2016

Android 开发无障碍指南

中国信息无障碍产品联盟&信息无障碍研究会　译制

20160615

翻译声明

翻译机构：信息无障碍研究会（ARA） 中国信息无障碍产品联盟（CAPA）

译者：刘辉、李鸿利、张昆

审阅：杨骅、刘彪、蔡勇斌、朱广锐、沈广荣、王孟琦

本文档翻译自谷歌官方文档《Android guide: Accessibility》。如您对翻译文档内容有异议，请将原文文档做为主要参考，原文版权由 Google 持有并保留。

该翻译文档同时也有网页版及中英文混排的网页版，便于您的审阅。

目录

简介

很多 Android 用户因能力差异，需要使用不同的方式与 Android 设备进行交互。视觉障碍、肢体残疾和与年龄有关的障碍用户，不能充分的看到或使用触摸屏，还有听力障碍用户因听力损失而无法获取音频信息和警报。

Android 提供了无障碍特性和服务，帮助这些用户更轻松地浏览他们的设备，包括文本转语音、触觉反馈、手势导航、轨迹球和定向垫导航。 Android 应用程序开发者可以利用这些服务，使他们的应用程序更加无障碍。

Android 开发者还可以建立自己的无障碍服务，提供更强的可用性特性，如音频提示、物理反馈以及替代导航方式。无障碍服务可以为所有应用、一组应用或单一的 app 提供这些增强特性。

以下主题展示了如何使用 Android 框架，使应用程序更加无障碍：

让应用无障碍

开发实践和 API 特性可以确保应用程序对残障人群无障碍。

无障碍开发清单

一个帮助开发人员确保他们的应用无障碍的清单。

构建无障碍服务

如何使用 API 特性来构建无障碍服务，使其他应用程序对用户更加无障碍。

让应用无障碍

当视觉、肢残和老龄化用户激活设备中的无障碍服务和特性时，Android 应用将会更加无障碍。无障碍服务让应用更加无障碍，即使在代码中不做任何的无障碍修改。但是，需要采取以下步骤来评估应用的无障碍性能，确保所有用户都有愉快的使用体验。

保证所有用户的无障碍体验需要做到以下几步，特别是使用 Android 框架提供的组件创建用户界面时。如果应用只使用标准组件，步骤如下：

1. 使用 android:contentDescription 属性为应用中的交互控件添加描述性文本，特别是 ImageButton、 ImageView 和 CheckBox。

2. 保证所有可以接收输入（触摸和输入）的用户界面元素都可以使用定向控制到达，例如：轨迹球、D-pad（物理或虚拟）、导航手势。

3. 保证在音频反馈的同时伴有视觉反馈或通知，帮助听障用户使用应用。

4. 使用无障碍导航服务和特性测试应用。打开 TalkBack 和触摸浏览（Explore by Touch），然后尝试使用定向控制操作应用。更多无障碍测试信息，详见无障碍测试清单（Accessibility Testing Checklist）。

如果自定义控件继承了 View 类，需要另外的工作来保证组件无障碍。该章节讨论的是怎样让自定义控件拥有无障碍特性。

注意：该章节的实现步骤描述了为视障用户对应用进行无障碍改造的需求。在无障碍开发清单中需要复查视障用户的无障碍需求。

标记用户界面元素

很多用户界面控件依靠视觉提示告知用户含义和作用。例如：一个记事本程序可能会使用一个带有加号图片的 ImageButton 来提示用户可以通过它添加一

个新的记录。或者，一个 EditText 元素的附近可能会有一个标签来提示目的。视障用户不能清晰看到这些提示并遵循它们，导致这些提示无效。

可以使用 android:contentDescription XML 布局属性来让这些控件更加无障碍。该属性中的文本不会出现在屏幕上，但如果用户启用了可以提供音频反馈的无障碍服务，当用户导航到这些控件的时候，描述文本将会被读出。

因此，在应用的用户界面中需要为每个 ImageButton、ImageView、CheckBox 控件设置 android:contentDescription 属性，同时也需要为视障用户在那些可能需要额外信息的输入控件中添加描述文本。

例如，下面的 ImageButton 使用 add_note 字符串资源为加号按钮设置内容描述，该资源在英文界面可能被定义为“Add note”：

```
<ImageButton
    android:id=”@+id/add_note_button”
    android:src=”@drawable/add_note”
    android:contentDescription=”@string/add_note” />
```

通过给图像按钮添加描述文本，当用户将焦点移到该按钮，或者鼠标悬停时，提供语音反馈的无障碍服务可以读出“Add note”。

注意:对于 EditText 区域，提供 android:hint 属性***代替*** 内容描述，文本区域为空的时候此属性帮助用户理解应该输入什么样的内容。当文本区域填充上内容，TalkBack 将会读出输入的文本，而不会读出提示文本。

启用焦点导航

焦点导航允许残障用户使用定向控制器一步步的浏览用户界面控件。定向控制器可以是物理的，如轨迹球、定向垫（D-pad）、方向键，或虚拟的，如非视觉键盘（Eyes-Free Keyboard）、和 Android4.1 以及以上可用的手势导航。定向控制器是很多 Android 用户导航的首选方式。

为了保证用户可以使用定向控制器导航应用，需要验证应用中所有用户界面

输入控件，在不使用触屏的情况下可以到达和激活。应该验证点击定向控制器的中心按钮（或 OK 按钮）和触屏的触摸操作有同样的效果。更多关于测试定向控制的信息，详见测试焦点导航。

启用视图焦点

当用户界面元素的 android:focusable 属性设置为 true 时，可以使用定向控制到达该元素。该设置允许用户使用定向控制聚焦元素并与之交互。 Android 框架提供的用户界面控件默认可聚焦，并通过改变控件的外观可直观表明其聚焦。

