使用 Lint 检查改进您的代码

除了通过构建测试来确保您的应用符合其功能要求之外，还务必通过 Lint 运行代码来确保您的代码不存在结构问题。结构不合理的代码会影响 Android 应用的可靠性和效率，并使您的代码更难以维护，而 Lint 工具有助于找到这些代码。

例如，如果 XML 资源文件包含未使用的命名空间，这样不仅占用空间，而且还会导致不必要的处理。其他结构问题（如使用目标 API 版本不支持的已弃用的元素或 API 调用）可能会导致代码无法正常运行。Lint 可帮助您解决这些问题。

要进一步提高 Lint 检查的性能，还应向您的代码添加注解 (https://developer.android.com/studio/write/annotations.html?hl=zh-CN)。

概览

Android Studio 提供了一个名为 Lint 的代码扫描工具，可帮助您发现并更正代码结构质量的问题，而无需您实际执行应用，也不必编写测试用例。系统会报告该工具检测到的每个问题并提供问题的描述消息和严重级别，以便您可以快速确定需要优先进行的关键改进。此外，您还可以降低问题的严重级别以忽略与项目无关的问题，或者提高严重级别以突出特定问题。

Lint 工具可以检查您的 Android 项目源文件是否有潜在的错误，以及在正确性、安全性、性能、易用性、无障碍性和国际化方面是否需要优化改进。使用 Android Studio 时，无论何时编译应用，都会运行配置的 Lint 和 IDE 检查。不过，您可以手动运行检查 (#manuallyRunInspections)或从命令行运行 Lint (#commandline)。

注意：在 Android Studio 中编译代码时，会运行额外的 IntelliJ 代码检查 (https://www.jetbrains.com/help/idea/2018.3/code-inspection.html)以简化代码审核。

图 1 显示了 Lint 工具如何处理应用源文件。

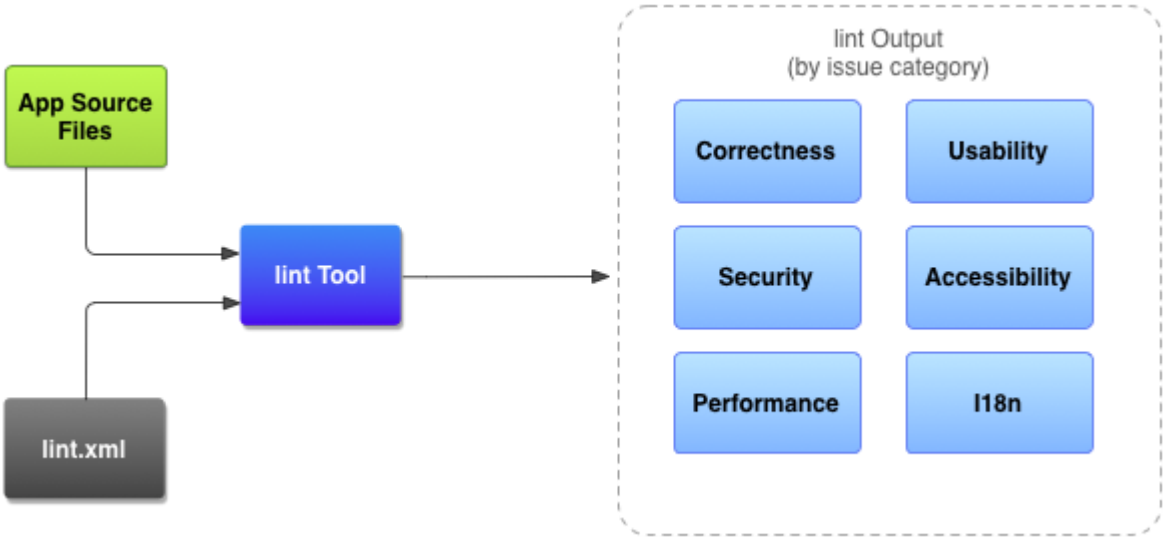

图 1. Lint 工具的代码扫描工作流

应用源文件

源文件包含组成 Android 项目的文件，包括 Java、Kotlin 和 XML 文件、图标以及 ProGuard 配置文件。

`lint.xml` 文件

一个配置文件，可用于指定要排除的任何 Lint 检查以及自定义问题严重级别。

Lint 工具

一个静态代码扫描工具，您可以从命令行或在 Android Studio 中对 Android 项目运行该工具（请参阅手动运行检查 (#manuallyRunInspections)）。Lint 工具检查可能会影响 Android 应用的质量和性能的代码结构问题。强烈建议您先更正 Lint 检测到的所有错误，然后再发布您的应用。

Lint 检查结果

您可以在控制台或 Android Studio 的 **Inspection Results** 窗口中查看 Lint 检查结果。请参阅手动运行检查 (#manuallyRunInspections)。

从命令行运行 Lint

如果您当前使用 Android Studio 或 Gradle，您可以通过从项目的根目录输入以下某个命令，使用 Gradle 封装容器 (https://docs.gradle.org/current/userguide/gradle_wrapper.html)对项目调用 `lint` 任务：

- 在 Windows 上：

```
gradlew lint
```

- 在 Linux 或 Mac 上：

```
./gradlew lint
```

您应看到类似于以下内容的输出：

```
> Task :app:lint
Ran lint on variant release: 5 issues found
Ran lint on variant debug: 5 issues found
Wrote HTML report to file:<path-to-project>/app/build/reports/lint-results
Wrote XML report to file:<path-to-project>/app/build/reports/lint-results
```

