



Лекция 6

Мобильная разработка

Пирская Любовь Владимировна,
к.т.н., доцент кафедры МОП ЭВМ
lpirskaya@sfedu.ru

Уведомления

Виды уведомлений

- DialogFragment
- Toast & snackbars
- Notification

Типы назначений уведомлений

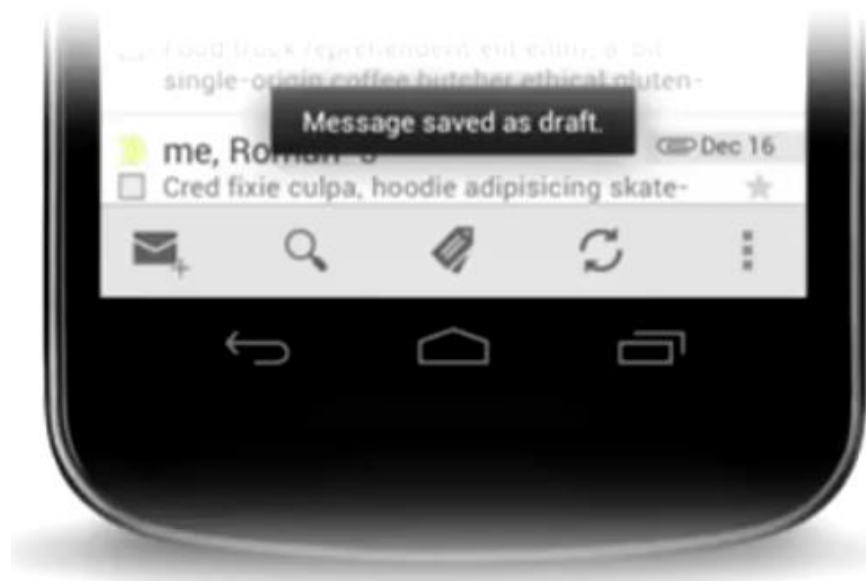
- DialogFragment & toast & snackbars
 - Обратная связь
 - Только когда приложение на экране
- Notification
 - Важные уведомления
 - Приложение работает в фоне
 - Push