Android 提供了几个 API 让开发者决定用户界面控件是否可聚焦，甚至请求给控件赋予焦点：

- `setFocusable()`
- `isFocusable()`
- `requestFocus()`

如果视图不是默认聚焦，可以在布局文件中设置 android:focusable 属性为 true，或者调用 setFocusable()方法让视图可聚焦。

控制焦点顺序

当用户在任何方向使用定向控制器导航时，焦点从一个用户界面元素（视图）传递到另一个由焦点顺序指定的用户界面元素（视图）。焦点顺序是以一种在某一特定方向上寻找相邻元素的算法为基础的。在极少数情况下，默认的算法可能不匹配开发者定义的顺序，或可能对于用户不符合逻辑。在这些情况下，可以在布局文件中使用下列的 xml 属性明确地覆盖焦点顺序：

android:nextFocusDown
当用户向下导航时，定义下一个接收焦点的视图；

android:nextFocusLeft
当用户向左导航时，定义下一个接收焦点的视图；

android:nextFocusRight

当用户向右导航时，定义下一个接收焦点的视图；

android:nextFocusUp

当用户向上导航时，定义下一个接收焦点的视图。

下面的 XML 布局示例中展示了两个可聚焦的用户界面元素，这两个元素的 android:nextFocusDown 和 android:nextFocusUp 属性被明确地设置。TextView 位于 EditText 的右边。但是，因为设置了这些属性，当 EditText 聚焦时，按下下光标可以将焦点移到 TextView 元素上。

```
<LinearLayout android:orientation="horizontal"
            ... >
      <EditText android:id="@+id/edit"
            android:nextFocusDown=" @+id/text"
            ... />
      <TextView android:id="@+id/text"
            android:focusable=" true"
            android:text="Hello, I am a focusable TextView"
            android:nextFocusUp=" @id/edit"
            ... />
</LinearLayout>
```

当修改焦点顺序时，确保每一个用户界面控件的所有方向的导航能按照预期工作，尤其是反向导航的时候（从来的路径返回）。

注意：开发者可以使用诸如 setNextFocusDownId() 和 setNextFocusRightId()的方法，在程序运行的时候修改用户界面组件的聚焦顺序。

创建可访问自定义视图

如果应用需要一个自定义视图组件，开发者必须做一些额外的工作来确保自

定义视图是可访问的。这些都是确保视图可访问性的主要任务：

- 处理定向控制器的点击；
- 实现无障碍 API 方法；
- 发送特定的 AccessibilityEvent 对象到自定义视图；
- 为视图填充 AccessibilityEvent 和 AccessibilityNodeInfo；

处理定向控制器的点击

在大多数的 Android 设备上，使用定向控制器单击视图会发送一个带有 KEYCODE_DPAD_CENTER 的 KeyEvent 事件到当前的焦点视图。所有的标准 Android 视图已经适当地处理 KEYCODE_DPAD_CENTER。当构建一个自定义 View 控件，确保这个事件的效果跟在触摸屏上触摸视图的效果一样。

自定义控件也应该将 KEYCODE_ENTER 事件作为 KEYCODE_DPAD_CENTER 事件处理，这种方法使全键盘用户的交互更加容易。

实现无障碍 API 方法

无障碍事件是用户在应用中与视觉界面元素的交互消息。这些消息是由无障碍服务（Accessibility Services）处理的，这些服务使用这些事件中的信息去产生补充反馈和提示。在 Android4.0（API 级别 14）和更高的系统中，生成无障碍事件的方法被扩展到比 Android1.6(API 级别 4)的 AccessibilityEventSource 提供更多的详细信息。扩展无障碍方法是 View 类和 View.AccessibilityDelegate 类的一部分。这些方法如下：

sendAccessibilityEvent()

(API 级别 4)当用户操作视图时，这个方法被调用。该事件使用用户操作类型进行分类，例如 TYPE_VIEW_CLICKED。开发者一般不需要实现这个方法，除非创建了自定义视图。

sendAccessibilityEventUnchecked()

(API 级别 4)当调用的代码需要直接检查设备是否激活无障碍特性(AccessibilityManager.isEnabled())的时候，调用该方法。如果实现该方法，必须执行无障碍激活的检查，而不管系统真正的设置。一般的，不需要为自定义视图实现该方法。

`dispatchPopulateAccessibilityEvent()`

(API 级别 4)当自定义视图产生无障碍事件时，系统调用这个方法。在 API 级别 14 中,该方法的默认实现是为视图调用 `onPopulateAccessibilityEvent()`，然后为该视图的子元素调用 `dispatchPopulateAccessibilityEvent()`。为了支持 Android4.0（API 级别 14）***之前*** 的版本，开发者***必须*** 重写该方法，使用自定义视图的描述性文本填充 `getText()`，描述性文本将会被无障碍服务读出，例如 TalkBack。

`onPopulateAccessibilityEvent()`

(API 级别 14)该方法为视图的 `AccessibilityEvent` 设置朗读文本提示。如果视图是另一个生成无障碍事件视图的子视图，调用该方法。

注意： 通过该方法修改文本以外的属性可能会被其他方法重写。虽然在该方法中开发者可以修改无障碍事件的属性,但是应该做到这些改变仅限于文本内容，并使用 `onInitializeAccessibilityEvent()`方法修改事件的其他属性。

注意： 如果该事件的实现完全重写了输出文本，而不允许布局的其他部分修改这个内容，不要在代码中调用该方法的父类实现。

`onInitializeAccessibilityEvent()`

(API 级别 14)系统调用此方法来获取超出文本内容的视图状态的附加信息。如果自定义视图提供简单 `TextView` 或 `Button` 以外的交互控制,开发者应该重写该方法,并且使用该方法设置该视图的额外信息到事件中,如密码区域类型,复选框类型或提供用户交互或反馈的状态。如果重写这个方法,开发者必须调用它父类的实现方法，然后只修改那些父类中尚未设置的属性。

