KotlinConf

Kotlin Static Analysis with Android Lint

Tor Norbye
@tornorbye

Outline

- Lint Philosophy
- Lint Features
- How to write a lint check
  - Basics
  - Testing
  - Kotlin
- Lint Infrastructure Features
- Gotchas
- Futures

Android Lint

A static analyzer

1.0 released in 2011.

Android Lint Guiding Philosophy #1:

False positives are better than false negatives.

Lint Features

Ability to Suppress Issues

In XML with attributes: tools:ignore="MyId"

In Java & Kotlin with annotation: @SuppressLint("MyId")

In Java & Kotlin on statements: //noinspection MyId

On any location, or regular expression, with lint.xml file

Via quickfix in the IDE

Android Lint Guiding Philosophy #2:

Focus on Android issues — Leave general coding issues to the IDEs

Annotation Checks

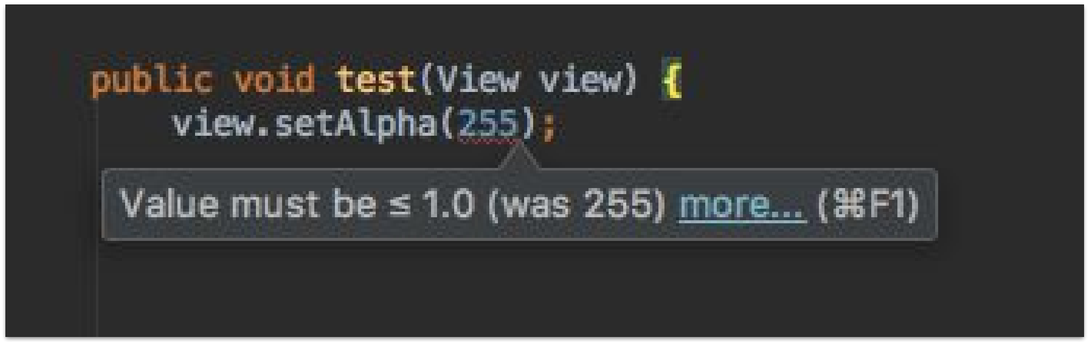
@StringRes, @DrawableRes, ...

@UiThread, @WorkerThread, …

@IntDef, @StringDef, …

@CheckResult, @CallSuper, @RestrictTo

@Size, @IntRange, @FloatRange



Android Lint not just for Android
(as of version 3.1)

classpath 'com.android.tools.build:gradle:3.1.0-alpha01'
apply plugin 'kotlin'
apply plugin 'com.android.lint'

```
$ ./gradlew lint
```

Lint Features

Multi Disciplinary

.xml, AndroidManifest

.java, .kt, .gradle

.class

.pro/.cfg

.png, .webp, .jpg, ...

DesignerNewsStory.java:

```
final TextView title = (TextView) findViewById(R.id.story_title);
title.setText(story.title);
```

designer_news_story_item.xml

```
android {
<io.plaidapp.ui.widget.BaselineGridTextView
    android:id="@+id/story_title"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_marginStart="@dimen/padding_normal"
    android:layout_marginTop="@dimen/padding_normal"
```

DesignerNewsStory.java:

```
final TextView title = (TextView) findViewById(R.id.story_title);
title.setText(story.title);
```

designer_news_story_item.xml

```
android {
<io.plaidapp.ui.widget.BaselineGridTextView
    android:id="@+id/story_title"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_marginStart="@dimen/padding_normal"
    android:layout_marginTop="@dimen/padding_normal"
```

DesignerNewsStory.java:

```
final TextView title = (TextView) findViewById(R.id.story_title);
title.setText(story.title);
```

designer_news_story_item.xml

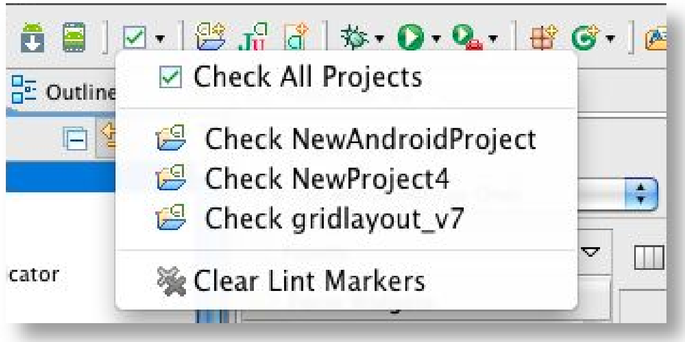
```
android {
<io.plaidapp.ui.widget.BaselineGridTextView
    android:id="@+id/story_title"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_marginStart="@dimen/padding_normal"
    android:layout_marginTop="@dimen/padding_normal"
```

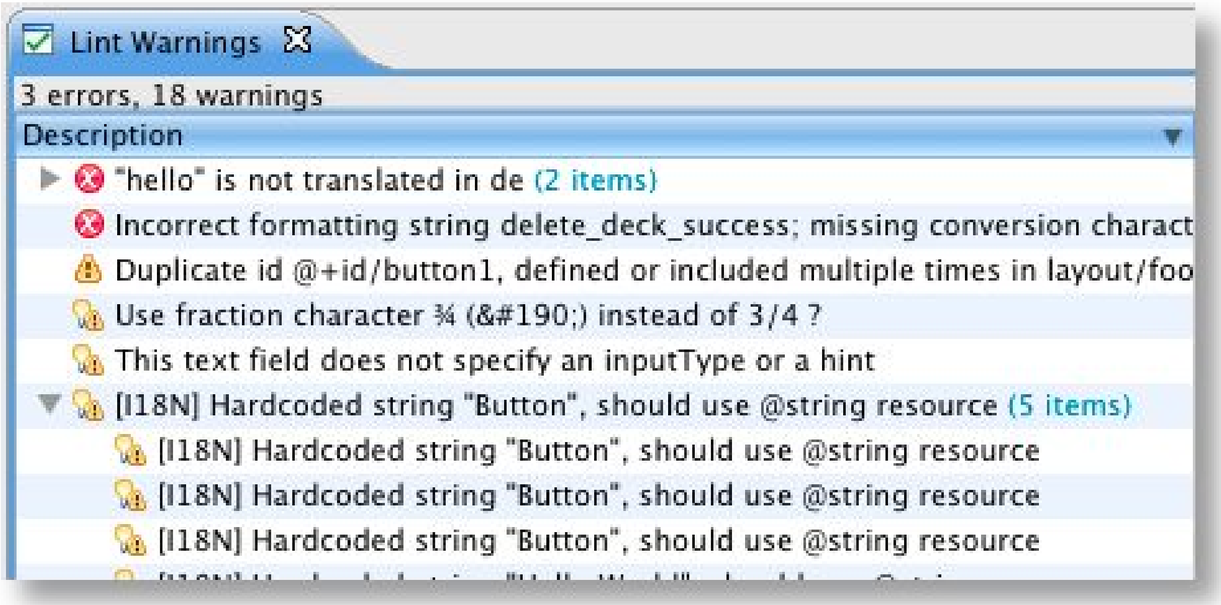
Lint Features

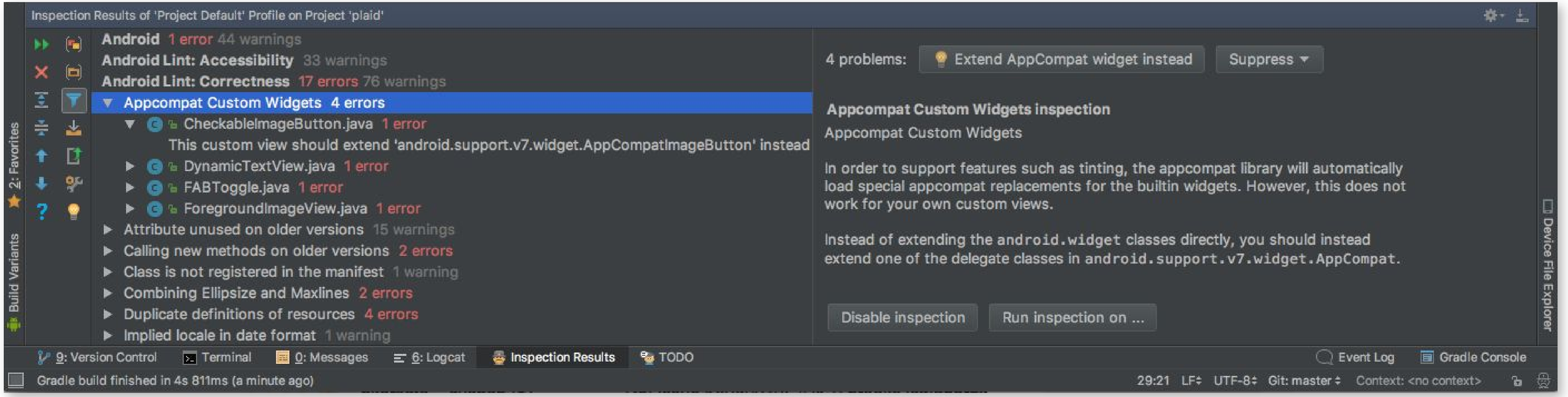
Designed for tool integration

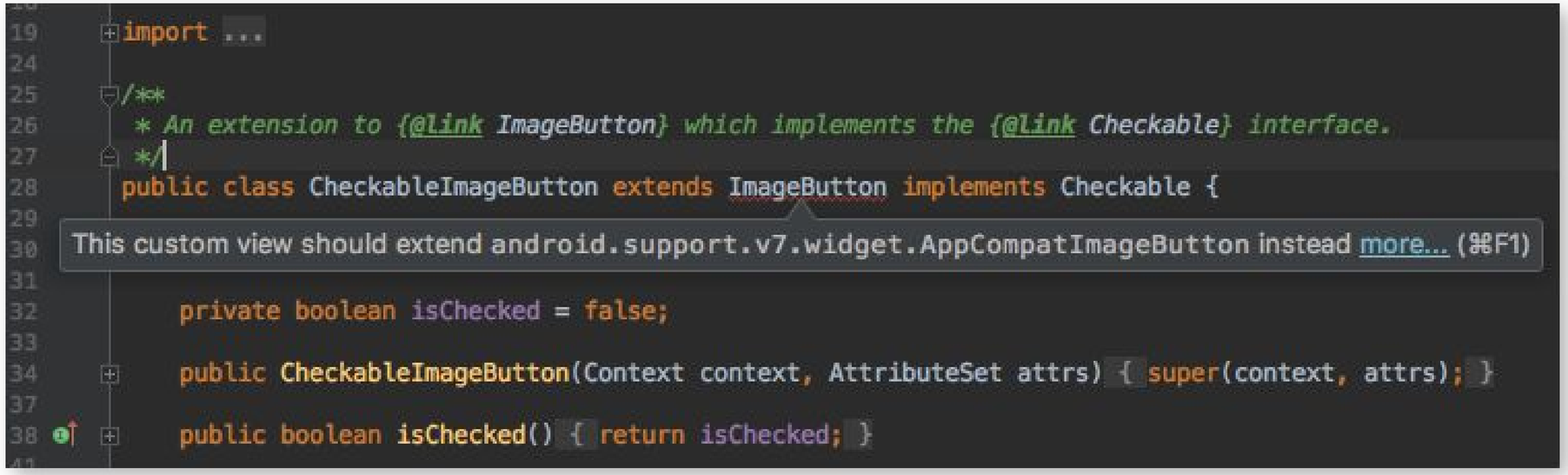
Not a command line binary

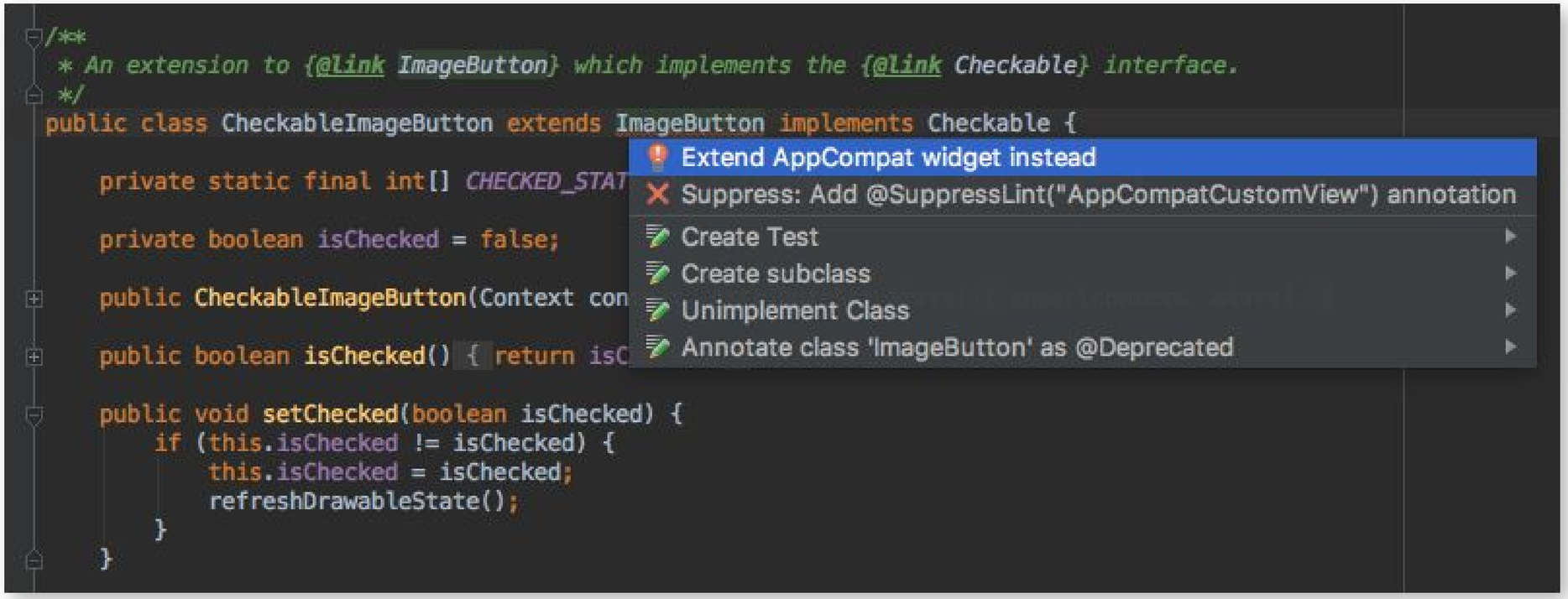
Packaged within each integration


Outline
Check All Projects
Check NewAndroidProject
Check NewProject4
Check gridlayout_v7
Clear Lint Markers
cator

