода при изменении условий.

Чтение предпочтений

По умолчанию все предпочтения вашего приложения сохраняются в файле, который доступен из любого места вашего приложения посредством вызова статического метода [PreferenceManager.getDefaultSharedPreferences()](https://developer.android.com/reference/android/preference/PreferenceManager.html?hl=ru" \l "getDefaultSharedPreferences(android.content.Context)). Он возвращает объект [SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html?hl=ru), содержащий все пары «ключ-значение», связанные с объектами [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru), использованными в вашей операции [PreferenceActivity](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru).

В качестве примера показано чтение одного из значений предпочтений из любой другой операции в вашем приложении:

SharedPreferences sharedPref = PreferenceManager.getDefaultSharedPreferences(this);  
String syncConnPref = sharedPref.getString(SettingsActivity.KEY\_PREF\_SYNC\_CONN, "");

Отслеживание изменений предпочтений

Существует несколько причин, по которым вы можете захотеть получать уведомления, как только пользователь изменяет одно из предпочтений. Чтобы получать обратный вызов при изменении любого из предпочтений, реализуйте интерфейс [SharedPreference.OnSharedPreferenceChangeListener](https://developer.android.com/reference/android/content/SharedPreferences.OnSharedPreferenceChangeListener.html?hl=ru) и зарегистрируйте приемник для объекта [SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html?hl=ru) посредством вызова [registerOnSharedPreferenceChangeListener()](https://developer.android.com/reference/android/content/SharedPreferences.html?hl=ru" \l "registerOnSharedPreferenceChangeListener(android.content.SharedPreferences.OnSharedPreferenceChangeListener)).

Этот интерфейс содержит только один метод обратного вызова, [onSharedPreferenceChanged()](https://developer.android.com/reference/android/content/SharedPreferences.OnSharedPreferenceChangeListener.html?hl=ru" \l "onSharedPreferenceChanged(android.content.SharedPreferences,%20java.lang.String)), и вы, вероятно, сочтете его самым простым способом реализации интерфейса в составе своей операции. Например:

public class SettingsActivity extends PreferenceActivity  
                              implements OnSharedPreferenceChangeListener {  
    public static final String KEY\_PREF\_SYNC\_CONN = "pref\_syncConnectionType";  
    ...  
  
    public void onSharedPreferenceChanged(SharedPreferences sharedPreferences,  
        String key) {  
        if (key.equals(KEY\_PREF\_SYNC\_CONN)) {  
            Preference connectionPref = findPreference(key);  
            // Set summary to be the user-description for the selected value  
            connectionPref.setSummary(sharedPreferences.getString(key, ""));  
        }  
    }  
}

В этом примере метод проверяет, выполнено ли изменение настройки для известного ключа предпочтений. Он вызывает [findPreference()](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru" \l "findPreference(java.lang.CharSequence)) для получения объекта [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru), который был изменен, поэтому он может изменить сводку пункта , описывающего выбор пользователя. То есть, когда настройка представляет собой [ListPreference](https://developer.android.com/reference/android/preference/ListPreference.html?hl=ru) или другую настройку с несколькими вариантами выбора, при изменении этой настройки вы должны вызвать [setSummary()](https://developer.android.com/reference/android/preference/Preference.html?hl=ru" \l "setSummary(int)) для отображения текущего состояния (например, настройка спящего режима, показанная на рисунке 5).

**Примечание.** В соответствии с рекомендациями раздела [Настройки](https://developer.android.com/design/patterns/settings.html?hl=ru) руководства «Дизайн для Android», мы рекомендуем вам обновлять сводку для **[ListPreference](https://developer.android.com/reference/android/preference/ListPreference.html?hl=ru)** при каждом изменении предпочтения пользователем, чтобы описать текущую настройку.