`onInitializeAccessibilityNodeInfo()`

(API 级别 14)该方法为无障碍服务提供视图的状态信息。默认 `View` 的实现包含一组标准的视图属性，但是如果自定义视图提供了超出 `TextView` 或 `Button` 的交互，开发者应该重写该方法，并在 `AccessibilityNodeInfo` 对象中设置视图的额外信息，该对象被该方法处理。

`onRequestSendAccessibilityEvent()`

(API 级别 14)当视图中的一个子视图生成 `AccessibilityEvent` 时，系统调用这个方法。该步骤允许父视图使用额外信息修复无障碍事件。当自定义视图有子视图，且其父视图可以为无障碍事件提供对无障碍服务有用的上下文信息的时候，此时开发者应该实现这个方法。

为了在自定义视图中支持这些无障碍方法，应该采取下列方法中的一种：

- 如果应用程序目标版本是 Android4.0(API 级别 14)或更高，就直接在自定义视图类中重写并实现上面列出的无障碍方法。
- 如果自定义视图的目的是要兼容 Android1.6(API 级别 4)及以上，在项目中添加版本 5 或更高的支持库。然后，在自定义视图类中，调用 `ViewCompat.setAccessibilityDelegate()` 方法来实现上面的无障碍方法。对于这种方法的示例，请参阅支持库（版本 5 或更高）AccessibilityDelegateSupportActivity 例子在(<sdk>/extras/android/support/v4/samples/Support4Demos/)。

在任何一种情况下，需要为自定义视图类实现下面的无障碍方法：

- `dispatchPopulateAccessibilityEvent()`
- `onPopulateAccessibilityEvent()`
- `onInitializeAccessibilityEvent()`
- `onInitializeAccessibilityNodeInfo()`

更多信息实现这些方法，请参阅填充无障碍事件。

发送无障碍事件

根据自定义视图的特性，不是由默认实现处理的事件，可能需要在不同时刻发送 AccessibilityEvent 对象。View 类为这些事件类型提供了默认的实现方法：

- 自 API 级别 4 引入：
  - TYPE_VIEW_CLICKED
  - TYPE_VIEW_LONG_CLICKED
  - TYPE_VIEW_FOCUSED
- 自 API 级别 14 引入：
  - TYPE_VIEW_SCROLLED
  - TYPE_VIEW_HOVER_ENTER
  - TYPE_VIEW_HOVER_EXIT

注意：悬停（Hover）事件与触摸浏览相关联，触摸浏览使用这些事件作为触发器，为用户界面元素提供音频反馈。

一般来说，每当自定义视图的内容发生变化时，应该发送一个 AccessibilityEvent 事件。例如，如果创建了一个自定义的滑动条，用户可以按下左边或右边的箭头选择数值，当滑动条的值发生改变时，自定义视图应该发出一个 TYPE_VIEW_TEXT_CHANGED 类型的事件。下面的示例代码演示了使用 sendAccessibilityEvent() 方法报告该事件：

```
@Override
public boolean onKeyUp (int keyCode, KeyEvent event) {
      if (keyCode == KeyEvent.KEYCODE_DPAD_LEFT) {
            mCurrentValue--;
            sendAccessibilityEvent(AccessibilityEvent.TYPE_VIEW_TEXT_CHANGED);
            return true;
      }
```

```
        ...
}
```

填充无障碍事件

每个 AccessibilityEvent 有一组必需的属性，这些属性描述当前视图的状态。这些属性包括视图类名称、内容描述和检查状态等信息。每个事件类型必须的特定属性被描述在 AccessibilityEvent 参考文档中。View 实现提供这些属性的默认值。包含类名和事件时间标记在内的很多值会被自动提供。如果正在创建一个自定义视图组件，必须提供一些关于视图内容和特性的信息。这些信息可能是简单的按钮标签，但是也可能包含想要添加到事件中的额外的状态信息。

一个自定义视图为无障碍服务提供信息的最低要求是实现 dispatchPopulateAccessibilityEvent() 方法。系统调用这个方法，为 AccessibilityEvent 请求信息，让自定义视图兼容 Android1.6(API 级别 4)以上的无障碍服务。下面的示例代码展示了该方法的基本实现：

```
@Override
public void dispatchPopulateAccessibilityEvent(AccessibilityEvent
event) {
        super.dispatchPopulateAccessibilityEvent(event);
        //在 API 级别 14 及以上版本中，调用父类实现将文本填充到事件；
        //在低于 API 级别 14 的系统中，检查事件的文本内容，并为自定义视
图添加合适的文本描述；
        CharSequence text = getText();
        if (!TextUtils.isEmpty(text)) {
                event.getText().add(text);
        }
}
```

在 Android 4.0(API 级别 14)和更高的系统中，使用 onPopulateAccessibilityEvent()和 onInitializeAccessibilityEvent()方法来填充或修改 AccessibilityEvent 事件中的信息。

onPopulateAccessibilityEvent()方法可专门用来为事件添加或修改文本内容，这些信息会被如 TalkBack 的无障碍服务转化为音频反馈。使用 onInitializeAccessibilityEvent()方法添加事件的其他信息，比如视图的选择状态。

此外，还应该实现 onInitializeAccessibilityNodeInfo()方法。该方法填充 AccessibilityNodeInfo 对象，视图层次在接收此事件后生成无障碍事件，无障碍服务使用 AccessibilityNodeInfo 对象访问该视图层次，获得更多的上下文信息并为用户提供合适的反馈。

