

1ヶ月で

ANDROIDカメラアプリ開発

@Trunk  
酒本伸也

2016/3/13

はじめに

# 自己紹介

酒本伸也

石川県

@sakebook

Android 3年くらい

iOS 半年ちょっと

ライフログ好き



# 本日の流れ

- 作るアプリのデモ
- Androidについて
- 開発環境設定
- 開発環境解説
- アプリのデバッグ
- ソースコード管理
- Adobe SDK導入



作るアプリのデモ

ANDROID

# ANDROIDとは

- Googleが開発するオープンソースなモバイル向けプラットフォーム

みんなちがうから、世界はたのしい。



brett  
medford, nj



heather  
los angeles, ca



derek  
nairobi, kenya



sara  
austin, tx



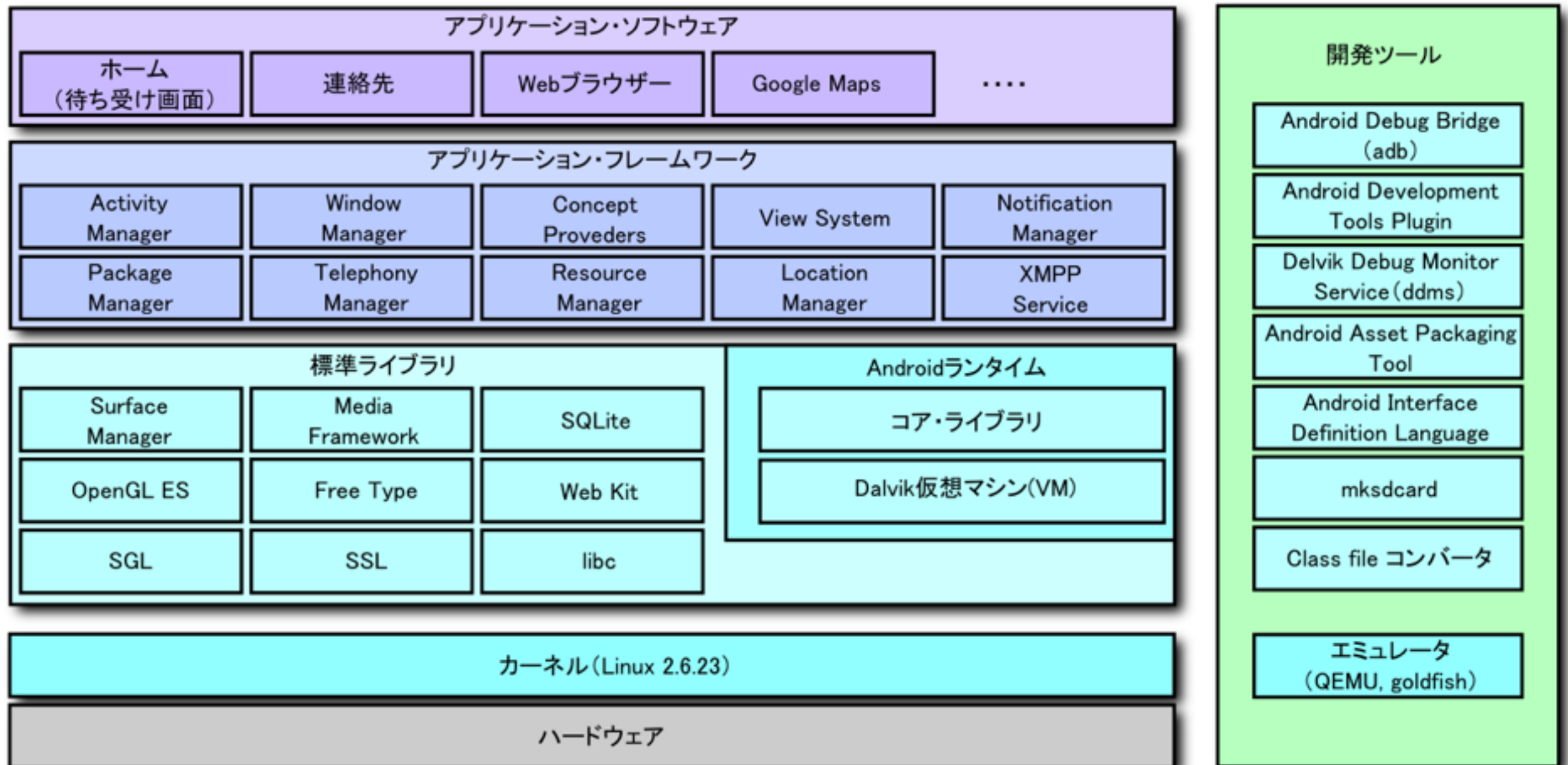
dan  
seoul, south korea



massimo  
philadelphia, pa

# アーキテクチャ

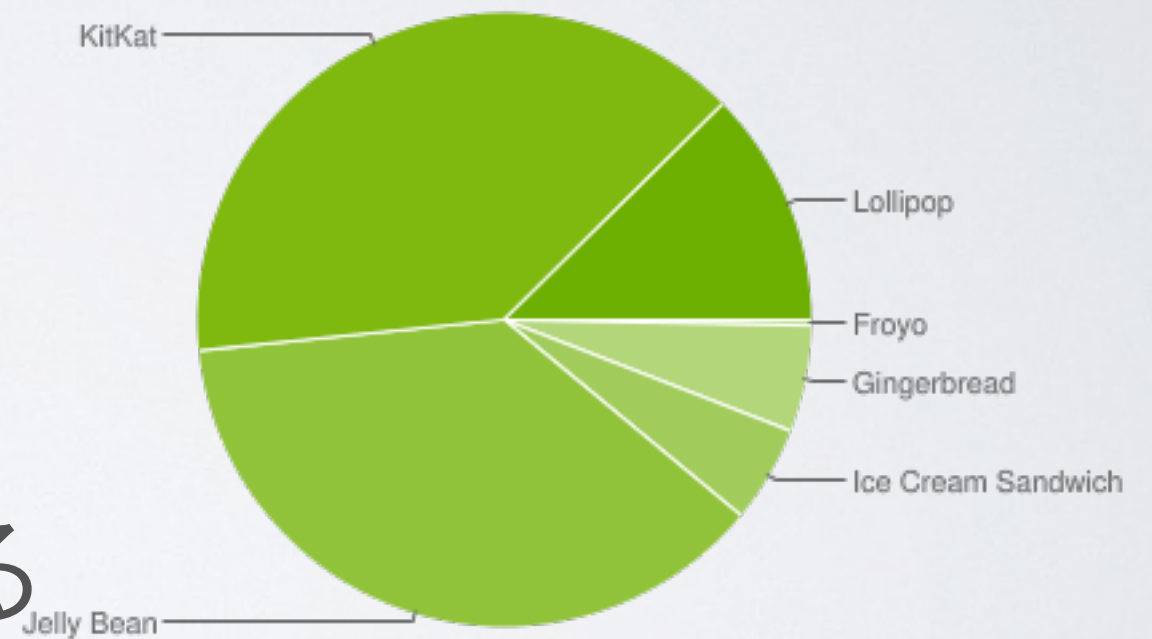
## Androidのアーキテクチャ





# バージョン

- 細かいものを含めると  
20以上もの種類  
(API LEVEL)
- 4.1以降が90%弱を占める  
(2015/6/1)



# APK

- アプリの拡張子
- Application PaKage  
からきている
- 開発したアプリは〇〇.apk  
という形になる



開發環境

# ANDROID STUDIO

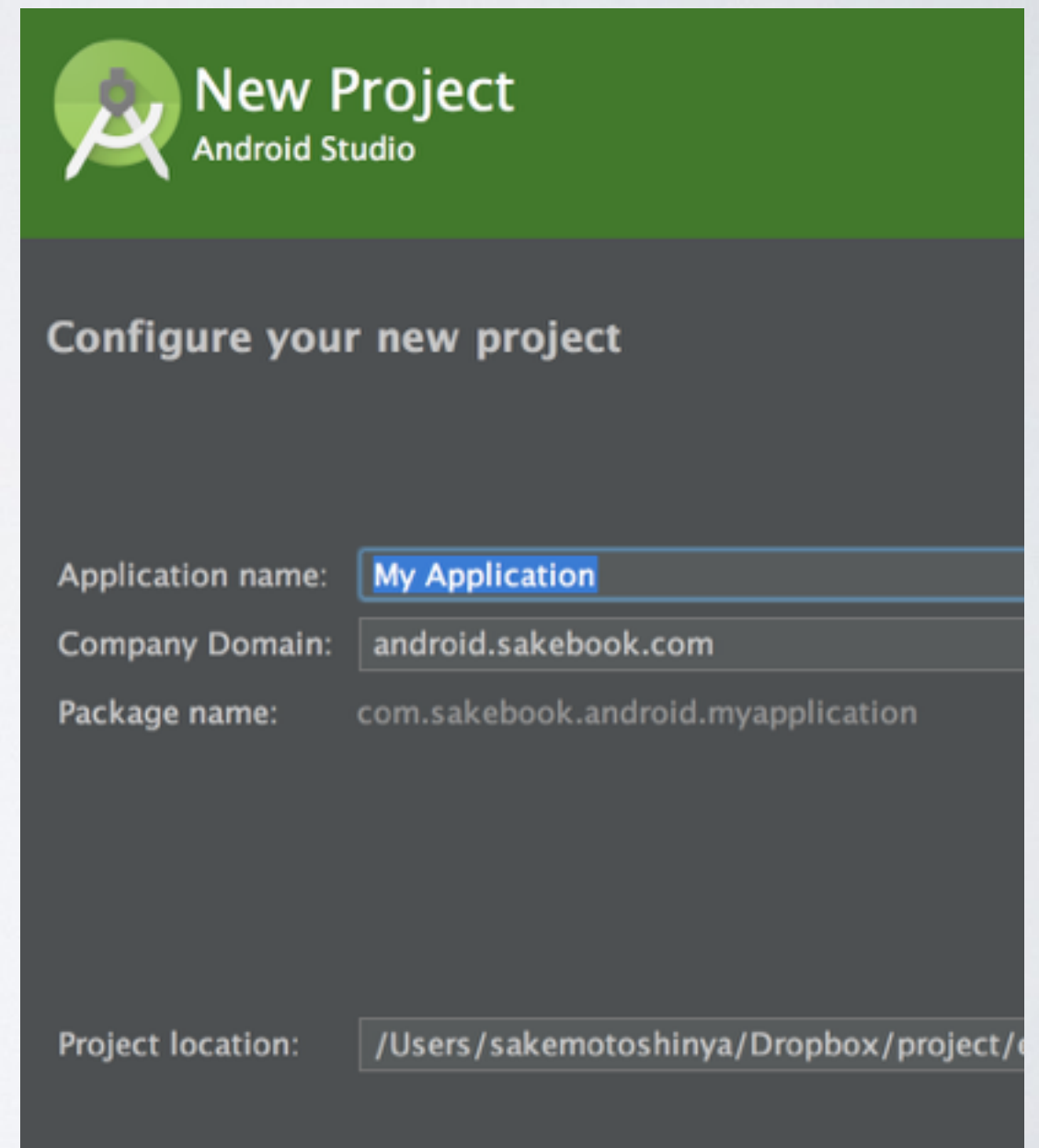
- IntelliJ IDEAをベースにした統合開発環境(IDE)
- Googleが開発
- Androidの開発に特化
- Eclipseは卒業してください(ADTが開発終了)



