测试支持库

本文内容

- 测试支持库功能
  - AndroidJUnitRunner
  - Espresso
  - UI Automator
- 测试支持库设置

另请参阅

- 测试支持库 API 参考
- 代码示例

Android 测试支持库提供了大量用于测试 Android 应用的框架。此库提供了一组 API，让您可以为应用快速构建何运行测试代码，包括 JUnit 4 和功能性用户界面 (UI) 测试。您可以从 Android Studio IDE 或命令行运行使用这些 API 创建的测试。

Android 测试支持库通过 Android SDK 管理器提供。如需了解详细信息，请参阅测试支持库设置

本页介绍了 Android 测试支持库提供了哪些工具、如何在测试环境中使用这些工具，以及库版本的相关信息。

测试支持库功能

Android 测试支持库包括以下自动化测试工具：

- **AndroidJUnitRunner**：适用于 Android 且与 JUnit 4 兼容的测试运行器
- **Espresso**：UI 测试框架；适合应用中的功能性 UI 测试
- **UI Automator**：UI 测试框架；适合跨系统和已安装应用的跨应用功能性 UI 测试

AndroidJUnitRunner

`AndroidJUnitRunner` 类是一个 JUnit 测试运行器，可让您在 Android 设备上运行 JUnit 3 或 JUnit 4 样式测试类，包括使用 Espresso 和 UI Automator 测试框架的设备。测试运行器可以将测试软件包和要测试的应用加载到设备、运行测试并报告测试结果。此类将替换 `InstrumentationTestRunner` 类，后者仅支持 JUnit 3 测试。

此测试运行器的主要功能包括：

- JUnit 支持
- 访问仪器信息
- 测试筛选
- 测试分片

要求 Android 2.2（API 级别 8）或更高版本。

JUnit 支持

测试运行器与 JUnit 3 和 JUnit 4（最高版本为 JUnit 4.10）测试兼容。不过，请勿在同一软件包中混用 JUnit 3 和 JUnit 4 测试代码，因为这可能会导致意外结果。如果要创建一个 JUnit 4 仪器测试类以在设备或模拟器上运行，则测试类必须以 `@RunWith(AndroidJUnit4.class)` 注解作为前缀。

以下代码段显示了如何编写 JUnit 4 仪器测试来验证 `CalculatorActivity` 类中的 *add* 操作是否正常工作。

```
import android.support.test.runner.AndroidJUnit4;
import android.support.test.runner.AndroidJUnitRunner;
import android.test.ActivityInstrumentationTestCase2;

@RunWith(AndroidJUnit4.class)
public class CalculatorInstrumentationTest
        extends ActivityInstrumentationTestCase2<CalculatorActivity> {

    @Before
    public void setUp() throws Exception {
        super.setUp();

        // Injecting the Instrumentation instance is required
        // for your test to run with AndroidJUnitRunner.
        injectInstrumentation(InstrumentationRegistry.getInstrumentation());
        mActivity = getActivity();
    }

    @Test
    public void typeOperandsAndPerformAddOperation() {
        // Call the CalculatorActivity add() method and pass in some operand values, then
        // check that the expected value is returned.
    }

    @After
    public void tearDown() throws Exception {
        super.tearDown();
    }
}
```

访问仪器信息

您可以使用 `InstrumentationRegistry` 类访问与测试运行相关的信息。此类包括 `Instrumentation` 对象、目标应用 `Context` 对象、测试应用 `Context` 对象，以及传递到测试中的命令行参数。使用 UI Automator 框架编写测试或编写依赖于 `Instrumentation` 或 `Context` 对象的测试时，此数据非常有用。

测试筛选

在 JUnit 4.x 测试中，您可以使用注解对测试运行进行配置。此功能可将向测试中添加样板文件和条件代码的需求降至最低。除了 JUnit 4 支持的标准注解外，测试运行器还支持 Android 特定的注解，包括：

- `@RequiresDevice`：指定测试仅在物理设备而不在模拟器上运行。
- `@SdkSupress`：禁止在低于给定级别的 Android API 级别上运行测试。例如，要禁止在低于 18 的所有 API 级别上运行测试，请使用注解 `@SDKSupress(minSdkVersion=18)`。
- `@SmallTest`、`@MediumTest` 和 `@LargeTest`：指定测试的运行时长以及运行频率。