下面的示例代码显示了如何使用 ViewCompat.setAccessibilityDelegate()重写这三种方法。注意，此示例代码要求添加 API 级别 4（或更高）的 Android 支持库(Support Library)到项目。

```
ViewCompat.setAccessibilityDelegate(new
AccessibilityDelegateCompat() {
      @Override
       public    void    onPopulateAccessibilityEvent(View    host,
AccessibilityEvent event) {
            super.onPopulateAccessibilityEvent(host, event);
           //调用父类实现来填充事件的文本。然后添加父类中不存在的文
本。
           //常常只需要在自定义视图中添加这些文本。
            CharSequence text = getText();
            if (!TextUtils.isEmpty(text)) {
                  event.getText().add(text);
            }
      }
      @Override
      public    void    onInitializeAccessibilityEvent(View    host,
AccessibilityEvent event) {
            super.onInitializeAccessibilityEvent(host, event);
             //调用父类实现让父类设置事件属性。然后添加父类不支持的
新属性。
```

```
            event.setChecked(isChecked());
        }
        @Override
        public void onInitializeAccessibilityNodeInfo(View host,
                    AccessibilityNodeInfoCompat info) {
            super.onInitializeAccessibilityNodeInfo(host, info);
            //调用父类实现让父类设置适当的 info 属性。然后添加父类不
支持的属性（checkable and checked)。
            info.setCheckable(true);
            info.setChecked(isChecked());
            //经常只需要添加自定义视图的文本。
            CharSequence text = getText();
            if (!TextUtils.isEmpty(text)) {
                info.setText(text);
            }
        }
}
```

在 Android 4.0(API 级别 14)和更高的应用程序中，可以在自定义视图类中直接实现这些方法。这种方法的另一个例子，请参阅 Android 支持库(Support Library)(版本 5 或更高)的示例，AccessibilityDelegateSupportActivity 样本在 (<sdk>/extras/android/support/v4/samples/Support4Demos/)。

提供自定义无障碍内容

在 Android4.0（API 级别 14）中，Android 框架得到改善，可以允许无障碍服务访问可生成无障碍事件的用户界面组件包含的视图层次。此改善允许无障碍服务提供一组更丰富的可以帮助用户的上下文信息。

在某些情况下，无障碍服务不能从视图层次中获得足够的信息。一个自定义界面控件样例包含两个以上的可点击区域，例如一个日历控件。这种情况下，无障碍服务无法获得足够的信息，因为可点击部分不是视图层次的一部分。

图 1 可选择日期的自定义日历视图。

在图 1 所示的例子中，整个日历是作为独立 View 实现，因此如果不采取其他措施，无障碍服务就不能获得有关视图和视图中用户选择的足够信息。例如，如果用户点击了包含 17 的日期，无障碍框架只能接收到整个日历控件的描述信息。这种情况下，TalkBack 无障碍服务会简单读出“日历”，或者稍好一些的读出“四月日历”，而用户将会不知道自己选择了哪天。

在像这样的情况下，为了给无障碍服务提供足够的上下文信息，Android 框架提供了一种方法——指定虚拟视图层次。应用开发者可以使用***虚拟视图层次***方法为无障碍服务提供一个附加视图层次，该层次能最大限度的接近屏幕上的真实信息。这种方法允许无障碍服务给用户提供更有用的上下文信息。

另一个需要虚拟视图继承结构的情况是，用户界面包含一组功能密切相关的控件（视图），操作一个控件会影响一个或多个组内其他元素的内容，例如增加按钮和减少按钮分开的数值选择器。在这种情况下，无障碍服务无法获得足够的信息，因为一个控件的操作会改变另一个控件的内容，且无障碍服务无法获得这种关系。为了处理这种情况，将包含视图的相关控件进行分组，从该容器中提供虚拟视图继承，清楚呈现控件提供的信息和行为。

为了提供虚拟视图继承，在自定义视图或者视图组中重写 getAccessibilityNodeProvider()方法，并返回 AccessibilityNodeProvider 的实现。该无障碍特性实现的一个例子，详见 ApiDemos 样例工程中的 AccessibilityNodeProviderActivity。在 Android1.6 和以后的版本中，可以使

用 Support Library 中的 ViewCompat.getAccessibilityNodeProvider()方法，并提供 AccessibilityNodeProviderCompat 实现，就可以兼容虚拟视图继承。

处理自定义触摸事件

自定义视图控件可能需要非标准的触摸事件行为。例如自定义控件可能使用 onTouchEvent(MotionEvent)监听方法来检测 ACTION_DOWN 和 ACTION_UP 事件，并触发一个特殊的点击事件。为了保证无障碍服务的兼容性，处理自定义点击事件的代码必须如下：

1. 为解释点击操作生成适当的 AccessibilityEvent；

2. 启用无障碍服务来为不能使用触摸屏的用户执行自定义单击操作。

要有效的处理这些需求，代码应该重写 performClick()方法，需要调用该方法的父类实现，然后执行点击事件需要的任何操作。当检测到自定义点击操作时，代码应该调用 performClick()。下列代码示例展示了这一模式：

```
class CustomTouchView extends View {

      public CustomTouchView(Context context) {
            super(context);
      }

      boolean mDownTouch = false;

      @Override
      public boolean onTouchEvent(MotionEvent event) {
            super.onTouchEvent(event);

            //监听向下向上触摸事件
            switch (event.getAction()) {
                  case MotionEvent.ACTION_DOWN:
```

```
                        mDownTouch = true;
                        return true;

                case MotionEvent.ACTION_UP:
                        if (mDownTouch) {
                                mDownTouch = false;
                                performClick(); //调用该方法处理响应，
启动无障碍服务为那些无法点击的用户执行操作。
                                return true;
                        }
                }
                return false; //为其他触摸事件返回假。
        }

        @Override
        public boolean performClick() {
                //调用父类实现，父类会生成无障碍事件，并且在视图上调用
onClick()监听器，如果任何；
                super.performClick();

                //在这里处理自定义点击操作

                return true;
        }
}
```