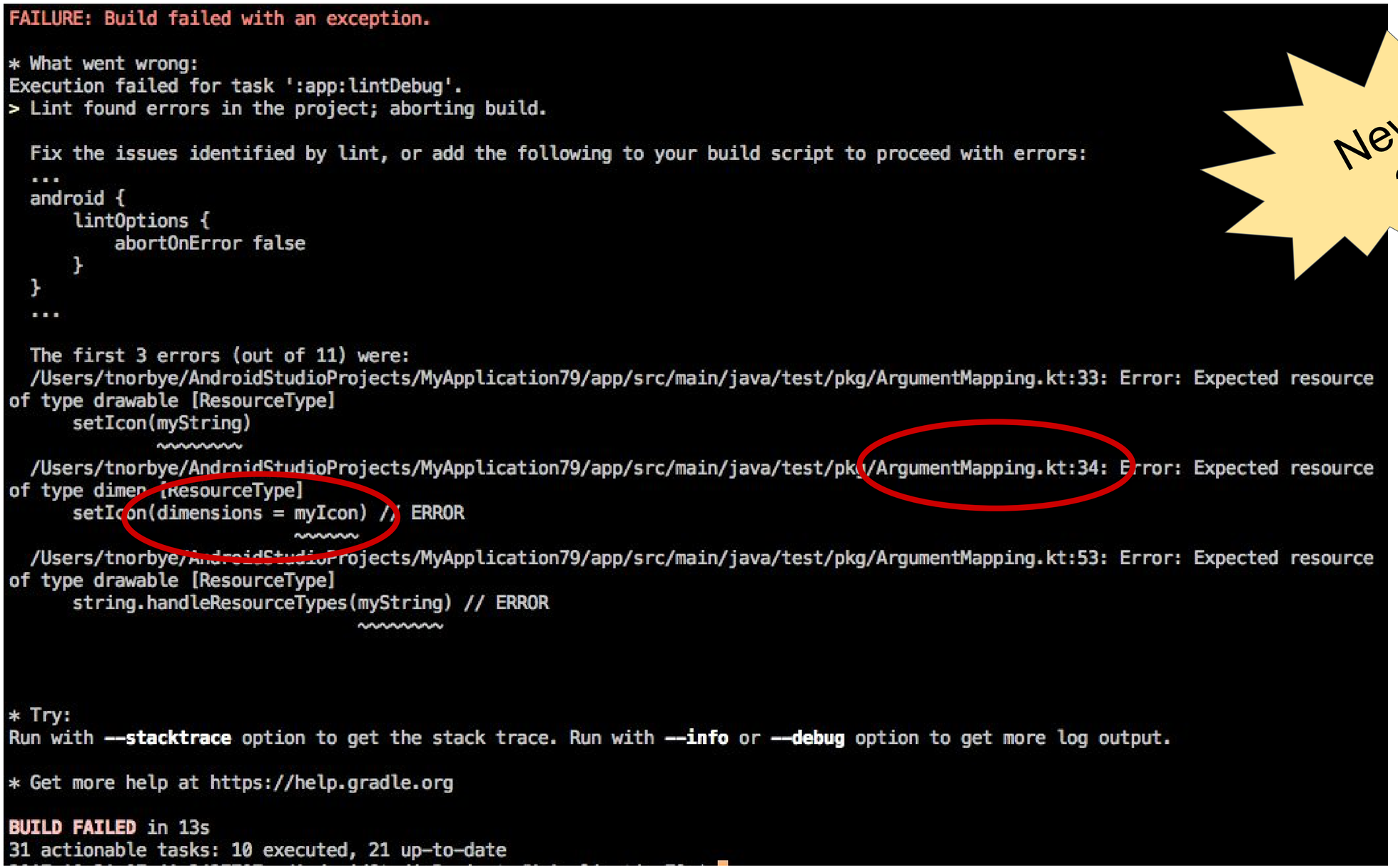

Lint Warnings
3 errors, 18 warnings
Description
"hello" is not translated in de (2 items)
Incorrect formatting string delete_deck_success; missing conversion charact
Duplicate id @+id/button1, defined or included multiple times in layout/foo
Use fraction character ¾ (¾) instead of 3/4 ?
This text field does not specify an inputType or a hint
[I18N] Hardcoded string "Button", should use @string resource (5 items)
[I18N] Hardcoded string "Button", should use @string resource
[I18N] Hardcoded string "Button", should use @string resource
[I18N] Hardcoded string "Button", should use @string resource


Inspection Results of 'Project Default' Profile on Project 'plaid'
Android 1 error 44 warnings
Android Lint: Accessibility 33 warnings
Android Lint: Correctness 17 errors 76 warnings
Appcompat Custom Widgets 4 errors
CheckableImageButton.java 1 error
This custom view should extend 'android.support.v7.widget.AppCompatImageButton' instead
DynamicTextView.java 1 error
FABToggle.java 1 error
ForegroundImageView.java 1 error
Attribute unused on older versions 15 warnings
Calling new methods on older versions 2 errors
Class is not registered in the manifest 1 warning
Combining Ellipsize and Maxlines 2 errors
Duplicate definitions of resources 4 errors
Implied locale in date format 1 warning
4 problems:
Extend AppCompat widget instead
Suppress
Appcompat Custom Widgets inspection
Appcompat Custom Widgets
In order to support features such as tinting, the appcompat library will automatically load special appcompat replacements for the builtin widgets. However, this does not work for your own custom views.
Instead of extending the android.widget classes directly, you should instead extend one of the delegate classes in android.support.v7.widget.AppCompat.
Disable inspection
Run inspection on ...
2: Favorites
Build Variants
Device File Explorer
9: Version Control
Terminal
0: Messages
6: Logcat
Inspection Results
TODO
Event Log
Gradle Console
Gradle build finished in 4s 811ms (a minute ago)
29:21
LF
UTF-8
Git: master
Context: <no context>


19 import ...
24
25 /**
26 * An extension to {@link ImageButton} which implements the {@link Checkable} interface.
27 */
28 public class CheckableImageButton extends ImageButton implements Checkable {
29
30 This custom view should extend android.support.v7.widget.AppCompatImageButton instead more... (⌘F1)
31
32 private boolean isChecked = false;
33
34 public CheckableImageButton(Context context, AttributeSet attrs) { super(context, attrs); }
37
38 public boolean isChecked() { return isChecked; }


/**
* An extension to {@link ImageButton} which implements the {@link Checkable} interface.
*/
public class CheckableImageButton extends ImageButton implements Checkable {
private static final int[] CHECKED_STAT
private boolean isChecked = false;
public CheckableImageButton(Context con
public boolean isChecked() { return isC
public void setChecked(boolean isChecked) {
if (this.isChecked != isChecked) {
this.isChecked = isChecked;
refreshDrawableState();
}
}
Extend AppCompat widget instead
Suppress: Add @SuppressLint("AppCompatCustomView") annotation
Create Test
Create subclass
Unimplement Class
Annotate class 'ImageButton' as @Deprecated

```
14 errors, 189 warnings
Wrote HTML report to file:///Users/tnorbye/dev/samples/plaid/app/build/reports/lint-results-debug.html
Wrote XML report to file:///Users/tnorbye/dev/samples/plaid/app/build/reports/lint-results-debug.xml


FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':app:lintDebug'.
> Lint found errors in the project; aborting build.

  Fix the issues identified by lint, or add the following to your build script to proceed with errors:
  ...
  android {
      lintOptions {
          abortOnError false
      }
  }
  ...

* Try:
Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output.

* Get more help at https://help.gradle.org

BUILD FAILED in 25s
49 actionable tasks: 3 executed, 46 up-to-date
```


FAILURE: Build failed with an exception.
* What went wrong:
Execution failed for task ':app:lintDebug'.
> Lint found errors in the project; aborting build.
Fix the issues identified by lint, or add the following to your build script to proceed with errors:
...
android {
lintOptions {
abortOnError false
}
}
...
The first 3 errors (out of 11) were:
/Users/tnorbye/AndroidStudioProjects/MyApplication79/app/src/main/java/test/pkg/ArgumentMapping.kt:33: Error: Expected resource of type drawable [ResourceType]
setIcon(myString)
/Users/tnorbye/AndroidStudioProjects/MyApplication79/app/src/main/java/test/pkg/ArgumentMapping.kt:34: Error: Expected resource of type dimen [ResourceType]
setIcon(dimensions = myIcon) // ERROR
/Users/tnorbye/AndroidStudioProjects/MyApplication79/app/src/main/java/test/pkg/ArgumentMapping.kt:53: Error: Expected resource of type drawable [ResourceType]
string.handleResourceTypes(myString) // ERROR
* Try:
Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output.
* Get more help at https://help.gradle.org
BUILD FAILED in 13s
31 actionable tasks: 10 executed, 21 up-to-date

New in 3.1

Lint Report: 14 errors and 189 warnings

Check performed at Fri Oct 27 13:36:19 PDT 2017

Implied default locale in case conversion

src/main/java/io/plaidapp/ui/DesignerNewsLogin.java:227: Implicitly using the default locale is a common source of bugs: Use `toLowerCase(Locale)` instead. For strings meant to be internal use `Locale.ROOT`, otherwise `Locale.getDefault()`.

```
224   final Toast confirmLogin = new Toast(getApplicationContext());
225   final View v = LayoutInflater.from(DesignerNewsLogin.this).inflate(R.layout
226           .toast_logged_in_confirmation, null, false);
227   ((TextView) v.findViewById(R.id.name)).setText(user.display_name.toLowerCase());
228   // need to use app context here as the activity will be destroyed shortly
229   GlideApp.with(getApplicationContext())
230           .load(user.portrait_url)
```

src/main/java/io/plaidapp/ui/DesignerNewsStory.java:818: Implicitly using the default locale is a common source of bugs: Use `toLowerCase(Locale)` instead. For strings meant to be internal use `Locale.ROOT`, otherwise `Locale.getDefault()`.

```
815                     }
816                 });
817                 if (comment.user_display_name != null) {
818                     holder.author.setText(comment.user_display_name.toLowerCase());
819                 }
820                 holder.author.setOriginalPoster(isOP(comment.user_id));
821                 if (comment.created_at != null) {
```

+ 9 MORE OCCURRENCES...

DefaultLocale | Correctness | Warning | Priority 6/10

EXPLAIN DISMISS

Duplicate definitions of resources

src/main/res/values/styles.xml:134: `android:statusBarColor` has already been defined in this `<style>`

```
131         <item name="android:windowAnimationStyle">@null</item>
132         <item name="android:colorPrimary">@color/designer_news</item>
133         <item name="android:colorAccent">@color/designer_news</item>
134         <item name="android:statusBarColor">@color/designer_news_super_dark</item>
135         <item name="android:navigationBarColor">@color/designer_news_super_dark</item>
136         <item name="android:colorButtonNormal">@color/designer_news_button</item>
137         <item name="android:colorControlActivated">@color/designer_news</item>
```

src/main/res/values/styles.xml:129: Previously defined here

```
126     </style>
127
128     <style name="Plaid.Translucent.DesignerNewsLogin">
129         <item name="android:statusBarColor">@android:color/transparent</item>
130         <item name="android:navigationBarColor">@android:color/transparent</item>
131         <item name="android:windowAnimationStyle">@null</item>
132         <item name="android:colorPrimary">@color/designer_news</item>
```

src/main/res/values/styles.xml:135: `android:navigationBarColor` has already been defined in this `<style>`

Lint Report: 14 errors and 189 warnings

Check performed at Fri Oct 27 13:36:19 PDT 2017

Implied default locale in case conversion

src/main/java/io/plaidapp/ui/DesignerNewsLogin.java:227: Implicitly using the default locale is a common source of bugs: Use `toLowerCase(Locale)` instead. For strings meant to be internal use `Locale.ROOT`, otherwise `Locale.getDefault()`.

```
224   final Toast confirmLogin = new Toast(getApplicationContext());
225   final View v = LayoutInflater.from(DesignerNewsLogin.this).inflate(R.layout
226           .toast_logged_in_confirmation, null, false);
227   ((TextView) v.findViewById(R.id.name)).setText(user.display_name.toLowerCase());
228   // need to use app context here as the activity will be destroyed shortly
229   GlideApp.with(getApplicationContext())
230           .load(user.portrait_url)
```

src/main/java/io/plaidapp/ui/DesignerNewsStory.java:818: Implicitly using the default locale is a common source of bugs: Use `toLowerCase(Locale)` instead. For strings meant to be internal use `Locale.ROOT`, otherwise `Locale.getDefault()`.

```
815                     }
816                 });
817                 if (comment.user_display_name != null) {
818                     holder.author.setText(comment.user_display_name.toLowerCase());
819                 }
820                 holder.author.setOriginalPoster(isOP(comment.user_id));
821                 if (comment.created_at != null) {
```

+ 9 MORE OCCURRENCES...

