2016

Android——实现无障碍

中国信息无障碍产品联盟&信息无障碍研究会　译制

刘辉　李鸿利　20160715

原文地址

目录

1.简介

当说到想要达到尽可能广泛的用户基数，重视 Android 应用的无障碍性能很重要。用户界面上的提示可能对大多数用户有用，例如当一个按钮被按下后有视觉变化，如果用户有视力障碍，这就不是最优的方案。

该课程展示了怎样充分利用内置 Android 框架的无障碍功能。该课程包含了怎样利用焦点导航和内容描述等平台特性，从无障碍方面对应用进行优化；也包含怎样构建无障碍服务，方便用户与任何 Android 应用进行交互，不只是自己开发的应用。

1.1 课程[1]

开发无障碍应用

学习让 Android 应用无障碍。允许键盘或定向垫的简单导航，设置标签并发送可以被无障碍服务解释的事件，以获得流畅的用户体验。

开发无障碍服务

开发一个监听无障碍事件的无障碍服务，在这些事件中获得事件类型和内容描述等的信息，并使用这些信息与用户交流。样例将会使用文本转语音引擎讲述给用户。

无障碍测试清单

学习怎样测试应用的无障碍特性。

[1] 本课程的相关性和先决条件为：Android 2.0（API 级别 5）或更高版本；

2.开发无障碍应用

Android 有几个集成到平台的焦点无障碍特性，这让为视觉和肢体障碍的用户进行应用优化更为简单。但是，什么是正确的优化，或利用平台达到目的的最简单方式，并不总是显而易见。该课程展示了怎样实现这些方法和平台特性，并使用这些方法和特性可以开发出支持无障碍的 Android 应用。

2.1 添加内容描述

一个精心设计的用户界面（UI）经常有不需要精确标签向用户标识目的的元素。在任务列表应用中，一个复选框后跟随一个条目，有非常明显的目的，在文件管理应用中却是个垃圾桶。但是，对于视觉障碍的用户，却需要其他 UI 提示。

幸运的是，在应用中给 UI 元素添加标签非常容易，这些标签可以被像 TalkBack 一样基于语音的无障碍服务大声朗读给用户。如果有标签在应用的生命周期内不会改变（例如“暂停”或“购买”），可以通过在 XML 布局文件中设置 UI 元素的 android:contentDescription 属性添加，如本样例中所示：

```
<Button
    android:id=”@+id/pause_button”
    android:src=”@drawable/pause”
    android:contentDescription=”@string/暂停”/>
```

但是，在很多情景下，需要根据上下文确定内容描述，例如开关按钮的状态、或列表条目等的可选择数据。使用 setContentDescription()在运行时编辑内容描述，如下：

```
String contentDescription = "Select " + strValues[position];
label.setContentDescription(contentDescription);
```

将这些加入代码中，是提升应用无障碍特性的最简单的方式，但只是最有效

的方法之一。尝试在所有有有用信息的地方添加内容描述，但是应该避免标记无用信息的误区。例如，不要给应用图标设置内容描述为“应用图标”。这只会增加用户在导航时从界面中抽取有用信息的噪音。

试试吧！下载 TalkBack（Google 发布的一个无障碍服务），并在“设置>无障碍>TalkBack”中打开，然后导航自己的应用，并听取 TalkBack 提供的音频提示。

2.2 焦点导航设计

应用应该支持多种导航方式，不只是触摸屏幕。很多 Android 设备支持硬件导航，例如 D-pad、方向键、或轨迹球，不只是触摸屏。另外，后期发布的 Android 也支持连接外部设备，例如通过 USB 或蓝牙连接的键盘。

为了支持这种导航模式，用户应该能够导航到的所有可导航的元素，都需要被设置为可聚焦。该修改可以在应用运行时在对应 UI 控件上使用 View.setFocusable() 方法实现，或者通过在 XML 布局文件中设置 android:focusable 属性实现。

同时，每个 UI 控件有 4 个属性，android:nextFocusUp、android:nextFocusDown、android:nextFocusLeft 和 android:nextFocusRight，当用户在特定方向导航时，可以用来指定下一个接收焦点的视图。虽然平台会自动根据布局临近自动决定导航顺序，但当该导航顺序不适合应用时，开发者可以使用这些属性重写导航顺序，如下例，呈现了一个按钮和标签，且都可聚焦，例如向下滑，焦点从按钮转移到文本视图，向上滑，焦点回到按钮。

```
<Button android:id="@+id/doSomething"
    android:focusable="true"
    android:nextFocusDown=”@id/label”
    ... />
<TextView android:id="@+id/label"
    android:focusable=”true”
    android:text="@string/标签文本"
```

```
android:nextFocusUp=”@id/doSomething”
... />
```