测试分片

测试运行器支持将单一测试套件拆分成多个*碎片*，因此您可以将属于同一碎片的测试作为一个组在同一 `Instrumentation` 实例下运行。每个分片由一个索引号进行标识。运行测试时，使用 `-e numShards` 选项指定要创建的独立分片数量，并使用 `-e shardIndex` 选项指定要运行哪个分片。

例如，要将测试套件拆分成 10 个分片，且仅运行第二个碎片中的测试，请使用以下命令：

```
adb shell am instrument -w -e numShards 10 -e shardIndex 2
```

要详细了解如何使用此测试运行器，请参阅 API 参考。

Espresso

Espresso 测试框架提供了一组 API 来构建 UI 测试，用于测试应用中的用户流。利用这些 API，您可以编写简洁、运行可靠的自动化 UI 测试。Espresso 非常适合编写*白盒*自动化测试，其中测试代码将利用所测试应用的实现代码详情。

Espresso 测试框架的主要功能包括：

- 灵活的 API，用于目标应用中的视图和适配器匹配。如需了解详细信息，请参阅视图匹配。
- 一组丰富的操作 API，用于自动化 UI 交互。如需了解详细信息，请参阅操作 API。
- UI 线程同步，用于提升测试可靠性。如需了解详细信息，请参阅 UI 线程同步。

要求 Android 2.2（API 级别 8）或更高版本。

视图匹配

利用 `Espresso.onView()` 方法，您可以访问目标应用中的 UI 组件并与之交互。此方法接受 `Matcher` 参数并搜索视图层次结构，以找到符合给定条件的相应 `View` 实例。您可以通过指定以下条件来优化搜索：

- 视图的类名称
- 视图的内容描述
- 视图的 `R.id`
- 在视图中显示的文本

例如，要找到 ID 值为 `my_button` 的按钮，可以指定如下匹配器：

```
onView(withId(R.id.my_button));
```

如果搜索成功，`onView()` 方法将返回一个引用，让您可以执行用户操作并基于目标视图对断言进行测试。

适配器匹配

在 `AdapterView` 布局中，布局在运行时由子视图动态填充。如果目标视图位于某个布局内部，而该布局是从 `AdapterView`（例如 `ListView` 或 `GridView`）派生出的子类，则 `onView()` 方法可能无法工作，因为只有布局视图的子集会加载到当前视图层次结构中。

因此，请使用 `Espresso.onData()` 方法访问目标视图元素。`Espresso.onData()` 方法将返回一个引用，让您可以执行用户操作并根据 `AdapterView` 中的元素对断言进行测试。

操作 API

通常情况下，您可以通过根据应用的用户界面执行某些用户交互来测试应用。借助 `ViewActions` API，您可以轻松地实现这些操作的自动化。您可以执行多种 UI 交互，例如：

- 视图点击
- 滑动
- 按下按键和按钮
- 键入文本
- 打开链接

例如，要模拟输入字符串值并按下按钮以提交该值，您可以像下面一样编写自动化测试脚本。`ViewInteraction.perform()` 和 `DataInteraction.perform()` 方法采用一个或多个 `ViewAction` 参数，并以提供的顺序运行操作。

```
// Type text into an EditText view, then close the soft keyboard
onView(withId(R.id.editTextUserInput))
    .perform(typeText(STRING_TO_BE_TYPED), closeSoftKeyboard());

// Press the button to submit the text change
onView(withId(R.id.changeTextBt)).perform(click());
```

UI 线程同步

由于计时问题，Android 设备上的测试可能随机失败。此测试问题称为*测试不稳定*。在 Espresso 之前，解决方法是在测试中插入足够长的休眠或超时期或添加代码，以便重试失败的操作。Espresso 测试框架可以处理 `Instrumentation` 与 UI 线程之间的同步；这就消除了对之前的计时解决方法的需求，并确保测试操作与断言更可靠地运行。

要详细了解如何使用 Espresso，请参阅 API 参考和测试单个应用的 UI 培训。

UI Automator

UI Automator 测试框架提供了一组 API 来构建 UI 测试，用于在用户应用和系统应用中执行交互。利用 UI Automator API，您可以执行在测试设备中打开“设置”菜单或应用启动器等操作。UI Automator 测试框架非常适合编写*黑盒*自动化测试，其中的测试代码不依赖于目标应用的内部实现详情。

UI Automator 测试框架的主要功能包括：

- 用于检查布局层次结构的查看器。如需了解详细信息，请参阅 UI Automator 查看器。
- 在目标设备上检索状态信息并执行操作的 API。如需了解详细信息，请参阅访问设备状态。
- 支持跨应用 UI 测试的 API。如需了解详细信息，请参阅 UI Automator API。