DefaultLocale | Correctness | Warning | Priority 6/10

EXPLAIN DISMISS

Duplicate definitions of resources

src/main/res/values/styles.xml:134: `android:statusBarColor` has already been defined in this `<style>`

```
131         <item name="android:windowAnimationStyle">@null</item>
132         <item name="android:colorPrimary">@color/designer_news</item>
133         <item name="android:colorAccent">@color/designer_news</item>
134         <item name="android:statusBarColor">@color/designer_news_super_dark</item>
135         <item name="android:navigationBarColor">@color/designer_news_super_dark</item>
136         <item name="android:colorButtonNormal">@color/designer_news_button</item>
137         <item name="android:colorControlActivated">@color/designer_news</item>
```

src/main/res/values/styles.xml:129: Previously defined here

```
126     </style>
127
128     <style name="Plaid.Translucent.DesignerNewsLogin">
129         <item name="android:statusBarColor">@android:color/transparent</item>
130         <item name="android:navigationBarColor">@android:color/transparent</item>
131         <item name="android:windowAnimationStyle">@null</item>
132         <item name="android:colorPrimary">@color/designer_news</item>
```

src/main/res/values/styles.xml:135: `android:navigationBarColor` has already been defined in this `<style>`

Using inlined constants on older versions

src/main/java/io/plaidapp/ui/widget/CollapsingTitleLayout.java:60: Field requires API level 23 (current min is 21): `android.text.Layout#BREAK_STRATEGY_HIGH_QUALITY`

```
57 public class CollapsingTitleLayout extends FrameLayout implements ReflowText.Reflowable {
58
59     // constants
60     private static final int BREAK_STRATEGY = Layout.BREAK_STRATEGY_HIGH_QUALITY;
61
62     // configurable attributes
63     private int titleInsetStart;
```

src/main/java/io/plaidapp/ui/PlayerActivity.java:129: Field requires API level 26 (current min is 21): `android.view.View#SYSTEM_UI_FLAG_LIGHT_NAVIGATION_BAR`

```
126             View.SYSTEM_UI_FLAG_LAYOUT_STABLE
127             | View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
128             | View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
129             | View.SYSTEM_UI_FLAG_LIGHT_NAVIGATION_BAR);
130     draggableFrame.setOnApplyWindowInsetsListener(new View.OnApplyWindowInsetsListener() {
131         @Override
132         public WindowInsets onApplyWindowInsets(View v, WindowInsets insets) {
```

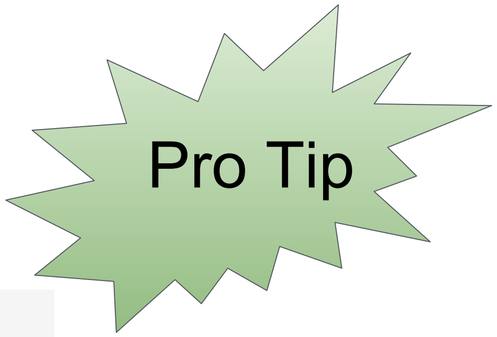
This check scans through all the Android API field references in the application and flags certain constants, such as static final integers and Strings, which were introduced in later versions. These will actually be copied into the class files rather than being referenced, which means that the value is available even when running on older devices. In some cases that's fine, and in other cases it can result in a runtime crash or incorrect behavior. It depends on the context, so consider the code carefully and decide whether it's safe and can be suppressed or whether the code needs to be guarded.

If you really want to use this API and don't need to support older devices just set the `minSdkVersion` in your `build.gradle` or `AndroidManifest.xml` files.
If your code is **deliberately** accessing newer APIs, and you have ensured (e.g. with conditional execution) that this code will only ever be called on a supported platform, then you can annotate your class or method with the `@TargetApi` annotation specifying the local minimum SDK to apply, such as `@TargetApi(11)`, such that this check considers 11 rather than your manifest file's minimum SDK as the required API level.
Note: This issue has an associated quickfix operation in Android Studio and IntelliJ IDEA.
To suppress this error, use the issue id "InlinedApi" as explained in the Suppressing Warnings and Errors section.

InlinedApi | Correctness | Warning | Priority 6/10

DISMISS

$ ./gradlew lintDebug -Dlint.html.prefs=theme=darcula



ScrollView size validation

src/main/res/layout/about_icon.xml:28: This LinearLayout should use `android:layout_height="wrap_content"`

```
25
26      <LinearLayout
27          android:layout_width="match_parent"
28          android:layout_height="match_parent"
29          android:orientation="vertical"
30          android:background="@color/background_light"
31          android:elevation="@dimen/z_card">
```

src/main/res/layout/about_plaid.xml:29: This LinearLayout should use `android:layout_height="wrap_content"`

```
26
27      <LinearLayout
28          android:layout_width="match_parent"
29          android:layout_height="match_parent"
30          android:orientation="vertical">
31
32          <io.plaidapp.ui.widget.CutoutTextView
```

ScrollViewSize | Correctness | Warning | Priority 7/10

EXPLAIN DISMISS

Check Release Builds

lintVital target

Classic problem: You had lint, but didn't run it

Android Gradle plugin separates "debug" and "release";
  Gradle **build** release target depends on lintVital

LintVital = fatal-severity checks

42/315, but user configurable

```
/* build.gradle */
android {
    lintOptions {
        // Promote //STOPSHIP comment detector, and the API check, to fatal
        fatal 'StopShip', 'NewApi'

        // Demote missing translations; we're okay with these gaps right now
        warning 'MissingTranslation'
    }
}
```

Jenkins

Pages

SPACE SHORTCUTS

How-to articles

CHILD PAGES

Plugins

Android Lint Plugin

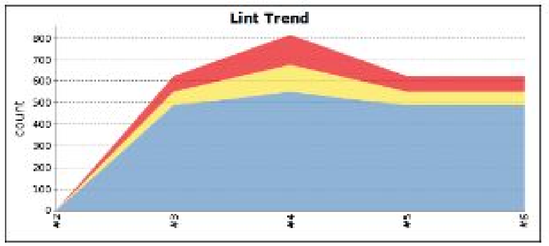
+ Create child page

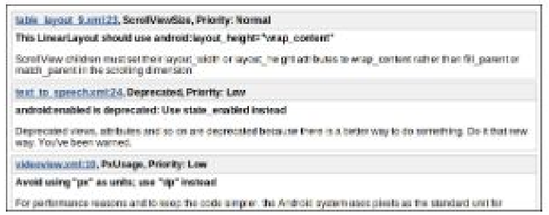
Features

Android Lint is a tool which scans your Android projects and reports on potential bugs, performance, security and translation issues, plus more.

This Jenkins plugin parses XML reports produced by running `lint`, analyses them and displays the results for each build.
Information shown includes a build summary, trend graphs, display of warnings in context, and dashboard portlets.

This plugin builds on the work of the static analysis core plugin; see the Static Code Analysis Plug-ins page for a fuller list of features.





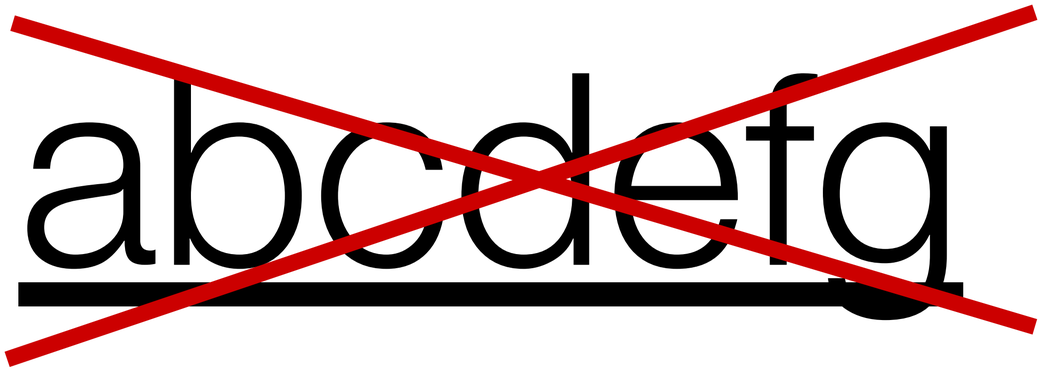
Requirements

- Jenkins 2.7 or newer
- The Static Analysis Utilities plugin — this will be automatically installed when you install this plugin
- Android SDK Tools r17 or newer, but at least r21 is recommended

If the Dashboard View plugin is also installed, you will be able to add Lint-specific portlets to your dashboard views.

Lint Features

Baselines



```
/* build.gradle */
android {
    lintOptions {
        baseline file('baseline.xml')
    }
}
```

```
$ ./gradlew lint
…

14 errors, 189 warnings
Created baseline file /Users/tnorbye/dev/samples/plaid/app/baseline.xml

...

FAILURE: Build failed with an exception.

$
```

```
$ head app/baseline.xml
<?xml version="1.0" encoding="UTF-8"?>
<issues format="4" by="lint 3.1.0-dev">

    <issue
        id="FontValidationWarning"
        message="For `minSdkVersion`=21 only `app:` attributes should be used"
        errorLine1="    android:fontProviderAuthority=&quot;com.google.android.gms.fonts&quot;"
        errorLine2="    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~">
        <location
            file="src/main/res/font/roboto_mono.xml"
            line="18"
            column="5"/>
    </issue>
```

```
$ ./gradlew lint
baseline.xml: Information: 14 errors and 189 warnings were filtered out because they are listed in the baseline file, baseline.xml  [LintBaseline]
0 errors, 1 warnings (14 errors, 189 warnings filtered by baseline baseline.xml)


BUILD SUCCESSFUL in 20s
49 actionable tasks: 3 executed, 46 up-to-date

$
```

```
$ ./gradlew lint
baseline.xml: Information: 14 errors and 188 warnings were filtered out because they are listed in the baseline file, baseline.xml [LintBaseline]
src/main/java/io/plaidapp/ui/DesignerNewsStory.java:826: Warning: Implicitly using the default locale is a common source of bugs: Use toLowerCase(Locale) instead. For strings meant to be internal use Locale.ROOT, otherwise Locale.getDefault(). [DefaultLocale]
                                .toString().toLowerCase());
                                            ~~~~~~~~~~~

0 errors, 2 warnings (14 errors, 188 warnings filtered by baseline baseline.xml)

FAILURE: Build failed with an exception.

$
```

Lint Features

Custom Checks

Write your own checks!

~/.android/lint/*.jar

$ANDROID_LINT_JARS

Lint Features
Custom Checks

compile 'com.android.support:appcompat-v7:27.0.0'
compile 'com.android.support:appcompat-v7:27.0.0**@aar**'

AAR (Android ARchive files) :

Code, manifest, resources, extra proguard rules, …

...and lint checks!

```
$ jar tvf ~/.gradle/caches/.../com.jakewharton.timber/timber/4.5.1/?/timber-4.5.1.aar
   216 Fri Jan 20 14:45:28 PST 2017 AndroidManifest.xml
  8533 Fri Jan 20 14:45:28 PST 2017 classes.jar
 10111 Fri Jan 20 14:45:28 PST 2017 lint.jar
    39 Fri Jan 20 14:45:28 PST 2017 proguard.txt
     0 Fri Jan 20 14:45:24 PST 2017 aidl/
     0 Fri Jan 20 14:45:28 PST 2017 assets/
     0 Fri Jan 20 14:45:28 PST 2017 jni/
     0 Fri Jan 20 14:45:28 PST 2017 res/
     0 Fri Jan 20 14:45:28 PST 2017 libs/
```

```
// build.gradle:
dependencies {
  compile 'com.jakewharton.timber:timber:4.5.1'
}
```

Lint Features

Custom Checks via lintChecks()

New packaging mechanism in 3.0

```
// Custom checks project named 'checks'
apply plugin: 'java-library'
apply plugin: 'kotlin'
dependencies {
    compileOnly "com.android.tools.lint:lint-api:$lintVersion"
    ...
}

// Usage of lint:
// If app: Analyze this project using the lint rules in the checks project.
// If library: Package the given lint checks library into this AAR .
dependencies {
    compile project(':foo')
    lintChecks project(':checks')
}
```

KotlinConf

Writing a Lint check

KotlinConf

Warning: Unofficial API!

APIs have and will change, etc etc etc.

<blink>
You've been warned!
</blink>

Write your checks in Kotlin

```
/**
 * Returns a suitable location for storing cache files. Note that the
 * directory may not exist. You can override the default location
 * using `$ANDROID_SDK_CACHE_DIR` (though note that specific
 * lint integrations may not honor that environment variable; for example,
 * in Gradle the cache directory will **always** be build/intermediates/lint-cache/.)
 *
 * @param create if true, attempt to create the cache dir if it does not
 *               exist
 *
 * @return a suitable location for storing cache files, which may be null if
 *         the create flag was false, or if for some reason the directory
 *         could not be created
 *
 */
@Deprecated("Use {@link #getCacheDir(String, boolean)} instead",
        ReplaceWith("getCacheDir(null, create)"))
open fun getCacheDir(create: Boolean): File? = getCacheDir(null, create)
```

Custom Check Project

```
apply plugin: 'java-library'
apply plugin: 'kotlin'
dependencies {
    compileOnly "com.android.tools.lint:lint-api:$lintVersion"
    compileOnly "com.android.tools.lint:lint-checks:$lintVersion"
    testCompile "junit:junit:4.12"
    testCompile "com.android.tools.lint:lint:$lintVersion"
    testCompile "com.android.tools.lint:lint-tests:$lintVersion"
    testCompile "com.android.tools:testutils:$lintVersion"
}
```

Custom Check Project

```
buildscript {
    ext {
        gradlePluginVersion = '3.1.0-alpha01'
        lintVersion =                '26.1.0-alpha01'
    }
}
```

lintVersion = gradlePluginVersion + 23.0.0

Client API vs Detector API

Lint has 2 APIs:

Client API: Integrate (and run) lint from within a tool

Detector API: Implement a new lint check

Client API: LintClient

Interface implemented by the tool (IDE, Gradle, etc.)

```
/** Report the given issue. This method will only be called if the configuration
* provided by [.getConfiguration] has reported the corresponding
* issue as enabled and has not filtered out the issue with its
* [Configuration.ignore] method.
*
* @param context  the context used by the detector when the issue was found
* @param issue    the issue that was found
* @param severity the severity of the issue
* @param location the location of the issue
* @param message  the associated user message
* @param format   the format of the description and location descriptions
* @param fix      an optional quick fix descriptor
*/
abstract fun report(
        context: Context,
        issue: Issue,
        severity: Severity,
        location: Location,
        message: String,
        format: TextFormat = TextFormat.RAW,
        fix: LintFix? = null)
```

```kotlin
/**
* Reads the given text file and returns the content as a string
*
* @param file the file to read
* @return the string to return, never null (will be empty if there is an
*         I/O error)
*/
abstract fun readFile(file: File): CharSequence

/**
* Reads the given binary file and returns the content as a byte array.
* By default this method will read the bytes from the file directly,
* but this can be customized by a client if for example I/O could be
* held in memory and not flushed to disk yet.
*
* @param file the file to read
* @return the bytes in the file, never null
*/
@Throws(IOException::class)
open fun readBytes(file: File): ByteArray = Files.toByteArray(file)
```

```kotlin
/**
* Opens a URL connection.
*
* Clients such as IDEs can override this to for example consider the user's IDE proxy
* settings.
*
* @param url     the URL to read
* @param timeout the timeout to apply for HTTP connections (or 0 to wait indefinitely)
* @return a [URLConnection] or null
* @throws IOException if any kind of IO exception occurs including timeouts
*/
@Throws(IOException::class)
open fun openConnection(url: URL, timeout: Int): URLConnection? {
```

```
// Set up exactly the expected maven.google.com network output to ensure stable
// version suggestions in the tests
.networkData("https://maven.google.com/master-index.xml", ""
        + "<?xml version='1.0' encoding='UTF-8'?>\n"
        + "<metadata>\n"
        + "  <com.android.tools.build/>"
        + "</metadata>")
.networkData("https://maven.google.com/com/android/tools/build/group-index.xml", ""
        + "<?xml version='1.0' encoding='UTF-8'?>\n"
        + "<com.android.tools.build>\n"
        + "  <gradle versions=\"2.3.3,3.0.0-alpha1\"/>\n"
        + "</com.android.tools.build>");
```

Creating a lint check

- Create an **Issue**, and return it from an **IssueRegistry**
- Implement a new **Detector** which reports the issue
- Write a test for the detector

Issue

Static metadata about a class of problems.

