How_to_test_iauto_framework

Last edited by **caoquanli** 1 month ago

如何为iauto.framework编写单体测试

Table of Contents

- 1. Instrument iauto.framework的代码
- 2. 为iauto.framework编写测试代码
- 3. 编译生成iauto.framework测试APK
- 4. 使用TF框架运行单体测试并生成单体测试报告

如果仅仅是为iauto.framework编写单体测试，那么直接按照普通的jar测试手顺就可以了,请参考 Android jar 单体测试手顺. 本文章的步骤是为了为单体测试生成code coverage report。

1. Instrument iauto.framework的代码

因为要为iauto.framework生成单体测试覆盖率，所以必须首先对其代码进行instrumention，同时为这个jar包生成em文件。

1. 在build/core/tasks/check_boot_jars/package_whitelist.txt中添加如下jar包：

```
 org\.objectweb\.asm.*
org\.jacoco\.agent\.rt.*
org\.jacoco\.agent.*
org\.jacoco\.core\.analysis.*
org\.jacoco\.core\.instr\.Instrumenter.*
org\.jacoco\.core\.instr.*
org\.jacoco\.core\.tools.*
org\.jacoco\.core\.runtime.*
org\.jacoco\.core.*
com\.vladium\.emma\.rt.*
com\.vladium\.emma.*
```

   在为iauto.framework生成instrumention code的时候会依赖这些包.

2. 在ivi/frameworks/api/Android.mk文件中添加如下两行三行代码:

```
 ifeq ($(EMMA_INSTRUMENT_FRAMEWORK),true)
LOCAL_EMMA_INSTRUMENT := true
endif
```

3. 运行 EMMA_INSTRUMENT_STATIC=true EMMA_INSTRUMENT_FRAMEWORK=true make -j8 全编译系统,然后烧卡，这样板子上运行的系统包含的iauto.framework就已经包含instrumention相关信息了。我们可以在out目录下面运行如下命令 find ./ -name "*.em",可以找到有多少代码是被instrument过。编译的过程中可能会报packages/apps/DocumentsUI的错误，可以修改其Android.mk文件，注释掉下面的两行代码：

```
 #LOCAL_JACK_FLAGS := \
#  -D jack.optimization.inner-class.accessors=true
```

2. 为iauto.framework编写测试代码

在 ivi/frameworks/api 目录下创建一个tests/目录用来存放单体测试的代码。tests下面有以下几个文件：

- iautoframeworkstests/src存放各个模块的代码，src下面的目录结构和 ivi/frameworks/api/java 下面的目录结构保持一致。
- 在tests目录下的Android.mk 文件构建如下：

```
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)

# Include the sub-makefiles
include $(call all-makefiles-under,$(LOCAL_PATH))
```

- 在iautoframeworkstests目录下构建AndroidTest.xml文件，此文件主要用于TF框架,具体的信息可以参考: Android Trade Federation(TF)框架单体测试.iauto.framework ivi/frameworks/api/tests/iautoframeworkstests 目录下的AndroidTest.xml构建如下：

```
<configuration description="Runs Tests for iauto framework">
    <target_preparer class="com.android.tradefed.targetprep.InstallBuildEnvApkSetup">
        <option name="apk-name" value="iautoFrameworksTests.apk" />
    </target_preparer>
     <option name="test-suite-tag" value="iautotest" />
    <test class="com.android.tradefed.testtype.CodeCoverageTest" >
        <option name="package" value="com.iauto" />
        <option name="runner" value="android.support.test.runner.AndroidJUnitRunner" />
        <option name="metadata-file"
         value="out/target/common/obj/JAVA_LIBRARIES/iauto.framework_intermediates/covera
ge.em" />
        <option name="source-dir" value="ivi/frameworks/api" />
    </test>
</configuration>
```