Lint 工具完成其检查后，会提供 XML 和 HTML 版 Lint 报告的路径。然后，您可以转到 HTML 报告并在浏览器中将其打开，如图 2 所示。

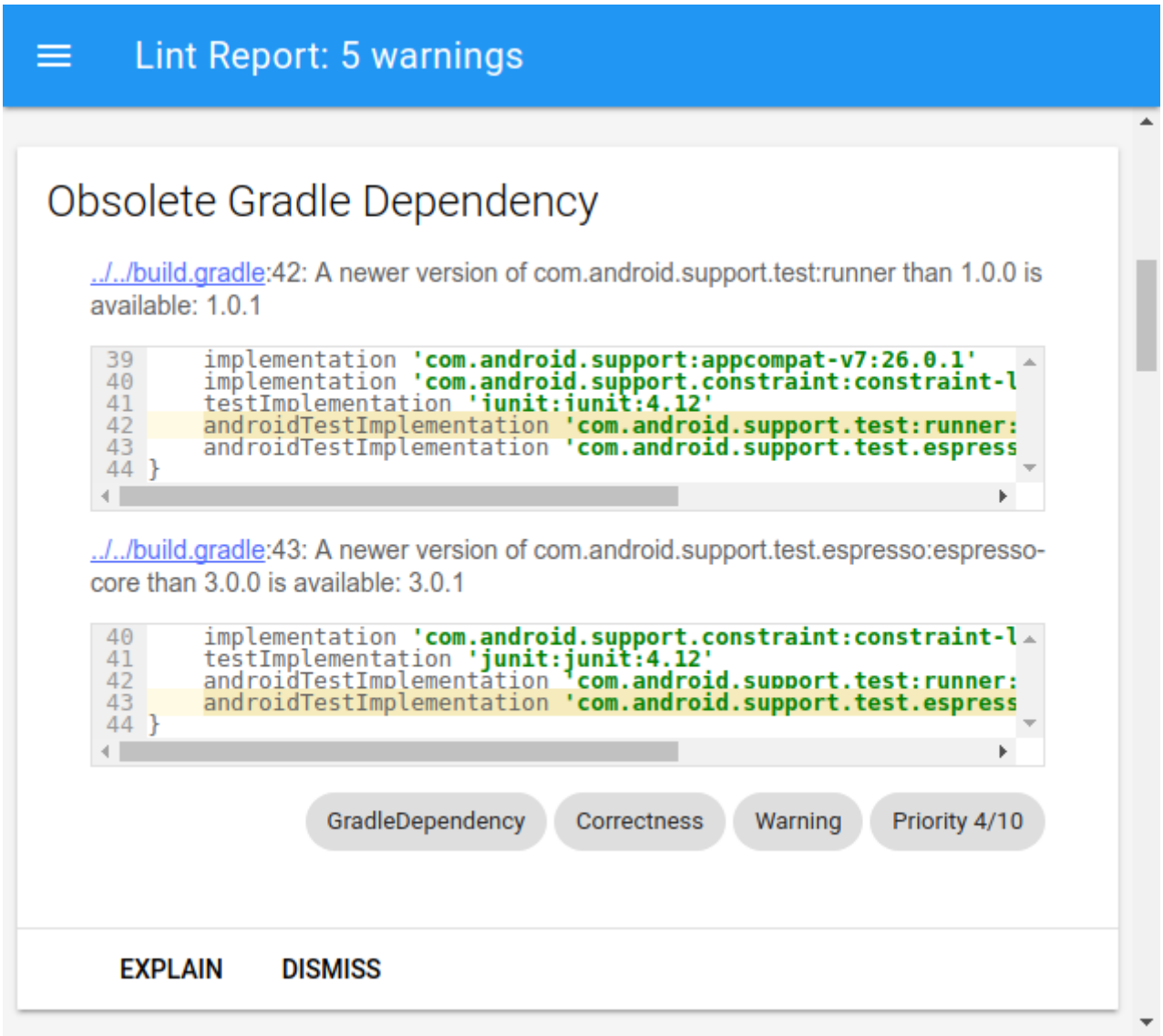

图 2. HTML Lint 报告示例

如果您的项目包含一些编译变体 (https://developer.android.com/studio/build/build-variants.html?hl=zh-CN)，而您希望仅对某个特定的编译变体运行 `lint` 任务，则必须将该变体名称的首字母大写，并在其前面加上 `lint` 作为前缀。

```
gradlew lintDebug
```

要详细了解如何从命令行运行 Gradle 任务，请阅读从命令行编译您的应用 (https://developer.android.com/studio/build/building-cmdline.html?hl=zh-CN)。

使用独立工具运行 Lint

如果您当前未使用 Android Studio 或 Gradle，您可以在通过 SDK 管理器 (https://developer.android.com/studio/command-line/sdkmanager.html?hl=zh-CN)安装 Android SDK Tools (https://developer.android.com/studio/command-line/index.html?hl=zh-CN#tools-sdk) 后使用独立 Lint 工具。安装该组件后，您可以在 ***android_sdk*/tools/** 目录中找到 Lint 工具。

要对项目目录中的文件列表运行 Lint，请使用以下命令：

```
lint [flags] <project directory>
```

例如，您可以发出以下命令来扫描 **myproject** 目录及其子目录下的文件。问题 ID **MissingPrefix** 提示 Lint 仅扫描缺少 Android 命名空间前缀的 XML 属性。

```
lint --check MissingPrefix myproject
```

要查看该工具支持的标志和命令行参数的完整列表，请使用以下命令：

```
lint --help
```

以下示例显示了对一个名为 Earthquake 的项目运行 Lint 命令时的控制台输出。

```
$ lint Earthquake

Scanning Earthquake: ...................................................
Scanning Earthquake (Phase 2): .......
AndroidManifest.xml:23: Warning: <uses-sdk> tag appears after <applicatio
  <uses-sdk android:minSdkVersion="7" />
  ^
AndroidManifest.xml:23: Warning: <uses-sdk> tag should specify a target AP
  <uses-sdk android:minSdkVersion="7" />
  ^
res/layout/preferences.xml: Warning: The resource R.layout.preferences app
res: Warning: Missing density variation folders in res: drawable-xhdpi [Ic
0 errors, 4 warnings
```

以上输出未列出任何错误，但列出了四条警告：其中三条警告（**ManifestOrder**、**UsesMinSdkAttributes** 和 **UnusedResources**）出现在项目的 **AndroidManifest.xml** 文件中，一条警告 (**IconMissingDensityFolder**) 出现在 **Preferences.xml** 布局文件中。

将 Lint 配置为抑制警告

默认情况下，当您运行 Lint 扫描时，Lint 工具会检查它支持的所有问题。您也可以限制 Lint 要检查的问题，并为这些问题分配严重级别。例如，您可以禁止对与项目无关的特定问题进行 Lint 检查，也可以将 Lint 配置为以较低的严重级别报告非关键问题。

您可以配置不同级别的 Lint 检查：

- 全局（整个项目）
- 项目模块
- 生产模块
- 测试模块
- 打开的文件
- 类层次结构
- 版本控制系统 (VCS) 范围

在 Android Studio 中配置 Lint

在您使用 Android Studio 时，内置的 Lint 工具会检查您的代码。您可以通过以下两种方式查看警告和错误：

- 作为代码编辑器中的弹出文本查看。Lint 发现问题后，会用黄色突出显示有问题的代码，而对于更严重的问题，则会在代码下面添加红色下划线。
- 依次点击 **Analyze > Inspect Code** 后，在 Lint **Inspection Results** 窗口中查看。请参阅手动运行检查 (#manuallyRunInspections)。