Combining Ellipsize and Maxlines

src/main/res/layout/account_dropdown_item.xml:26: Combining `ellipsize=marquee` and `maxLines=1` can lead to crashes. Use `singleLine=true` instead.

```
23      android:paddingEnd="@dimen/padding_normal"
24      android:paddingStart="@dimen/padding_normal"
25      android:background="@drawable/mid_grey_bounded_ripple"
26      android:maxLines="1"
27      android:ellipsize="marquee"
28      android:textAppearance="?android:attr/textAppearanceLargePopupMenu"
29      style="?android:attr/dropDownItemStyle" />
```

+ 1 Additional Locations...

Combining `ellipsize` and `maxLines=1` can lead to crashes on some devices. Earlier versions of lint recommended replacing `singleLine=true` with `maxLines=1` but that should not be done when using `ellipsize`.
More info: https://issuetracker.google.com/issues/36950033
To suppress this error, use the issue id "EllipsizeMaxLines" as explained in the Suppressing Warnings and Errors section.

EllipsizeMaxLines | Correctness | Error | Priority 4/10

DISMISS

Combining Ellipsize and Maxlines

src/main/res/layout/account_dropdown_item.xml:26: Combining `ellipsize=marquee` and `maxLines=1` can lead to crashes. Use `singleLine=true` instead.

```
23      android:paddingEnd="@dimen/padding_normal"
24      android:paddingStart="@dimen/padding_normal"
25      android:background="@drawable/mid_grey_bounded_ripple"
26      android:maxLines="1"
27      android:ellipsize="marquee"
28      android:textAppearance="?android:attr/textAppearanceLargePopupMenu"
29      style="?android:attr/dropDownItemStyle" />
```

+ 1 Additional Locations...

Combining `ellipsize` and `maxLines=1` can lead to crashes on some devices. Earlier versions of lint recommended replacing `singleLine=true` with `maxLines=1` but that should not be done when using `ellipsize`.
More info: https://issuetracker.google.com/issues/36950033
To suppress this error, use the issue id "EllipsizeMaxLines" as explained in the Suppressing Warnings and Errors section.

EllipsizeMaxLines | Correctness | Error | Priority 4/10

DISMISS

Issue ID

Unique

Typically Upper Camel Case

Used to suppress:
```
@SuppressWarnings("MyIssueId")
//noinspection MyIssueId
```

Used to configure issues in gradle:
```
android.lintOptions.error 'MyIssueId'
```

Combining Ellipsize and Maxlines

src/main/res/layout/account_dropdown_item.xml:26: Combining `ellipsize=marquee` and `maxLines=1` can lead to crashes. Use `singleLine=true` instead.

```
23      android:paddingEnd="@dimen/padding_normal"
24      android:paddingStart="@dimen/padding_normal"
25      android:background="@drawable/mid_grey_bounded_ripple"
26      android:maxLines="1"
27      android:ellipsize="marquee"
28      android:textAppearance="?android:attr/textAppearanceLargePopupMenu"
29      style="?android:attr/dropDownItemStyle" />
```

+ 1 Additional Locations...

Combining `ellipsize` and `maxLines=1` can lead to crashes on some devices. Earlier versions of lint recommended replacing `singleLine=true` with `maxLines=1` but that should not be done when using `ellipsize`.
More info: https://issuetracker.google.com/issues/36950033
To suppress this error, use the issue id "EllipsizeMaxLines" as explained in the Suppressing Warnings and Errors section.

EllipsizeMaxLines | Correctness | Error | Priority 4/10

DISMISS

Combining Ellipsize and Maxlines

src/main/res/layout/account_dropdown_item.xml:26: Combining `ellipsize=marquee` and `maxLines=1` can lead to crashes. Use `singleLine=true` instead.

```
23      android:paddingEnd="@dimen/padding_normal"
24      android:paddingStart="@dimen/padding_normal"
25      android:background="@drawable/mid_grey_bounded_ripple"
26      android:maxLines="1"
27      android:ellipsize="marquee"
28      android:textAppearance="?android:attr/textAppearanceLargePopupMenu"
29      style="?android:attr/dropDownItemStyle" />
```

+ 1 Additional Locations...

Combining `ellipsize` and `maxLines=1` can lead to crashes on some devices. Earlier versions of lint recommended replacing `singleLine=true` with `maxLines=1` but that should not be done when using `ellipsize`.
More info: https://issuetracker.google.com/issues/36950033
To suppress this error, use the issue id "EllipsizeMaxLines" as explained in the Suppressing Warnings and Errors section.

EllipsizeMaxLines | Correctness | Error | Priority 4/10

DISMISS

Combining Ellipsize and Maxlines

src/main/res/layout/account_dropdown_item.xml:26: Combining `ellipsize=marquee` and `maxLines=1` can lead to crashes. Use `singleLine=true` instead.

```
23      android:paddingEnd="@dimen/padding_normal"
24      android:paddingStart="@dimen/padding_normal"
25      android:background="@drawable/mid_grey_bounded_ripple"
26      android:maxLines="1"
27      android:ellipsize="marquee"
28      android:textAppearance="?android:attr/textAppearanceLargePopupMenu"
29      style="?android:attr/dropDownItemStyle" />
```

+ 1 Additional Locations...

Combining `ellipsize` and `maxLines=1` can lead to crashes on some devices. Earlier versions of lint recommended replacing `singleLine=true` with `maxLines=1` but that should not be done when using `ellipsize`.
More info: https://issuetracker.google.com/issues/36950033
To suppress this error, use the issue id "EllipsizeMaxLines" as explained in the Suppressing Warnings and Errors section.

EllipsizeMaxLines | Correctness | Error | Priority 4/10

DISMISS

Combining Ellipsize and Maxlines

src/main/res/layout/account_dropdown_item.xml:26: Combining `ellipsize=marquee` and `maxLines=1` can lead to crashes. Use `singleLine=true` instead.

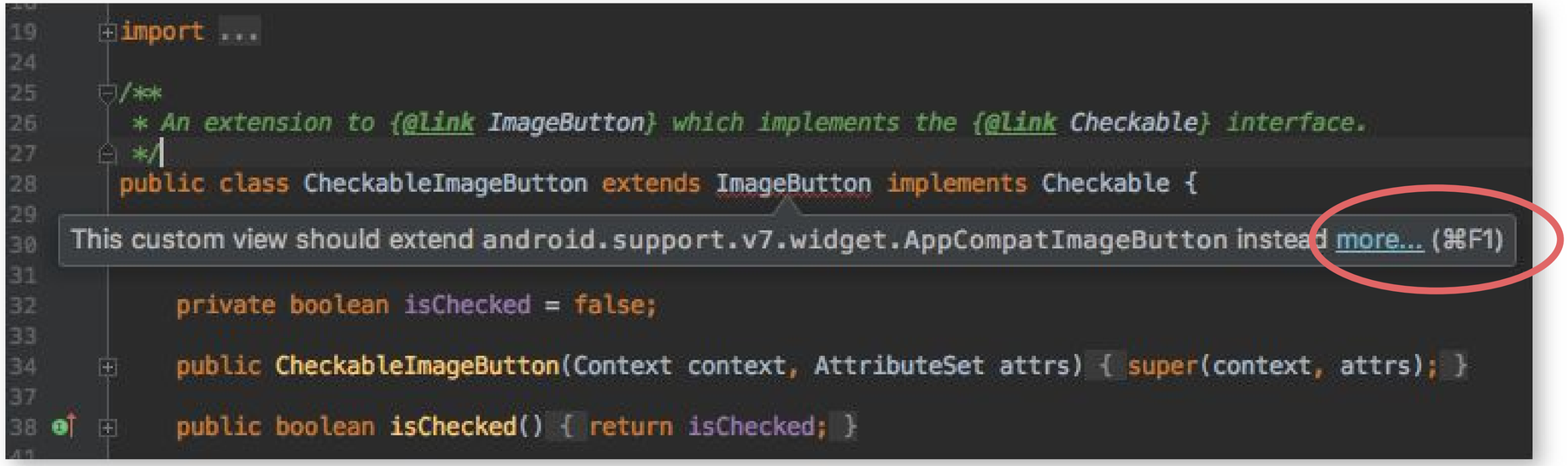
```
23      android:paddingEnd="@dimen/padding_normal"
24      android:paddingStart="@dimen/padding_normal"
25      android:background="@drawable/mid_grey_bounded_ripple"
26      android:maxLines="1"
27      android:ellipsize="marquee"
28      android:textAppearance="?android:attr/textAppearanceLargePopupMenu"
29      style="?android:attr/dropDownItemStyle" />
```

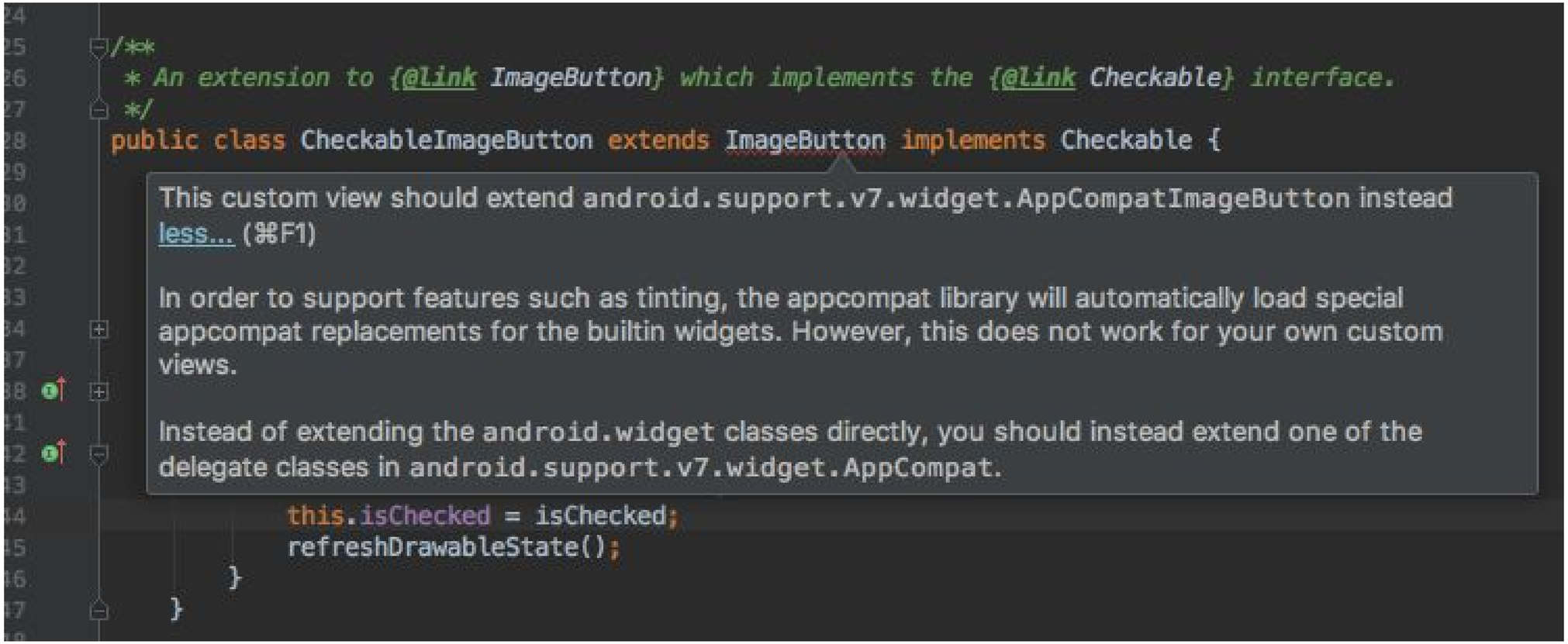
+ 1 Additional Locations...

Combining `ellipsize` and `maxLines=1` can lead to crashes on some devices. Earlier versions of lint recommended replacing `singleLine=true` with `maxLines=1` but that should not be done when using `ellipsize`.
More info: https://issuetracker.google.com/issues/36950033
To suppress this error, use the issue id "EllipsizeMaxLines" as explained in the Suppressing Warnings and Errors section.

EllipsizeMaxLines | Correctness | Error | Priority 4/10

DISMISS


import ...
/**
* An extension to {@link ImageButton} which implements the {@link Checkable} interface.
*/
public class CheckableImageButton extends ImageButton implements Checkable {
This custom view should extend android.support.v7.widget.AppCompatImageButton instead more... (⌘F1)
private boolean isChecked = false;
public CheckableImageButton(Context context, AttributeSet attrs) { super(context, attrs); }
public boolean isChecked() { return isChecked; }


/**
* An extension to {@link ImageButton} which implements the {@link Checkable} interface.
*/
public class CheckableImageButton extends ImageButton implements Checkable {
This custom view should extend android.support.v7.widget.AppCompatImageButton instead
less... (⌘F1)
In order to support features such as tinting, the appcompat library will automatically load special appcompat replacements for the builtin widgets. However, this does not work for your own custom views.
Instead of extending the android.widget classes directly, you should instead extend one of the delegate classes in android.support.v7.widget.AppCompat.
this.isChecked = isChecked;
refreshDrawableState();
}
}

Text Format

This is a `code symbol` → This is a `code symbol`

This is *italics* → This is *italics*

This is **bold** → This is **bold**

http://, https:// → http://, https://

not italics → *not italics*

Combining Ellipsize and Maxlines

src/main/res/layout/account_dropdown_item.xml:26: Combining `ellipsize=marquee` and `maxLines=1` can lead to crashes. Use `singleLine=true` instead.

```
23      android:paddingEnd="@dimen/padding_normal"
24      android:paddingStart="@dimen/padding_normal"
25      android:background="@drawable/mid_grey_bounded_ripple"
26      android:maxLines="1"
27      android:ellipsize="marquee"
28      android:textAppearance="?android:attr/textAppearanceLargePopupMenu"
29      style="?android:attr/dropDownItemStyle" />
```

+ 1 Additional Locations...