- 在ivi/frameworks/api/tests/iautoframeworkstests目录下构建AndroidManifest.xml文件，构建如下：

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.iauto" > <!--声明测试APK的包名-->
    <application>
        <uses-library android:name="android.test.runner" />
    </application>
    <instrumentation android:name="android.support.test.runner.AndroidJUnitRunner"
        android:label="Tests for iauto frameworks">
    </instrumentation>
</manifest>
```

- 在ivi/frameworks/api/tests/iautoframeworkstests目录下构建 Android.mk文件，构建如下：

```
LOCAL_PATH:= $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE_TAGS := tests
LOCAL_STATIC_JAVA_LIBRARIES := android.test.runner
LOCAL_SRC_FILES := $(call all-java-files-under, src)
#修改APK的名字
LOCAL_PACKAGE_NAME:=iautoFrameworksTests
LOCAL_STATIC_JAVA_LIBRARIES := \
      legacy-android-test \
      junit \
      android-support-test \
      mockito-target \
include $(BUILD_PACKAGE)
```

3. 编译生成iauto.framework测试APK

在源码目录下运行mmma -j8 ivi/frameworks/api，在out/target/product/mt2712/data/app/iautoFrameworksTests目录下生成测试iautoFrameworksTests.apk.

4. 使用TF框架运行单体测试并生成单体测试报告

在源码路径下面运行 tradefed.sh，然后输入 run ivi/frameworks/api/tests/iautoframeworkstests/AndroidTest.xml 命令:

- host log（PC端屏幕显示的log）如下：

```
 tf >run ivi/frameworks/api/tests/iautoframeworkstests/AndroidTest.xml
 10-29 09:56:14 I/TestInvocation: Invocation was started with cmd: ivi/frameworks/api/tes
ts/iautoframeworkstests/AndroidTest.xml
10-29 09:56:14 I/TestInvocation: Starting invocation for 'iautoFrameworksTests' with '[ B
uildInfo{bid=0, target=stub, serial=GFEDCBA0987654321} on device 'GFEDCBA0987654321']
10-29 09:56:14 I/InstallBuildEnvApkSetup: Installing iautoFrameworksTests.apk on GFEDCBA0
987654321
10-29 09:56:18 I/TextResultReporter: Saved device_logcat_setup log to /tmp/0/iautoFramewo
rksTests/inv_3115591015950281148/device_logcat_setup_8053052627035344923.zip
10-29 09:56:18 I/RemoteAndroidTest: Running am instrument -w -r   -e timeout_msec 300000
-e coverage true com.iauto/android.support.test.runner.AndroidJUnitRunner on unknown-car_
on_mt2712-GFEDCBA0987654321
10-29 09:56:21 I/InvocationToJUnitResultForwarder: run com.iauto started: 4 tests
10-29 09:56:21 I/InvocationToJUnitResultForwarder: run ended 80 ms
tf >

10-29 09:56:21 I/CodeCoverageTest: coverage file from device: /tmp/coverage_3102971821509
```

```
838806.ec
10-29 09:56:21 I/TextResultReporter: Saved com.iauto_runtime_coverage log to /tmp/0/iauto
FrameworksTests/inv_3115591015950281148/com.iauto_runtime_coverage_6064133704473222269.zip
10-29 09:56:21 I/CodeCoverageTest: create report dir: out/target/testcases/coverage/com.i
auto successfully
10-29 09:56:22 I/CodeCoverageTest: Jacoco code coverage report generation success
10-29 09:56:22 I/TextResultReporter: Saved device_logcat_test log to /tmp/0/iautoFramewor
ksTests/inv_3115591015950281148/device_logcat_test_2000454087332328405.zip
10-29 09:56:22 I/TextResultReporter: Saved device_logcat_teardown log to /tmp/0/iautoFram
eworksTests/inv_3115591015950281148/device_logcat_teardown_4826941461855767547.zip
10-29 09:56:22 I/TextResultReporter: Saved host_log log to /tmp/0/iautoFrameworksTests/in
v_3115591015950281148/host_log_648889028270277263.zip
```

- 备注： 覆盖率生成的报告在out/target/testcases/coverage目录下.