さあはじめよう

# NEW PROJECT

- Application name
  - アプリの名前
- Company Domain
  - 自分をユニークにするドメイン



New Project  
Android Studio

Configure your new project

Application name: My Application

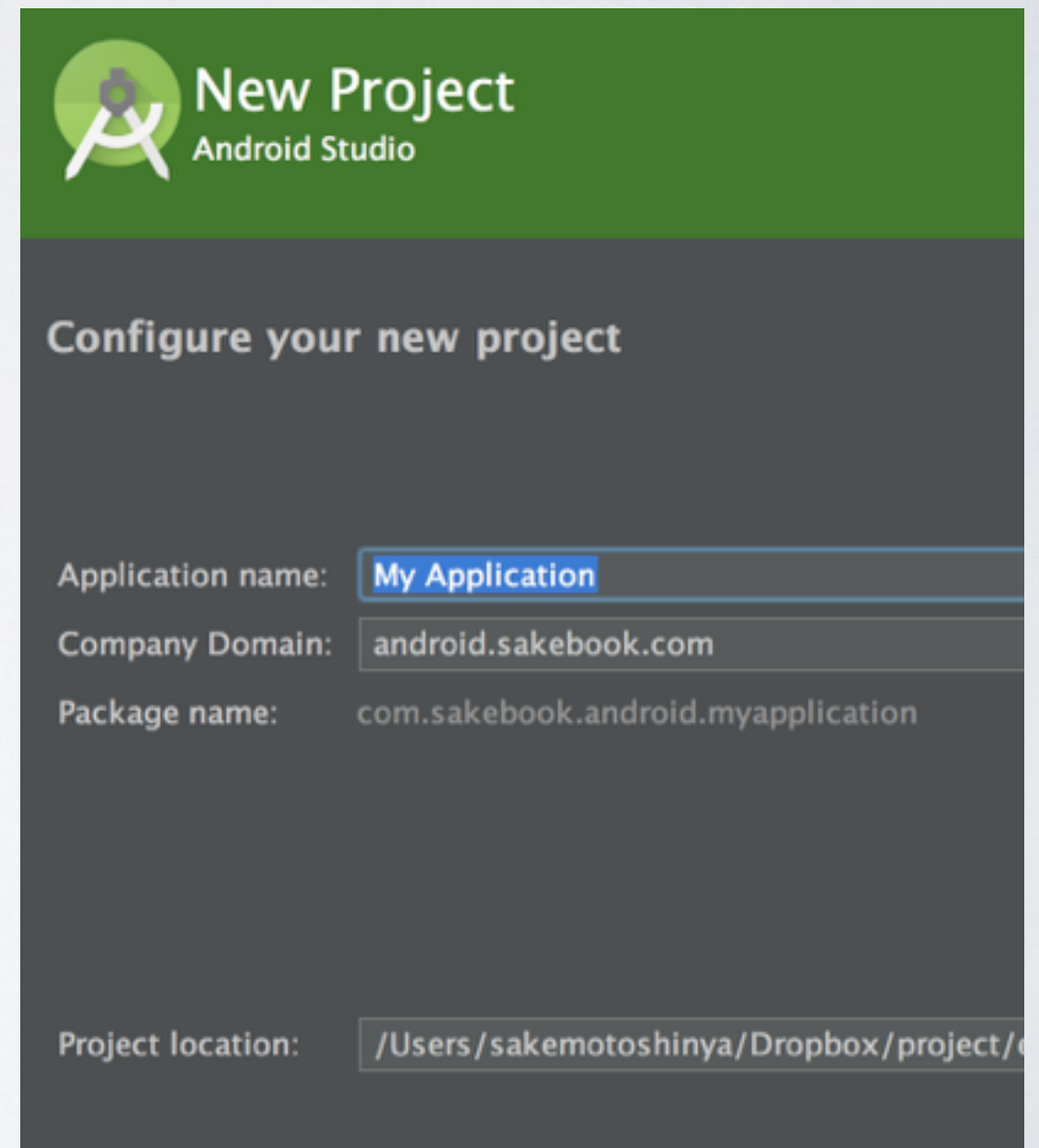
Company Domain: android.sakebook.com

Package name: com.sakebook.android.myapplication

Project location: /Users/sakemotoshinya/Dropbox/project/

# NEW PROJECT

- Package name
  - アプリをユニークにする文字列
  - 他のアプリと重複禁止
- Project location
  - PC内のプロジェクトパス



New Project  
Android Studio

Configure your new project

Application name: My Application

Company Domain: android.sakebook.com

Package name: com.sakebook.android.myapplication

Project location: /Users/sakemotoshinya/Dropbox/project/

# NEW PROJECT

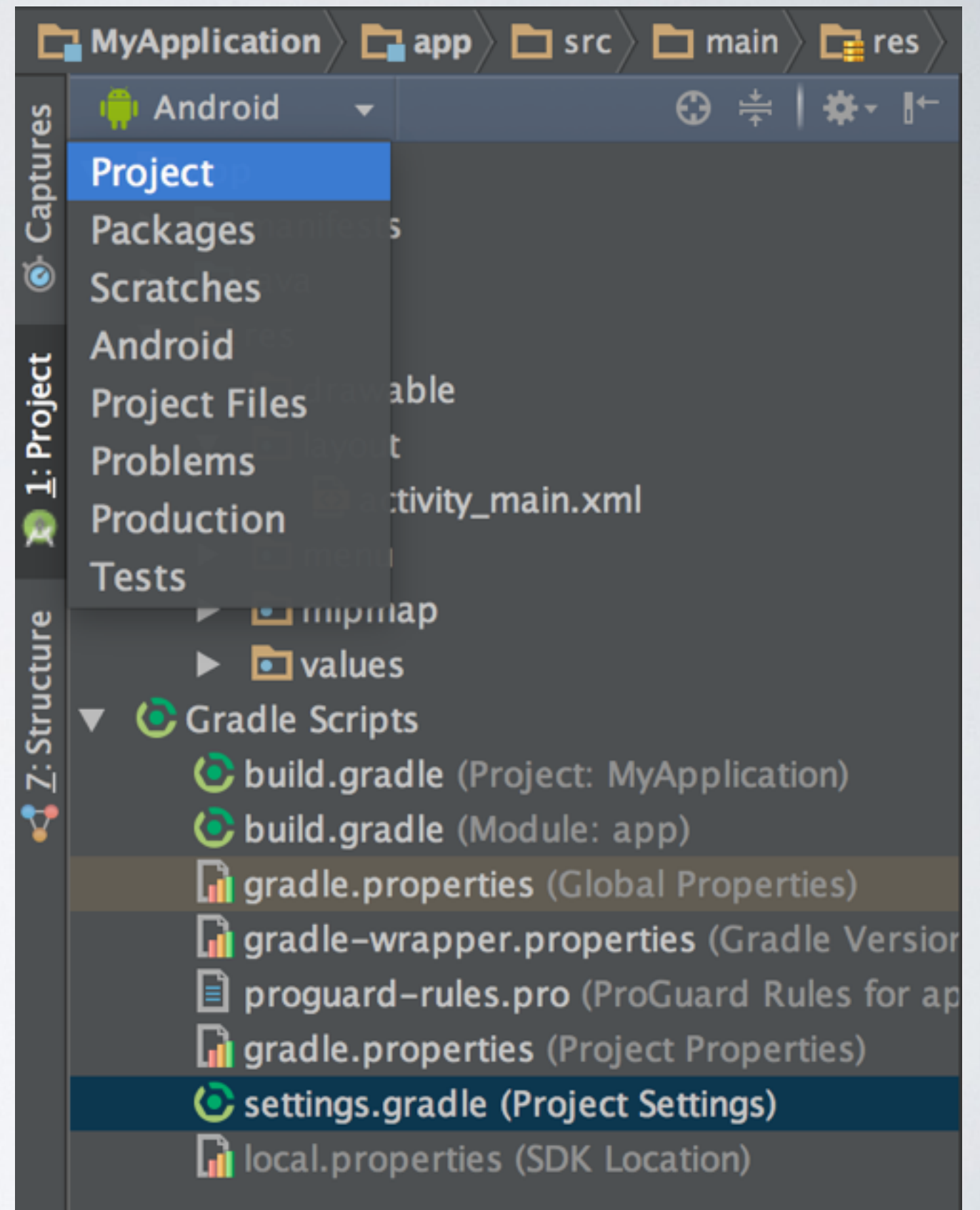
- Phone and Tablet
  - API 16(4.1)
- Empty Activity
- Activity ≡ 画面
- そのまま(後から変更可能)