DialogFragment

- Fragment
- Dialog
- Поддержка в SupportLibrary

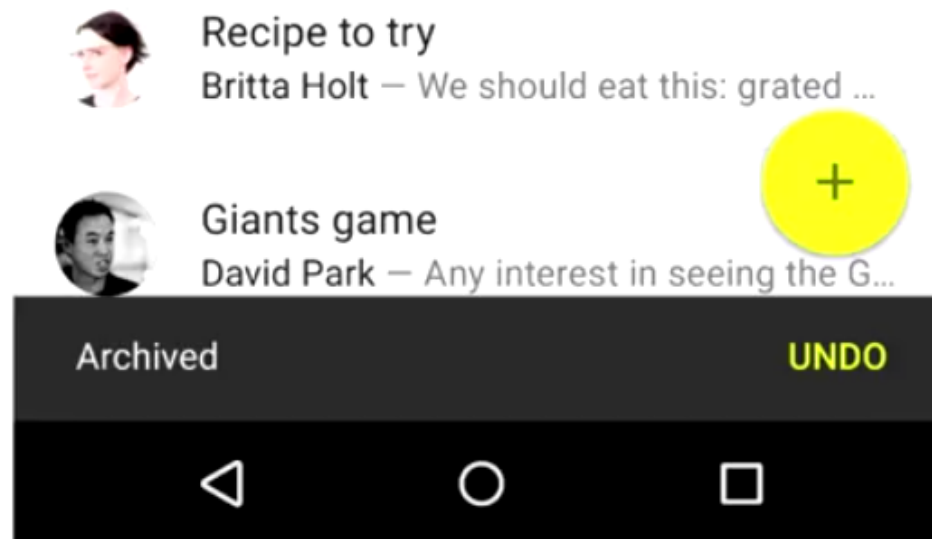
Toast

- Сообщение, не требующее действия от пользователя
- Короткое время показа



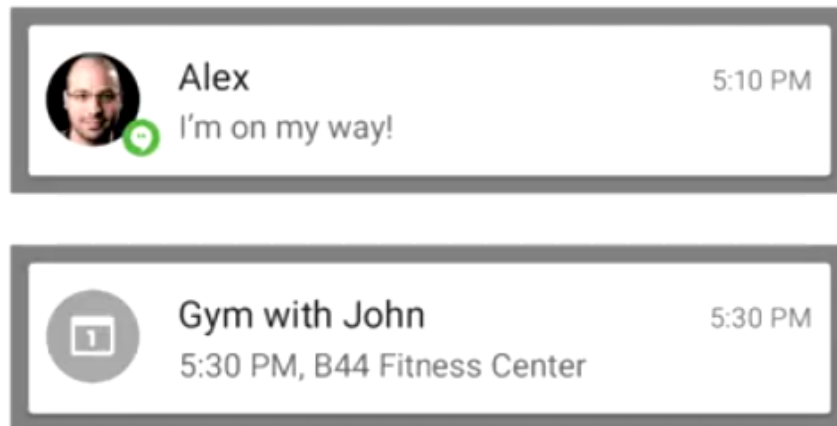
Snackbar

- Feedback о действии
- Action кнопка
- Swipe off - удалить с экрана



Notifications

- Действие внутри UI нотификации
- Дополнительная информация при раскрытии UI нотификации
- Совмещение нескольких нотификаций



Notifications. Когда использовать?

- Если события привязаны ко времени: календарь, чат
- Если событие действительно важное: разрядка батареи, входящие звонки
- Уведомления о фоновых процессах:
- Музыкальный плеер, скачивание файла и т.д.

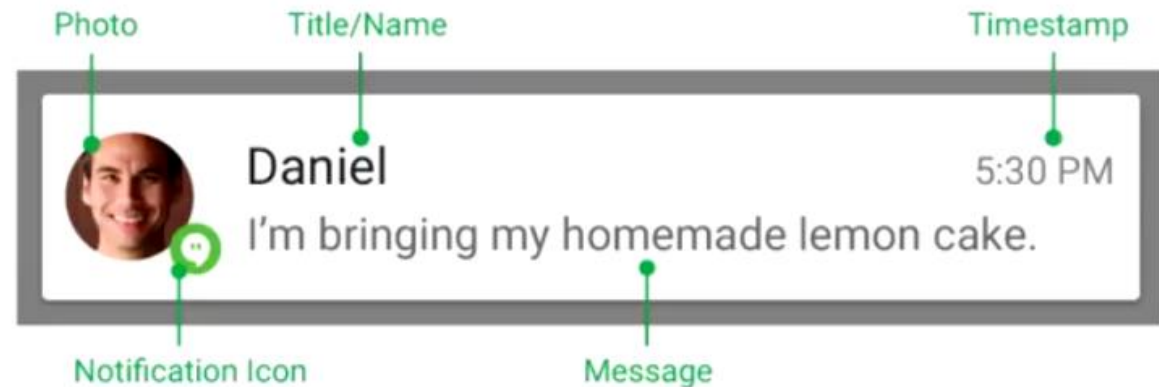
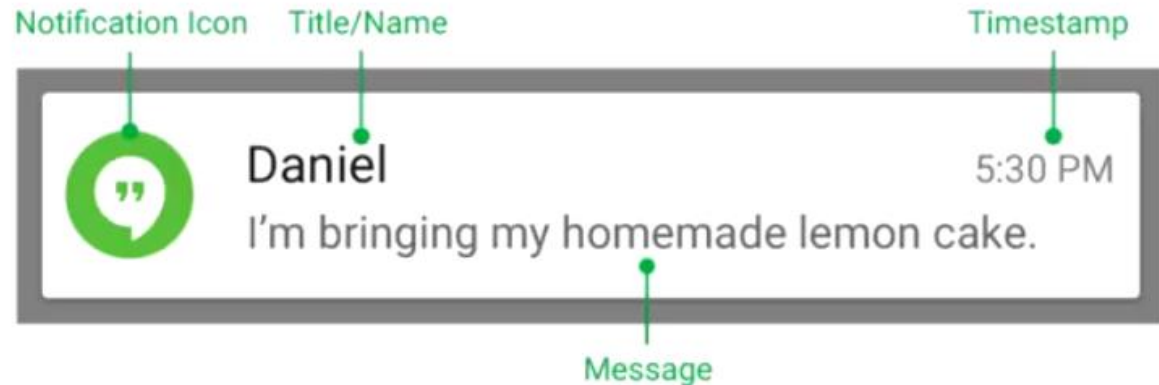
Notifications. Когда НЕ использовать?