要求 Android 4.3（API 级别 18）或更高版本。

UI Automator 查看器

`uiautomatorviewer` 工具提供了一个方便的 GUI，可以扫描和分析 Android 设备上当前显示的 UI 组件。您可以使用此工具检查布局层次结构，并查看在设备前台显示的 UI 组件属性。利用此信息，您可以使用 UI Automator（例如，通过创建与特定可见属性匹配的 UI 选择器）创建控制更加精确的测试。

`uiautomatorviewer` 工具位于 `<android-sdk>/tools/` 目录中。

访问设备状态

UI Automator 测试框架提供了一个 `UiDevice` 类，用于在目标应用运行的设备上访问和执行操作。您可以调用其方法来访问设备属性，如当前屏幕方向或显示尺寸。`UiDevice` 类还可用于执行以下操作：

- 更改设备旋转
- 按 D-pad 按钮
- 按“返回”、“主屏幕”或“菜单”按钮
- 打开通知栏
- 对当前窗口进行屏幕截图

例如，要模拟按下“主屏幕”按钮，请调用 `UiDevice.pressHome()` 方法。

UI Automator API

利用 UI Automator API，您可以编写稳健可靠的测试，而无需了解目标应用的实现详情。您可以使用这些 API 在多个应用中捕获和操作 UI 组件：

- `UiCollection`：枚举容器的 UI 元素以便计算子元素个数，或者通过可见的文本或内容描述属性来指代子元素。
- `UiObject`：表示设备上可见的 UI 元素。
- `UiScrollable`：为在可滚动 UI 容器中搜索项目提供支持。
- `UiSelector`：表示在设备上查询一个或多个目标 UI 元素。
- `Configurator`：允许您设置运行 UI Automator 测试所需的关键参数。

例如，以下代码显示了如何编写可在设备中调用默认应用启动器的测试脚本：

```
// Initialize UiDevice instance
mDevice = UiDevice.getInstance(getInstrumentation());

// Perform a short press on the HOME button
mDevice.pressHome();

// Bring up the default launcher by searching for
// a UI component that matches the content-description for the launcher button
UiObject allAppsButton = mDevice
        .findObject(new UiSelector().description("Apps"));

// Perform a click on the button to bring up the launcher
allAppsButton.clickAndWaitForNewWindow();
```

要详细了解如何使用 UI Automator，请参阅 API 参考和测试多个应用的 UI 培训。

测试支持库设置

Android 测试支持库软件包在最新版本的 Android 支持存储库中提供，后者可作为辅助组件通过 Android SDK 管理器下载。

要通过 SDK 管理器下载 Android 支持存储库，请执行以下操作：

1. 启动 Android SDK 管理器。
2. 在 SDK 管理器窗口中，滚动到 *Packages* 列表末尾，找到 *Extras* 文件夹并展开（如有必要）以显示其内容。
3. 选择 **Android Support Repository** 项。
4. 点击 **Install packages...** 按钮。

下载后，此工具会将支持存储库文件安装到您现有的 Android SDK 目录中。库文件位于 SDK 的以下子目录中：`<sdk>/extras/android/m2repository` 目录。

Android 测试支持库的类位于 `android.support.test` 软件包中。

要在 Gradle 项目中使用 Android 测试支持库，请在 `build.gradle` 文件中添加这些依赖关系：

```
dependencies {
  androidTestCompile 'com.android.support.test:runner:0.4'
  // Set this dependency to use JUnit 4 rules
  androidTestCompile 'com.android.support.test:rules:0.4'
  // Set this dependency to build and run Espresso tests
  androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.1'
  // Set this dependency to build and run UI Automator tests
  androidTestCompile 'com.android.support.test.uiautomator:uiautomator-v18:2.1.2'
}
```

要将 `AndroidJUnitRunner` 设置为 Gradle 项目中的默认测试仪器运行器，请在 `build.gradle` 文件中指定此依赖关系：

```
android {
    defaultConfig {
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
}
```

强烈建议将 Android 测试支持库与 Android Studio IDE 搭配使用。Android Studio 提供支持测试开发的功能，例如：

- 基于 Gradle 的灵活构建系统，为测试代码提供依赖关系管理支持
- 单一的项目结构，包含单元与仪器测试代码以及应用源代码
- 支持在虚拟或物理设备上通过命令行或图形用户界面部署和运行测试

如需了解有关 Android Studio 及其下载的详细信息，请参阅下载 Android Studio 和 SDK 工具。