上面的代码通过使用 `performClick()`方法保证自定义点击事件与无障碍服务兼容，生成无障碍事件并且为无障碍服务提供入口，并代替用户执行自定义点击事件。

注意：如果自定义视图有不同的可点击区域，例如自定义日历视图，必须在自定义视图中重写 `getAccessibilityNodeProvider()`实现虚拟视图继承(virtual view hierarchy)，保证与无障碍服务兼容。

测试无障碍

测试应用的无障碍是保证良好用户体验的重要部分。开发者可以通过在应用程序启用语音反馈和仅使用定向控制在应用程序内导航来测试最重要的无障碍特性。更多无障碍测试信息，详见无障碍测试清单(Accessibility Testing Checklist)。

无障碍开发清单

对应用进行无障碍优化是对应用的易用性、更精确的获取细节、取悦用户等方面更深层次的延伸。该文档提供了无障碍需求、建议和注意事项的清单，可以帮助开发者确保应用程序无障碍。遵循该清单，不能保证应用无障碍，但却是个通往无障碍的开端。

创建无障碍应用并不只是开发者的责任，需要将设计和测试小组加入进来，让他们了解应用开发其他阶段的无障碍指南：

- android 设计：无障碍
- 无障碍测试清单

在大多数案例中，创建 android 无障碍应用并不需要严格的代码重构。相反，无障碍优化作用于用户如何与应用程序交互的微妙细节，开发者可以提供给用户可以感知和理解的反馈。该清单帮助开发者专注于主要的开发问题，保证应用的无障碍细节正确。

无障碍需求

必须完成以下步骤来保证应用最低级别的无障碍特性。

1. **描述用户界面控件：**为没有视觉文本的用户界面组件提供内容描述，特别是图像按钮(ImageButton)，图像视图(ImageView)和复选框(CheckBox)组件。使用 XML 布局属性 android:contentDescription 或者使用 setContentDescription(CharSequence)方法来为无障碍服务提供内容描述。（例外：装饰性图像）

2. **保证基于焦点的导航：**保证用户在使用基于硬件或软件定向控制(D-pads，轨迹球，键盘和导航手势）时，可以导航屏幕布局。在一些情况下，需要保证用户界面可聚焦 ，或者改变焦点顺序使用户交互更具逻辑性。

3. **自定义视图控件：**如果为应用程序创建了自定义界面控件，为自定义视图实现无障碍接口并提供内容描述。对于需要与 android 最低版本 1.6 保持一致性的自定义控件，使用支持库（Support Library）实现最新的无障碍特性。

4. **纯音频反馈替代：**音频反馈必须有第二种反馈机制来支持听障用户的使用。例如，消息铃声作为系统通知必须伴随触觉反馈（如果可能的话）或其他视觉警告。

5. **测试：**使用定向控制导航 android 应用，和使用 TalkBack 提供的非视觉导航来测试无障碍性能。更多关于无障碍测试的信息，详见无障碍测试清单。

无障碍建议

以下步骤被推荐用来确保应用程序无障碍。如果不执行这些操作，这可能会影响应用程序整体的无障碍和质量。

1. **Android 设计无障碍指南：**在创建布局前，复查和遵守设计指南中提供的无障碍方案。

2. **框架提供的控件：**尽可能使用 Android 标准用户界面控件，这些控件默认提供无障碍支持。

3. **临时或自我隐藏控件和通知：**避免使用一段时间之后，变暗或消失的用户界面控件。如果该功能对应用很重要，为该功能提供一个替代的交互方式。

特殊情况和注意事项

以下列表描述了需要采取措施保证应用无障碍的特殊情况。检查此列表，查看这些特殊情况和注意事项是否适用于正在开发的应用程序，如果适用，请执行

相应的操作。

1. **文本区域提示：**对于 EditText 区域，使用 android:hint 属性*代替*内容描述，当文本区域为空的时候帮助用户了解期望输入的内容，当 EditText 被填充时，允许填充内容被朗读出。

2. **高视觉特性的自定义控件：**如果应用程序包含一个具有高视觉特性的自定义控件（例如日历控件），默认无障碍服务处理不会为用户提供充足的内容描述，开发者可以考虑使用 AccessibilityNodeProvider 为控件提供虚拟视图层次。

3. **自定义控件和点击处理：**如果应用中的自定义控件操作手势特殊，例如使用 onTouchEvent(MotionEvent) 监听 MotionEvent.ACTION_DOWN 和 MotionEvent.ACTION_UP ，并作为点击事件对待，开发者必须触发一个等效于点击的 AccessibilityEvent，并为用户无障碍服务提供一种方式模拟执行这项操作。更多信息，详见处理自定义触摸事件。

4. **有功能改变的控件：**如果用户在正常使用应用的过程中，应用中的按钮或其他控件的功能会发生改变（例如，一个按钮从**播放**变为**暂停**），保证按钮的 android:contentDescription 会相应改变。

5. **相关联控件提示：**提供独立功能的一组控件，例如日期选择器（DatePicker），当用户与相关联控件中的个别控件交互的时候，提供有效的音频反馈。

6. **视频播放和字幕：**如果应用程序提供视频播放，必须为聋人和重听用户提供字幕和注释来帮助他们理解。视频播放控件必须明确说明字幕是否可用，并提供简单的方式启用字幕。