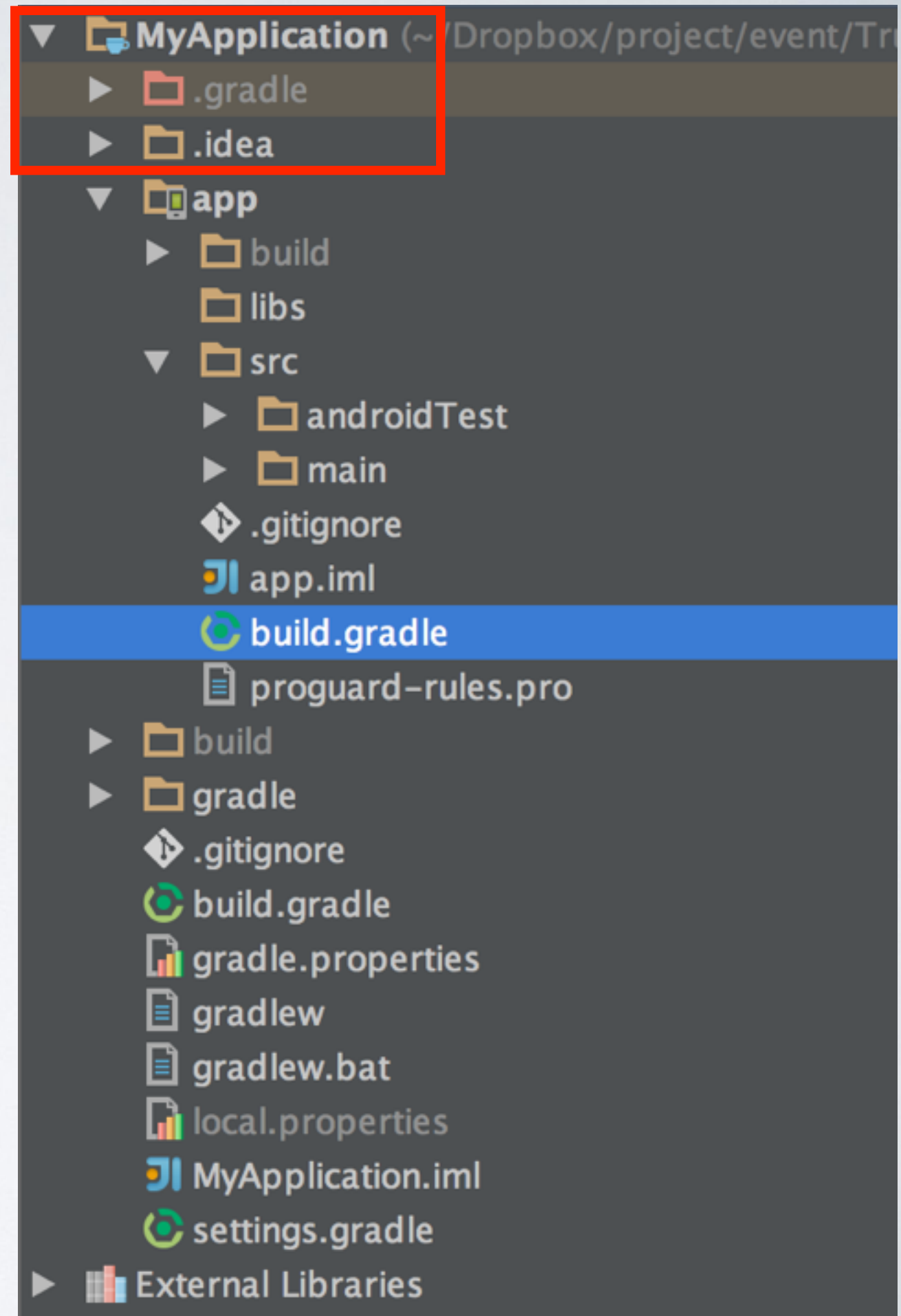
# プロジェクト構成

- プルダウンを

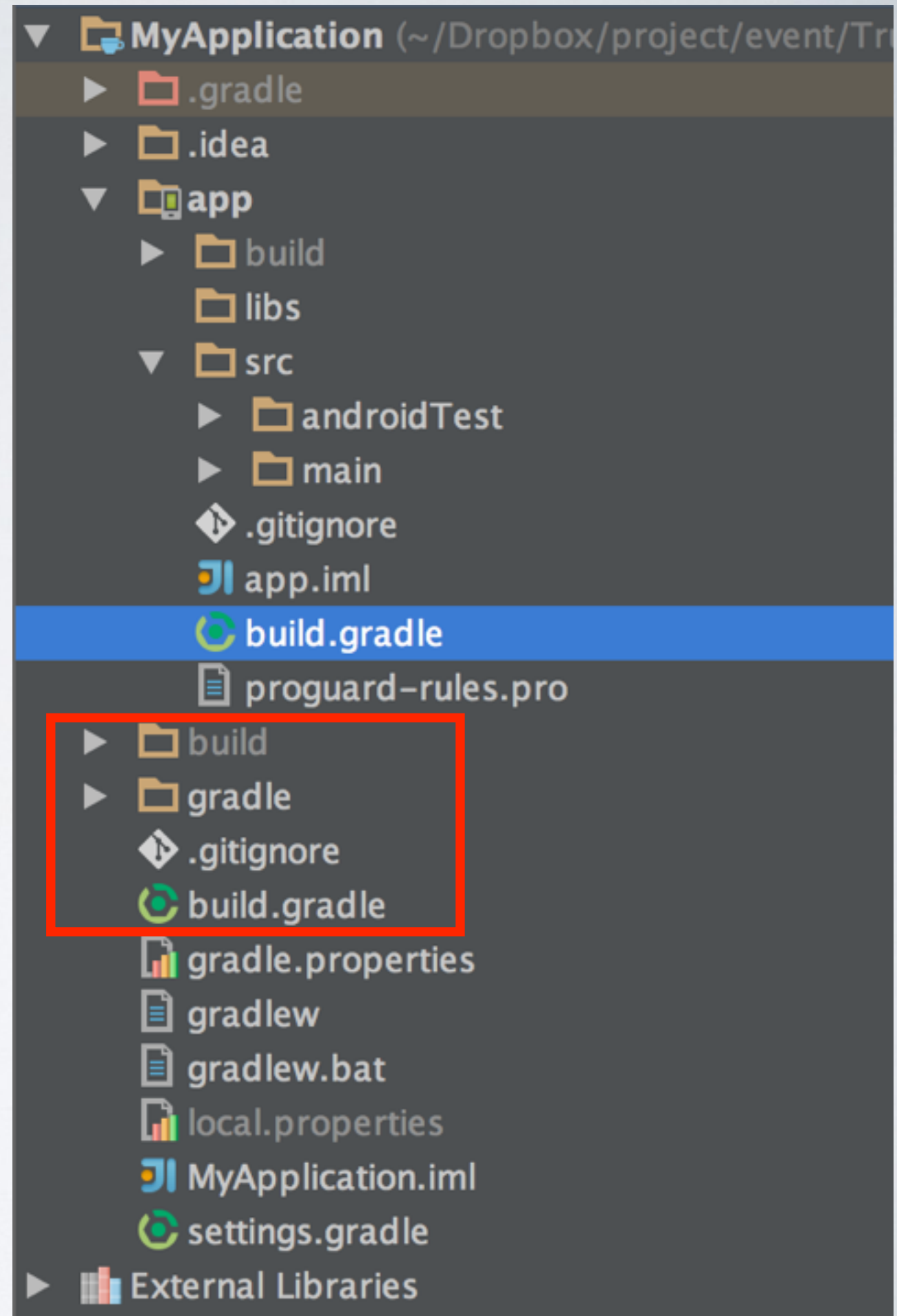
Android -> Project  
に変更



- MyApplication(Root)
- プロジェクトルート
- .gradle/
- ローカルリポジトリ
- .idea/
- IDEの設定状況

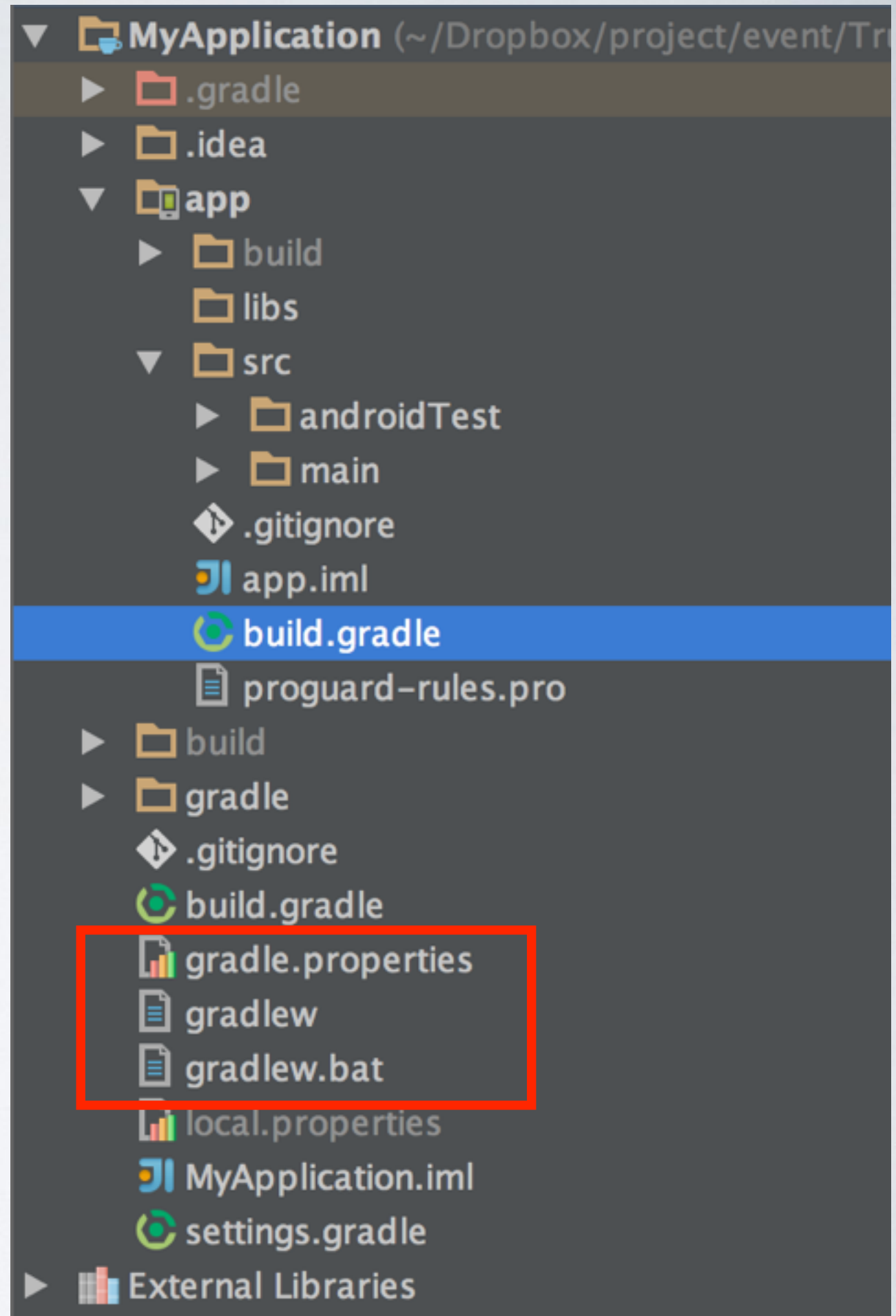


- build/
  - ビルド時の  
ファイル置き場
- gradle/
  - ビルドに必要な  
Gradle置き場
- .gitignore
  - gitで管理しないものを  
指定
- build.gradle
- プロジェクトの  
依存関係を記述