Для правильного управления жизненным циклом в операции мы рекомендуем вам регистрировать или отменять регистрацию вашего приемника [SharedPreferences.OnSharedPreferenceChangeListener](https://developer.android.com/reference/android/content/SharedPreferences.OnSharedPreferenceChangeListener.html?hl=ru) во время выполнения обратных вызовов [onResume()](https://developer.android.com/reference/android/app/Activity.html?hl=ru" \l "onResume()) и [onPause()](https://developer.android.com/reference/android/app/Activity.html?hl=ru" \l "onPause()) соответственно:

@Override  
protected void onResume() {  
    super.onResume();  
    getPreferenceScreen().getSharedPreferences()  
            .registerOnSharedPreferenceChangeListener(this);  
}  
  
@Override  
protected void onPause() {  
    super.onPause();  
    getPreferenceScreen().getSharedPreferences()  
            .unregisterOnSharedPreferenceChangeListener(this);  
}

**Внимание!** Когда вы вызываете приемник **[registerOnSharedPreferenceChangeListener()](https://developer.android.com/reference/android/content/SharedPreferences.html?hl=ru" \l "registerOnSharedPreferenceChangeListener(android.content.SharedPreferences.OnSharedPreferenceChangeListener))**, диспетчер предпочтений не сохраняет строгую ссылку на приемник. Вы должны сохранить строгую ссылку на приемник, в противном случае она будет чувствительной к очистке памяти. Мы рекомендуем хранить ссылку на приемник в данных экземпляра объекта , который будет существовать, пока вам нужен приемник.

Например, в следующем коде вызывающий объект не сохраняет ссылку на приемник. В результате этого приемник будет удален при очистке памяти и через некоторое время приведет к сбою:

prefs.registerOnSharedPreferenceChangeListener(  
  // Bad! The listener is subject to garbage collection!  
  new SharedPreferences.OnSharedPreferenceChangeListener() {  
  public void onSharedPreferenceChanged(SharedPreferences prefs, String key) {  
    // listener implementation  
  }  
});

Вместо этого сохраните ссылку на приемник в поле данных экземпляра объекта , который будет существовать, пока нужен приемник:

SharedPreferences.OnSharedPreferenceChangeListener listener =  
    new SharedPreferences.OnSharedPreferenceChangeListener() {  
  public void onSharedPreferenceChanged(SharedPreferences prefs, String key) {  
    // listener implementation  
  }  
};  
prefs.registerOnSharedPreferenceChangeListener(listener);

Контроль использования сети

Начиная с версии Android 4.0, системное приложение «Настройки» позволяет пользователям просматривать использование сетевых данных приложениями, работающими на переднем плане и в фоновом режиме. После этого пользователи могут отключить использование данных в фоновом режиме для отдельных приложений. Для того, чтобы пользователи не отключали доступ вашего приложения к данным в фоновом режиме, вы должны эффективно использовать подключение в режиме передачи данных и предоставить пользователям возможность настройки использования данных вашим приложением посредством настроек приложения.

Например, вы можете позволить пользователям управлять частотой синхронизации данных приложения, выполнением загрузки только в режиме подключения по Wi-Fi, использованием данных в роуминге и т. д. Когда эти возможности управления доступны, пользователи с меньшей вероятностью отключат доступ вашего приложения к данным, когда оно достигает установленных в системных настройках лимитов, поскольку вместо отключения они могут точно контролировать объем данных, который использует ваше приложение.

После добавления необходимых предпочтений в вашу операцию [PreferenceActivity](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru) для управления поведением вашего приложения в отношении данных вы должны добавить фильтр намерений для [ACTION\_MANAGE\_NETWORK\_USAGE](https://developer.android.com/reference/android/content/Intent.html?hl=ru#ACTION_MANAGE_NETWORK_USAGE) в вашем файле манифеста. Например:

<activity android:name="SettingsActivity" ... >  
    <intent-filter>  
       <action android:name="android.intent.action.MANAGE\_NETWORK\_USAGE" />  
       <category android:name="android.intent.category.DEFAULT" />  
    </intent-filter>  
</activity>

Этот фильтр манифеста указывает системе, что эта операция управляет использованием данных вашим приложением. Так, когда пользователь проверяет объем использованных приложением данных в системном приложении «Настройки», отображается кнопка *Просмотреть настройки приложения*, которая запускает вашу операцию [PreferenceActivity](https://developer.android.com/reference/android/preference/PreferenceActivity.html?hl=ru), чтобы пользователь мог уточнить, сколько данных использует ваше приложение.