7. **补充无障碍音频反馈：**仅使用 Android 无障碍框架为应用程序提供无障碍音频反馈。像 TalkBack 类的无障碍服务应该是为应用提供无障碍音频反馈的唯一方式。使用 android:contentDescription XML 布局属性或使用无障碍框架 APIs 动态地添加该属性来为应用提供提示信息。例如，如果开发者想要把应用程序执行的操作告知用户，如书籍自动翻页功能，

使用 announceForAccessibility(CharSequence)方法让无障碍服务为用户读出该信息。

8. **复杂视觉交互的自定义控件：**对于提供复杂或非标准视觉交互的自定义控件，使用 AccessibilityNodeProvider 为自定义控件提供虚拟视图层次，AccessibilityNodeProvider 允许无障碍服务为用户提供简化交互模型。如果该方式不可行，考虑提供一个无障碍的替代视图。

9. **小控件组：**如果控件比推荐的触摸尺寸小，可以考虑使用 ViewGroup 将这些控件组合起来，并使用 android:contentDescription 为该组合提供内容描述。

10. **装饰性图像和图形：**应用屏幕中的不提供任何内容或用户无法操作的纯装饰性元素，不应该为这些元素提供无障碍内容描述。

构建无障碍服务

无障碍服务是一个为残疾人或可能暂时无法与设备完全互动的人提供用户界面扩展功能的应用程序。例如，在开车、照顾孩子或参加一个非常喧闹的宴会时，用户可能需要另外的交互反馈作为补充或替代。

Android 提供了标准的无障碍服务，包括 TalkBack，开发人员可以创建和发布自己的服务。 本文档介绍了构建无障碍服务的基本知识。

构建和部署无障碍服务的能力在 Android 1.6（API 等级 4）被推出，并在 Android 4.0（API 等级 14）明显改善其功能。Android 支持库也被更新，Android 4.0 发布的支持库可以使得这些无障碍特性也能支持 android 1.6。鼓励那些致力于实现广泛可兼容的无障碍服务开发者们使用支持库，并且在 Android4.0 版本中开发更多先进的无障碍特性。

manifest 声明和权限

提供无障碍服务的应用，必须在应用的声明（manifest）文件中明确声明，以便 android 系统把此应用程序作为无障碍服务处理。本节介绍无障碍服务的必需和可选设置。

无障碍服务声明

为了能被看作一个无障碍服务，必须在声明文件中的应用程序(application)元素中包含服务（service）元素（而不是活动（activity）元素）。此外，在服务（service）元素内，还必须包括无障碍服务的目标过滤器。为了兼容 Android 4.1 及更高版本的系统，声明文件中必须请求 `BIND_ACCESSIBILITY_SERVICE` 的权限，如下例所示：

```
<manifest>
   ...
```

```
    <uses-permission ... />
    ...
    <application>
        ...
        <service android:name=".MyAccessibilityService"
                android:label="@string/accessibility_service_label"
                android:permission="android.permission.BIND_ACCESSIBILI
TY_SERVICE">
            <intent-filter>
                <action
android:name="android.accessibilityservice.AccessibilityService" />
            </intent-filter>
        </service>
        <uses-permission
android:name="android.permission.BIND_ACCESSIBILITY_SERVICE" />
    </application>
</manifest>
```

在 Android 1.6（API 等级 4）和更高版本的系统中，所有无障碍服务都需要部署这些声明。

无障碍服务配置

无障碍服务还必须提供一个配置，该配置指定了无障碍服务处理的无障碍事件类型和有关无障碍服务的附加信息。无障碍服务的配置被包含在 `AccessibilityServiceInfo` 类中。无障碍服务可以使用该类的实例创建一个配置，并在运行时使用 `setServiceInfo()` 设置此配置。但是，使用该方法并不能配置所有的配置项。

从 Android 4.0 开始，可以在声明文件中包含一个<meta-data>元素，该元素指向一个配置文件。该配置文件允许开发者设置无障碍服务的所有内容，具体如下例所示：

```
<service android:name=".MyAccessibilityService">
   ...
   <meta-data
      android:name="android.accessibilityservice"
      android:resource="@xml/accessibility_service_config" />
</service>
```

该 meta-data 元素指向的是开发者在应用资源目录下创建的 XML 文件(<project_dir>/res/xml/accessibility_service_config.xml)。下面的代码展示了无障碍服务配置文件的样例：

```
<accessibility-service
xmlns:android="http://schemas.android.com/apk/res/android"
      android:description="@string/accessibility_service_description"
      android:packageNames="com.example.android.apis"
      android:accessibilityEventTypes="typeAllMask"
      android:accessibilityFlags="flagDefault"
      android:accessibilityFeedbackType="feedbackSpoken"
      android:notificationTimeout="100"
      android:canRetrieveWindowContent="true"
      android:settingsActivity="com.example.android.accessibility.ServiceSettingsActivity"/>
```

无障碍服务配置文件使用的 XML 属性，点击以下链接查看更多信息：

- android:description
- android:packageNames
- android:accessibilityEventTypes
- android:accessibilityFlags

- android:accessibilityFeedbackType
- android:notificationTimeout
- android:canRetrieveWindowContent
- android:settingsActivity

哪些配置设置可以在运行时动态地设置的详细信息，请参阅 AccessibilityServiceInfo 参考文档。

注册无障碍事件

无障碍服务配置参数最重要的功能之一是允许开发者指定无障碍服务能处理的无障碍事件类型。成功指定该信息使无障碍服务间能相互合作，并允许开发者灵活地处理特定应用的特定事件类型。事件过滤可以包含以下规则：

- **包名(PackageNames)：** - 指定想要无障碍服务处理的无障碍事件的应用程序包名。如果省略该参数，无障碍服务将被视为服务于任何应用程序的无障碍事件。在无障碍服务配置文件中，该参数可使用 android:packageNames 属性被设置为一个逗号分隔的列表，或使用 AccessibilityServiceInfo.packageNames 成员设置。