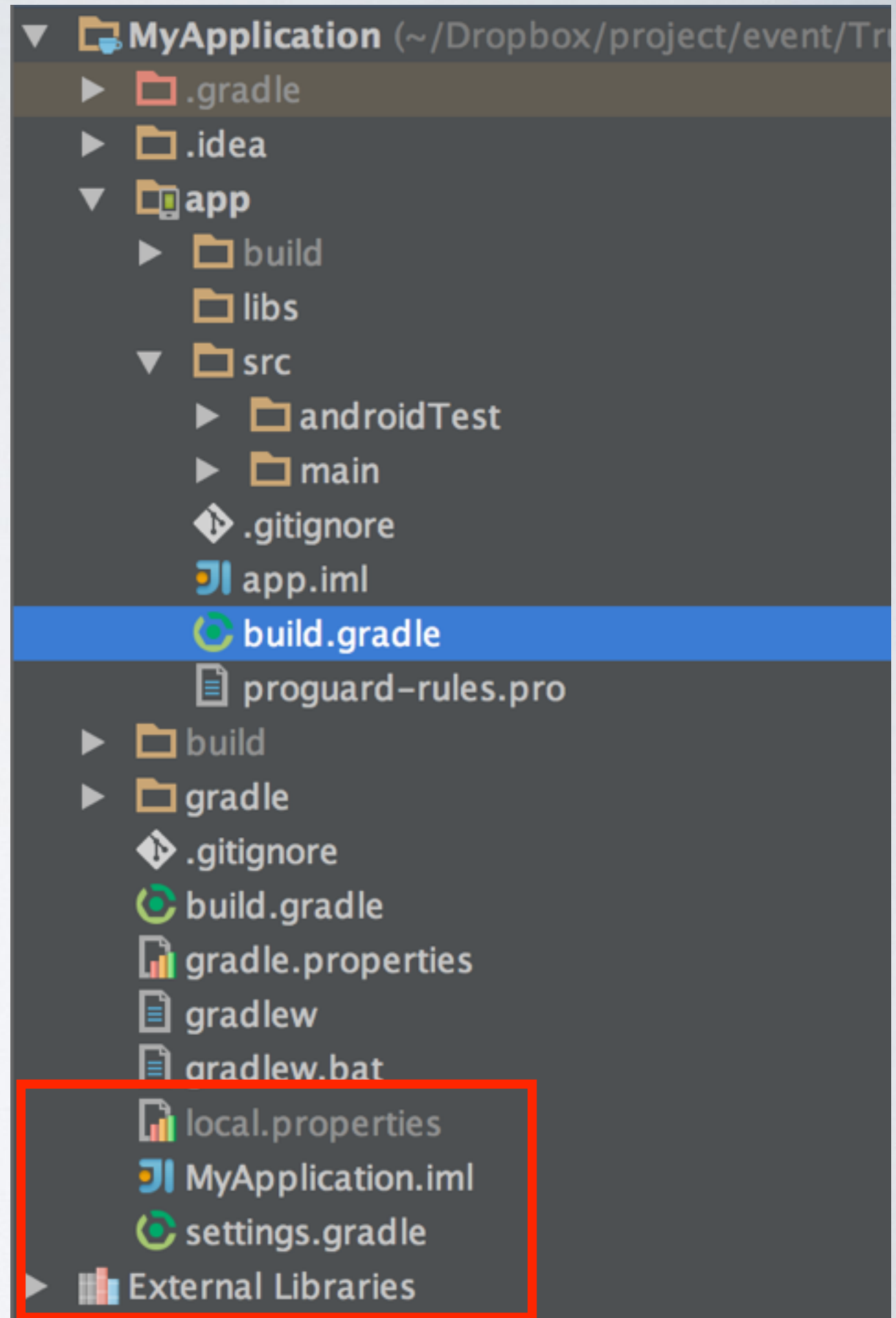




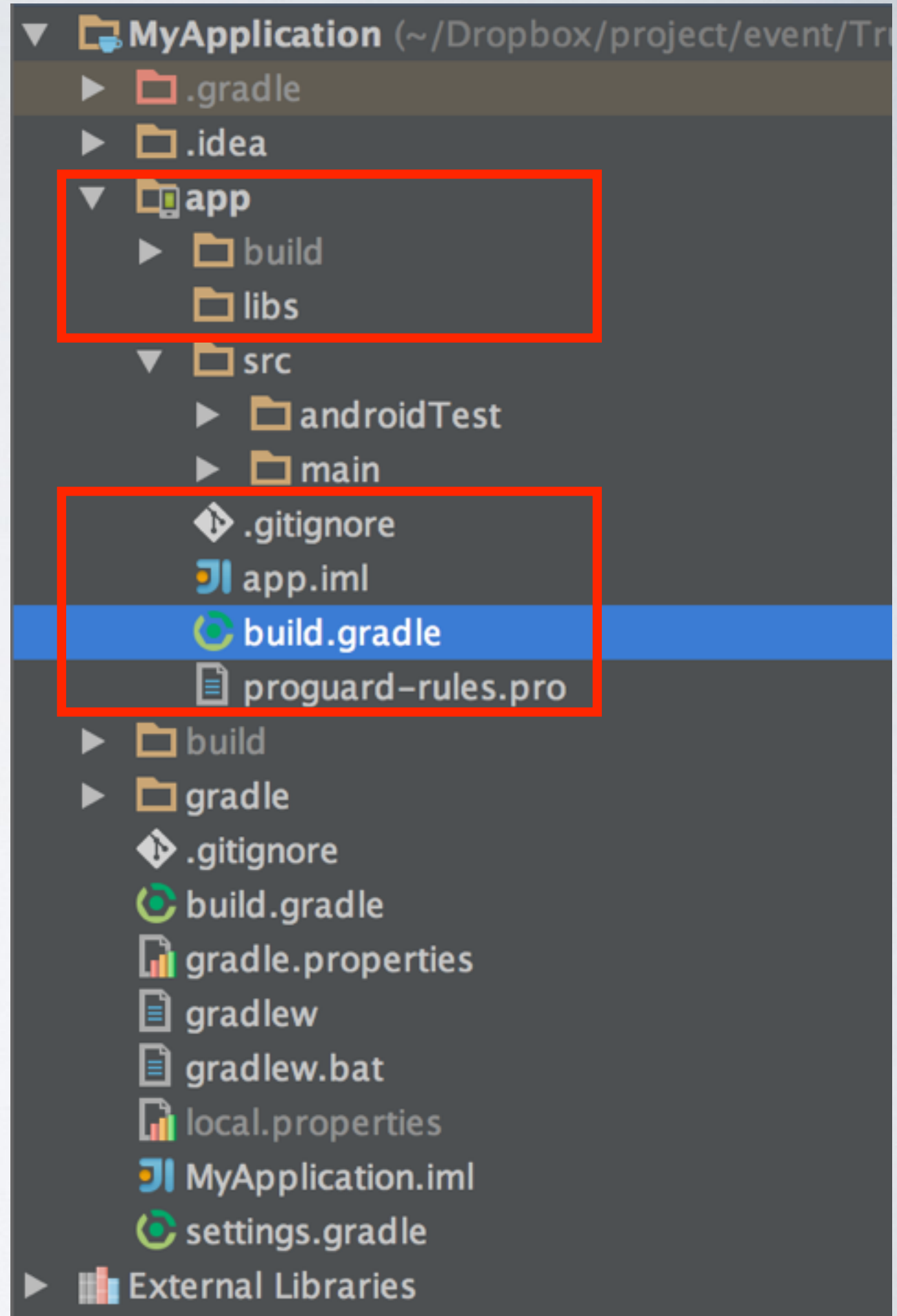
- gradle.properties
  - Gradleが  
参照するプロパティ
- gradlew
  - Linuxの  
Gradle実行ファイル
- gradlew.bat
  - Windowsの  
Gradle実行ファイル



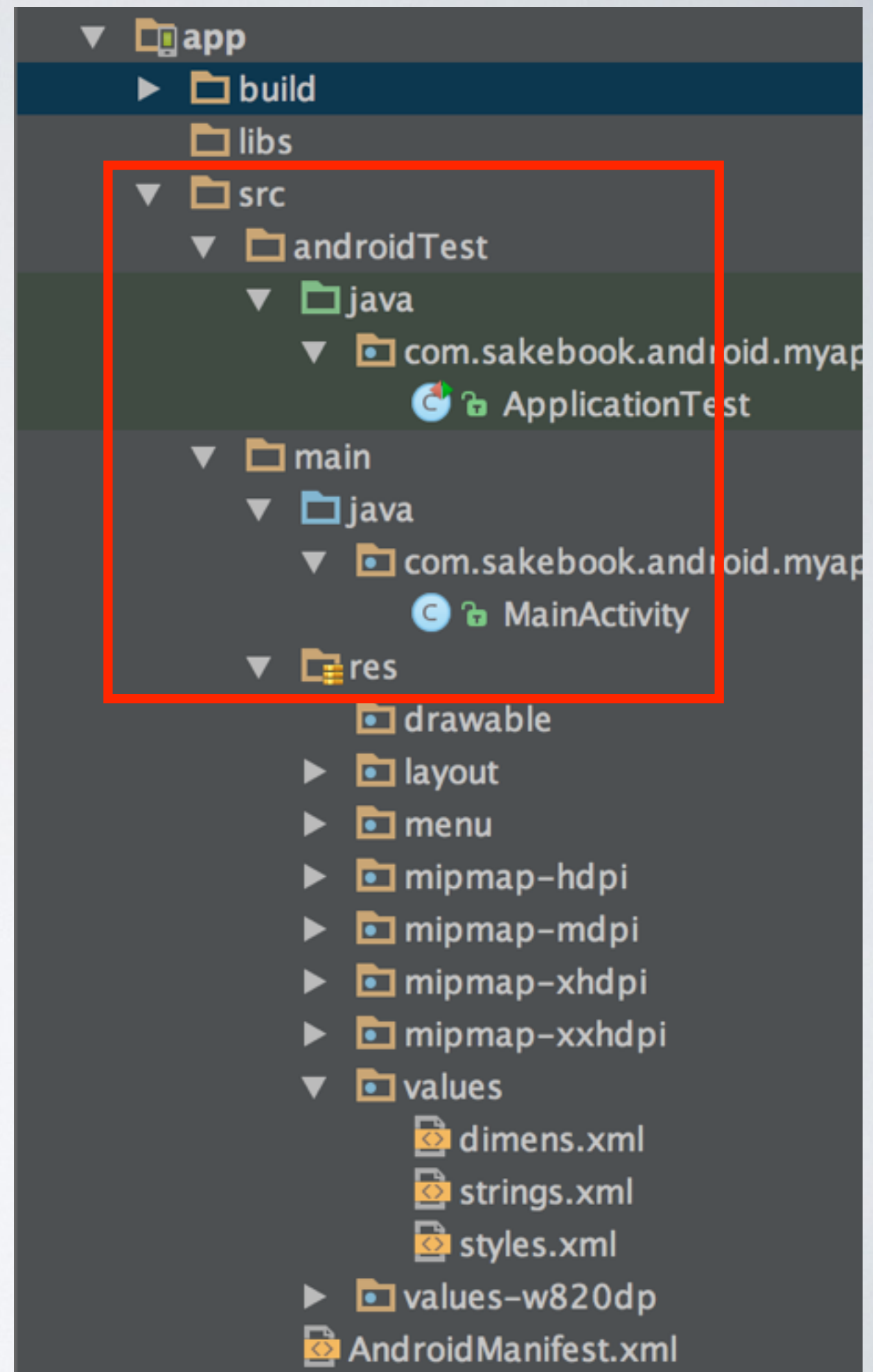
- local.properties
  - 個人の環境のSDKのパスの設定
- MyApplication.iml
  - IntelliJのファイル
- setting.gradle
  - Gradleのビルドモジュールを記述
- External Libraries
  - プロジェクトの外から参照しているライブラリ



- app/
  - モジュール
- app/libs
  - ローカルライブラリ置き場
- app/build.gradle
  - モジュールのビルドの設定や依存関係を記述
- app/proguard-rules.pro
  - 難読化に用いる

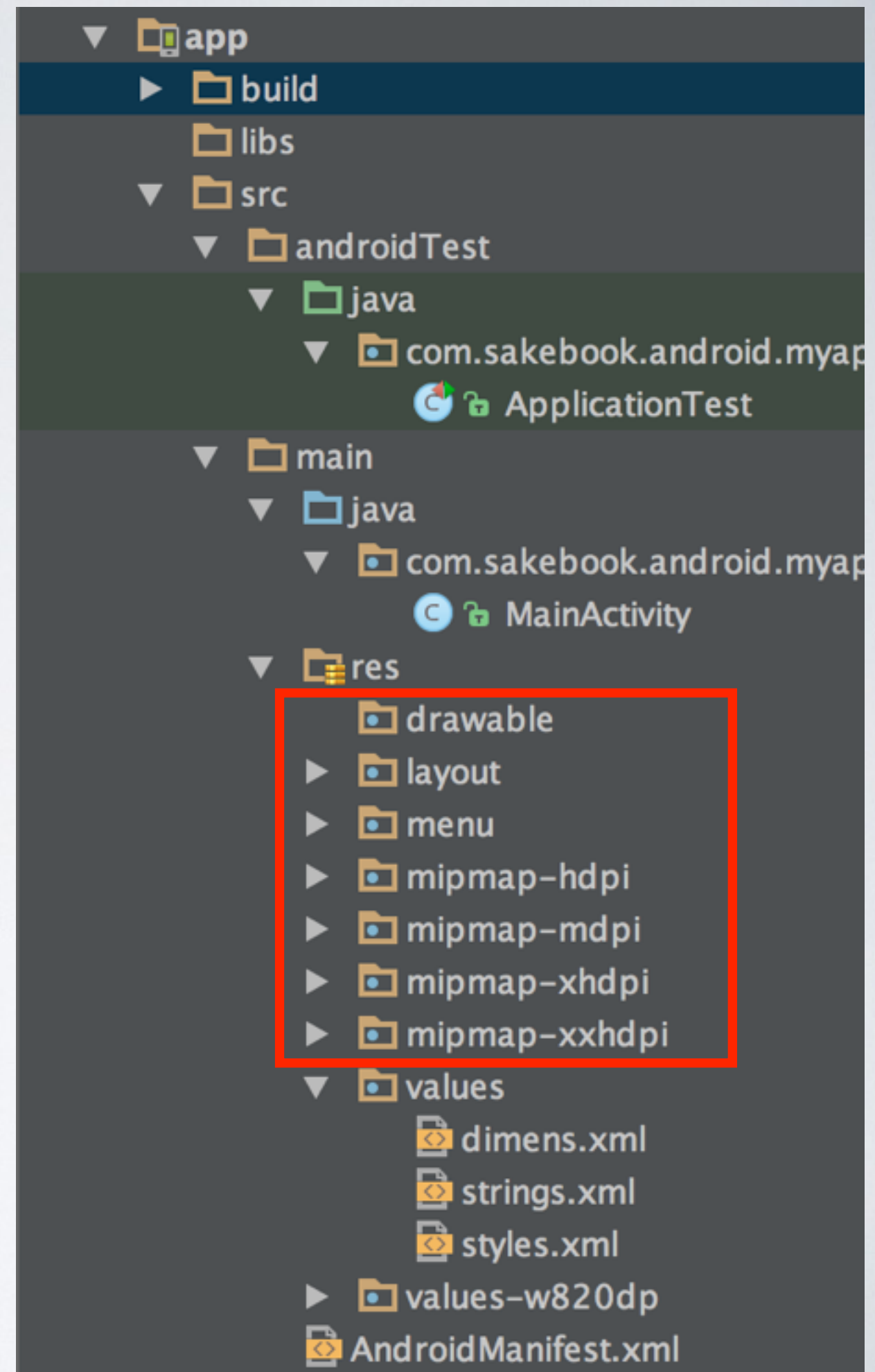


- src
  - コンパイル対象の  
ファイル置き場
- src/androidTest
  - テストファイル置き場
- src/main/java
  - ソースファイル置き場
- src/main/res
  - リソースファイル置き場