- Сообщение об обновлении/сохранении...
- Если запущено приложение, событие отображается в UI
- Сообщения об ошибках, которые не требуют реакции пользователя
- Реклама приложения, без полезного содержимого
- Отображение постоянно обновляемой информации (использовать виджеты)

Notifications. Взаимодействие

- Отображаются пиктограммами в области уведомлений
- Свайпом – удаляются. Как правило, уведомления о фоновых процессах удалить нельзя
- По клику переход в связанное приложение

Notifications. Отображение



Notifications. Создание действия

```
/**
 * Create PendingIntent
 *
 * @return PendingIntent
 */
private PendingIntent getPendingIntent() {
    int requestID = (int) System.currentTimeMillis();

    Intent firstIntent = new Intent(this, FirstActivity.class);
    Intent secondIntent = new Intent(this, SecondActivity.class);

    TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
    stackBuilder.addParentStack(FirstActivity.class);
    stackBuilder.addNextIntent(firstIntent);
    stackBuilder.addNextIntent(secondIntent);

    return stackBuilder.getPendingIntent(requestID, PendingIntent.FLAG_UPDATE_CURRENT);
}
```

Notifications. Создание

```
public static final int NOTIF_ID = 1;

/**
 * Create
 */
private void createNotification() {
    NotificationCompat.Builder builder =
        new NotificationCompat.Builder(this)
            .setSmallIcon(R.drawable.ic_launcher)
            .setAutoCancel(true)
            .setContentTitle(getString(R.string.title_notification))
            .setContentText(getString(R.string.text_notification))
            .setContentIntent(getPendingIntent());

    NotificationManager mNotificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
    mNotificationManager.notify(NOTIF_ID, builder.build());
}
```


Notifications. Обновление

```
/**
 * Update
 */
private void updateNotification() {
    NotificationCompat.Builder updateBuilder =
        new NotificationCompat.Builder(this)
            .setWhen(System.currentTimeMillis())
            .setAutoCancel(true)
            .setSmallIcon(R.drawable.ic_action_event)
            .setContentTitle(getString(R.string.title_update_notification))
            .setContentText(getString(R.string.text_update_notification))
            .setContentIntent(getPendingIntent());

    NotificationManager mNotificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
    mNotificationManager.notify(NOTIF_ID, updateBuilder.build());
}
```

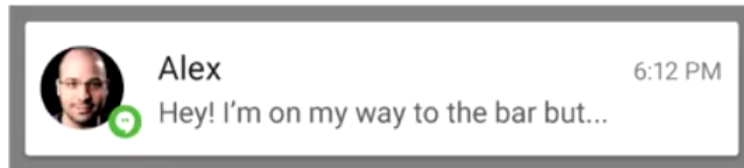

Notifications. Удаление

```
/**
 * Remove
 */
public static void removeNotification(final Context context) {
    NotificationManager mNotificationManager = (NotificationManager) context
        .getSystemService(Context.NOTIFICATION_SERVICE);
    mNotificationManager.cancel(NOTIF_ID);
}
```

Notifications. Удаление

```
/**
 * Remove
 */
public static void removeNotification(final Context context) {
    NotificationManager mNotificationManager = (NotificationManager) context
        .getSystemService(Context.NOTIFICATION_SERVICE);
    mNotificationManager.cancel(NOTIF_ID);
}
```