Построение пользовательского предпочтения

Система Android содержит множество подклассов [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru), которые позволяют вам строить пользовательский интерфейс для нескольких различных типов настроек. Тем не менее, вы можете обнаружить, что для нужной вам настройки нет встроенного решения, например, для выбора числа или даты. В таком случае вам потребуется создать нестандартное предпочтение путем наследования класса [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru)или одного из других подклассов.

При наследовании класса [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) нужно выполнить несколько важных пунктов:

* Укажите пользовательский интерфейс, который должен отображаться при выборе этой настройки пользователем.
* При необходимости сохраните значение настройки.
* Инициализируйте [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) текущим значением (или значением по умолчанию), когда предпочтение отображается.
* Укажите значение по умолчанию в ответ на запрос системы.
* Если [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) содержит свой собственный пользовательский интерфейс (например, диалоговое окно), сохраните и восстановите состояние для обработки изменений жизненного цикла (например, когда пользователь поворачивает экран).

В следующих разделах описано выполнение каждой из этих задач.

Указание пользовательского интерфейса

Если вы наследуете класс [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) непосредственно, вы должны реализовать метод [onClick()](https://developer.android.com/reference/android/preference/Preference.html?hl=ru" \l "onClick()), чтобы задать действие, происходящее при выборе пункта пользователем. Однако большая часть нестандартных настроек наследует [DialogPreference](https://developer.android.com/reference/android/preference/DialogPreference.html?hl=ru), чтобы отобразить диалоговое окно, что упрощает процедуру. Когда вы наследуете [DialogPreference](https://developer.android.com/reference/android/preference/DialogPreference.html?hl=ru), вы должны вызвать [setDialogLayoutResourcs()](https://developer.android.com/reference/android/preference/DialogPreference.html?hl=ru" \l "setDialogLayoutResource(int)), находясь в конструкторе класса, чтобы указать макет диалогового окна.

В качестве примера показан конструктор нестандартного диалогового окна [DialogPreference](https://developer.android.com/reference/android/preference/DialogPreference.html?hl=ru), в котором объявляется макет и указывается текст для положительной и отрицательной кнопок диалога по умолчанию:

public class NumberPickerPreference extends DialogPreference {  
    public NumberPickerPreference(Context context, AttributeSet attrs) {  
        super(context, attrs);  
  
        setDialogLayoutResource(R.layout.numberpicker\_dialog);  
        setPositiveButtonText(android.R.string.ok);  
        setNegativeButtonText(android.R.string.cancel);  
  
        setDialogIcon(null);  
    }  
    ...  
}

Сохранение значения настройки

Вы можете сохранить значение настройки в любой момент, вызвав один из методов persist\*() класса [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru), например, [persistInt()](https://developer.android.com/reference/android/preference/Preference.html?hl=ru" \l "persistInt(int)), если настройка имеет целое значение, или [persistBoolean()](https://developer.android.com/reference/android/preference/Preference.html?hl=ru" \l "persistBoolean(boolean)) для сохранения логического значения.

**Примечание.** Каждое предпочтение **[Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru)** может сохранять только один тип данных, поэтому вы должны использовать метод **persist\*()**, соответствующий типу данных, используемых вашим пользовательским предпочтением **[Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru)**.

Выбор метода сохранения настройки может зависеть от наследованного класса [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru). Если вы наследуете [DialogPreference](https://developer.android.com/reference/android/preference/DialogPreference.html?hl=ru), вы должны сохранять значение только при закрытии диалога с положительным результатом (пользователь нажал кнопку «OK»).

Когда [DialogPreference](https://developer.android.com/reference/android/preference/DialogPreference.html?hl=ru) закрывается, система вызывает метод [onDialogClosed()](https://developer.android.com/reference/android/preference/DialogPreference.html?hl=ru" \l "onDialogClosed(boolean)). Этот метод содержит логический аргумент, который указывает, является ли результат пользователя «положительным» — если аргумент имеет значение true, значит пользователь выбрал положительную кнопку и вы должны сохранить новое значение. Например:

@Override  
protected void onDialogClosed(boolean positiveResult) {  
    // When the user selects "OK", persist the new value  
    if (positiveResult) {  
        persistInt(mNewValue);  
    }  
}

В этом примере mNewValue — это член класса, который содержит текущее значение настройки. При вызове [persistInt()](https://developer.android.com/reference/android/preference/Preference.html?hl=ru" \l "persistInt(int)) значение сохраняется в файл [SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html?hl=ru) (с автоматическим использованием ключа, указанного в XML-файле для этого предпочтения [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru)).