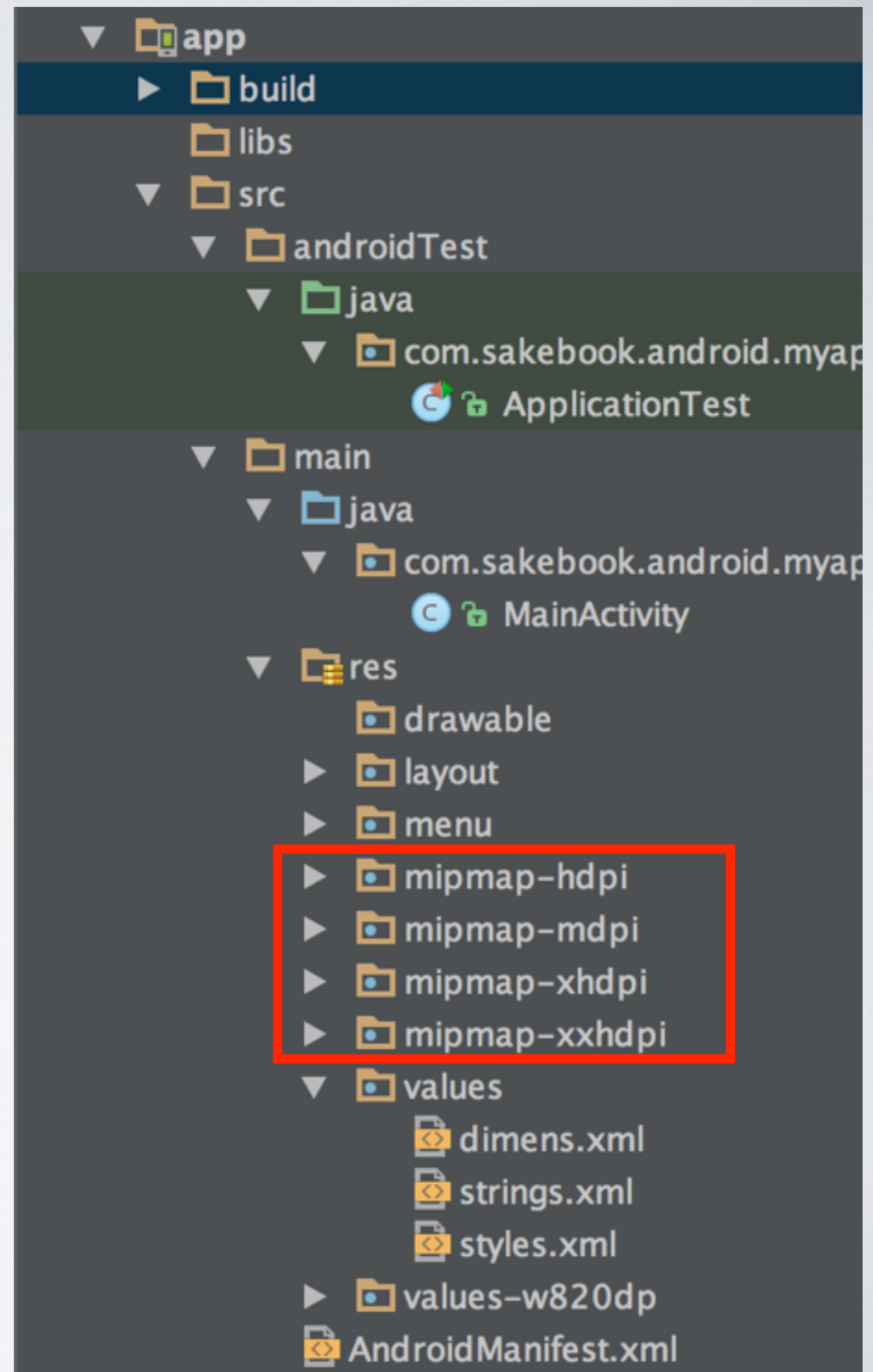




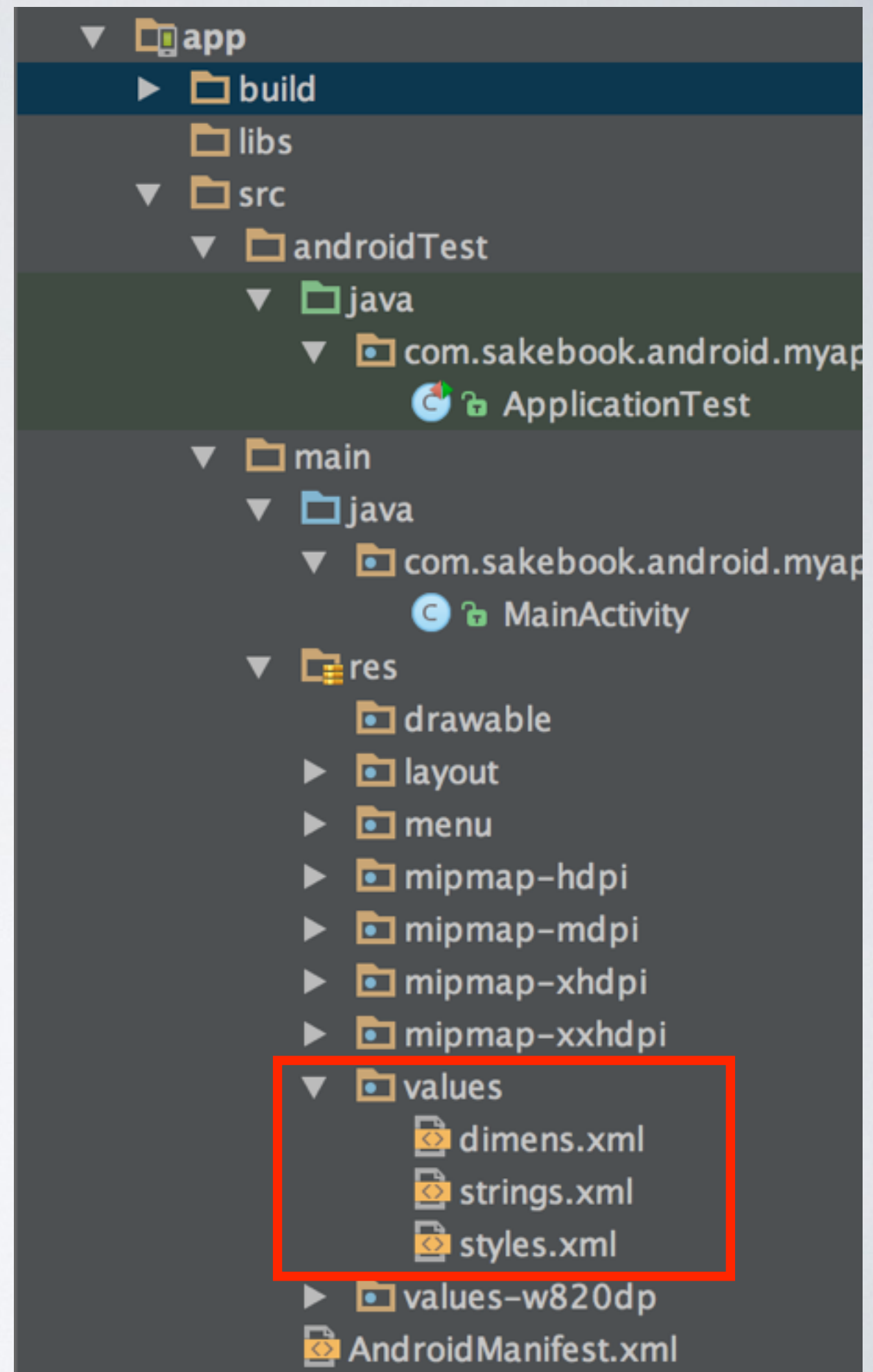
- drawable
  - 画像やxmlを置く場所
- layout
  - レイアウトのxmlを置く場所
- menu
  - メニューのxmlを置く場所
- mipmap
  - アイコン画像を置く場所



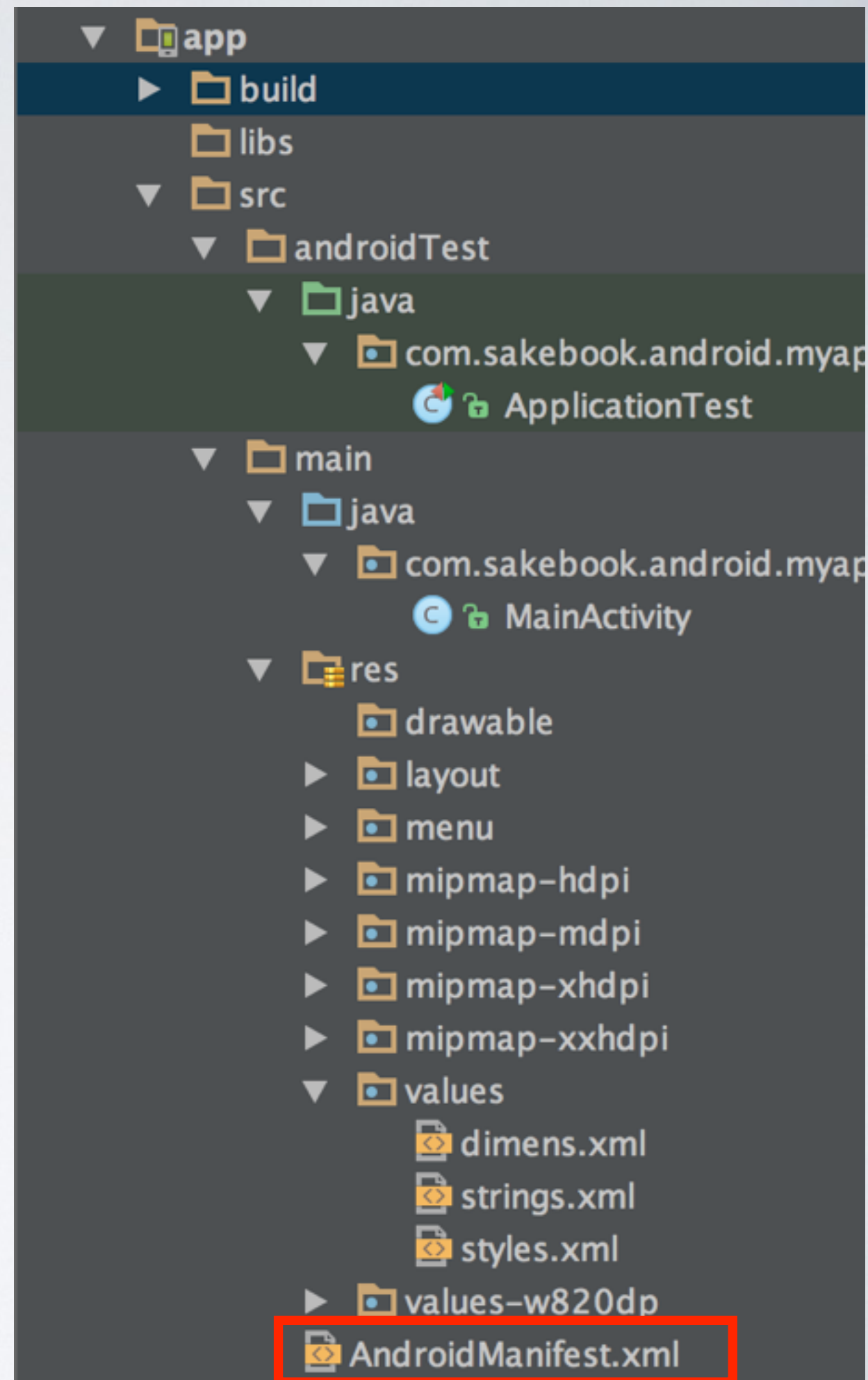
- dpi  
(Dot Per Inch: 画面密度)
- mdpi(48x48)
- hdpi(72x72)
- xhdpi(96x96)
- xxhdpi(144x144)



- `dimens.xml`
  - 文字のサイズや余白の値などを定義できる
- `strings.xml`
  - テキストを定義できる
- `styles.xml`
  - アプリのテーマを記述できる



- AndroidManifest.xml
  - アプリと端末間のPermissionやActivityなどの振る舞い、Package nameなど様々なことを設定できる





# ANDROID STUDIO SETTING

# コード補完

- 途中まで入力すると予測で  
現在利用可能な候補を表示してくれる
- Android Studio -> Preference ->  
Keymap -> 「Completion」  
(好きなコマンドを割り当てる)

# 行数表示

- ファイルの左側に現在の行数を表示してくれる
- Android Studio -> Preference -> Editor -> General -> Appearance -> 「Show line numbers」  
(チェックを入れる)

# GENERATE

- 自動生成できるものへのショートカット

(⌘+n)