配置 Lint 文件

您可以在 **`lint.xml`** 文件中指定 Lint 检查偏好设置。如果您是手动创建此文件，请将其放置在 Android 项目的根目录下。

`lint.xml` 文件由封闭的 **`<lint>`** 父标记组成，此标记包含一个或多个 **`<issue>`** 子元素。Lint 会为每个 **`<issue>`** 定义唯一的 **`id`** 属性值。

```
<?xml version="1.0" encoding="UTF-8"?>
    <lint>
        <!-- list of issues to configure -->
```

```
</lint>
```

您可以通过在 **<issue>** 标记中设置严重级别属性来更改某个问题的严重级别或对该问题停用 Lint 检查。

提示：如需查看 Lint 支持的问题及其对应的问题 ID 的完整列表，请运行 **lint --list** 命令。

lint.xml 文件示例

以下示例显示了 **lint.xml** 文件的内容。

```
<?xml version="1.0" encoding="UTF-8"?>
<lint>
    <!-- Disable the given check in this project -->
    <issue id="IconMissingDensityFolder" severity="ignore" />

    <!-- Ignore the ObsoleteLayoutParam issue in the specified files -->
    <issue id="ObsoleteLayoutParam">
        <ignore path="res/layout/activation.xml" />
        <ignore path="res/layout-xlarge/activation.xml" />
    </issue>

    <!-- Ignore the UselessLeaf issue in the specified file -->
    <issue id="UselessLeaf">
        <ignore path="res/layout/main.xml" />
    </issue>

    <!-- Change the severity of hardcoded strings to "error" -->
    <issue id="HardcodedText" severity="error" />
</lint>
```

配置 Java、Kotlin 和 XML 源文件的 Lint 检查

您可以禁止 Lint 检查 Java、Kotlin 和 XML 源文件。

提示：您可以在 **Default Preferences** 对话框中管理 Lint 检查 Java、Kotlin 或 XML 源文件的功能。只需依次选择 **File > Other Settings** > **Default Settings**，然后在 **Default Preferences** 对话框的左侧窗格中依次选择 **Editor > Inspections**。

配置 Java 或 Kotlin 的 Lint 检查

要专门对 Android 项目中的某个类或方法停用 Lint 检查，请向该代码添加 `@SuppressLint` 注解。

以下示例展示了如何对 `onCreate` 方法中的 `NewApi` 问题关闭 Lint 检查。Lint 工具会继续检查该类的其他方法中的 `NewApi` 问题。

KOTLIN JAVA (#JAVA)

```
@SuppressLint("NewApi")
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.main)
```

以下示例展示了如何对 `FeedProvider` 类中的 `ParserError` 问题关闭 Lint 检查：

KOTLIN JAVA (#JAVA)

```
@SuppressLint("ParserError")
class FeedProvider : ContentProvider() {
```

要禁止 Lint 检查文件中的所有问题，请使用 `all` 关键字，如下所示：

KOTLIN JAVA (#JAVA)

```
@SuppressLint("all")
```

配置 XML 的 Lint 检查

您可以使用 `tools:ignore` 属性对 XML 文件的特定部分停用 Lint 检查。在 `lint.xml` 文件中添加以下命名空间值，以便 Lint 工具能够识别该属性：

```
namespace xmlns:tools="http://schemas.android.com/tools"
```

以下示例展示了如何对 XML 布局文件的 **<LinearLayout>** 元素中的 **UnusedResources** 问题关闭 Lint 检查。如果某个父元素声明了 **ignore** 属性，则该元素的子元素会继承此属性。在本例中，也会对 **<TextView>** 子元素停用 Lint 检查。

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

    <TextView
        android:text="@string/auto_update_prompt" />
</LinearLayout>
```

要禁止检查多个问题，请使用以逗号分隔的字符串列出要禁止检查的问题。例如：

```
tools:ignore="NewApi,StringFormatInvalid"
```

要禁止 Lint 检查 XML 元素中的所有问题，请使用 **all** 关键字，如下所示：

```
tools:ignore="all"
```

通过 Gradle 配置 Lint 选项

通过 Android Plugin for Gradle，您可以使用模块级 **build.gradle** 文件中的 **lintOptions {}** (http://google.github.io/android-gradle-dsl/current/com.android.build.gradle.internal.dsl.LintOptions.html#com.android.build.gradle.internal.dsl.LintOptions) 块配置某些 Lint 选项，如要运行或忽略的检查。以下代码段展示了您可以配置的部分属性：

```
android {
  ...
  lintOptions {
    // Turns off checks for the issue IDs you specify.
    disable 'TypographyFractions','TypographyQuotes'
    // Turns on checks for the issue IDs you specify. These checks are in
    // addition to the default lint checks.
    enable 'RtlHardcoded','RtlCompat', 'RtlEnabled'
    // To enable checks for only a subset of issue IDs and ignore all othe
    // list the issue IDs with the 'check' property instead. This property
    // any issue IDs you enable or disable using the properties above.
    check 'NewApi', 'InlinedApi'
    // If set to true, turns off analysis progress reporting by lint.
    quiet true
    // if set to true (default), stops the build if errors are found.
    abortOnError false
    // if true, only report errors.
    ignoreWarnings true
  }
}
...
```

创建警告基准

您可以为项目的当前警告集创建快照，然后将该快照用作将来运行检查的基准，以便只报告新问题。有了基准快照，您便可开始使用 Lint 使编译失败，而不必先返回并解决所有现有问题。

要创建基准快照，请修改项目的 **build.gradle** 文件，如下所示。

```
android {
  lintOptions {
    baseline file("lint-baseline.xml")
  }
}
```

首次添加此代码行时，系统会创建 **lint-baseline.xml** 文件以建立基准。此后，Lint 工具仅读取该文件以确定基准。如果要创建新基准，请手动删除该文件并再次运行 Lint 以重新创建它。

然后，从 IDE（依次选择 **Analyze > Inspect Code**）或从命令行运行 Lint，如下所示。系统会输出 **lint-baseline.xml** 文件的位置。您的设置的文件位置可能与此处显示的有所不同。

```
$ ./gradlew lintDebug

...