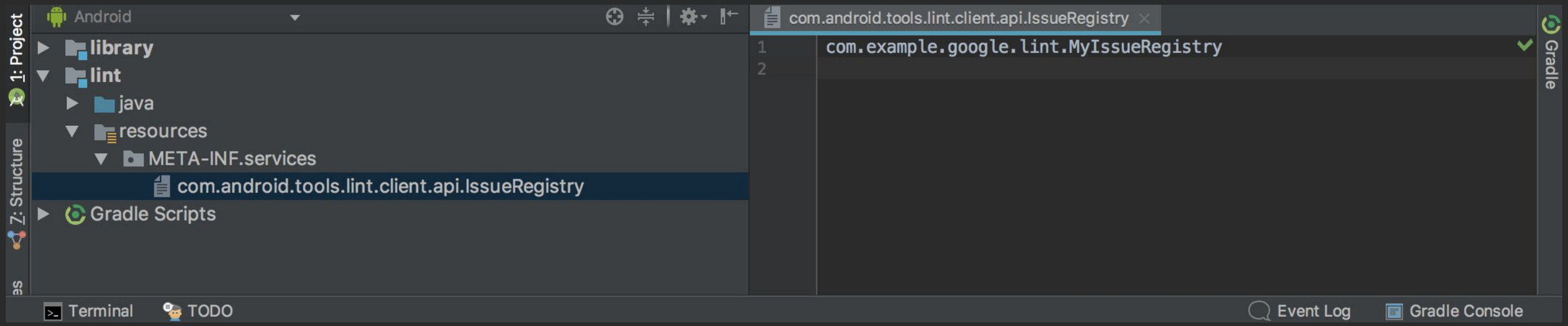
Combining `ellipsize` and `maxLines=1` can lead to crashes on some devices. Earlier versions of lint recommended replacing `singleLine=true` with `maxLines=1` but that should not be done when using `ellipsize`.
More info: https://issuetracker.google.com/issues/36950033
To suppress this error, use the issue id "EllipsizeMaxLines" as explained in the Suppressing Warnings and Errors section.

EllipsizeMaxLines | Correctness | Error | Priority 4/10

DISMISS

```
val ISSUE = Issue.create(
        "MyId",
        "Short title for my issue",

        """
        This is a longer explanation of the issue.
        Many paragraphs here, with links, **emphasis**, etc.
        """.trimIndent(),

        Category.CORRECTNESS,
        2,
        Severity.ERROR,
        Implementation(
                MyDetector::class.java,
                Scope.MANIFEST_SCOPE))
        .addMoreInfo("https://issuetracker.google.com/12345")
```

```kotlin
import com.android.tools.lint.client.api.IssueRegistry

class MyIssueRegistry : IssueRegistry() {
  override fun getIssues() = listOf(ISSUE)
}
```



```
val ISSUE = Issue.create(
        "MyId",
        "Short title for my issue",

        """
        This is a longer explanation of the issue.
        Many paragraphs here, with links, **emphasis**, etc.
        """.trimIndent(),

        Category.CORRECTNESS,
        2,
        Severity.ERROR,
        Implementation(
                MyDetector::class.java,
                Scope.MANIFEST_SCOPE))
        .addMoreInfo("https://issuetracker.google.com/12345")
```

Detector

Responsible for detecting occurrences of an issue in the source code

Detector class registered via an Issue

Multiple issues can register the same detector

```
class MyDetector : Detector() {
    override fun run(context: Context) {
        context.report(ISSUE, Location.create(context.file), "I complain a lot")
    }
}
```

Detector Interfaces

There are a number of Detector specializations:

- **XmlScanner** - XML files (visit with DOM)
- **UastScanner** - Java and Kotlin files (visit with UAST)
- **ClassScanner** - .class files (bytecode, visit with ASM)
- **BinaryResourceScanner** - binaries like images
- **ResourceFolderScanner** - android res folders
- **GradleScanner** - Gradle build scripts
- **OtherFileScanner** - Other files

XmlScanner

- getApplicableElements(): List<String>
- visitElement(element: org.w3c.dom.Element)

- getApplicableAttributes(): List<String>
- visitAttribute(attribute: org.w3c.dom.Attr)

- visitDocument(org.w3c.dom.Document)

(XmlScanner.ALL: Visit all attributes or all elements)

```
import com.android.tools.lint.detector.api.Detector
import com.android.tools.lint.detector.api.Detector.XmlScanner
import com.android.tools.lint.detector.api.Location
import com.android.tools.lint.detector.api.XmlContext
import org.w3c.dom.Element


class MyDetector : Detector(), XmlScanner {
    override fun getApplicableElements() = listOf("placeholder")

    override fun visitElement(context: XmlContext, element: Element) {
        context.report(ISSUE, context.getLocation(element), "I complain a lot")
    }
}
```

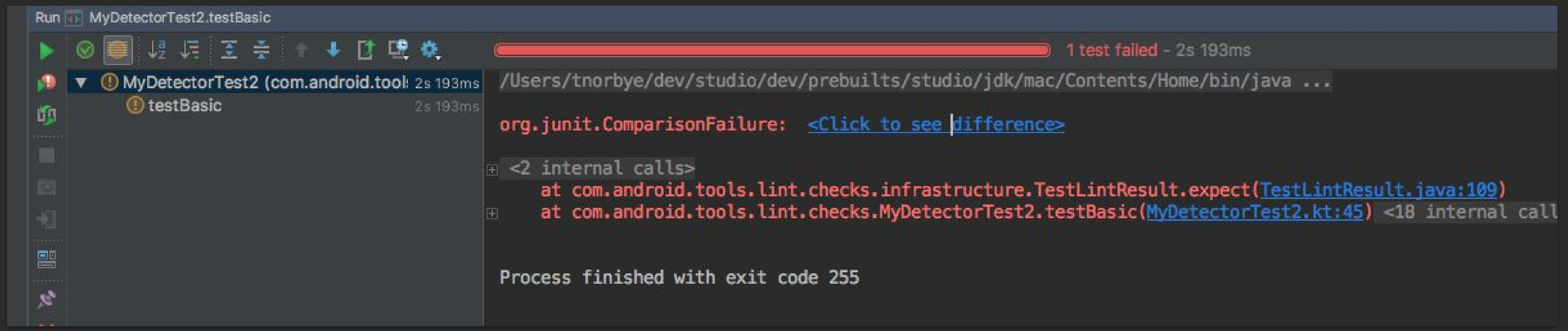
```
import com.android.tools.lint.checks.infrastructure.LintDetectorTest
import com.android.tools.lint.checks.infrastructure.LintDetectorTest.*
import com.android.tools.lint.checks.infrastructure.TestLintTask.*
import org.junit.Test

class MyDetectorTest {
    @Test
    fun `Check basic scenario`() {
        lint().files(
                manifest("""
                    <manifest xmlns:android="http://schemas.android.com/apk/res/android">
                        <placeholder android:targetSdkVersion="23" />
                    </manifest>
                """).indented())
                .issues(ISSUE)
                .run()
                .expect("")
    }
}
```

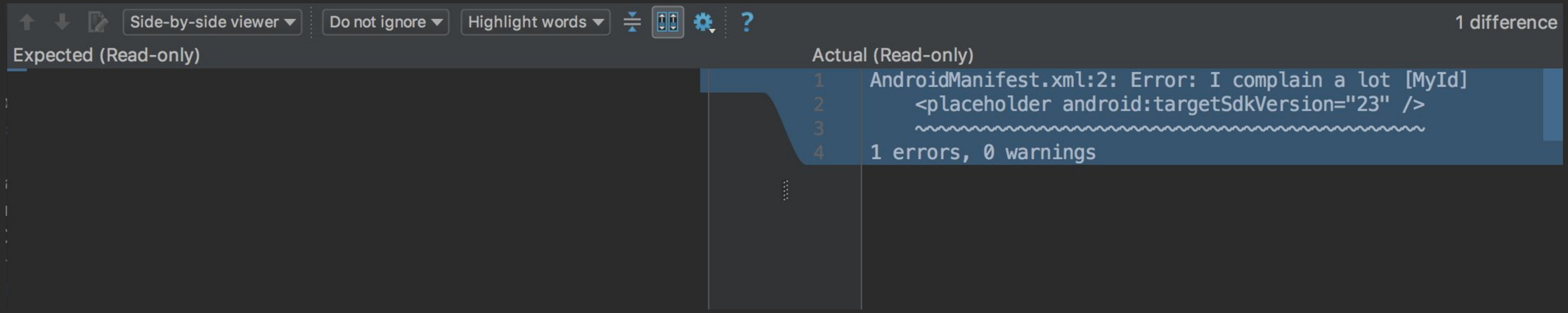
```
class MyDetectorTest {
    @Test
    fun `Check basic scenario`() {
        lint().files(
                manifest("""
                    <manifest xmlns:android="http://schemas.android.com/apk/res/android">
                        <placeholder android:targetSdkVersion="23" />
                    </manifest>
                """).indented())
                .issues(ISSUE)
                .run()
                .expect("")
    }
}
```

```
    }

    public void testKotlinPropertySyntax() {
        lint().files(
                manifest().minSdk(1),
                kotlin("package test.pkg\n" +
                        //"@Suppress(\"UsePropertyAccessSyntax\")\n" +
                        "fun testApiCheck(calendar: java.util.Calendar) {\n" +
                        "    calendar.weekYear\n" +
                        "    calendar.getWeekYear()\n" +
                        "}    \n"))
                .run()
                .expect("src/test/pkg/test.kt:4: Error: Call requires API level 24 (c
                        "    calendar.weekYear\n" +
                        "             ~~~~~~~~\n" +
                        "src/test/pkg/test.kt:5: Error: Call requires API level 24 (c
                        "    calendar.getWeekYear()\n" +
                        "             ~~~~~~~~~~~\n" +
                        "2 errors, 0 warnings\n");
    }
```

Use property access syntax more... (⌘F1)

```
import com.android.tools.lint.checks.infrastructure.LintDetectorTest
import com.android.tools.lint.checks.infrastructure.LintDetectorTest.*
import com.android.tools.lint.checks.infrastructure.TestLintTask.*
import org.junit.Test

class MyDetectorTest {
    @Test
    fun `Check basic scenario`() {
        lint().files(
                manifest("""
                    <manifest xmlns:android="http://schemas.android.com/apk/res/android"
                        package="test.pkg.library" >
                        <placeholder android:targetSdkVersion="23" />
                    </manifest>
                """).indented())
                .issues(ISSUE)
                .run()
                .expect("")
    }
}
```


Run MyDetectorTest2.testBasic
1 test failed - 2s 193ms
MyDetectorTest2 (com.android.tool 2s 193ms
testBasic 2s 193ms
/Users/tnorbye/dev/studio/dev/prebuilts/studio/jdk/mac/Contents/Home/bin/java ...
org.junit.ComparisonFailure: <Click to see difference>
<2 internal calls>
at com.android.tools.lint.checks.infrastructure.TestLintResult.expect(TestLintResult.java:109)
at com.android.tools.lint.checks.MyDetectorTest2.testBasic(MyDetectorTest2.kt:45) <18 internal call
Process finished with exit code 255


Side-by-side viewer
Do not ignore
Highlight words
1 difference
Expected (Read-only)
Actual (Read-only)
1 AndroidManifest.xml:2: Error: I complain a lot [MyId]
2 <placeholder android:targetSdkVersion="23" />
3 ~~~~~~
4 1 errors, 0 warnings

```
class MyDetectorTest {
    @Test
    fun `Check basic scenario`() {
        lint().files(
            manifest("""
            <manifest xmlns:android="http://schemas.android.com/apk/res/android">
                <placeholder android:targetSdkVersion="23" />
            </manifest>
            """).indented())
            .issues(ISSUE)
            .run()
            .expect("""
            AndroidManifest.xml:2: Error: I complain a lot [MyId]
                <placeholder android:targetSdkVersion="23" />
                ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
            1 errors, 0 warnings""")
    }
}
```

```
class MyDetectorTest {
    @Test
    fun `Check basic scenario`() {
        lint().files(
            manifest("""
            <manifest xmlns:android="http://schemas.android.com/apk/res/android">
                <placeholder android:targetSdkVersion="23" />
            </manifest>
            """).indented())
            .issues(ISSUE)
            .run()
            .expectWarningCount(0) // Avoid
            .expectErrorCount(1)
            .check { it.contains("Warning 2") }
    }
}
```

```
    override fun visitElement(context: XmlContext, element: Element) {
        context.report(ISSUE, context.getLocation(element), "I complain a lot")
        context.report(ISSUE, context.getNameLocation(element), "I complain a lot")
    }

    override fun visitAttribute(context: XmlContext, attribute: Attr) {
        context.report(ISSUE, context.getLocation(attribute), "Warning 1")
        context.report(ISSUE, context.getNameLocation(attribute), "Warning 2")
        context.report(ISSUE, context.getValueLocation(attribute), "Warning 3")
    }
}
```

```
class MyDetectorTest {
    @Test
    fun `Check basic scenario`() {
        lint().files(
            manifest("""
            <manifest xmlns:android="http://schemas.android.com/apk/res/android">
                <placeholderandroid:targetSdkVersion="23" />
            </manifest>
            """).indented())
            .issues(ISSUE)
            .run()
            .expect("""
            AndroidManifest.xml:2: Error: I complain a lot [MyId]
                <placeholder android:targetSdkVersion="23" />
                ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
            AndroidManifest.xml:2: Error: I complain a lot [MyId]
                <placeholder android:targetSdkVersion="23" />
                 ~~~~~~~~~~~~
            2 errors, 0 warnings
            """)
    }
}
```