# 困った時

- (Alt + Enter)でその場でできることをサジェストしてくれる

# 他にもたくさん

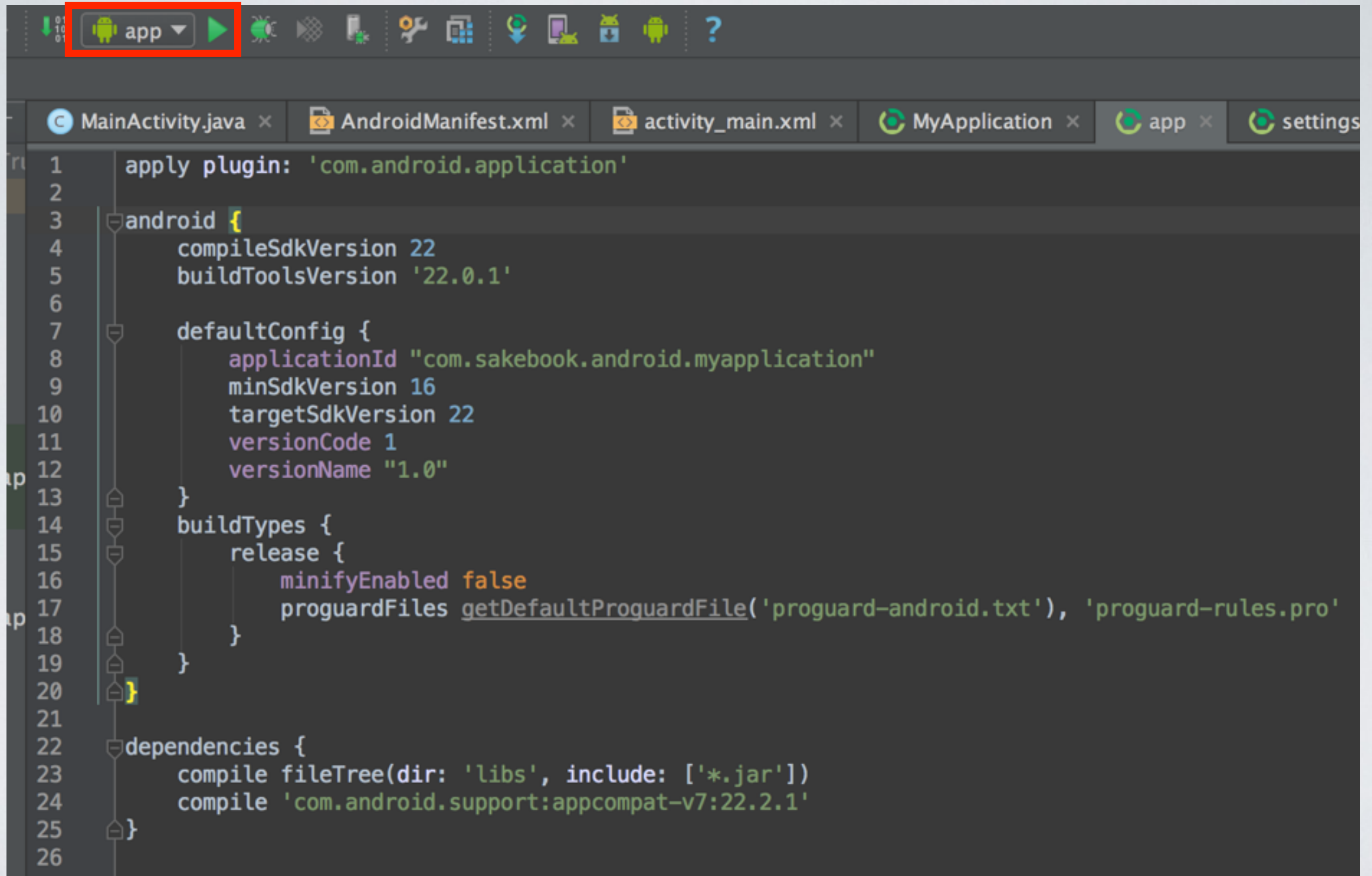
- [http://qiita.com/sugoi\\_wada/items/db449d5cbb5c83cb586c](http://qiita.com/sugoi_wada/items/db449d5cbb5c83cb586c)
- <http://qiita.com/takke/items/5cbc629f7f65d6a49906>

ビルド

# ビルドシステム

- Gradle
  - Groovyで記述できる
  - setting.gradleで対象となるモジュールを指定
  - app/build.gradleにビルド時のタスクを記述
- コマンドライン or ASの機能でビルド実行





# コマンドライン

- `./gradlew assembleDebug`
- BuildTypeがDebugでビルドされたapkを  
app/build/outputs/apk に作成
- `./gradlew installDebug`
- 対象のデバイスにapkをインストール

実機 OR エミュレータ

# 実機がなければエミュレータ

- 純正のものはあくびが出るほど遅い
  - 2.xから早くなるらしい。
- Genymotionを使う



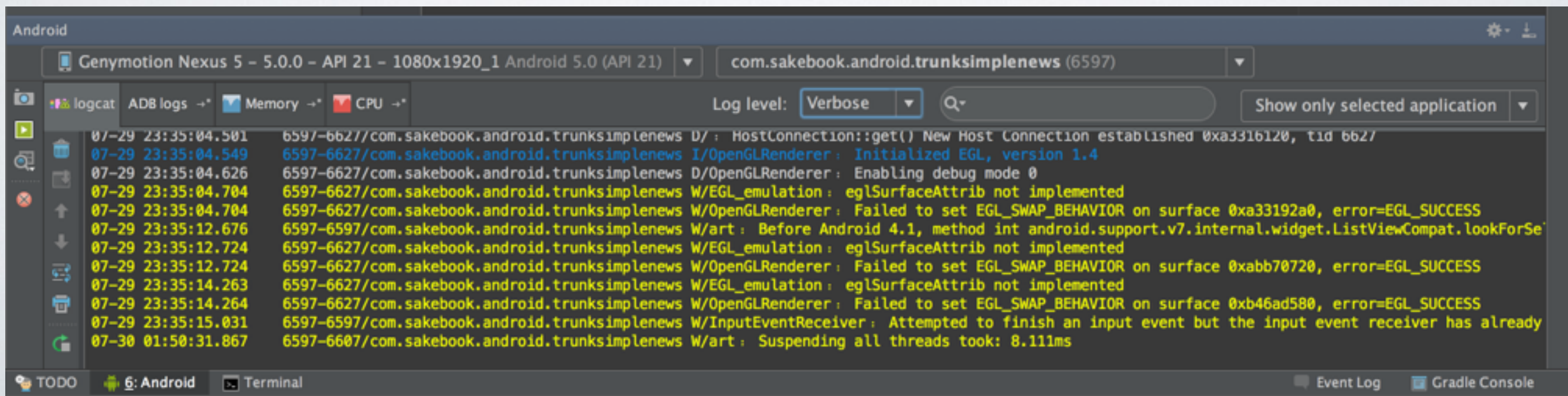
# GENYMOTION

- 爆速のエミュレータ
- 無料版だと制限があるが、  
通常利用なら十分
- <https://www.genymotion.com/>

LOGCAT

# LOGCAT

- 接続されているデバイスが流しているログを  
収集・表示する機能



# 表示させる

- やっとコードに触る！？
- 14行目に

```
Log.d("Trunk", "onCreate");
```



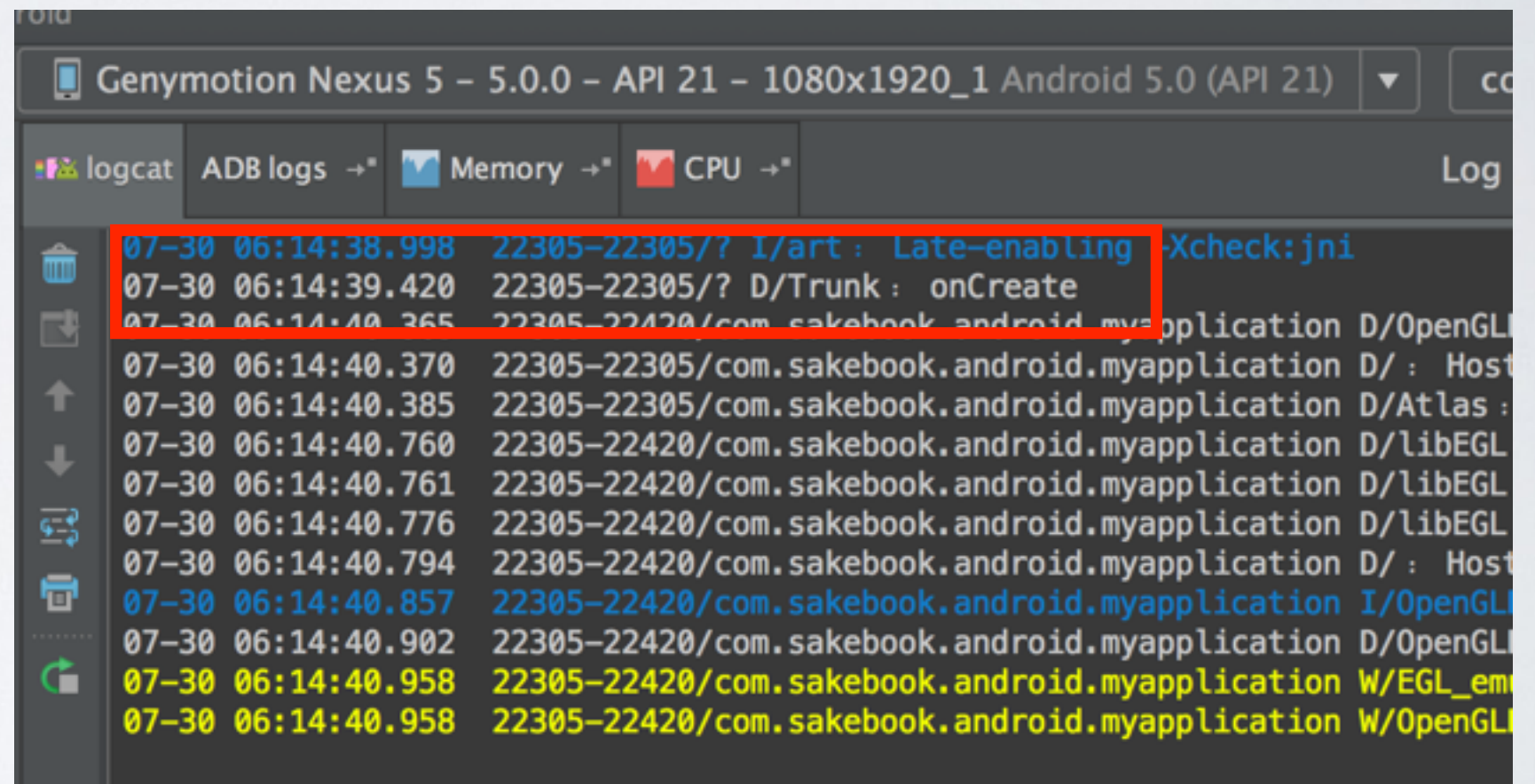
# こういうときが

- Alt + Enterの出番！

```
10      @Override
11      protected void onCreate(Bundle savedInstanceState) {
12          ? android.util.Log? ↵↵
13          setContentView(R.layout.activity_main);
14          Log.d("MainActivity", "onCreate");
15      }
16
17      @Override
```