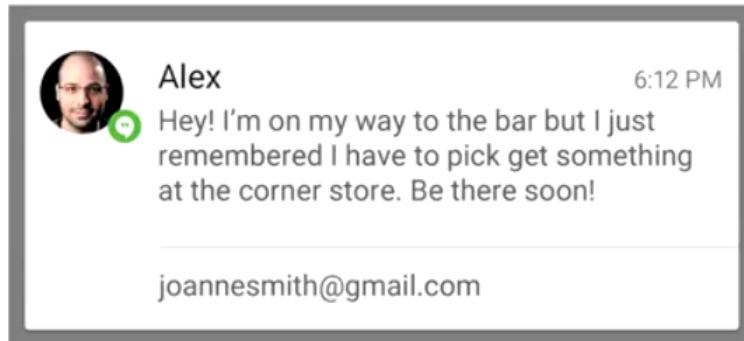
Notifications. Расширенный вид



A notification card for a contact named Alex. It features a circular profile picture on the left, the name 'Alex' in bold, and the time '6:12 PM' on the right. The message text is 'Hey! I'm on my way to the bar but...'. A small green icon with a white 'm' is visible next to the profile picture.

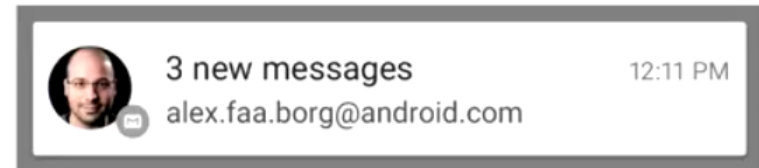
EXPAND
↓

↑
CONTRACT



An expanded version of the notification card for Alex. It shows the full message text: 'Hey! I'm on my way to the bar but I just remembered I have to pick get something at the corner store. Be there soon!'. Below the message, the email address 'joannesmith@gmail.com' is displayed. The card is enclosed in a grey border.

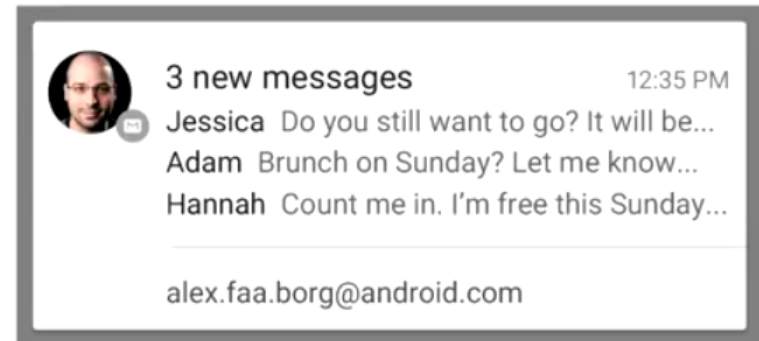
TEXT



A notification card for a contact named alex.faa.borg@android.com. It features a circular profile picture on the left, the text '3 new messages' in bold, and the time '12:11 PM' on the right. The message text is 'alex.faa.borg@android.com'. A small grey icon with a white 'm' is visible next to the profile picture.

EXPAND
↓

↑
CONTRACT



An expanded version of the notification card for alex.faa.borg@android.com. It shows the full message text: 'Jessica Do you still want to go? It will be... Adam Brunch on Sunday? Let me know... Hannah Count me in. I'm free this Sunday...'. Below the message, the email address 'alex.faa.borg@android.com' is displayed. The card is enclosed in a grey border.

INBOX

Notifications. Расширенный вид

- BigTextStyle
 - Расширенное текстовое описание
- BigPictureStyle
 - Отображение большой картинки
- InboxStyle
 - Отображение списка до 5 строк
- MediaStyle
 - Взаимодействие с музыкальным плеером

Notifications. Расширенный вид

```
/**
 * Get expanded
 *
 * @return NotificationCompat.InboxStyle
 */
private NotificationCompat.InboxStyle getExpanded() {
    NotificationCompat.InboxStyle inboxStyle = new NotificationCompat.InboxStyle();

    inboxStyle.setBigContentTitle(getString(R.string.big_title_notification));
    for (int i = 1; i <= 3; i++) {
        inboxStyle.addLine(getString(R.string.big_line_notification) + " " + i);
    }

    return inboxStyle;
}

/**
 * Expanded
 */
private void expandedNotification() {
    NotificationCompat.Builder expandedBuilder =
        new NotificationCompat.Builder(this)
            .setWhen(System.currentTimeMillis())
            .setAutoCancel(true)
            .setSmallIcon(R.drawable.ic_launcher)
            .setContentTitle(getString(R.string.title_notification))
            .setContentText(getString(R.string.text_notification))
            .setContentIntent(getPendingIntent())
            .setStyle(getExpanded());

    NotificationManager mNotificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
    mNotificationManager.notify(NOTIF_ID, expandedBuilder.build());
}
```