Wrote XML report to file:///app/lint-baseline.xml
Created baseline file /app/lint-baseline.xml
```

运行 **lint** 会将所有当前问题记录在 **lint-baseline.xml** 文件中。当前问题集称为“基准”，如果要与其他人共享 **lint-baseline.xml** 文件，您可以将其签入版本控制。

自定义基准

如果要将某些问题类型（而不是全部）添加到基准，您可以通过修改项目的 **build.gradle** 文件来指定要添加的问题，如下所示。

```
android {
  lintOptions {
    check 'NewApi', 'HandlerLeak'
    baseline file("lint-baseline.xml")
   }
}
```

创建基准后，如果向代码库添加任何新警告，Lint 将仅列出新引入的错误。

基准警告

实行基准时，您会收到一条信息性警告，告知您一个或多个问题已被过滤掉，因为它们已在基准中列出。之所以发出这条警告，是为了帮您记住您已配置基准，因为理想情况下，您希望在某一时刻解决所有问题。

这条信息性警告不仅会告知您过滤掉的错误和警告的确切数量，而且还会跟踪不再报告的问题。此信息可让您知道是否确实解决了问题，以便您可以选择性地重新创建基准，以防止错误再次出现时检测不到。

注意：在 IDE 中以批处理模式运行检查时，会启用基准；但对于修改文件时在后台运行的编辑器中的检查，会忽略基准。原因是基准适用于这样一种情况：代码库具有大量的现有警告，但您确实希望在处理代码时在本地解决相关问题。

手动运行检查

您可以通过依次选择 **Analyze > Inspect Code**，手动运行配置的 Lint 及其他 IDE 检查。检查结果将显示在 **Inspection Results** 窗口中。

设置检查范围和配置文件

选择要分析的文件（检查范围）和要运行的检查（检查配置文件），具体操作步骤如下：

1. 在 **Android** 视图中，打开您的项目，然后选择要分析的项目、文件夹或文件。
2. 从菜单栏中，依次选择 **Analyze > Inspect Code**。