验证应用在这些情景下的工作符合直觉。最简单的方式是在 Android 模拟器中简单地运行应用，并使用模拟方向键导航 UI，使用 OK 按钮代替触摸选择 UI 控件。

2.3 发送无障碍事件

如果在 Android 框架中使用视图组件，无论何时选择一个条目或改变 UI 中的焦点时，都会创建一个 AccessibilityEvent。这些事件被无障碍服务检测，让它能提供文本转语音类的服务给用户。

如果写了一个自定义视图，保证在适当的时候发送事件。通过调用具有一个呈现已发生事件类型参数的 sendAccessibilityEvent(int)生成事件。当前支持的事件类型完全列表可在 AccessibilityEvent 参考文档中找到。

作为一个样例，如果想要将一个图片扩展为聚焦时可使用键盘输入标题，需要发送一个 TYPE_VIEW_TEXT_CHANGED 事件，尽管是不正常地输入到 image 视图。生成事件的代码如下所示：

```
public void onTextChanged(String before, String after) {
    ...
    if(AccessibilityManager.getInstance(mContext).isEnabled()) {
    sendAccessibilityEvent(AccessibilityEvent.TYPE_VIEW_TEXT_CHANGED
);
        }
        ...
    }
```

2.4 测试应用

当在应用中加入无障碍功能时，必须去测试它。为了测试内容描述和无障碍事件，安装和启用无障碍服务。一个选择是 TalkBack，一个免费的、开源的、在 Google Play 上可获得的屏幕阅读器。启用服务，测试应用中的所有导航流程并听取语音反馈。

同时，尝试使用定向控制器而不是触摸屏幕，导航应用。如果可获得，可以使用 D-pad 或轨迹球等物理设备。如果没有，可使用 Android 模拟器模拟键盘控制。

在服务提供反馈和定向导航应用时，应该有一种感觉，就像是在没有任何视觉提示的情况下导航应用。当问题出现时及时修复，测试结束后将会得到一个更加无障碍的 Android 应用。

3.开发无障碍服务

无障碍服务是 Android 框架的一个特性，被设计用来代替安装在 Android 设备上的应用，为用户提供替代导航反馈。无障碍服务可以代表应用与用户进行交流，例如，当用户访问屏幕上的重要位置时，将文本转化为语音或提供触觉反馈。本课程覆盖了如何创建无障碍服务、处理从应用接收到的信息，并把信息报告给用户。

3.1 创建无障碍服务

一个无障碍服务可以被捆绑在一个普通应用中，或被作为一个独立 Android 工程进行创建。在任何情况下，创建该服务的步骤是相同的。在工程内，创建一个继承自 AccessibilityService 的类。

```
package com.example.android.apis.accessibility;
import android.accessibilityservice.AccessibilityService;
public class MyAccessibilityService extends AccessibilityService {
...
    @Override
    public void onAccessibilityEvent(AccessibilityEvent event) {
    }

    @Override
    public void onInterrupt() {
    }

...
}
```

像其他服务一样，也应该在清单文件（manifest file）中声明该服务。记得指明想要该服务处理的 android.accessibilityservice intent，这样当应用发送

AccessibilityEvent 时该服务可被调用。

```
<application ...>
  ...
  <service android:name=".MyAccessibilityService">
     <intent-filter>
        <action android:name="android.accessibilityservice.AccessibilityService" />
     </intent-filter>
            . . .
  </service>
  ...
</application>
```

如果为该服务创建了一个新工程，且不打算创建一个应用，可以从源代码中移除启动 Activity 类（通常被称为 MainActivity.java）。同时也要记得从清单文件中移除相应的 activity 元素。

3.2 配置无障碍服务

为无障碍服务设置配置变量，告知系统想要运行服务的方式和时间。哪些事件类型是想要处理的？服务应该被所有应用激活还是只被特定的包名激活？服务会使用哪些不同的反馈类型？

