# Tools attributes reference

Android Studio supports a variety of XML attributes in the `tools` namespace that enable design-time features (such as which layout to show in a fragment) or compile-time behaviors (such as which shrinking mode to apply to your XML resources). When you build your app, the build tools remove these attributes so there is no effect on your APK size or runtime behavior.

To use these attributes, add the `tools` namespace to the root element of each XML file where you'd like to use them, as shown here:

```
<RootTag xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" >
```

## Error handling attributes

The following attributes help suppress lint warning messages.

### `tools:ignore`

*Intended for: Any element*

*Used by: Lint*

This attribute accepts a comma-separated list of lint issue ID's that you'd like the tools to ignore on this element or any of its decendents.

For example, you can tell the tools to ignore the `MissingTranslation` error:

```
<string name="show_all_apps" tools:ignore="MissingTranslation">All</string>
```

### `tools:targetApi`

*Intended for: Any element*

*Used by: Lint*

This attribute works the same as the `@TargetApi` (/reference/android/annotation/TargetApi) annotation in Java code: it lets you specify the API level (either as an integer or as a code name) that supports this element.

This tells the tools that you believe this element (and any children) will be used only on the specified API level or higher. This stops lint from warning you if that element or its attributes are not available on the API level you specify as your `minSdkVersion`.

For example, you might use this because `GridLayout` (/reference/android/widget/GridLayout) is available only on API level 14 and higher, but you know this layout is not used for any lower versions:

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:targetApi="14" >
```

However, you should instead use `GridLayout` from the support library (/reference/androidx/gridlayout/widget/GridLayout).

### `tools:locale`

*Intended for:* `<resources>`

*Used by: Lint, Android Studio editor*

This tells the tools what the default language/locale is for the resources in the given `<resources>` element (because the tools otherwise assume English) in order to avoid warnings from the spell checker. The value must be a valid locale qualifier (/guide/topics/resources/providing-resources#LocaleQualifier).

For example, you can add this to your `values/strings.xml` file (the default string values) to indicate that the language used for the default strings is Spanish rather than English:

```
<resources xmlns:tools="http://schemas.android.com/tools"
    tools:locale="es">
```

## Design-time view attributes

The following attributes define layout characteristics that are visible only in the Android Studio layout preview.

## `tools:` instead of `android:`

*Intended for:* `<View>`

*Used by: Android Studio layout editor*

You can insert sample data in your layout preview by using the `tools:` prefix instead of `android:` with any `<View>` attribute from the Android framework. This is useful when the attribute's value isn't populated until runtime but you want to see the effect beforehand, in the layout preview.

For example, if the `android:text` attribute value is set at runtime or you want to see the layout with a value different than the default, you can add `tools:text` to specify some text for the layout preview only.


**Figure 1.** The `tools:text` attribute sets "Google Voice" as the value for the layout preview

You can add both the `android:` namespace attribute (which is used at runtime) and the matching `tools:` attribute (which overrides the runtime attribute in the layout preview only).

You can also use a `tools:` attribute to *unset* an attribute only for the layout preview. For example, if you have a `FrameLayout` with multiple children but you want to see only one child in the layout preview, you can set one of them to be invisible in the layout preview, as shown here:

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="First" />

<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:text="Second"
    tools:visibility="invisible"  />
```

When using the Layout Editor (/studio/write/layout-editor) in design view, the **Properties** window also allows you to edit some design-time view attributes. Each design-time attribute is indicated with a wrench icon

next to the attribute name to distinguish it from the real attribute of the same name.

## `tools:context`

*Intended for: Any root* `<View>`

*Used by: Lint, Android Studio layout editor*

This attribute declares which activity this layout is associated with by default. This enables features in the editor or layout preview that require knowledge of the activity, such as what the layout theme should be in the preview and where to insert `onClick` handlers when you make those from a quickfix (figure 2).