- **事件类型(Event Types)** - 指定开发者想要无障碍服务处理的无障碍事件的类型。在无障碍服务配置文件中，该参数可使用 android:accessibilityEventTypes 属性设置为由“ | ”分隔的列表（例如 accessibilityEventTypes="typeViewClicked|typeViewFocused"），或者使用 AccessibilityServiceInfo.eventTypes 成员设置。

当创建无障碍服务时，开发者需要全面考虑无障碍服务能处理的无障碍事件，并只注册这些事件。因为用户可能同时激活多个无障碍服务，开发者创建的服务不能浪费在该服务不能处理的事件上。记住，其他的服务也可能会为了提供用户

体验来处理这些事件。

注：如果服务提供不同反馈类型(feedback types)，Android 框架可以为无障碍事件分派一个以上的无障碍服务。但是，如果为两个或更多的服务提供相同的反馈类型，只有第一个注册的服务才能接收事件。

无障碍服务方法

一个无障碍服务必须继承 AccessibilityService 类，重写该类的以下方法。这些方法被 Android 系统调用时按顺序呈现，从服务被启动（onServiceConnected()），一直运行（onAccessibilityEvent()、onInterrupt()），到服务被关闭（onUnbind()）。

- onServiceConnected()

-（可选）当系统成功连接到无障碍服务时，调用该方法。使用该方法为无障碍服务提供任何一次性的启动步骤，包含连接到用户反馈系统服务，例如音频管理或设备震动。如果想要在应用运行时设置无障碍服务的配置或做一次性的调整，可以在该方法中调用 setServiceInfo()来实现。

- onAccessibilityEvent()

-（必选）当系统检测到一个符合无障碍服务设定的无障碍事件过滤参数的事件时，该方法被系统回调。例如，当用户在应用程序中点击一个按钮或聚焦一个用户界面控件时，你的无障碍服务会为此应用程序提供反馈。此时，系统调用这个方法，传递相关联的无障碍事件，无障碍服务就可以翻译和使用该事件为用户提供反馈。在无障碍服务的生命周期中，该方法会被调用多次。

- onInterrupt()

-（必选）当系统想要打断无障碍服务提供的反馈时该方法被调用，一般情况下是响应用户操作，如移动焦点到一个不同的控件。在无障碍服务的生命周期中，该方法会被调用很多次。

- onUnbind()

-（可选）当系统想要关闭无障碍服务的时候，调用该方法。使用该方法去完成一次性的关闭进程，包括解除分配用户反馈系统服务，例如音频管理或者设备震动。

这些回调方法为无障碍服务提供基本架构。在 AccessibilityEvent 对象中怎样处理由 Android 系统提供的数据并为用户提供反馈，是开发者自己决定的。获取无障碍事件信息的更多内容，详见实现无障碍（Implementing Accessibility）培训。

获得事件详情

Android 系统通过 AccessibilityEvent 对象为无障碍服务提供用户界面交互的信息。在 Android4.0 之前，可从无障碍事件中获得信息，且无障碍事件能够提供用户所选择的界面控件的大量详细信息，但这些信息仅包含有限的上下文信息。在很多情况下，丢失的上下文信息可能是理解已选定控件的意义的关键。

一个界面的上下文是很关键，例如，日历或每日计划。如果用户选择周一到周五的下午四点，无障碍服务告知下午四点，但是没有说明是哪个工作日，哪个月的哪一天，反馈结果就会令人非常困扰。在这种情况下，用户界面控件的上下文对那些做时间规划的人非常关键。

Android4.0 较大的扩展了无障碍服务能获取到的关于用户界面交互的信息量，这些信息是通过基于视图层次组合的无障碍事件提供。视图层次是用户界面元素的组合，该组合包含组件（父亲）和可能包含在其中的子元素（孩子）。因此，Android 系统可以提供无障碍事件的更丰富的详细信息，允许无障碍服务为用户提供更多有用的反馈。

系统将 AccessibilityEvent 传递到无障碍服务的 onAccessibilityEvent() 回调方法，因此，无障碍服务获得用户交互事件的信息。AccessibilityEvent 会提供事件的详细信息，包含 AccessibilityEvent 的执行类型、描述文本和其他信息。从 Android4.0 开始（Android4.0 以前的支持情况详见支持库（Support

Library）中的 AccessibilityEventCompat），可以通过调用以下方法获得更多信息。

- AccessibilityEvent.getRecordCount() 和 getRecord(int)

-这些方法允许开发者获得一组 AccessibilityRecord 对象，该对象作为 AccessibilityEvent 的一部分，通过系统传递给开发者。该详情的级别为无障碍服务触发的事件提供了更多上下文信息。

- AccessibilityEvent.getSource()

- 该方法返回 AccessibilityNodeInfo 对象。该对象允许开发者请求查看触发无障碍事件的组件视图布局层次（父亲和孩子）。该特性允许无障碍服务访问事件的所有上下文内容，包含任何闭合视图或子视图的内容和状态。

重要:从一个 AccessibilityEvent 访问视图层次的能力会将用户私人信息潜在的暴露给无障碍服务。基于该原因，无障碍服务必须在无障碍服务配置 XML 文件请求访问权限，该文件中需要包含 canRetrieveWindowContent 属性并将其设置为 true。如果 XML 文件不包含这些设置，调用 getSource()将会失败。