有两种选择来设置这些变量。向后兼容的选择是在代码中使用 setServiceInfo(android.accessibilityservice.AccessibilityServiceInfo)设置它们。要做到该选择，需要重写 onServiceConnected()方法并在该方法中配置无障碍服务。

```
@Override
public void onServiceConnected() {
    //设置此服务想要监听的事件类型。其他类型将不会传递到此服务。
     info.eventTypes = AccessibilityEvent.TYPE_VIEW_CLICKED |
                       AccessibilityEvent.TYPE_VIEW_FOCUSED;
    //如果只想该服务运行在指定的应用中，在这里设置包名。
```

```
    //否则当服务被激活，它将监听所有应用的事件。
    info.packageNames = new String[]{"com.example.android.myFirstApp",
"com.example.android.mySecondApp"};
    //设置服务将提供的反馈类型。
    info.feedbackType = AccessibilityServiceInfo.FEEDBACK_SPOKEN;
    //当没有为已生成 AccessibilityEvent 类型提供指定包名时，默认服务
    //才会被调用。该服务*是*指定应用的，所以该标志（flag）是不是必须的。
    //如果这是一个通用的服务，设置默认标志（flag）是值得考虑的。

    // info.flags = AccessibilityServiceInfo.DEFAULT;

    info.notificationTimeout = 100;

    this.setServiceInfo(info);
    }
```

第二种选择是使用 XML 文件配置服务。某些配置选项，如 canRetrieveWindowContent，只有在使用 XML 配置服务时才可用。与上述相同的配置选项，使用 XML 定义，如下：

```
<accessibility-service
        android:accessibilityEventTypes="typeViewClicked|typeViewFocused"
        android:packageNames="com.example.android.myFirstApp,
        com.example.android.mySecondApp"
        android:accessibilityFeedbackType="feedbackSpoken"
        android:notificationTimeout="100"
        android:settingsActivity="com.example.android.apis.accessibility.TestBac
        kActivity"
        android:canRetrieveWindowContent="true"
    />
```

如果走 XML 路线，通过在服务声明中添加指向 XML 文件的<meta-data>标签，保证在清单文件中引用该 XML 文件。如果将 XML 文件保存在

res/xml/serviceconfig.xml 中，新的标签如下：

```
<service android:name=".MyAccessibilityService">
    <intent-filter>
        <action android:name="android.accessibilityservice.AccessibilityService" />
    </intent-filter>
    <meta-data android:name="android.accessibilityservice"
            android:resource="@xml/serviceconfig" />
</service>
```

3.3 AccessibilityEvent 响应

现在无障碍服务已经启动运行和监听事件，写一些代码，让它知道当一个 AccessibilityEvent 真的到来时该做什么吧！首先重写 onAccessibilityEvent(AccessibilityEvent)方法。在此方法中，用 getEventType()确定事件类型，并使用 getContentDescription()抽取与发出事件的视图相关联的所有标签文本。

```
    @Override
    public void onAccessibilityEvent(AccessibilityEvent event) {
        final int eventType = event.getEventType();
        String eventText = null;
        switch(eventType) {
            case AccessibilityEvent.TYPE_VIEW_CLICKED:
                eventText = "Focused: ";
                break;
            case AccessibilityEvent.TYPE_VIEW_FOCUSED:
                eventText = "Focused: ";
                break;
        }
        eventText = eventText + event.getContentDescription();
       //优化文本，例如将组合的字符串朗读给用户。
```

```
        speakToUser(eventText);
        ...
    }
```

3.4 为获取更多上下文查询视图层次

该步骤是可选的，却是非常有用的。Android 平台为每个 AccessibilityService 提供了查询视图层次的能力，来收集生成事件的 UI 组件、以及其父视图和子视图的信息。为了做到这点，确保在 XML 配置文件中设置了以下代码：

```
android:canRetrieveWindowContent="true"
```

一旦这样做了，可使用 getSource()得到一个 AccessibilityNodeInfo 对象。如果生成该事件的窗口仍然是活跃窗口，此调用只返回一个对象。如果是非活跃窗口，将返回空，行为相应改变。下面的样例是一个代码片段，当服务接收到事件时，执行以下行为：