# 表示された

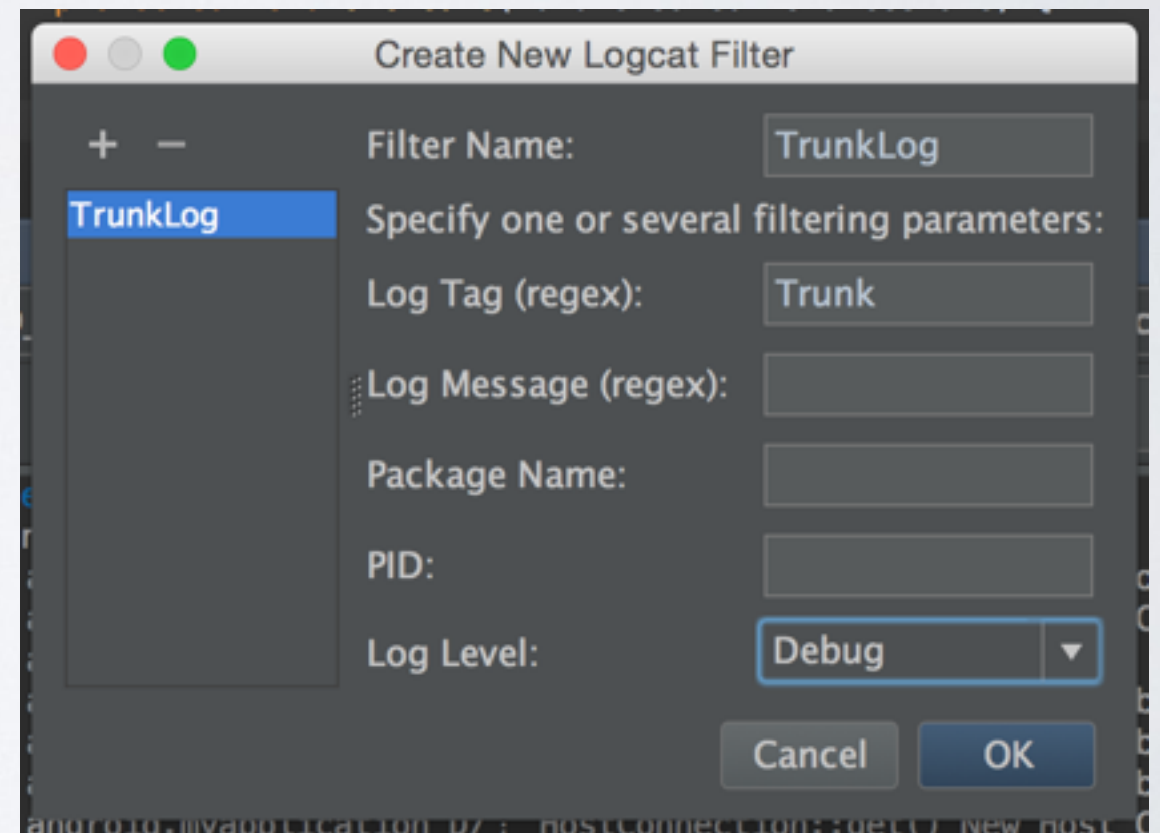
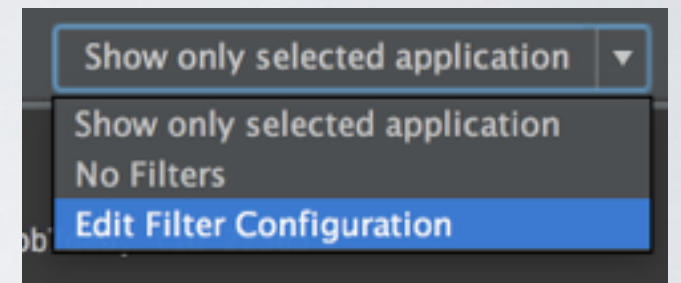
- 見分けが  
つくように  
なった



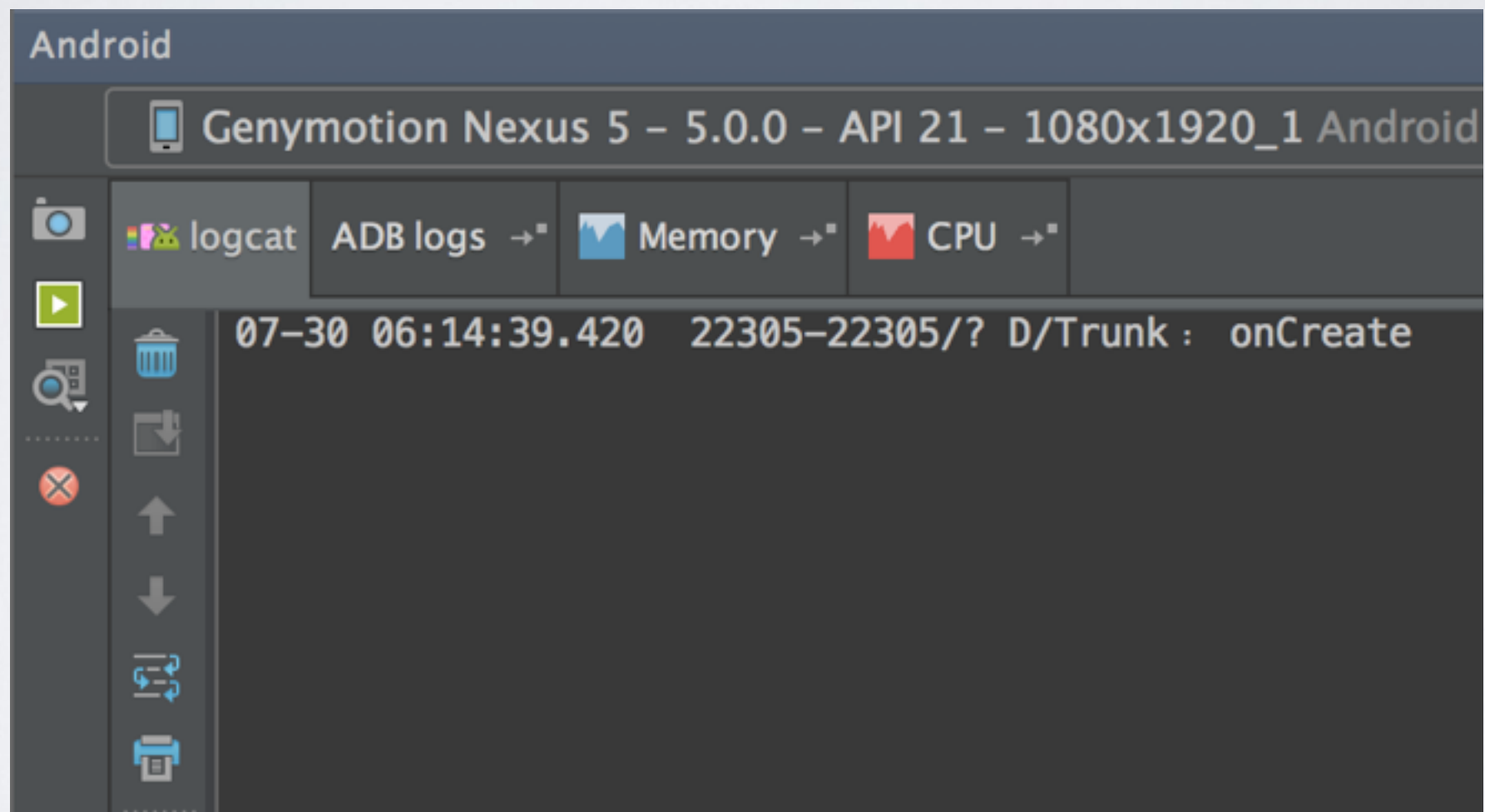
```
Genymotion Nexus 5 - 5.0.0 - API 21 - 1080x1920_1 Android 5.0 (API 21)
logcat ADB logs → Memory → CPU → Log
07-30 06:14:38.998 22305-22305/? I/art: Late-enabling -Xcheck:jni
07-30 06:14:39.420 22305-22305/? D/Trunk: onCreate
07-30 06:14:40.365 22305-22420/com.sakebook.android.myapplication D/OpenGL
07-30 06:14:40.370 22305-22305/com.sakebook.android.myapplication D/: Host
07-30 06:14:40.385 22305-22305/com.sakebook.android.myapplication D/Atlas:
07-30 06:14:40.760 22305-22420/com.sakebook.android.myapplication D/libEGL
07-30 06:14:40.761 22305-22420/com.sakebook.android.myapplication D/libEGL
07-30 06:14:40.776 22305-22420/com.sakebook.android.myapplication D/libEGL
07-30 06:14:40.794 22305-22420/com.sakebook.android.myapplication D/: Host
07-30 06:14:40.857 22305-22420/com.sakebook.android.myapplication I/OpenGL
07-30 06:14:40.902 22305-22420/com.sakebook.android.myapplication D/OpenGL
07-30 06:14:40.958 22305-22420/com.sakebook.android.myapplication W/EGL_emo
07-30 06:14:40.958 22305-22420/com.sakebook.android.myapplication W/OpenGL
```

# 邪魔なものを排除したい

- Filterを作成する



# 見やすくなった！





# 使い分け

- `Log.v(String tag, String msg)` 「冗長」
- `Log.d(String tag, String msg)` 「デバッグ」
- `Log.i(String tag, String msg)` 「情報」
- `Log.w(String tag, String msg)` 「警告」
- `Log.e(String tag, String msg)` 「エラー」