Инициализация текущего значения

Когда система добавляет ваше предпочтение [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) на экран, она вызывает метод [onSetInitialValue()](https://developer.android.com/reference/android/preference/Preference.html?hl=ru" \l "onSetInitialValue(boolean,%20java.lang.Object)), чтобы уведомить вас, имеет ли настройка сохраненное значение. Если сохраненного значения нет, этот вызов предоставляет вам значение по умолчанию.

Метод [onSetInitialValue()](https://developer.android.com/reference/android/preference/Preference.html?hl=ru" \l "onSetInitialValue(boolean,%20java.lang.Object)) передает логическое значение, restorePersistedValue, чтобы показать, было ли уже сохранено значение для настройки. Если значение равно true, вы должны извлечь сохраненное значение, вызвав один из методов getPersisted\*() класса [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru), например, [getPersistedInt()](https://developer.android.com/reference/android/preference/Preference.html?hl=ru" \l "getPersistedInt(int)) для целого значения. Обычно требуется извлечь сохраненное значение, чтобы можно было правильно обновить пользовательский интерфейс для отражения ранее сохраненного значения.

Если restorePersistedValue имеет значение false, вы должны использовать значение по умолчанию, которое передается во втором аргументе.

@Override  
protected void onSetInitialValue(boolean restorePersistedValue, Object defaultValue) {  
    if (restorePersistedValue) {  
        // Restore existing state  
        mCurrentValue = this.getPersistedInt(DEFAULT\_VALUE);  
    } else {  
        // Set default state from the XML attribute  
        mCurrentValue = (Integer) defaultValue;  
        persistInt(mCurrentValue);  
    }  
}

Каждый метод getPersisted\*() содержит аргумент, который указывает значение по умолчанию на случай, когда действительно нет сохраненного значения, или не существует ключ. В приведенном выше примере локальная константа служит для указания значения по умолчанию на случай, если [getPersistedInt()](https://developer.android.com/reference/android/preference/Preference.html?hl=ru" \l "getPersistedInt(int)) не может вернуть сохраненное значение.

**Внимание!** Вы **не можете** использовать **defaultValue** в качестве значения по умолчанию в методе **getPersisted\*()**, так как его значение всегда равно null, когда **restorePersistedValue** имеет значение **true**.

Предоставление значения по умолчанию

Если экземпляр вашего класса [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) указывает значение по умолчанию (с помощью атрибута android:defaultValue), система вызывает [onGetDefaultValue()](https://developer.android.com/reference/android/preference/Preference.html?hl=ru" \l "onGetDefaultValue(android.content.res.TypedArray,%20int)), когда она создает экземпляр объекта для извлечения значения. Вы должны реализовать этот метод, чтобы сохранить значение по умолчанию для системы в [SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html?hl=ru). Например:

@Override  
protected Object onGetDefaultValue(TypedArray a, int index) {  
    return a.getInteger(index, DEFAULT\_VALUE);  
}

Аргументы метода предоставляют все необходимое: массив атрибутов и указатель положения android:defaultValue, который вы должны извлечь. Причина, по которой вы должны реализовать этот метод, чтобы извлечь значение по умолчанию из атрибута, состоит в том, что вы должны указать локальное значение по умолчанию для атрибута в случае, когда значение не определено.

Сохранение и восстановление состояния предпочтений

Как и [View](https://developer.android.com/reference/android/view/View.html?hl=ru) в макете, ваш подкласс [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) отвечает за сохранение и восстановление своего состояния в случае перезапуска операции или фрагмента (например, когда пользователь поворачивает экран). Чтобы правильно сохранять и восстанавливать состояние вашего класса [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru), вы должны реализовать методы обратного вызова жизненного цикла [onSaveInstanceState()](https://developer.android.com/reference/android/preference/Preference.html?hl=ru" \l "onSaveInstanceState()) и [onRestoreInstanceState()](https://developer.android.com/reference/android/preference/Preference.html?hl=ru" \l "onRestoreInstanceState(android.os.Parcelable)).