1. 立即抓取生成事件视图的父视图；
2. 在此视图中，寻找作为子视图的标签和复选框；
3. 如果找到它们，创建一个表明标签和它是否被选中的字符串，报告给用户
4. 如果在遍历视图层次时，在任何点返回空值，此方法将被安静的放弃。

```
//使用 AccessibilityNodeInfo，代替 onAccessibilityEvent
@Override
public void onAccessibilityEvent(AccessibilityEvent event) {
    AccessibilityNodeInfo source = event.getSource();
    if (source == null) {
        return;
    }

    //抓取发送事件视图的父视图。
```

```
AccessibilityNodeInfo rowNode = getListItemNodeInfo(source);
if (rowNode == null) {
    return;
}

//使用父视图，得到标签和复选框两个子节点的引用。
AccessibilityNodeInfo labelNode = rowNode.getChild(0);
if (labelNode == null) {
    rowNode.recycle();
    return;
}

AccessibilityNodeInfo completeNode = rowNode.getChild(1);
if (completeNode == null) {
    rowNode.recycle();
    return;
}

//根据内部标签的文本和复选框的状态，来确定任务内容和任务是否完成。
if (rowNode.getChildCount() < 2 || !rowNode.getChild(1).isCheckable()) {
    rowNode.recycle();
    return;
}

CharSequence taskLabel = labelNode.getText();
final boolean isComplete = completeNode.isChecked();
String completeStr = null;

if (isComplete) {
    completeStr = getString(R.string.checked);
} else {
    completeStr = getString(R.string.not_checked);
}
```

```
        String reportStr = taskLabel + completeStr;
        speakToUser(reportStr);
}
```

现在已经有一个完整的能正常工作的无障碍服务。通过添加 Android 的文本转语音引擎，或使用振动器（Vibrator）提供触觉反馈，尝试配置服务与用户的交互方式吧！

4.无障碍测试清单

测试是实现应用对不同能力用户无障碍的一个重要部分。遵循无障碍设计和开发指南是实现该目标的重要步骤，但是无障碍测试可以发现在设计和开发期间不明显的用户交互问题。

该无障碍测试清单指导无障碍测试的重要方面，包含整体目标、必要测试步骤、推荐测试和特殊建议。该文档也讨论了为了测试目标怎样启动 Android 设备的无障碍特性。

4.1 测试目标

无障碍测试应该有以下高级别的目标：

- 在没有视觉帮助下启动和使用应用；
- 应用中的所有任务流程可以使用定向控制进行简单导航，并提供明确适当的反馈。

4.2 测试需求

为了保证应用最低级别的无障碍，必须完成以下测试。

1. **定向控制**：验证应用在不使用触摸屏的情况下可以被操作。尝试只使用定向控制器在应用中完成主要任务。在模拟器中使用键盘和 D-pad 控制或在 Android4.1（API 级别 16）或以上的设备上使用手势导航。

   注意：键盘和 D-pads 提供与无障碍手势不同的导航路径。手势允许用户聚焦屏幕上几乎所有的内容，键盘和 D-pad 导航只允许聚焦输入区域和按钮。

2. **TalkBack 音频反馈**：当启用 TalkBack 并且控件被聚焦时，验证提供信息的用户界面控件（图形或文本）或允许用户的操作有清楚明确的音频描述。使用定向控制在应用布局元素间移动焦点。

3. **触摸浏览提示**：当启用触摸浏览时，验证提供信息的用户界面控件（图形或文本）或允许用户的操作有清楚明确的音频描述。所有内容或控件区域都应该提供音频反馈。

4. **可触摸控件大小**：用户可选择或操作的所有控件最小宽高应为 48dp（大概 9mm），是 Android 设计中所推荐的。

5. **启用 TalkBack 时的手势响应**：验证应用特定手势，例如缩放图片、滚动列表、在页面间扫动或导航转盘控制，在开启 TalkBack 的情况下可持续工作。如果这些手势不起作用，必须为该操作提供一个替代交互。

6. **非纯音频反馈**：音频反馈必须总是有第二种反馈机制支持听障用户的使用，例如：信息到达时的声音警报应该伴随一个系统通知，触觉反馈（如果有）或其他视觉警告。

4.3 测试建议

推荐进行以下测试保证应用的无障碍特性。如果未测试这些条目，可能会影响应用整体的无障碍特性和质量。

1. **重复音频提示**：检查临近的相关控件（例如在列表中有多个组件的条目）没有机械地重复相同的音频反馈。例如，在包含联系人图像、文本名字和标题的联系人列表里面，每一个条目不应该机械地重复“鲍勃 • 史密斯”提示。