3. 在 **Specify Inspection Scope** 对话框中，查看设置。

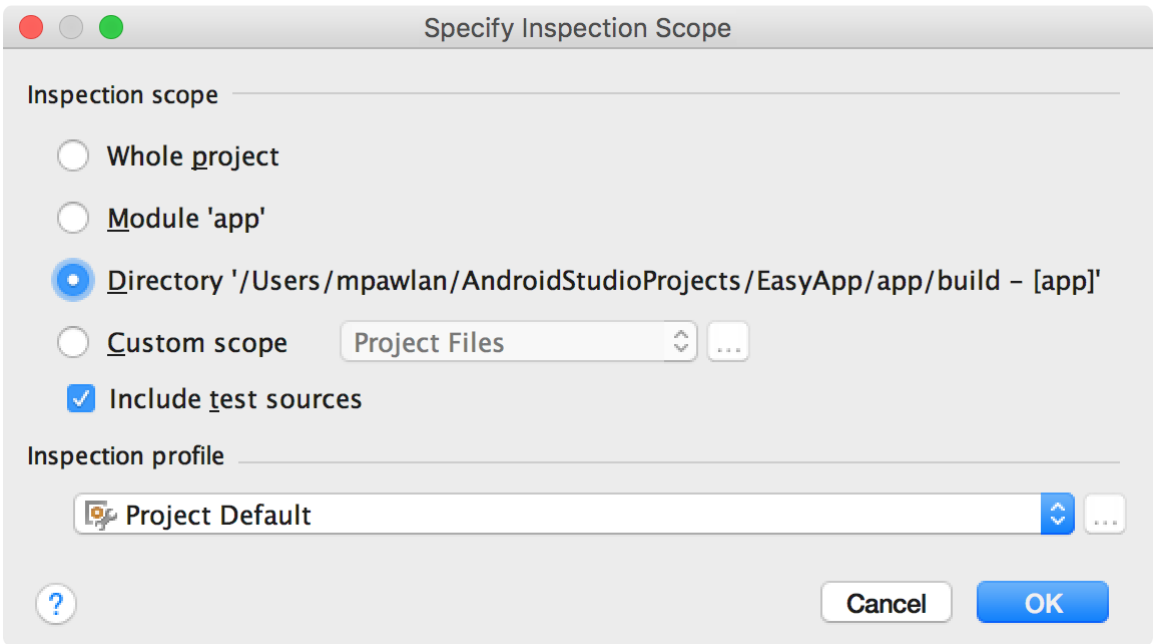

图 3. 查看检查范围设置

Specify Inspection Scope 对话框中显示的选项组合因您选择的是项目、文件夹还是文件而异。您可以通过选中其他某个单选按钮来更改要检查的内容。您可以在 Specify Inspection Scope (https://www.jetbrains.com/help/idea/2018.3/specify-inspection-scope-dialog.html) 对话框中查看 **Specify Inspection Scope** 对话框中可能显示的所有字段的说明。

- 如果您选择一个项目、文件或目录，**Specify Inspection Scope** 对话框中会显示选定**项目**、**文件**或**目录**的路径。
- 如果您选择多个项目、文件或目录，在 **Specify Inspection Scope** 对话框中，**选定的文件**旁边会显示一个选中的单选按钮。

4. 在 **Inspection profile** 下，保留默认配置文件 (**Project Default**)。
5. 点击 **OK** 以运行检查。图 4 显示了通过运行**检查代码**所生成的 Lint 及其他 IDE 检查结果：

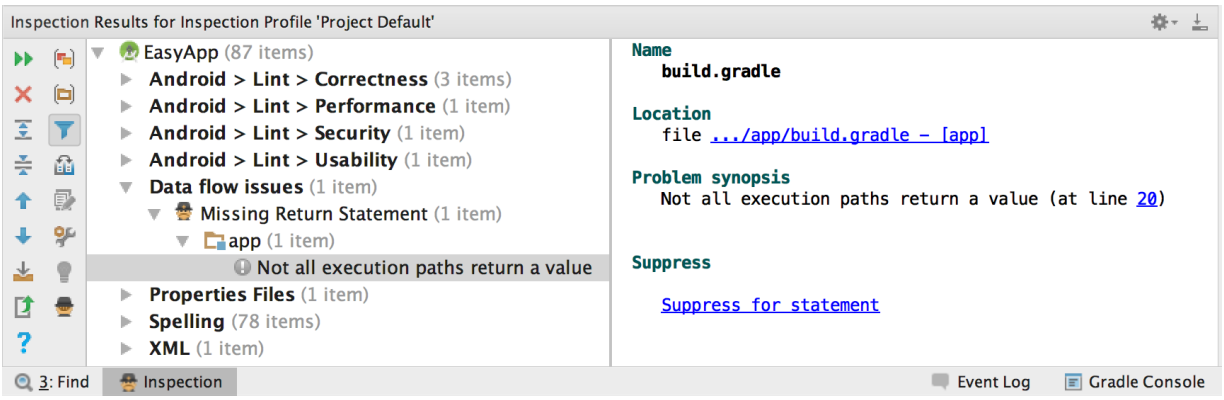

图 4. 选择问题以查看其解决方法

6. 在左侧窗格树状视图中，通过展开并选择错误类别、类型和问题来查看检查结果。右侧窗格显示选定错误类别、类型或问题的检查报告，并提供错误的名称和位置。在适用情况下，检查报告会显示问题概要等其他信息，以帮助您更正问题。
7. 在左侧窗格树状视图中，右键点击某个类别、类型或问题，以显示上下文菜单。根据上下文，您可以执行以下全部或部分操作：跳到源代码、排除和包含选定项、抑制问题、修改设置、管理检查警报和重新运行检查。

如需左侧工具栏按钮、上下文菜单项和检查报告字段的说明，请参阅检查工具窗口 (https://www.jetbrains.com/help/idea/2018.3/inspection-tool-window.html)。