Heads-up нотификации

- Высокий приоритет
- Короткое время
- В расширенном виде

- Например

- Входящий звонок
- Новое sms
- Низкий заряд батареи



Приоритет нотификаций

- MAX
- HIGH
- DEFAULT
- LOW
- MIN

Приоритет нотификаций. МАХ

- Срочные и зависящие от времени:
 - Входящий звонок
 - Ведение по маршруту
 - Оповещение о чрезвычайных ситуациях

Приоритет нотификаций. HIGH и DEFAULT

- HIGH
 - Важные email
 - Сообщения чата
 - Sms
- DEFAULT
 - Приоритет по умолчанию
 - Для всех legacy уведомлений

Приоритет нотификаций. LOW и MIN

- LOW
 - Обновление приложений
- MIN
 - Пропущенные события
 - Подсказки
 - Погода
 - Не отображаются в statusBar

Heads-up. Пример

- Максимальный приоритет и установлен рингтон или включена вибрация
- Режим fullscreen (`fullScreenIntent`)

Heads-up. Пример

```
/**
 * Heads-up
 */
@TargetApi(Build.VERSION_CODES.JELLY_BEAN)
private void headsUpNotification() {
    // close
    Intent closeIntent = new Intent(this, ActionNotificationReceiver.class);
    PendingIntent pendingIntentClose = PendingIntent
        .getBroadcast(this, 0, closeIntent, PendingIntent.FLAG_UPDATE_CURRENT);
    NotificationCompat.Action closeAction = new NotificationCompat.Action(0, getString(R.string.close),
        pendingIntentClose);

    // show
    NotificationCompat.Action showAction = new NotificationCompat.Action(0, getString(R.string.more),
        getPendingIntent());

    NotificationCompat.Builder actiondBuilder =
        new NotificationCompat.Builder(this)
            .setWhen(System.currentTimeMillis())
            .setSmallIcon(R.drawable.ic_launcher)
            .setContentTitle(getString(R.string.title_notification))
            .setContentText(getString(R.string.text_notification))
            .setAutoCancel(true)
            .addAction(showAction)
            .addAction(closeAction);

    // Heads-up
    actiondBuilder.setPriority(Notification.PRIORITY_MAX)
        .setDefaults(Notification.DEFAULT_SOUND | Notification.DEFAULT_VIBRATE);
    actiondBuilder.setFullScreenIntent(getPendingIntent(), true);

    NotificationManager mNotificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
    mNotificationManager.notify(NOTIF_ID, actiondBuilder.build());
}
```

Экран блокировки

- VISIBILITY_PUBLIC
 - Полное содержание (по умолчанию)
- VISIBILITY_PRIVATE
 - Содержание без личной информации
- VISIBILITY_SECRET
 - Минимальное содержание

Экран блокировки. Пример

```
/**
 * Lock Screen
 */
@TargetApi(Build.VERSION_CODES.LOLLIPOP)
private void lockScreenNotification() {
    // close
    Intent closeIntent = new Intent(this, ActionNotificationReceiver.class);
    PendingIntent pendingIntentClose = PendingIntent
        .getBroadcast(this, 0, closeIntent, PendingIntent.FLAG_UPDATE_CURRENT);
    NotificationCompat.Action closeAction = new NotificationCompat.Action(0, getString(R.string.close),
        pendingIntentClose);

    // show
    NotificationCompat.Action showAction = new NotificationCompat.Action(0, getString(R.string.more),
        getPendingIntent());

    NotificationCompat.Builder actiondBuilder =
        new NotificationCompat.Builder(this)
            .setWhen(System.currentTimeMillis())
            .setAutoCancel(true)
            .setSmallIcon(R.drawable.ic_launcher)
            .setContentTitle(getString(R.string.title_notification))
            .setContentText(getString(R.string.text_notification))
            .addAction(showAction)
            .addAction(closeAction)
            .setPriority(Notification.PRIORITY_DEFAULT);

    // Lock Screen
    actiondBuilder.setVisibility(Notification.VISIBILITY_PUBLIC);
    actiondBuilder.setVisibility(Notification.VISIBILITY_SECRET);
    actiondBuilder.setVisibility(Notification.VISIBILITY_PRIVATE);

    NotificationManager mNotificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
    mNotificationManager.notify(NOTIF_ID, actiondBuilder.build());
}
```