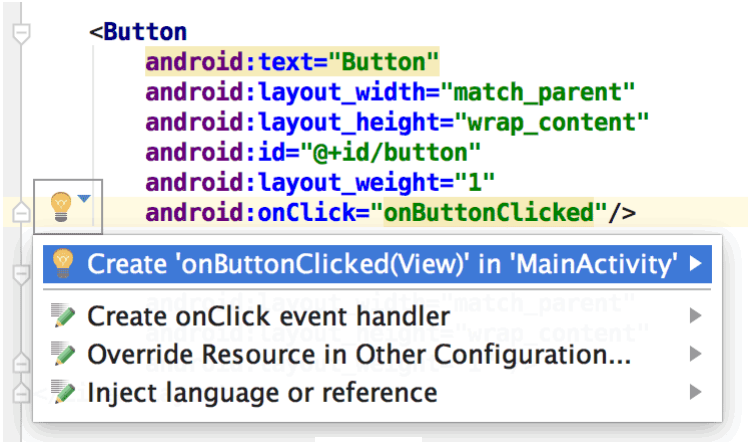


**Figure 2.** Quickfix for the `onClick` attribute works only if you've set `tools:context`

You can specify the activity class name using the same dot prefix as in the manifest file (excluding the full package name). For example:

```
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MainActivity" >
```

**Tip:** You can also select the theme for the layout preview from the Layout Editor toolbar (/studio/write/layout-editor#change-appearance).

## tools:itemCount

*Intended for:* `<RecyclerView>`

*Used by: Android Studio layout editor*

For a given `RecyclerView` (/reference/androidx/recyclerview/widget/RecyclerView), this attribute specifies the number of items the layout editor should render in the **Preview** window.

For example:

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/recyclerView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:itemCount="3"/>
```

## tools:layout

*Intended for:* `<fragment>`

*Used by: Android Studio layout editor*

This attribute declares which layout you want the layout preview to draw inside the fragment (because the layout preview cannot execute the activity code that normally applies the layout).

For example:

```
<fragment android:name="com.example.main.ItemListFragment"
    tools:layout="@layout/list_content" />
```

## tools:listitem / tools:listheader / tools:listfooter

*Intended for:* `<AdapterView>` *(and subclasses like* `<ListView>`*)*

*Used by: Android Studio layout editor*

These attributes specify which layout to show in the layout preview for a list's items, header, and footer. Any data fields in the layout are filled with numeric contents such as "Item 1" so that the list items are not repetitive.

For example:

```
<ListView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@android:id/list"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:listitem="@layout/sample_list_item"
    tools:listheader="@layout/sample_list_header"
    tools:listfooter="@layout/sample_list_footer" />
```

**Note:** These attributes don't work for `ListView` in Android Studio 2.2, but this is fixed in 2.3 (issue 215172 (https://code.google.com/p/android/issues/detail?id=215172)).

## `tools:showIn`

*Intended for: Any root* `<View>` *in a layout that's referred to by an* `<include>`

*Used by: Android Studio layout editor*

This attribute allows you to point to a layout that uses this layout as an include (/training/improving-layouts/reusing-layouts), so you can preview (and edit) this file as it appears while embedded in its parent layout.

For example:

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:text="@string/hello_world"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    tools:showIn="@layout/activity_main" />
```

Now the layout preview shows this `TextView` layout as it appears inside the `activity_main` layout.

## `tools:menu`

*Intended for: Any root* `<View>`

*Used by: Android Studio layout editor*

This attribute specifies which menu the layout preview should show in the app bar (/training/appbar). The value can be one or more menu IDs, separated by commas (without `@menu/` or any such ID prefix and without the `.xml` extension). For example:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:menu="menu1,menu2" />
```

## `tools:minValue / tools:maxValue`

*Intended for:* `<NumberPicker>`

*Used by: Android Studio layout editor*

These attributes set minimum and maximum values for a `NumberPicker` (/reference/android/widget/NumberPicker) view.

For example:

```
<NumberPicker xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/numberPicker"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    tools:minValue="0"
    tools:maxValue="10" />
```

### tools:openDrawer

*Intended for:* `<DrawerLayout>`

*Used by: Android Studio layout editor*

This attribute allows you to open a `DrawerLayout` (/reference/androidx/drawerlayout/widget/DrawerLayout) in the **Preview** pane of the layout editor. You can also modify how the layout editor renders the layout by passing one of the following values:

| Constant | Value | Description |
|---|---|---|
| end | 800005 | Push object to the end of its container, not changing its size. |
| left | 3 | Push object to the left of its container, not changing its size. |
| right | 5 | Push object to the right of its container, not changing its size. |
| start | 800003 | Push object to the beginning of its container, not changing its size. |

For example:

```
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:openDrawer="start" />
```

### "@tools:sample/*" resources

*Intended for: Any view that supports UI text or images.*

*Used by: Android Studio layout editor*

This attribute allows you to inject placeholder data or images into your view. For example, if you want to test how your layout behaves with text, but you don't yet have finalized UI text for your app, you can use placeholder text as follows:

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    tools:text="@tools:sample/lorem" />
```