Locations

File, start and end positions

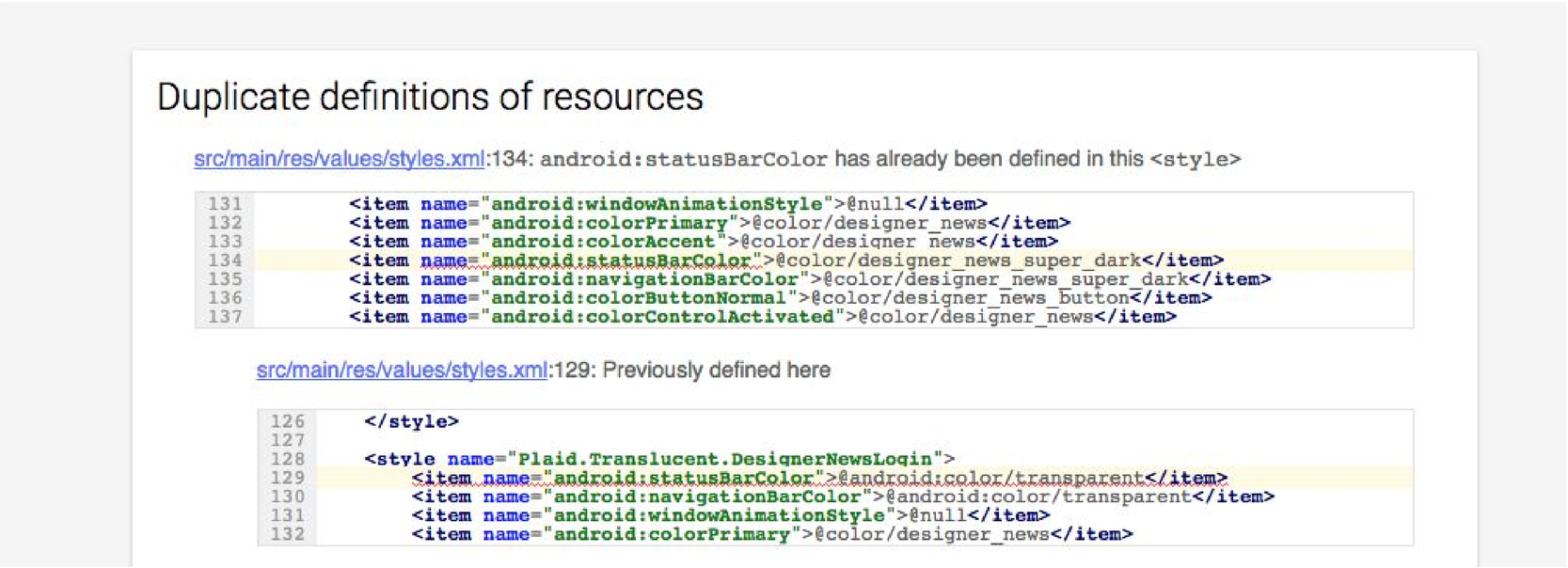
Typically created from "AST" nodes

Can be linked

Can be described

Can create location range between nodes +/- delta

Linked Locations




```
Location secondary = context.getLocation(previous);
secondary.setMessage(“Previously defined here");
location.setSecondary(secondary);
```

DesignerNewsStory.java:

```
final TextView title = (TextView) findViewById(R.id.story_title);
title.setText(story.title);
```

designer_news_story_item.xml

```
android {
<io.plaidapp.ui.widget.BaselineGridTextView
    android:id="@+id/story_title"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_marginStart="@dimen/padding_normal"
    android:layout_marginTop="@dimen/padding_normal"
```

DesignerNewsStory.java: UastScanner

MyDetector

```
final TextView title = (TextView) findViewById(R.id.story_title);
title.setText(story.title);
```

designer_news_story_item.xml: XmlScanner

```
android {
<io.plaidapp.ui.widget.BaselineGridTextView
    android:id="@+id/story_title"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_marginStart="@dimen/padding_normal"
    android:layout_marginTop="@dimen/padding_normal"
```

Detector Lifecycle

Issue registers Detector class, not Detector instance

New detector instantitated for each analysis run

You can stash data in the detector instance.

Predefined Iteration Order

- Manifest
- Android resources (alphabetical by folder type)
- Java & Kotlin
- Bytecode (.class files)
- Gradle files
- ProGuard files
- Property Files
- Other files

Detectors are invoked based on Issue scope registration.

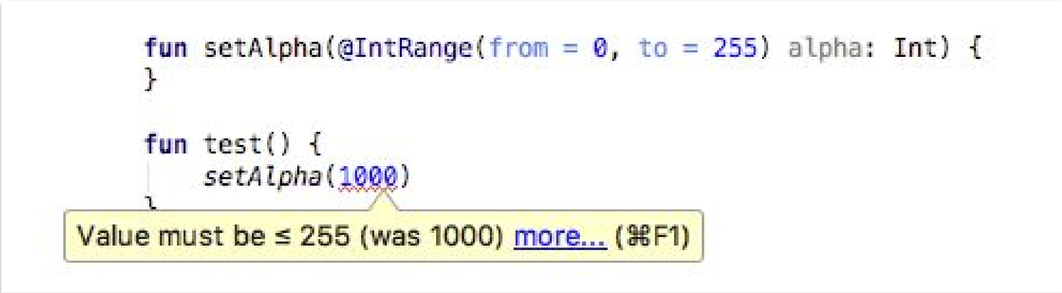
Multipass Analysis

If you want to process the sources more than once:

context.driver.requestRepeat(this, …)

```
override fun afterCheckProject(context: Context) {
    if (context.phase == 1 && haveUnusedResources()) {
        // Request another scan through the resources such that we can
        // gather the actual locations
        context.driver.requestRepeat(this, Scope.ALL_RESOURCES_SCOPE);
    }
}

override fun visitElement(context: XmlContext, element: Element) {
    int phase = context.phase

    Attr attribute = element.getAttributeNode(ATTR_NAME);
    if (attribute == null || attribute.getValue().isEmpty()) {
        if (phase == 2) {
            ....
```

Scopes: On the fly analysis



Checks run on the fly if analysis runs on a single file

Determined by issues scopes, not Detector interfaces!

```
public static final Issue ISSUE = Issue.create(
        "HardcodedText",
        "Hardcoded text",
        "Hardcoding text attributes directly in layout files is bad for several reasons:\n" +
        "* <description omitted on this slide>",
        Category.I18N, 5, Severity.WARNING,
        new Implementation(
                HardcodedValuesDetector.class,
                Scope.RESOURCE_FILE_SCOPE));

public static final Issue UNUSED_ISSUE = Issue.create(
        "UnusedResources",
        "Unused resources",
        "Unused resources make applications larger and slow down builds.",
        Category.PERFORMANCE, 3, Severity.WARNING,
        new Implementation(
                UnusedResourceDetector.class,
                EnumSet.of(Scope.MANIFEST, Scope.ALL_RESOURCE_FILES, Scope.ALL_JAVA_FILES,
                        Scope.BINARY_RESOURCE_FILE, Scope.TEST_SOURCES)));
```

Single file: On the fly possible

Multiple file scopes: Only batch mode

Scopes: On the fly analysis

Separate issues in detector can allow single file analysis

```
public Implementation(
        @NonNull Class<? extends Detector> detectorClass,
        @NonNull EnumSet<Scope> scope,
        @NonNull EnumSet<Scope>... analysisScopes) {

// ApiDetector.UNSUPPORTED issue registration
new Implementation(
    ApiDetector.class,
    EnumSet.of(Scope.JAVA_FILE, Scope.RESOURCE_FILE, Scope.MANIFEST),
    Scope.JAVA_FILE_SCOPE,
    Scope.RESOURCE_FILE_SCOPE,
    Scope.MANIFEST_SCOPE));
```

```
lint().files(
        xml("src/main/res/drawable/foo.xml", VECTOR),
        xml("src/main/res/layout/main_activity.xml", LAYOUT_SRC),
        gradle(""
                + "buildscript {\n"
                + "    dependencies {\n"
                + "        classpath 'com.android.tools.build:gradle:2.0.0'\n"
                + "    }\n"
                + "}\n"
                + "android.defaultConfig.vectorDrawables.useSupportLibrary = true\n"))
        .incremental("src/main/res/layout/main_activity.xml")
        .run()
        .expect(""
                + "src/main/res/layout/main_activity.xml:3: Error: When using VectorDrawableCompat, "
                + "you need to use app:srcCompat. [VectorDrawableCompat]\n"
                + "    <ImageView android:src=\"@drawable/foo\" />\n"
                + "               ~~~~~~~~~~~\n"
                + "1 errors, 0 warnings\n");
```

~~JavaScanner~~

Callback for Java sources

```
@Deprecated("Use UastScanner instead", ReplaceWith("UastScanner"))
class JavaScanner { … }
```

UAST

Universal Abstract Syntax Tree

Created by JetBrains

Describes superset of Java and Kotlin

Allows single analysis covering both

UAST

Hierarchy

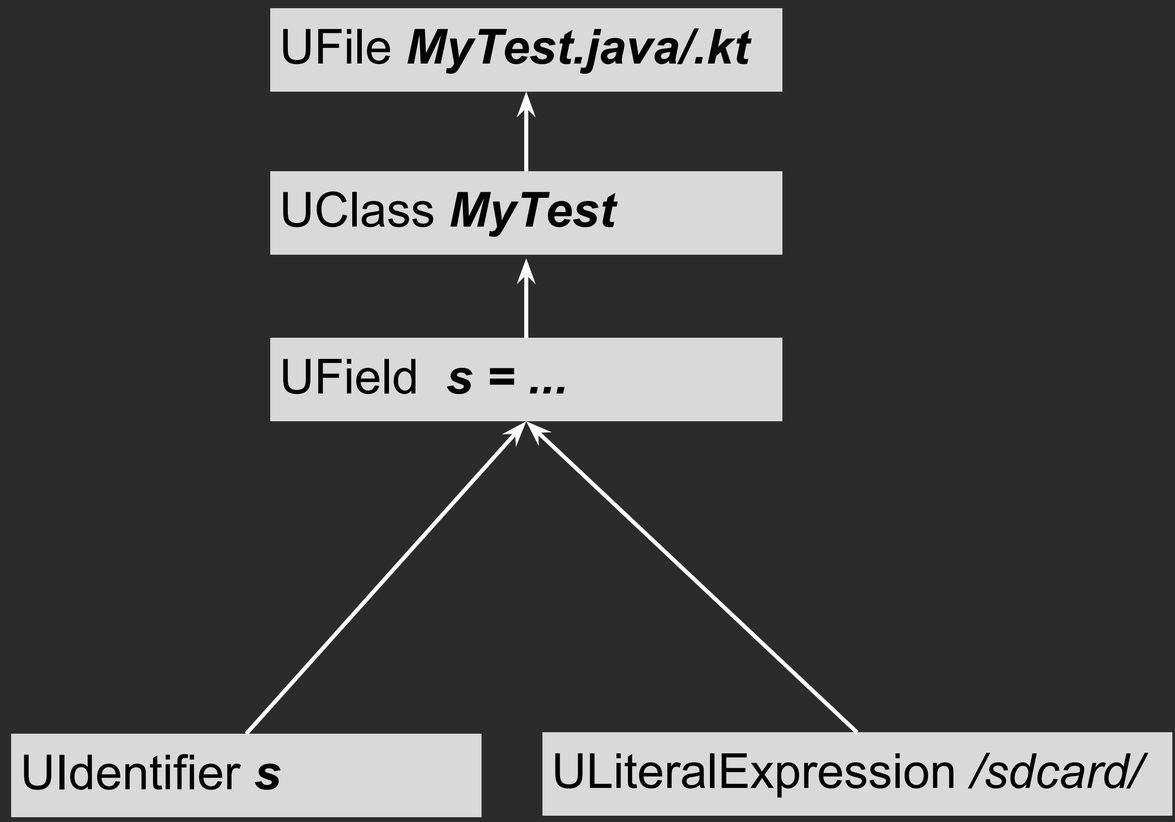
- UElement: Root of everything
- UFile: Compilation unit
- UClass: A class declaration
- UMember: A member such as a method or field
- UField: A field declaration
- UMethod: A method declaration

UAST
Hierarchy

- UComment
- UDeclaration
- UExpression
- UBlockExpression
- UCallExpression
- USwitchExpression
- ULoopExpression (UForEachExpression, UDoWhile...,)
- UReturnExpression
- ...

```java
// MyTest.java
package test.pkg;

public class MyTest {
   String s = "/sdcard/mydir";
}
```

```kotlin
// MyTest.kt
package test.pkg

class MyTest {
   val s: String = "/sdcard/mydir"
}
```



```
lint().files(
        kotlin(""
                + "package test.pkg\n"
                + "\n"
                + "class MyTest {\n"
                + "    val s: String = \"/sdcard/mydir\"\n"
                + "}\n"), ...

override fun createUastHandler(context: JavaContext): UElementHandler? {
   println(context.uastFile?.asRecursiveLogString())


UFile (package = test.pkg)
    UClass (name = MyTest)
        UField (name = s)
            UAnnotation (fqName = org.jetbrains.annotations.NotNull)
            ULiteralExpression (value = "/sdcard/mydir")
        UAnnotationMethod (name = getS)
        UAnnotationMethod (name = MyTest)
```

UAST
Resolving

References and calls can be resolved; for this call:

```
label.setText("myText")
```

```
val call: UCallExpression = ...
val resolved = call.resolve()
```

Returns the method, or field, or parameter, etc.

You can then look inside method, or at field initializer etc.

UAST

Resolving returns PSI!

```
val call: UCallExpression = ...
val resolved: PsiElement? = call.resolve()
```

UAST versus PSI

PSI: Program Structure Interface

Used in IntelliJ to model Java code, Groovy, XML, properties etc.

Many UElements also implement PSI interfaces:

```
interface UClass : UDeclaration, PsiClass {
interface UMethod : UDeclaration, PsiMethod {
interface UField : UVariable, PsiField {
interface UParameter : UVariable, PsiParameter {
```

PSI exterior, UAST interior

Use PSI outside methods. Use UAST inside methods.

When you resolve, you're in PSI space.