2. **音频提示超载或不足**：检查临近的有关控件提供适当级别的音频反馈，让用户理解和操作屏幕元素。提示太少或太多都会让用户在理解和使用控件时存在困难。

4.4 特殊实例和注意事项

以下列表描述了保证应用无障碍需要测试的特定场景。这里描述的某些或所有情况可能适用于正在开发的应用。复查该列表，验证这些特殊实例是否适用并采取合适的措施。

1. **复查开发特殊实例和注意事项**：复查无障碍开发的特殊实例列表并在应用中

测试已使用的实例。

2. **功能改变控件的提示**：根据应用上下文或流程改变功能的按钮或其他控件必须提供适用于当前功能的音频提示。例如，一个按钮，功能从播放视频变为暂停视频，应该提供适用于当前状态的音频提示。

3. **视频播放和字幕**：如果应用提供视频播放功能，验证该功能支持字幕来帮助听障用户。视频播放控件必须清楚说明视频字幕可获得，且提供简单方式以启用字幕。

4.5 测试无障碍特性

无障碍功能的测试，例如 TalkBack、触摸浏览和无障碍手势，需要启用测试设备中的这些功能。本章节描述了为了无障碍测试如何启动这些功能。

4.5.1 测试音频反馈

Android 设备上的无障碍音频反馈功能提供音频提示，在应用中导航时讲述屏幕内容。通过在 Android 设备中启用这个功能，可以测试视障用户的应用体验。

Android 中的用户音频反馈一般是由无障碍服务 TalkBack 和系统触摸浏览功能提供。TalkBack 无障碍服务会预置在大多数 Android 设备中，并且可以从 Google Play 免费下载。

4.5.1.1 TalkBack 测试

当用户将焦点移到控件上，TalkBack 无障碍服务会讲述出用户界面控件的内容。作为测试焦点导航和音频提示的一部分，该服务应该被启用。

启用 TalkBack 无障碍服务：

1. 打开“设置”应用；
2. 导航到“无障碍”目录并选择它；
3. 选择“无障碍”并启用它；
4. 选择“TalkBack”并启用它；

注意：TalkBack 是残障用户可用性最高的 Android 无障碍服务，其他无障碍

服务也可获得，并且可由用户安装。

更多关于使用 TalkBack 的信息，详见 TalkBack。

4.5.1.2 触摸浏览测试

触摸浏览系统功能在 Android4.0 和以后版本的设备可获得，启用一个特殊无障碍模式，允许用户在应用界面上拖动手指并听取屏幕内容的读出。该功能不需要使用定向控制器就可聚焦屏幕元素，并听取用户界面控件上的悬停事件。

启用触摸浏览：

1. 打开“设置”应用；
2. 导航到“无障碍”目录并选择它；
3. 选择“TalkBack”并启用它；

   注意：在 Android4.1（API 级别 16）和更高版本的设备中，系统提供一个弹出消息启用触摸浏览。在老版本中，必须遵守以下步骤。
4. 返回“无障碍”目录，选择触摸浏览并启用它。

   注意：必须先打开 TalkBack，否则不会出现该选项。

关于使用触摸浏览功能的更多信息，详见触摸浏览；

4.5.2 测试焦点导航

焦点导航是为了操作应用中的独立用户界面元素，使用定向控制器在独立用户界面元素间导航。视觉障碍用户或精细动作障碍用户经常使用这种导航模式，代替触摸导航。作为无障碍测试一部分，应该验证应用可以在只使用定向控制的情况下进行操作。

可以使用纯焦点控制测试应用的导航，即使要测试的设备没有定向控制器。Android 模拟器提供模拟定向控制器，可以用来测试导航。也可以使用基于软件的定向控制器，例如，在一个没有物理 D-pad 的测试设备上，由非视觉键盘提供一个定向控制器模拟 D-pad 的使用。

4.5.3 测试手势导航

手势导航是一个无障碍导航模式，允许用户使用特定手势导航 Android 设备

和应用。该导航模式在 Android4.1（API 级别 16）和更高版本中可获得。

注意：无障碍手势提供不同于键盘和 D-pad 的导航路径。手势允许用户聚焦屏幕上几乎所有的内容，键盘和 D-pad 导航只允许聚焦输入区域和按钮。

启用手势导航：

- 启用 TalkBack 和触摸浏览功能，被描述在“测试触摸浏览”。当启用了这两个功能，无障碍手势自动启动。
- 可以在“设置>无障碍>TalkBack>设置>管理快捷手势”中改变手势设置。

更多关于使用触摸浏览无障碍功能的信息，详见触摸浏览。

注意：无障碍服务，除 TalkBack 外，一个无障碍手势可能映射不同的用户操作。如果在测试过程中，手势没有产生预期结果，尝试在继续前关闭其他无障碍服务。