The following table describes the types of placeholder data you can inject into your layouts.

| Attribute value | Description of placeholder data |
|---|---|
| `@tools:sample/full_names` | Full names that are randomly generated from the combination of `@tools:sample/first_names` and `@tools:sample/last_names`. |
| `@tools:sample/first_names` | Common first names. |
| `@tools:sample/last_names` | Common last names. |
| `@tools:sample/cities` | Names of cities from across the world. |
| `@tools:sample/us_zipcodes` | Randomly generated US zipcodes. |
| `@tools:sample/us_phones` | Randomly generated phone numbers with the following format: `(800) 555-xxxx`. |
| `@tools:sample/lorem` | Placeholder text that is derived from Latin. |
| `@tools:sample/date/day_of_week` | Randomized dates and times for the specified format. |
| `@tools:sample/date/ddmmyy` | |
| `@tools:sample/date/mmddyy` | |
| `@tools:sample/date/hhmm` | |
| `@tools:sample/date/hhmmss` | |
| `@tools:sample/avatars` | Vector drawables that you can use as profile avatars. |

| | |
|---|---|
| `@tools:sample/backgrounds/scenic` | Images that you can use as backgrounds. |

## Resource shrinking attributes

The following attributes allow you to enable strict reference checks and declare whether to keep or discard certain resources when using resource shrinking (/studio/build/shrink-code#shrink-resources).

To enable resource shrinking, set the `shrinkResources` property to `true` in your `build.gradle` file (alongside `minifyEnabled` for code shrinking). For example:

```
android {
    ...
    buildTypes {
        release {
            shrinkResources true
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
                    'proguard-rules.pro'
        }
    }
}
```

### `tools:shrinkMode`

*Intended for:* `<resources>`

*Used by: Build tools with resource shrinking*

This attribute allows you to specify whether the build tools should use "safe mode" (play it safe and keep all resources that are explicitly cited and that *might* be referenced dynamically with a call to `Resources.getIdentifier()` (/reference/android/content/res/Resources#getIdentifier(java.lang.String,%20java.lang.String,%20java.lang.String)) ) or "strict mode" (keep only the resources that are explicitly cited in code or in other resources).

The default is to use safe mode (`shrinkMode="safe"`). To instead use strict mode, add `shrinkMode="strict"` to the `<resources>` tag as shown here:

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:tools="http://schemas.android.com/tools"
    tools:shrinkMode="strict" />
```

When you enable strict mode, you may need to use `tools:keep` (#toolskeep) to keep resources that were removed but that you actually want, and use `tools:discard` (#toolsdiscard) to explicitly remove even more resources.

For more information, see Shrink your resources (/studio/build/shrink-code#shrink-resources).

## tools:keep

*Intended for:* `<resources>`

*Used by: Build tools with resource shrinking*

When using resource shrinking to remove unused resources, this attribute allows you to specify resources to keep (typically because they are referenced in an indirect way at runtime, such as by passing a dynamically generated resource name to `Resources.getIdentifier()` (/reference/android/content/res/Resources#getIdentifier(java.lang.String,%20java.lang.String,%20java.lang.String)) ).

To use, create an XML file in your resources directory (for example, at `res/raw/keep.xml`) with a `<resources>` tag and specify each resource to keep in the `tools:keep` attribute as a comma-separated list. You can use the asterisk character as a wild card. For example:

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:tools="http://schemas.android.com/tools"
    tools:keep="@layout/used_1,@layout/used_2,@layout/*_3" />
```

For more information, see Shrink your resources (/studio/build/shrink-code#shrink-resources).

## tools:discard

*Intended for:* `<resources>`

*Used by: Build tools with resource shrinking*

When using resource shrinking to strip out unused resources, this attribute allows you to specify resources you want to manually discard (typically because the resource *is* referenced but in a way that does not affect your app, or because the Gradle plugin has incorrectly deduced that the resource is referenced).

To use, create an XML file in your resources directory (for example, at `res/raw/keep.xml`) with a `<resources>` tag and specify each resource to keep in the `tools:discard` attribute as a comma-separated list. You can use the asterisk character as a wild card. For example:

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:tools="http://schemas.android.com/tools"
    tools:discard="@layout/unused_1" />
```

For more information, see Shrink your resources (/studio/build/shrink-code#shrink-resources).



Last updated 2020-08-25 UTC.