使用自定义范围

您可以使用 Android Studio 中提供的某个自定义范围，具体操作步骤如下：

1. 在 **Specify Inspection Scope** 对话框中，点击 **Custom scope**。
2. 点击 **Custom scope** 下拉列表以显示选项。

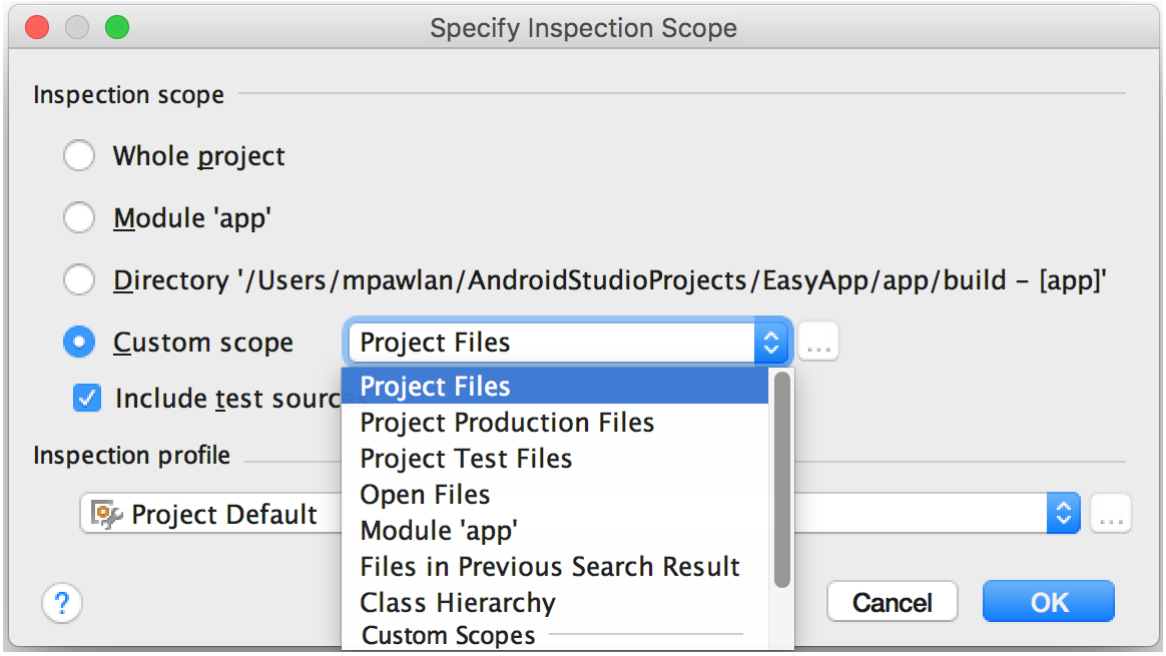

图 5. 选择要使用的自定义范围

- **Project Files**：当前项目中的所有文件。
- **Project Production Files**：仅限当前项目中的生产文件。
- **Project Test Files**：仅限当前项目中的测试文件。请参阅测试类型和位置 (https://developer.android.com/studio/test/index.html?hl=zh-CN#test_types_and_location) 。
- **Open Files**：仅限当前项目中已打开的文件。
- **Module <your-module>**：仅限当前项目中对应模块文件夹中的文件。
- **Current File**：仅限当前项目中的当前文件。当您选择了文件或文件夹时，会显示此选项。
- **Class Hierarchy**：如果您选择此选项并点击 **OK**，会出现一个对话框，其中显示当前项目中的所有类。您可以使用此对话框中的 **Search by Name** 字段过滤并选择要检查的类。如果未过滤类列表，代码检查将检查所有类。

3. 点击 **OK**。

创建自定义范围

如果您要检查的部分文件和目录不在当前可用的任何自定义范围内，您可以创建自定义范围。

1. 在 **Specify Inspection Scope** 对话框中，选择 **Custom scope**。
2. 点击 **Custom Scope** 下拉列表后面的三个点。

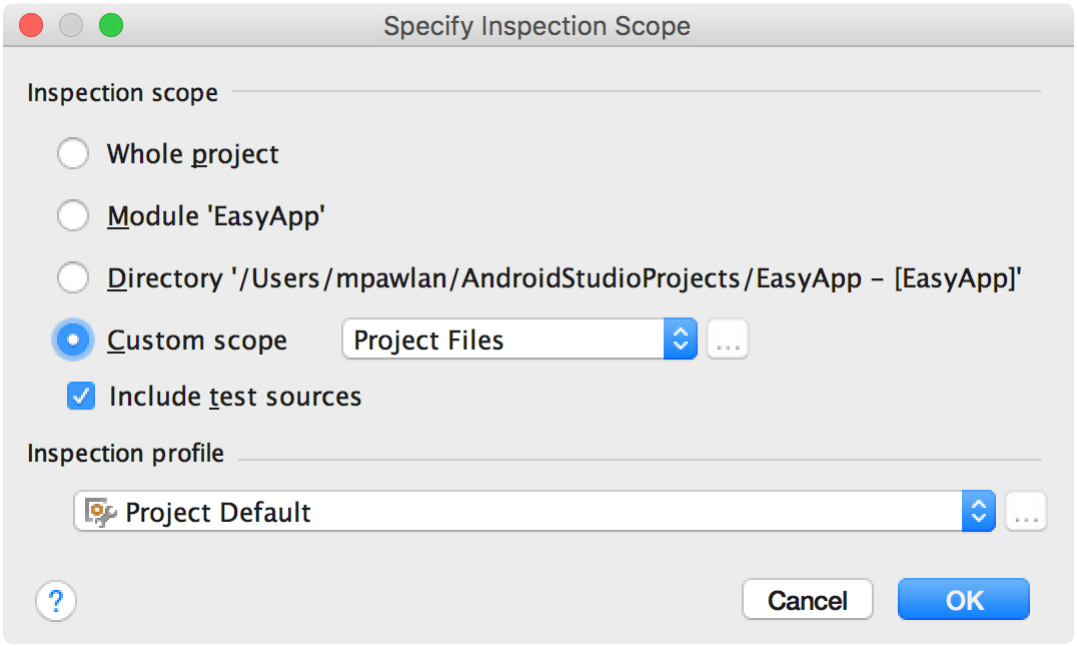

图 6. “Specify Inspection Scope”对话框

此时将显示 **Scopes** 对话框。