Do **not** call psiMethod.getBody() or psiField.getInitializer().
Use UastContext.getMethod(psiMethod),
getVariable(psiField), etc. For PsiAnnotation, for now use
JavaUAnnotation.wrap

```
fun analyze(file: UFile) {
    file.accept(object : AbstractUastVisitor() {
        override fun visitClass(node: UClass): Boolean {
            if (node.uastSuperTypes.any { it.getQualifiedName() == "android.view.View" }) {
                node.accept(object : AbstractUastVisitor() {
                    override fun visitMethod(node: UMethod): Boolean {
                        if (node.isConstructor && node.uastParameters.size == 2) {
                            // do something
                        }
                        return super.visitMethod(node)
                    }
                })
            }
            return super.visitClass(node)
        }
    })
}
```

UastScanner

Convenience callbacks to

- Check any calls to a method of a given name
- Check any instantiations of a given class
- Check any symbol reference of a given name
- Check any subclass declaration from super class names
- Check any Android resource reference
- Visit annotation usages for a given set of annotations
- Visit any AST nodes by type

```
// Sample code
public void test(AlarmManager alarmManager) {
   alarmManager.setRepeating(AlarmManager.ELAPSED_REALTIME, 100, 0, null);
```

```
// Detector code
class AlarmDetector : Detector(), Detector.UastScanner {

  override fun getApplicableMethodNames(): List<String>? = listOf("setRepeating")

  override fun visitMethod(context: JavaContext, node: UCallExpression, method: PsiMethod) {
     val evaluator = context.evaluator
      if (evaluator.isMemberInClass(method, "android.app.AlarmManager") &&
              evaluator.getParameterCount(method) == 4) {
         ...
```

```
// Sample code
public void test(AlarmManager alarmManager) {
   alarmManager.setRepeating(AlarmManager.ELAPSED_REALTIME, 100, 0, null);



// Detector code
class AlarmDetector : Detector(), Detector.UastScanner {

  override fun getApplicableMethodNames(): List<String>? = listOf("setRepeating")

  override fun visitMethod(context: JavaContext, node: UCallExpression, method: PsiMethod) {
     val evaluator = context.evaluator
      if (evaluator.isMemberInClass(method, "android.app.AlarmManager") &&
              evaluator.getParameterCount(method) == 4) {
         ...
```

Single Tree Iteration

Internally, lint builds up lookup tables:

Method names:
["setRepeating", {AlarmDetector}],
["setHostnameVerifier", {AllowAllHostnameDetector}],
["findViewById", {CutPasteDetector, ViewTypeDetector}],

Super types:
["android.app.Fragment", {FragmentDetector}],
["android.app.Activity", {OnClickDetector,
RegistrationDtor}

```
// Sample code
public static final int MY_DELAY = 200;
public void test(AlarmManager alarmManager) {
  alarmManager.setRepeating(AlarmManager.ELAPSED_REALTIME, MY_DELAY, 0, null);

// Detector code
class AlarmDetector : Detector(), Detector.UastScanner {

override fun visitMethod(context: JavaContext, node: UCallExpression,
                         method: PsiMethod) {
  val evaluator = context.evaluator
  if (evaluator.isMemberInClass(method, "android.app.AlarmManager") &&
          evaluator.getParameterCount(method) == 4) {
      val argument = node.valueArguments[1]
      val value = ConstantEvaluator.evaluate(context, argument)
      if (value is Number && value.toLong() < 5000L) {
          val message = "Value will be forced up to 5000 as of Android 5.1; " +
                  "don't rely on this to be exact"
          context.report(ISSUE, argument, context.getLocation(argument), message)
      }
  }
}
```

```
// Sample code
public static final int MY_DELAY = 200;
public void test(AlarmManager alarmManager) {
    alarmManager.setRepeating(AlarmManager.ELAPSED_REALTIME, MY_DELAY, 0, null);

// Detector code
class AlarmDetector : Detector(), Detector.UastScanner {

override fun visitMethod(context: JavaContext, node: UCallExpression,
                         method: PsiMethod) {
    val evaluator = context.evaluator
    if (evaluator.isMemberInClass(method, "android.app.AlarmManager") &&
            evaluator.getParameterCount(method) == 4) {
        val argument = node.valueArguments[1]
        val value = ConstantEvaluator.evaluate(context, argument)
        if (value is Number && value.toLong() < 5000L) {
            val message = "Value will be forced up to 5000 as of Android 5.1; " +
                    "don't rely on this to be exact"
            context.report(ISSUE, argument, context.getLocation(argument), message)
        }
    }
}
```

Important Helpers

JavaEvaluator

ConstantEvaluator

TypeEvaluator

ResourceEvaluator

Utils: LintUtils, SdkUtils, UastLintUtils, XmlUtils

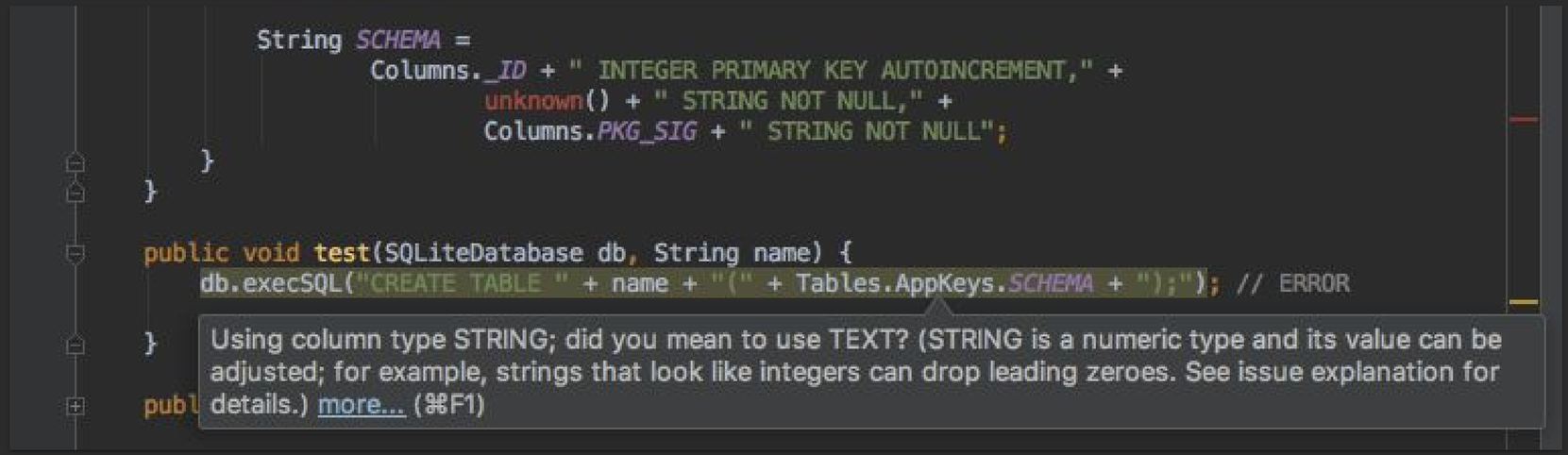
Java Evaluator

- Does class extend (even indirectly) some other class?
- Does class implement some interface?
- Is a given method a member (even indirectly) of a class?
- Does a method match these parameter types?
- Is the given method public/protected/final/static/etc?
- Find the super method of the given method
- Find the PsiClass for a given qualified name string
- Get the erasure of the given type (List<String> to List)
- Find the package containing the given element
- Compute argument mapping **...and more!**

Constant Evaluator

- Given an element, compute the constant
  - Looks up field constants
  - Combines them (+,-,!,etc)
- Optionally willing to use non-final initial field values
- Optionally willing to drop unknown values

Note that UAST has an evaluator framework; lint may soon use it.




```
// Try to resolve the String and look for STRING keys
UExpression argument = call.getValueArguments().get(0);
String sql = ConstantEvaluator.evaluateString(context, argument, true);
if (sql != null && (sql.startsWith("CREATE TABLE") || sql.startsWith("ALTER TABLE"))
        && sql.matches(".*\\bSTRING\\b.*")) {
    String message = "Using column type STRING; did you mean to use TEXT? "
            + "(STRING is a numeric type and its value can be adjusted; for example, "
            + "strings that look like integers can drop leading zeroes. See issue "
            + "explanation for details.)";
    context.report(ISSUE, call, context.getLocation(call), message);
}
```

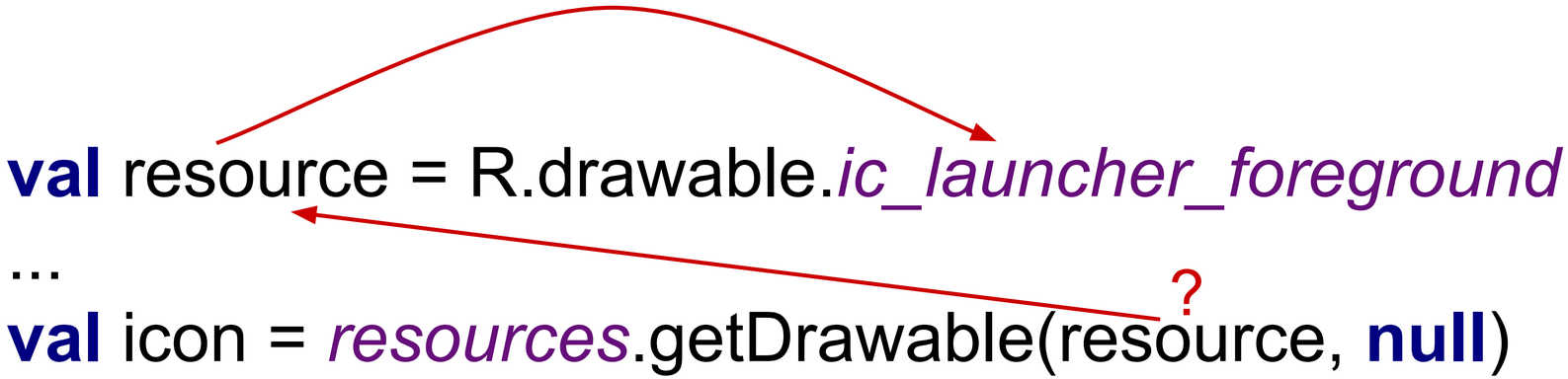
Type Evaluator

Given an element, try to guess the concrete type:

```
var view: Activity? = null
...
view = ListActivity()
...
println(view) // What is the type of view here?
```

Resource Evaluator

Given an element, figure out the resource that the element is referring to:



LintUtils

- editDistance(s, t): # of edits required to turn s into t (useful for "abcdf: Did you mean abcde?")
- isDataBindingExpression(s), isManifestPlaceholder(s)
- isReferenceMatch ("@+id/foo" == "@id/foo")
- etc

Others: SdkUtils, UastLintUtils, XmlUtils

Annotation Support

Specify applicable annotations

Callback for each annotation usage

- Method: Checks each exit point
- Parameter: Checks each variable reference and call argument
- Variable: Checks each reference and initialization

Includes annotations on member, class, package

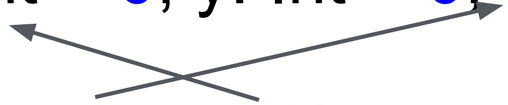
(Requires 3.1.)

```
class CheckResultDetector : AbstractAnnotationDetector(), Detector.UastScanner {


override fun applicableAnnotations(): List<String> = listOf(
        "android.support.annotation.CheckResult",
        "edu.umd.cs.findbugs.annotations.CheckReturnValue",       // findbugs
        "javax.annotation.CheckReturnValue",                      // JSR 305
        "com.google.common.annotations.CanIgnoreReturnValue"// errorprone
)
```

```
override fun visitAnnotationUsage(
        context: JavaContext,
        element: UElement,
        annotation: UAnnotation,
        qualifiedName: String,
        method: PsiMethod?,
        annotations: MutableList<UAnnotation>,
        allMemberAnnotations: MutableList<UAnnotation>,
        allClassAnnotations: MutableList<UAnnotation>,
        allPackageAnnotations: MutableList<UAnnotation>) {

    val expression = element.getParentOfType<UExpression>(
            UExpression::class.java, false) ?: return

    if (isExpressionValueUnused(expression)) {
        val message = String.format("The result of `%1\$s` is not used",
                getMethodName(expression))
        val location = context.getLocation(expression)
        report(context, CHECK_RESULT, expression, location, message)
    }
}
```

Kotlin Argument Mapping

```
fun adjust(x: Int = 0, y: Int = 0, w: Int = 0, h: Int = 0)

        adjust(w = 50, x = 0)
```



Automatically handled for annotations

```
override fun computeArgumentMapping(
   call: UCallExpression,
   method: PsiMethod)
   : Map<UExpression, PsiParameter> { … }
```

(Requires 3.1.)

Quickfixes

Automatic fix action in the IDE

LintFix: A simple **descriptor** of action to address the problem

Limited facility; more complex operations implemented directly in IDE.

More powerful support planned.

```
String name = method.getName();
String replace = null;
if (GET_ACTION_BAR.equals(name)) {
    replace = "getSupportActionBar";
} else if (START_ACTION_MODE.equals(name)) {
    replace = "startSupportActionMode";
} else if (SET_PROGRESS_BAR_VIS.equals(name)) {
    replace = "setSupportProgressBarVisibility";
} else if (REQUEST_WINDOW_FEATURE.equals(name)) {
    replace = "supportRequestWindowFeature";
}

if (replace != null) {
    String message = String.format("Should use `%1$s` instead of `%2$s` name", replace, name);
    LintFix fix = fix().name("Replace with " + replace + "()").replace()
            .text(name).with(replace).build();
    context.report(ISSUE, node, context.getLocation(node), message, fix);
}
```

```
.run()
.expect("""
src/test/pkg/AppCompatTest.java:5: Warning: Should use getSupportActionBar instead of getActionBar name
getActionBar();
~~~~~~~~~~
src/test/pkg/AppCompatTest.java:8: Warning: Should use startSupportActionMode instead of startActionMode
        startActionMode(null);
        ~~~~~~~~~~~~~~
0 errors, 2 warnings
""")
.expectFixDiffs("""
Fix for src/test/pkg/AppCompatTest.java line 4: Replace with getSupportActionBar():
@@ -5 +5
-         getActionBar();
+         getSupportActionBar();
Fix for src/test/pkg/AppCompatTest.java line 7: Replace with startSupportActionMode():
@@ -8 +8
-         startActionMode(null);
+         startSupportActionMode(null);
""");
```

Quickfixes

- fix().name("Quickfix description")

- composite(LintFix…)
- group(LintFix…)

- set(namespace, attribute, value)
- unset(namespace, attribute, value)

Quickfixes - String Replacements

- fix().replace():
  - .**all**() or .**text**(old: String)) or .**pattern**(pattern: String)
  - .**with**(replacement: String)
  - .**range**(location: Location)
  - .**shortenNames**()
  - .**reformat**()

Pro Tip:
Back references with \k<*n*>