The background features a complex, abstract design. The top and bottom edges are decorated with sharp, overlapping geometric shapes in a palette of yellow, orange, red, and magenta. The central portion of the image is a large, soft-focus area in shades of pink and light orange, creating a dreamy, ethereal atmosphere. The text 'Styles & Themes' is centered within this soft-focus area.

Styles & Themes

Что это?

- Стиль – xml файл, в который выносят атрибуты, относящиеся к стилизации того или иного виджета(Text View, ImageView и проч.)
- Тема – xml файл, содержащий ссылки на стили, применяемый к activity или приложению в целом

Применение

- Переиспользование
- Разделение(информации о разметке от информации о внешнем виде)
- Переопределение стандартного вида

Пример

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="#00FF00"
    android:typeface="monospace"
    android:text="@string/hello" />
```

```
<TextView
    style="@style/CodeFont"
    android:text="@string/hello" />
```

Создание стиля

- Файл styles.xml в папке values/
- Можно объявлять все, что можно объявить в layout'е

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="CodeFont" parent="@android:style/TextAppearance.Medium">
        <item name="android:layout_width">fill_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">#00FF00</item>
        <item name="android:typeface">monospace</item>
    </style>
</resources>
```

Наследование стилей

- Можно наследовать уже существующие стили(смотри предыдущий слайд)
- Можно создавать свою иерархию стилей
- Тег `parent` используется только для наследования стилей `android`, для наследования своих стилей используется точка(`MyStyle.InheritanceStyle`)
- Переопределение свойств
- Добавление свойств

Использование стилей и тем платформы

- Большое количество predetermined стилей
- Можно кастомизировать вид стандартных компонентов(диалоги, виджеты и проч.)
- Можно найти по следующим ссылкам
- `styles.xml`
- `themes.xml`
- `R.attr`

Известные проблемы

- Есть закрытые стили – нельзя переопределить
- Очень много атрибутов и стилей – сложно ориентироваться
- Сложно отлаживать(нельзя продебажить, неясно, почему не работает)

Ссылки на атрибуты в xml

- Вы можете ссылаться на конкретные атрибуты конкретной темы(той, которая установлена activity, в которой используется этот xml или той, которая установлена всему приложению)
- Специальный синтаксис
- `?android:attr/listPreferredItemHeight` – означает «использовать значение, определенное атрибутом `listPreferredItemHeight` из определенной темы»



Preferences

SharedPreferences

- Key-value хранилище(файл)
- Хранение данных, которые должны сохраняться между запусками приложения(например настройки)
- Android предоставляет простое но мощное api
- `getActivity().getSharedPreferences(name, mode)`
- `PreferenceManager.getDefaultSharedPreferences(context)`

SharedPreferences API

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPref.edit();
editor.putInt(getString(R.string.saved_high_score), newHighScore);
editor.commit();
```

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
int defaultValue = getResources().getInteger(R.string.saved_high_score_default);
long highScore = sharedPref.getInt(getString(R.string.saved_high_score), defaultValue);
```

Settings

- Используется SharedPreferences API
- Создание экрана настроек из xml
- Богатая библиотека разных компонентов(CheckboxPreference, ListPreference, TimePickerPreference и т.д.)
- Использование PreferenceFragment
- Инициализация дефолтных значений настроек
- PreferenceChangeListener

preference.xml

```
<?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>
```

PreferenceFragment

```
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);  
    }  
    ...  
}
```