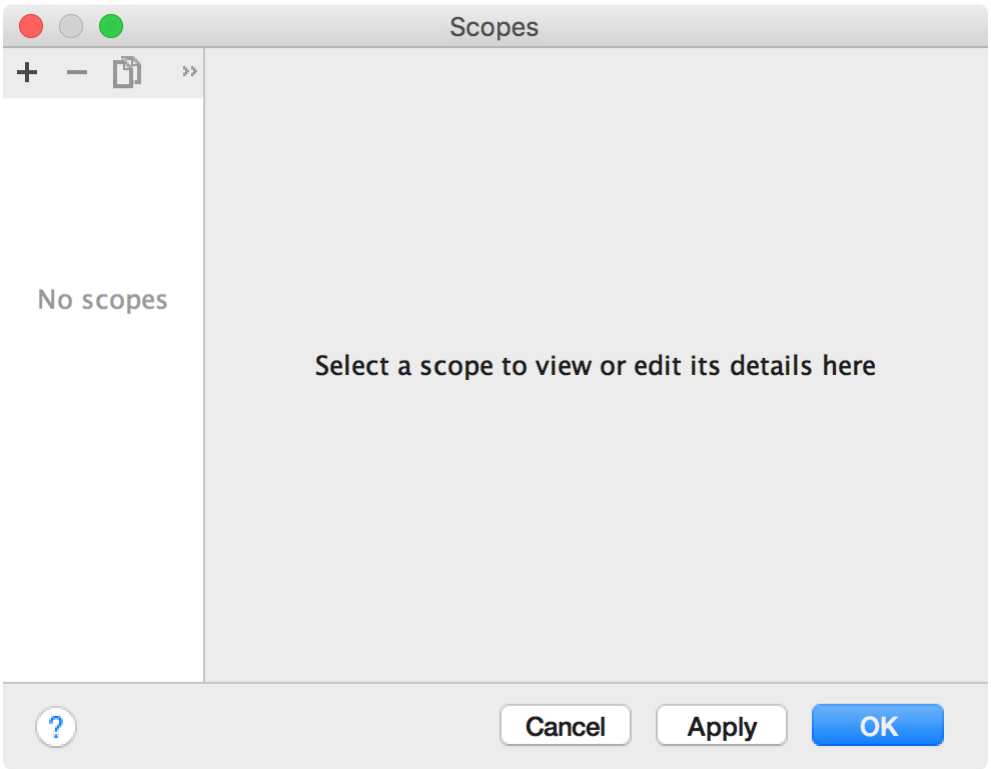

图 7. 创建自定义范围

3. 点击 **Add** 图标
   
   以定义新范围。

4. 在随即出现的 **Add Scope** 下拉列表中，选择 **Local**。

   局部范围和共享范围都在项目内用于**检查代码**功能。**共享**范围还可用于具有范围字段的其他项目功能。例如，当您点击 **Edit Settings** 图标

以更改 **Find Usages** 的设置时，随即出现的对话框中包含一个 **Scope** 字段，您可以在其中选择共享范围。

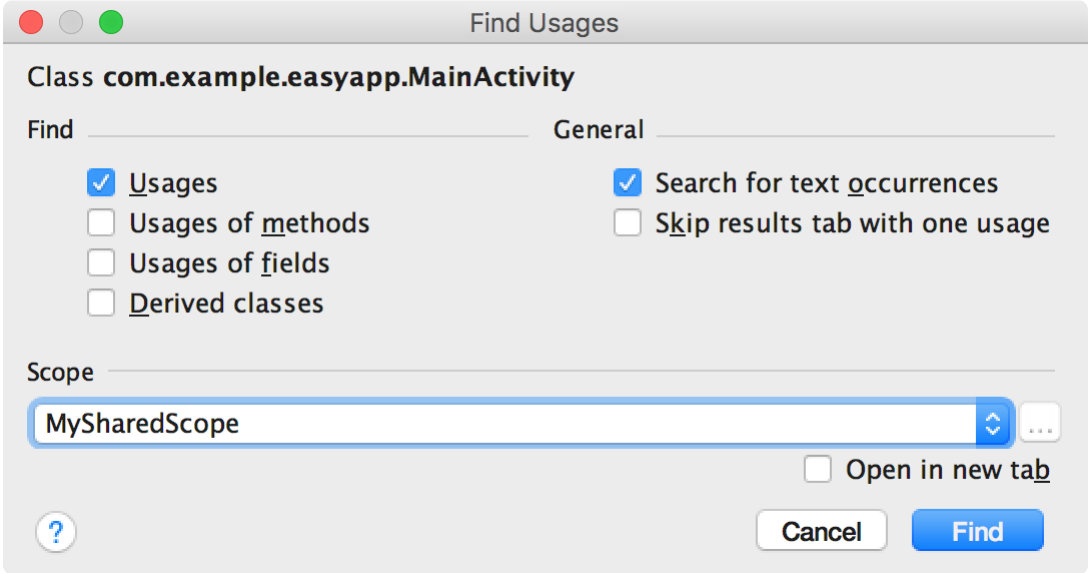

图 8. 从 **Find Usages** 对话框中选择共享范围

5. 为范围命名，然后点击 **OK**。

   Scopes 对话框的右侧窗格填充有可用于定义自定义范围的选项。

6. 从下拉列表中，选择 **Project**。

   此时将显示可用项目的列表。

★ **注意**：您可以为项目或软件包创建自定义范围。它们的步骤相同。

7. 展开项目文件夹，选择要添加到自定义范围的内容，然后点击右侧的某个按钮。

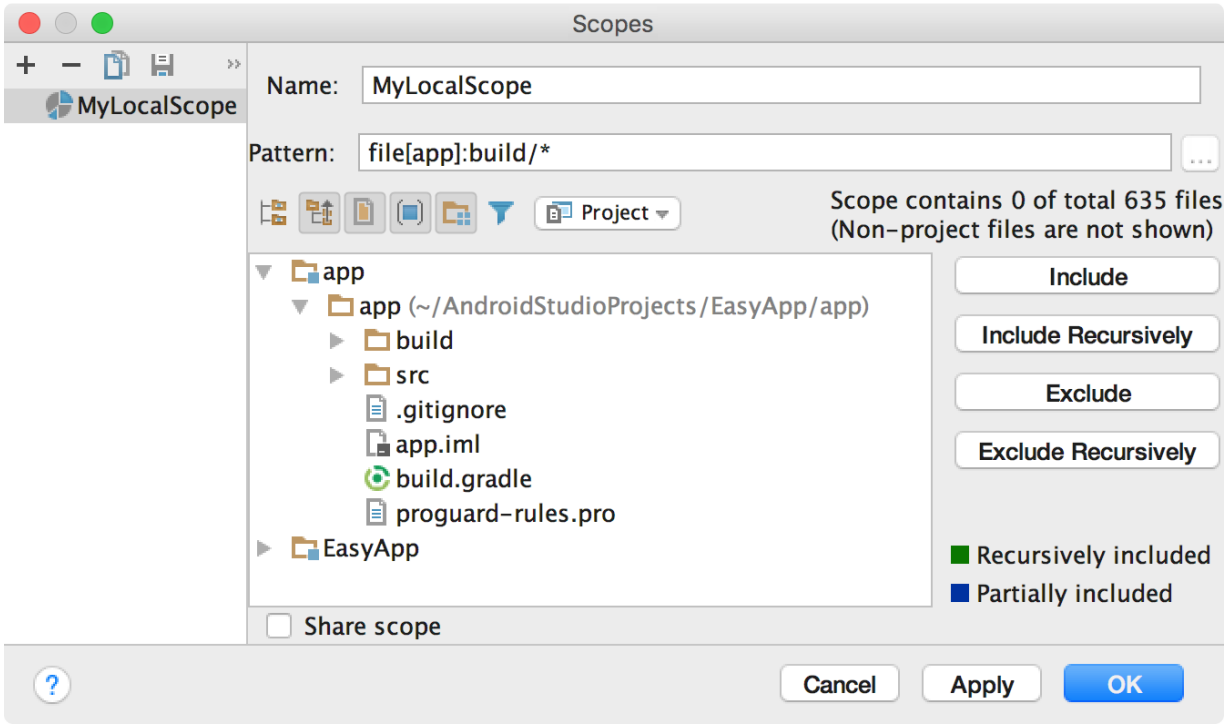

图 9. 定义自定义范围

- **Include**：包含此文件夹及其文件，但不包含其任何子文件夹。
- **Include Recursively**：包含此文件夹及其所有文件，以及子文件夹及其文件。
- **Exclude**：排除此文件夹及其文件，但不排除其任何子文件夹。
- **Exclude Recursively**：排除此文件夹及其所有文件，以及子文件夹及其文件。

图 10 显示包含 **main** 文件夹，并且递归包含 **java** 文件夹。蓝色表示部分包含的文件夹，而绿色表示递归包含的文件夹和文件。

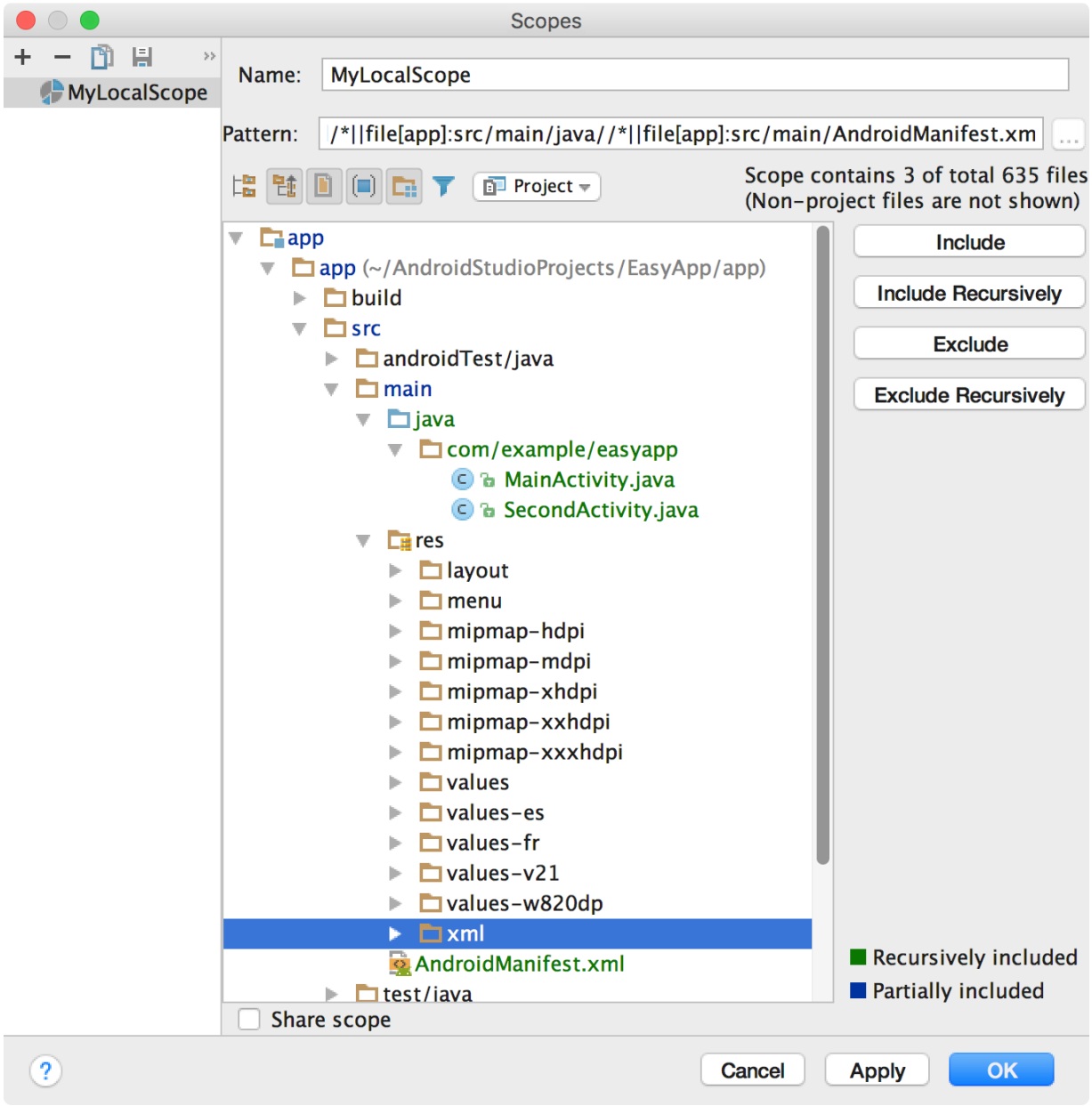

图 10. 自定义范围模式示例

- 如果您选择 **java** 文件夹并点击 **Exclude Recursively**，**java** 文件夹以及该文件夹下的所有文件夹和文件将不再用绿色突出显示。