Состояние вашего [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) определяется объектом, который реализует интерфейс [Parcelable](https://developer.android.com/reference/android/os/Parcelable.html?hl=ru). Система Android предоставляет вам такой объект в качестве начальной точки для определения вашего объекта состояния: класс [Preference.BaseSavedState](https://developer.android.com/reference/android/preference/Preference.BaseSavedState.html?hl=ru).

Чтобы определить, как ваш класс [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) сохраняет свое состояние, вы должны наследовать класс [Preference.BaseSavedState](https://developer.android.com/reference/android/preference/Preference.BaseSavedState.html?hl=ru). Вы должны переопределить лишь несколько методов и определить объект[CREATOR](https://developer.android.com/reference/android/preference/Preference.BaseSavedState.html?hl=ru#CREATOR).

Для большинства приложений вы можете скопировать следующую реализацию и просто изменить строки, которые обрабатывают value, если ваш подкласс [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru) сохраняет типы данных, отличные от целых.

private static class SavedState extends BaseSavedState {  
    // Member that holds the setting's value  
    // Change this data type to match the type saved by your Preference  
    int value;  
  
    public SavedState(Parcelable superState) {  
        super(superState);  
    }  
  
    public SavedState(Parcel source) {  
        super(source);  
        // Get the current preference's value  
        value = source.readInt();  // Change this to read the appropriate data type  
    }  
  
    @Override  
    public void writeToParcel(Parcel dest, int flags) {  
        super.writeToParcel(dest, flags);  
        // Write the preference's value  
        dest.writeInt(value);  // Change this to write the appropriate data type  
    }  
  
    // Standard creator object using an instance of this class  
    public static final Parcelable.Creator<SavedState> CREATOR =  
            new Parcelable.Creator<SavedState>() {  
  
        public SavedState createFromParcel(Parcel in) {  
            return new SavedState(in);  
        }  
  
        public SavedState[] newArray(int size) {  
            return new SavedState[size];  
        }  
    };  
}

После добавления показанной выше реализации [Preference.BaseSavedState](https://developer.android.com/reference/android/preference/Preference.BaseSavedState.html?hl=ru) в ваше приложение (обычно в качестве подкласса вашего подкласса [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru)), вам потребуется реализовать методы [onSaveInstanceState()](https://developer.android.com/reference/android/preference/Preference.html?hl=ru" \l "onSaveInstanceState()) и [onRestoreInstanceState()](https://developer.android.com/reference/android/preference/Preference.html?hl=ru" \l "onRestoreInstanceState(android.os.Parcelable)) для вашего подкласса [Preference](https://developer.android.com/reference/android/preference/Preference.html?hl=ru).

Например:

@Override  
protected Parcelable onSaveInstanceState() {  
    final Parcelable superState = super.onSaveInstanceState();  
    // Check whether this Preference is persistent (continually saved)  
    if (isPersistent()) {  
        // No need to save instance state since it's persistent,  
        // use superclass state  
        return superState;  
    }  
  
    // Create instance of custom BaseSavedState  
    final SavedState myState = new SavedState(superState);  
    // Set the state's value with the class member that holds current  
    // setting value  
    myState.value = mNewValue;  
    return myState;  
}  
  
@Override  
protected void onRestoreInstanceState(Parcelable state) {  
    // Check whether we saved the state in onSaveInstanceState  
    if (state == null || !state.getClass().equals(SavedState.class)) {  
        // Didn't save the state, so call superclass  
        super.onRestoreInstanceState(state);  
        return;  
    }  
  
    // Cast state to custom BaseSavedState and pass to superclass  
    SavedState myState = (SavedState) state;  
    super.onRestoreInstanceState(myState.getSuperState());  
  
    // Set this Preference's widget to reflect the restored state  
    mNumberPicker.setValue(myState.value);  
}