```
LintFix fix = fix().replace()
        .name("Add cast")
        .text("findViewById")
        .shortenNames()
        .reformat(true)
        .with("(android.view.View)findViewById").build();

context.report(ADD_CAST, context.getLocation(findViewByIdCall),
        "Add explicit cast here; won't compile with language level 1.8 "
                + "without it", fix);
```

Write-UAST

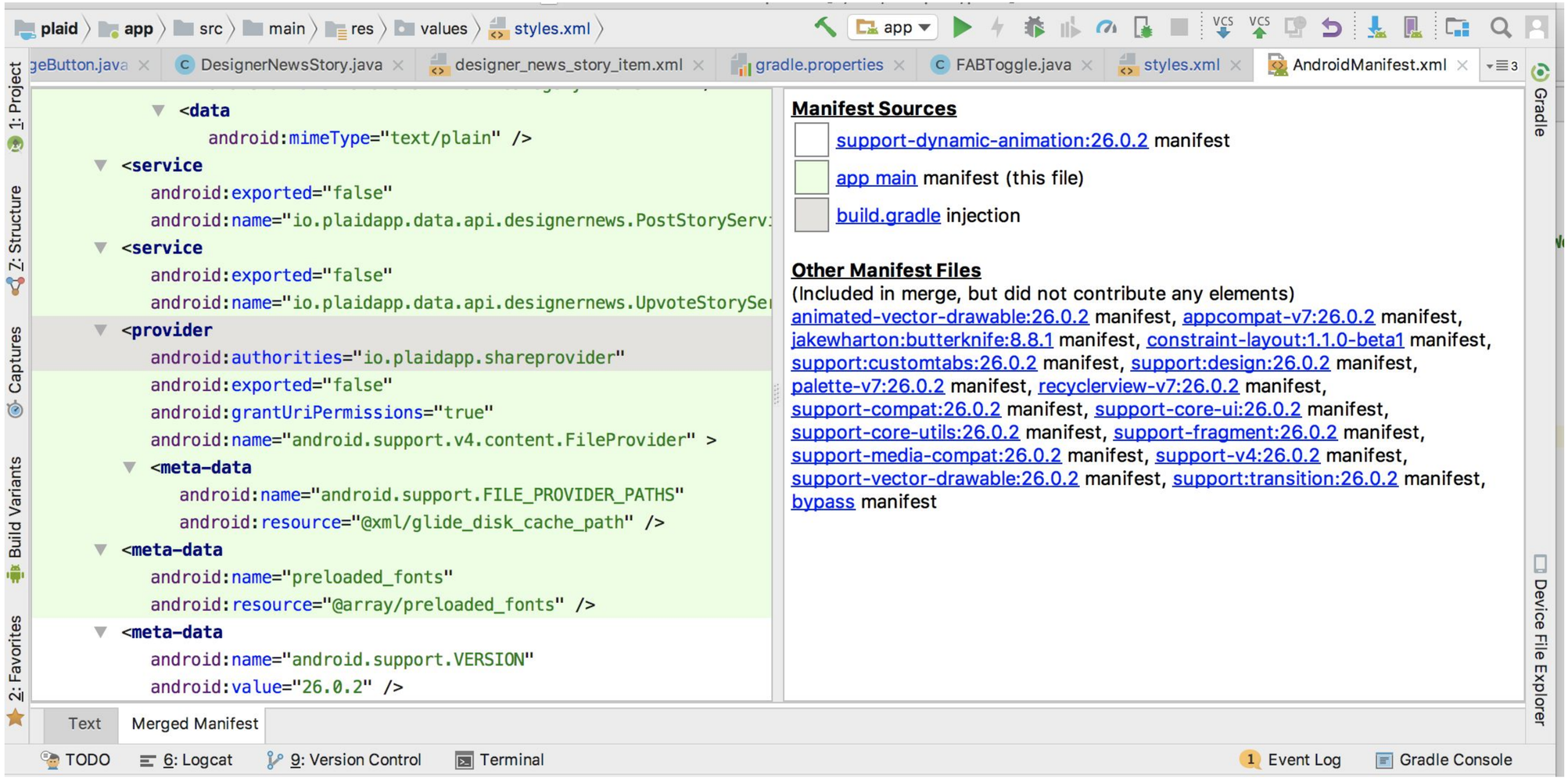
Not a thing.

A write-API for UAST is not in the works.

Convenience helpers are.

For now need to handle each UAST language separately.

Merged Manifest Support



Merged Manifest Support

Originally, scan through source manifests yourself

3.0: Project.getMergedManifest(): Document

Can report errors on merged manifest nodes

Locations mapped back to source

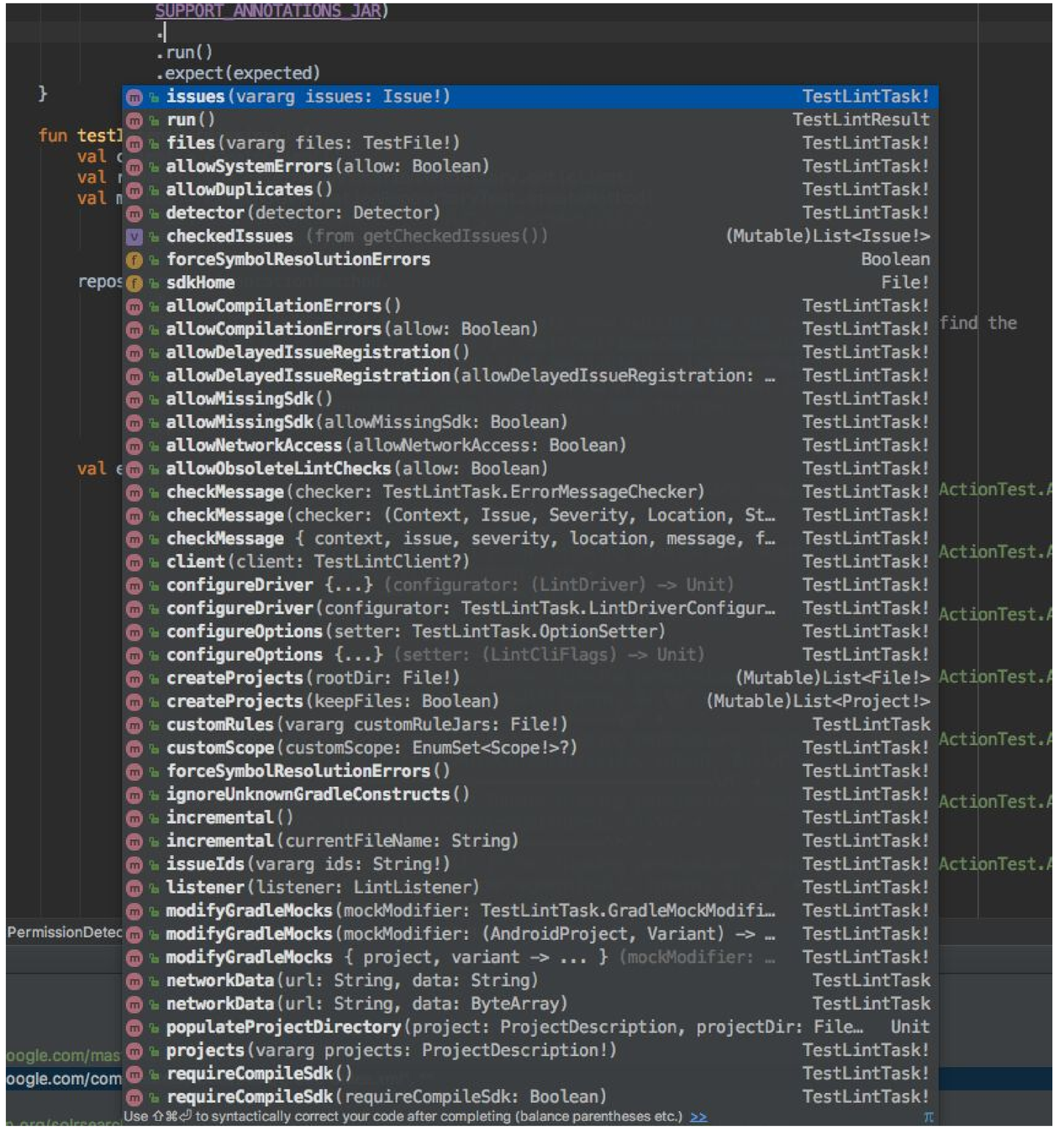
```
val project = context.getMainProject()
val mergedManifest = project.mergedManifest ?: return false
val manifest = mergedManifest.documentElement ?: return false
val application = getFirstSubTagByName(manifest, "application") ?: return false
var usesLibrary = getFirstSubTagByName(application, "uses-library")

while (usesLibrary != null) {
    val name = usesLibrary.getAttributeNS(ANDROID_URI, "name")
    if (name == "com.google.android.things") {
        // something
    }

    usesLibrary = getNextTagByName(usesLibrary, "uses-library")
}
```

Call Graph Support

New (and **experimental**) in 3.1

```
/**
* Whether this implementation wants to access the global call graph
* with a call to {@link #analyzeCallGraph(Context, CallGraphResult)}.
*/
boolean isCallGraphRequired();

/**
* Analyze the call graph requested with {@link #isCallGraphRequired()}
*/
void analyzeCallGraph(@NonNull Context context,
    @NonNull CallGraphResult callGraph);
```

Test DSL


SUPPORT_ANNOTATIONS_JAR)
.run()
.expect(expected)
}
issues(vararg issues: Issue!) TestLintTask!
run() TestLintResult
files(vararg files: TestFile!) TestLintTask!
allowSystemErrors(allow: Boolean) TestLintTask!
allowDuplicates() TestLintTask!
detector(detector: Detector) TestLintTask!
checkedIssues (from getCheckedIssues()) (Mutable)List<Issue!>
forceSymbolResolutionErrors Boolean
sdkHome File!
allowCompilationErrors() TestLintTask!
allowCompilationErrors(allow: Boolean) TestLintTask!
allowDelayedIssueRegistration() TestLintTask!
allowDelayedIssueRegistration(allowDelayedIssueRegistration: … TestLintTask!
allowMissingSdk() TestLintTask!
allowMissingSdk(allowMissingSdk: Boolean) TestLintTask!
allowNetworkAccess(allowNetworkAccess: Boolean) TestLintTask!
allowObsoleteLintChecks(allow: Boolean) TestLintTask!
checkMessage(checker: TestLintTask.ErrorMessageChecker) TestLintTask!
checkMessage(checker: (Context, Issue, Severity, Location, St… TestLintTask!
checkMessage { context, issue, severity, location, message, f… TestLintTask!
client(client: TestLintClient?) TestLintTask!
configureDriver {...} (configurator: (LintDriver) -> Unit) TestLintTask!
configureDriver(configurator: TestLintTask.LintDriverConfigur… TestLintTask!
configureOptions(setter: TestLintTask.OptionSetter) TestLintTask!
configureOptions {...} (setter: (LintCliFlags) -> Unit) TestLintTask!
createProjects(rootDir: File!) (Mutable)List<File!>
createProjects(keepFiles: Boolean) (Mutable)List<Project!>
customRules(vararg customRuleJars: File!) TestLintTask
customScope(customScope: EnumSet<Scope!>?) TestLintTask!
forceSymbolResolutionErrors() TestLintTask!
ignoreUnknownGradleConstructs() TestLintTask!
incremental() TestLintTask!
incremental(currentFileName: String) TestLintTask!
issueIds(vararg ids: String!) TestLintTask!
listener(listener: LintListener) TestLintTask!
modifyGradleMocks(mockModifier: TestLintTask.GradleMockModifi… TestLintTask!
modifyGradleMocks(mockModifier: (AndroidProject, Variant) -> … TestLintTask!
modifyGradleMocks { project, variant -> ... } (mockModifier: … TestLintTask!
networkData(url: String, data: String) TestLintTask
networkData(url: String, data: ByteArray) TestLintTask
populateProjectDirectory(project: ProjectDescription, projectDir: File… Unit
projects(vararg projects: ProjectDescription!) TestLintTask!
requireCompileSdk() TestLintTask!
requireCompileSdk(requireCompileSdk: Boolean) TestLintTask!
Use ⇧⌘⏎ to syntactically correct your code after completing (balance parentheses etc.) >>

Gotcha's

Things to look out for with Kotlin

Have at least one unit test for Kotlin sample source

Catch accidental usage of PSI (such as psiMethod.body())

Gotcha's

Things to look out for with Kotlin

When looking for return values, don't just look for
  UReturn nodes!

Expression body methods don't have return nodes

fun double(int: Int) = 2 * int

```
if (expressions.size() == 1) {
    UExpression statement = expressions.get(0);
    UExpression returnValue = null;
    if (statement instanceof UReturnExpression) {
        UReturnExpression returnStatement = (UReturnExpression) statement;
        returnValue = returnStatement.getReturnExpression();
    } else if (statement != null) {
        // Kotlin: may not have an explicit return statement
        returnValue = statement;
    }
```

Gotcha's

Things to look out for with Kotlin

Variables may not have type declarations; handle that gracefully.

```
val x = something()
```

Here the UVariable.typeReference() will be null

Gotcha's

Things to look out for with Kotlin

When handling UastBinaryOperator.**EQUALS**, don't forget **IDENTITY_EQUALS**

== versus ===

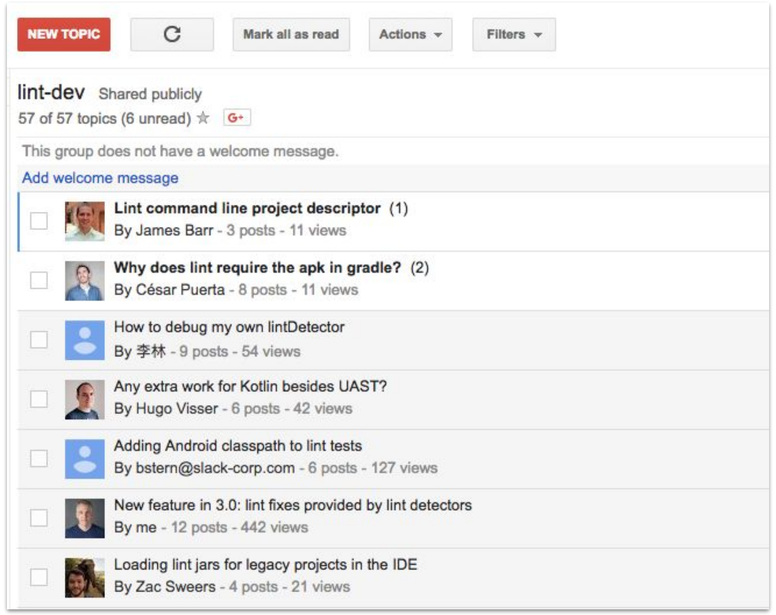
(Ditto for NOT_EQUALS and IDENTITY_NOT_EQUALS)

Lint 2.0 Plans

- UAST ✓
- More powerful quickfix API
- Improved registration API
- Resource repository lookup
- KTS support
- Simple detector options API (boolean, strings, ranges)
- Finish Callgraph and interprocedural API support
- **Stable API**
  - Rip out old support (Lombok, ResolvedNode, PSI)
  - Renaming (Java -> Uast)
  - Replacing *Utils with Kotlin extension methods

Community: https://groups.google.com/forum/#!forum/lint-dev

(or just search for lint-dev)



Thank you!

Tor Norbye
@tornorbye

#kotlinconf17