- 相反，如果您选择用绿色突出显示的 **MainActivity.java** 文件并点击“Exclude”，**MainActivity.java** 将不再用绿色突出显示，而 **java** 文件夹下的其他所有项目则会用绿色突出显示。

8. 点击 **OK**。该自定义范围将显示在下拉列表的底部。

查看和修改检查配置文件

Android Studio 附带了许多 Lint 及其他检查配置文件，这些配置文件可通过 Android 更新进行更新。您可以原封不动地使用这些配置文件，也可以修改它们的名称、说明、严重级别和范围。您还可以激活和禁用整组的配置文件或一组配置文件中的个别配置文件。

要访问 **Inspections** 对话框，请执行以下操作：

1. 依次选择 **Analyze > Inspect Code**。
2. 在 **Specify Scope** 对话框的 **Inspection Profile** 下，点击 **More**。

   此时将显示 **Inspections** 对话框，其中列出了支持的检查及其说明。

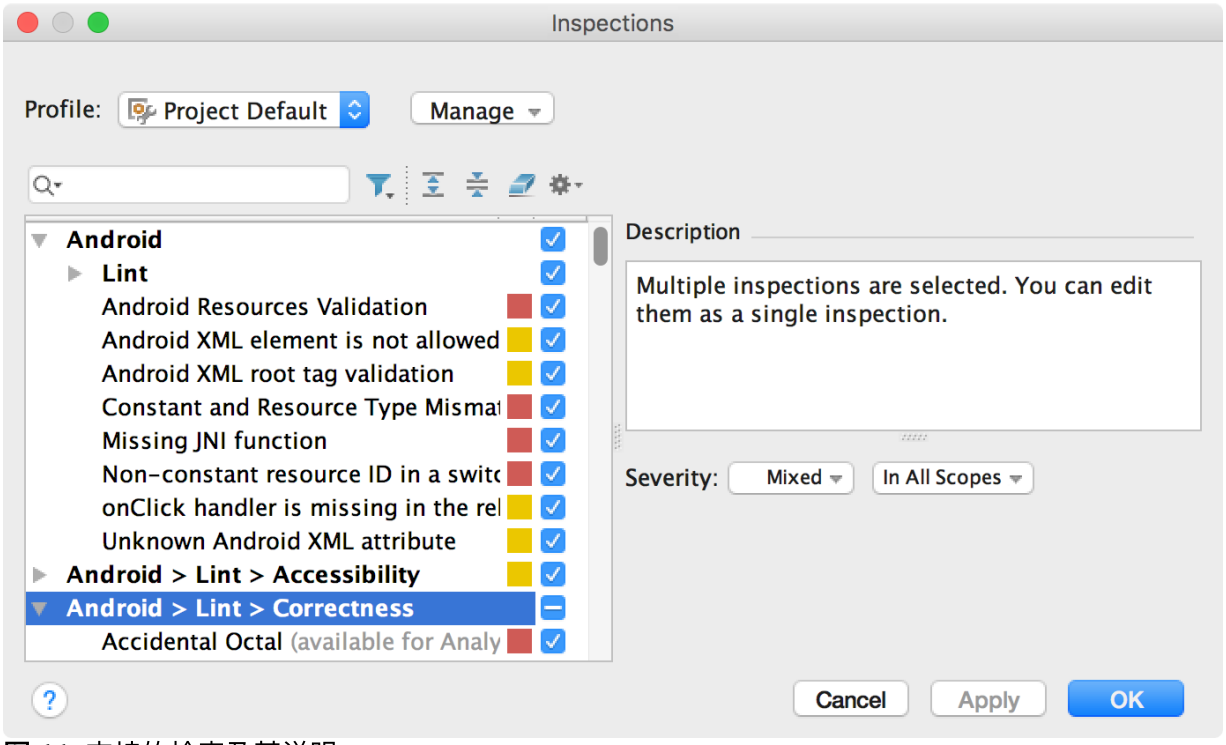

图 11. 支持的检查及其说明

3. 选择 **Profile** 下拉列表，以在 **Default** (Android Studio) 与 **Project Default**（活动项目）检查之间切换。如需了解详情，请参阅以下 IntelliJ 页面：“Specify Inspection Scope”对话框 (https://www.jetbrains.com/help/idea/2018.3/specify-inspection-scope-dialog.html)。
4. 在左侧窗格的 **Inspections** 对话框中，选择一个顶级配置文件类别，或展开一个组并选择特定的配置文件。选择一个配置文件类别后，您可以将该类别中的所有检查项目当作一个检查项目进行修改。
5. 选择 **Manage** 下拉列表，以复制检查项目、对检查项目进行重命名、向检查项目添加说明，以及导出和导入检查项目。
6. 操作完成后，点击 **OK**。