注意：在 Android4.1（API 级别 16）和以后的版本中，getSource()、AccessibilityNodeInfo.getChild()和 getParent()方法，只返回一个对无障碍来说相当重要的视图对象（这些视图画有内容，或者响应用户操作）。如果无障碍服务针对所有视图，可以通过设置无障碍服务的 AccessibilityServiceInfoflags 成员来请求，详见 flag 中的非重要视图（FLAG_INCLUDE_NOT_IMPORTANT_VIEWS）。

为用户采取措施

从 Android4.0（API 级别 14）开始，无障碍服务可以代表用户操作，包含改变输入焦点和选择（激活）用户界面元素。在 Android4.1（API 级别 16），操作的范围被扩展至包含滚动列表和与文本域交互。无障碍服务也可采取全局操作，如导航到主界面、按返回按钮、打开屏幕通知和最近应用列表。Android4.1 也包

含新焦点类型，***无障碍焦点***，该焦点类型可让所有视觉元素能够被无障碍服务所选择。

这些新的能力让开发者能够开发替代导航模式，如手势导航，提高残障用户对 Android 设备的控制。

手势监听

无障碍服务可以监听特定手势，并代表用户响应。该特性被添加在 Android4.1（API 级别 16）版本中，要求无障碍服务激活触摸浏览特性。无障碍服务可以通过设置 `AccessibilityServiceInfo` 实例的 `flags` 成员为 `FLAG_REQUEST_TOUCH_EXPLORATION_MODE`，请求该特性的激活，样例如下所示：

```
public class MyAccessibilityService extends AccessibilityService {
      @Override
      public void onCreate() {
                                        getServiceInfo().flags        =
AccessibilityServiceInfo.FLAG_REQUEST_TOUCH_EXPLORATION_MODE;
      }
      ...
}
```

一旦无障碍服务请求激活触摸浏览，如果该特性未打开，用户必须允许打开该特性。当该特性激活，无障碍服务将会通过 `onGesture()`回调方法接收无障碍手势的通知，并且为用户响应操作。

使用无障碍操作

无障碍服务可以代替用户操作，让用户与应用的交互更简单有效。无障碍服务的这个能力在 Android4.0（API 级别 14）被加入，在 Android4.1（API 级别 16）被显著扩展。

为了代替用户操作，无障碍服务必须注册来接收来自某些或者更多应用中的

事件，并在无障碍服务配置文件中设置 android:canRetrieveWindowContent 属性为 true，来请求查看应用程序的内容。当接收到事件，无障碍服务可以使用 getSource() 方法在事件中检索 AccessibilityNodeInfo 对象。使用 AccessibilityNodeInfo 对象你的无障碍服务可以浏览视图层次来判定采取什么操作并使用 performAction()为用户操作。

```
public class MyAccessibilityService extends AccessibilityService {

      @Override
      public void onAccessibilityEvent(AccessibilityEvent event) {
            // 获得事件的资源节点
             AccessibilityNodeInfo nodeInfo = event.getSource();

             //使用事件和节点信息判定为用户执行什么样的操作。
              nodeInfo.performAction(AccessibilityNodeInfo.ACTION_SCR
OLL_FORWARD);

             //循环 nodeInfo 对象
             nodeInfo.recycle();
      }
      ...
}
```

performAction()方法允许无障碍服务在应用中执行操作。如果无障碍服务需要去执行全局操作，例如导航到主屏幕、按返回键、打开屏幕通知或者最近应用列表等，使用 performGlobalAction()方法。

使用焦点模式

Android4.1（API 级别 16）引入了一种新的用户界面焦点，被称为***无障碍焦点（Accessibility Focus）***。这种类型的焦点可用于在无障碍服务中，选择任何视觉用户界面元素并操作该元素。该焦点类型不同于***输入焦点（Input Focus）***。当用户输入字符，点击键盘上的 enter 键或 D-pad 控制器的中心按钮时，输入焦

点决定屏幕上哪个元素接收输入。

无障碍焦点与输入焦点是完全分开和独立的。事实上，用户界面中的一个元素有输入焦点，同时另一个元素有无障碍焦点，这种情况是可能出现的。无障碍焦点的目的是，为无障碍服务提供一种与屏幕上视觉元素进行交互的方法，而不管该元素是否是系统层面的可输入聚焦元素。开发者可以在无障碍手势测试中看到无障碍焦点。更多关于该特性的信息，详见测试手势导航。

注：当元素具有输入焦点，使用无障碍焦点的无障碍服务有责任整合当前输入焦点。如果服务没有整合输入焦点和无障碍焦点，有可能会在应用中引起问题，问题是当采用特定操作时，需要将输入焦点固定在特定的位置。

无障碍服务可以使用 `AccessibilityNodeInfo.findFocus()` 方法来判定哪些用户界面元素有输入焦点或无障碍焦点，也可以使用 `focusSearch()` 方法来检索哪些元素可以用输入焦点选择。最后，可以使用 `performAction(AccessibilityNodeInfo.ACTION_SET_ACCESSIBILITY_FOCUS)` 方法来设置无障碍焦点。

样例代码

API Demo 项目包含两个样例，可以用来作为生成无障碍服务的起点(<sdk>/samples/<platform>/ApiDemos/src/com/example/android/apis/accessibility):

- ClockBackService

-该服务是以 `AccessibilityService` 的原始实现为基础，可以用来作为开发基层无障碍服务的参考，适用于 Android1.6（API 级别 4）和以后的版本。

- TaskBackService

-该服务基于升级版无障碍 APIs，该 APIs 引进于 Android4.0(API 级别 14)。但是，可以使用 Android 支持库（Support Libary）中的同等支持类包（例如，

AccessibilityRecordCompat, AccessibilityNodeInfoCompat）代替后面 API 级别中引进的类（例如，AccessibilityNodeInfo, AccessibilityNodeInfo），让这些样例可以作用于 Android1.6（API 级别 4）。