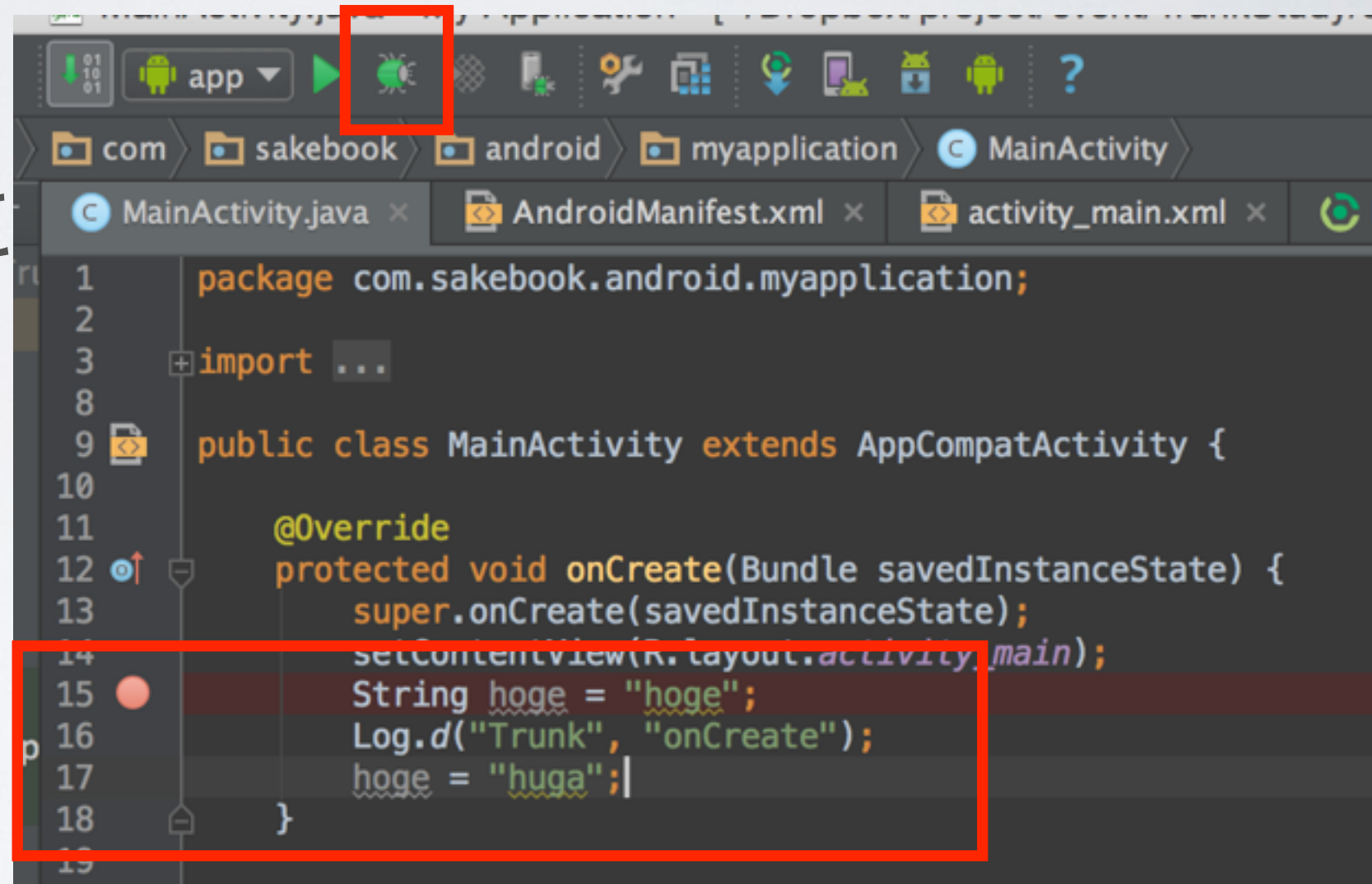
BREAKPOINT

# BREAK POINT

- Logcatだとクラッシュした場所で、他の値がどうなっているかなどがわからない
- クラッシュする付近から動きをつぶさに追いたい！

# DEBUG

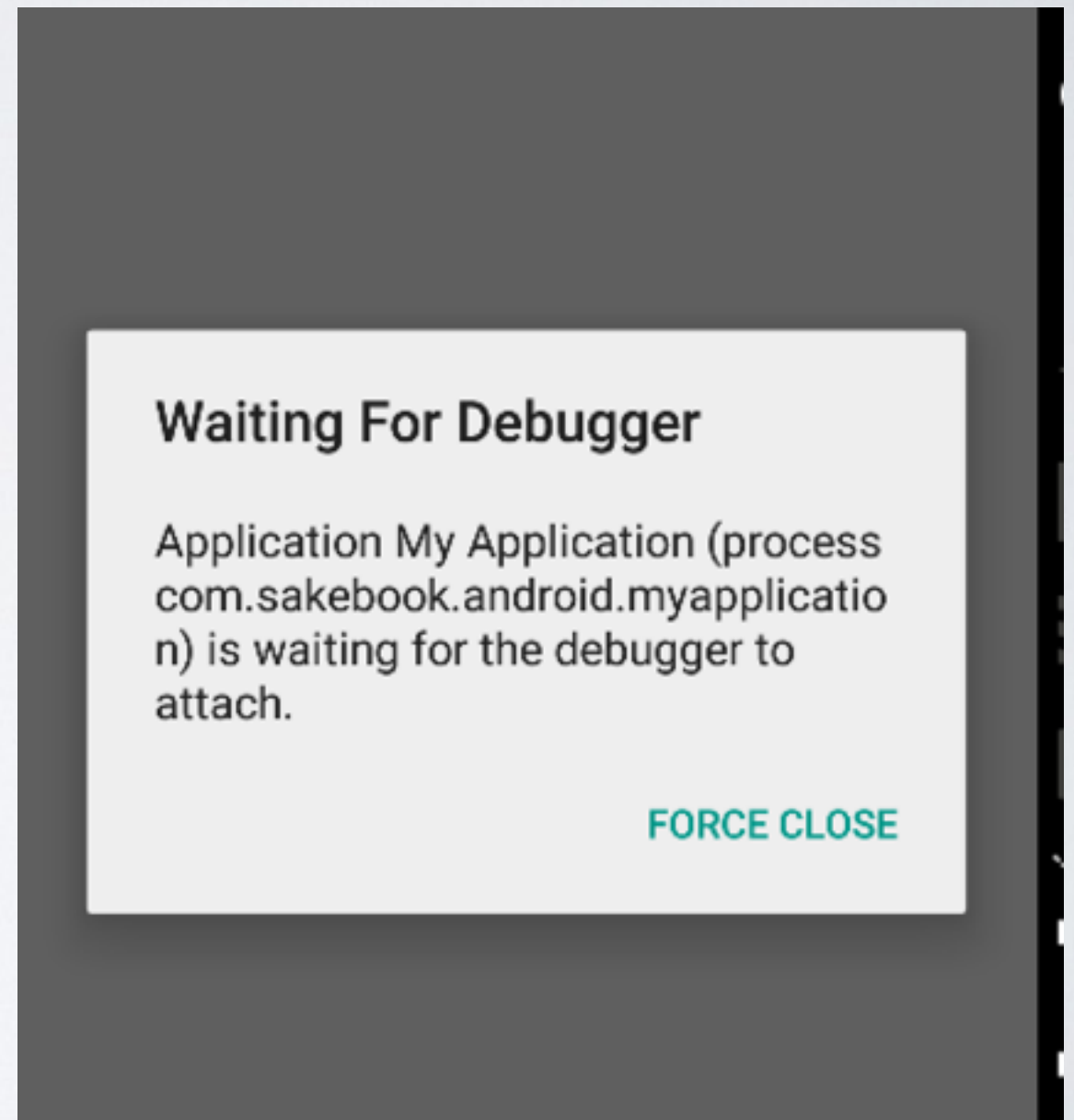
- プログラムを  
止めたい場所に  
赤丸を設置
- 「hoge」の  
値を試みる





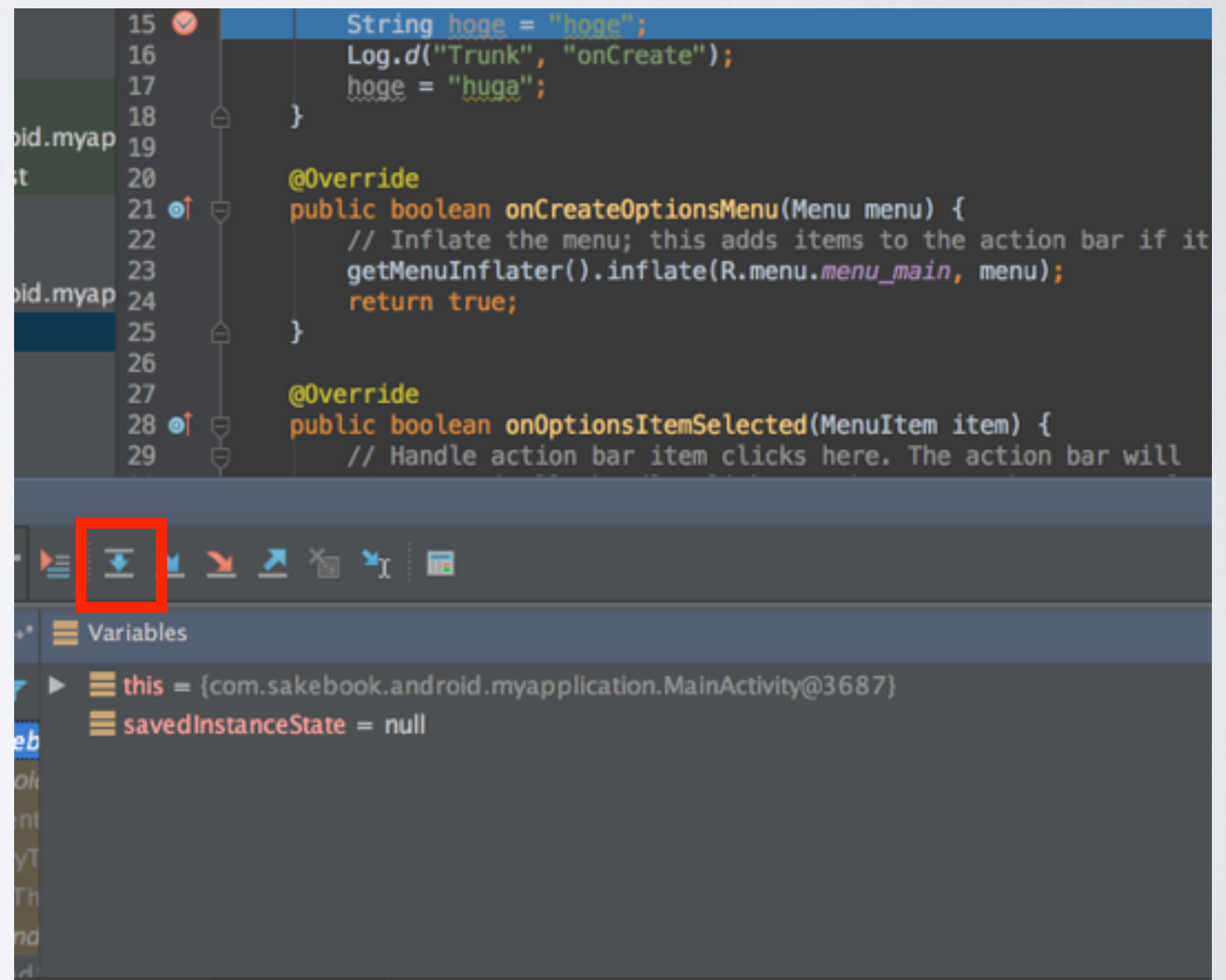
# お使いのデバイスは正常です

- デバイス側に表示がでるが待つ

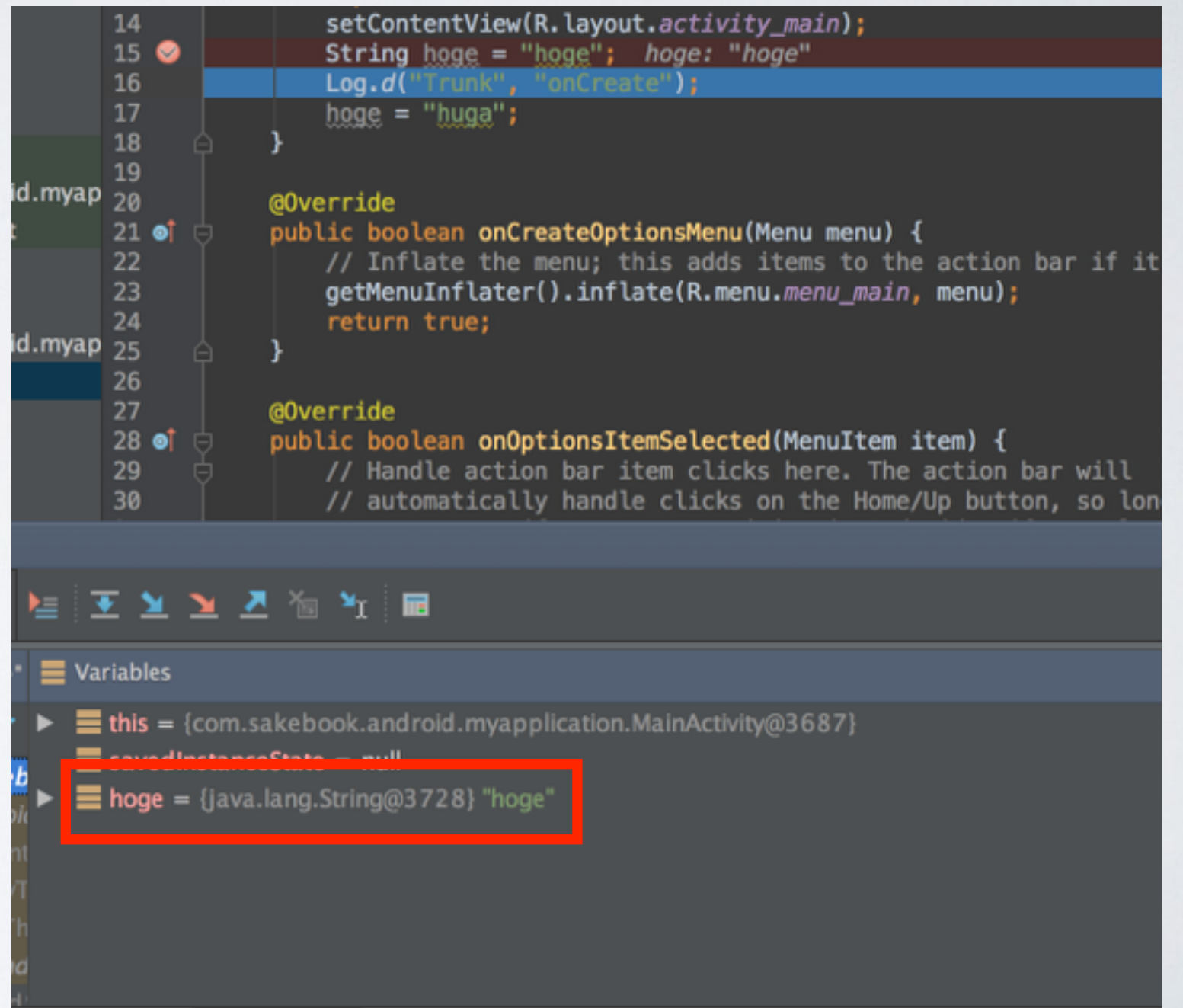


# ステップオーバー

- 15行目でBreak
- 止まった行は  
その時点では  
評価されない



- 「hoge」が定義されている



The screenshot shows the Android Studio IDE. The top pane displays Java code for an Android application. The code includes the following lines:

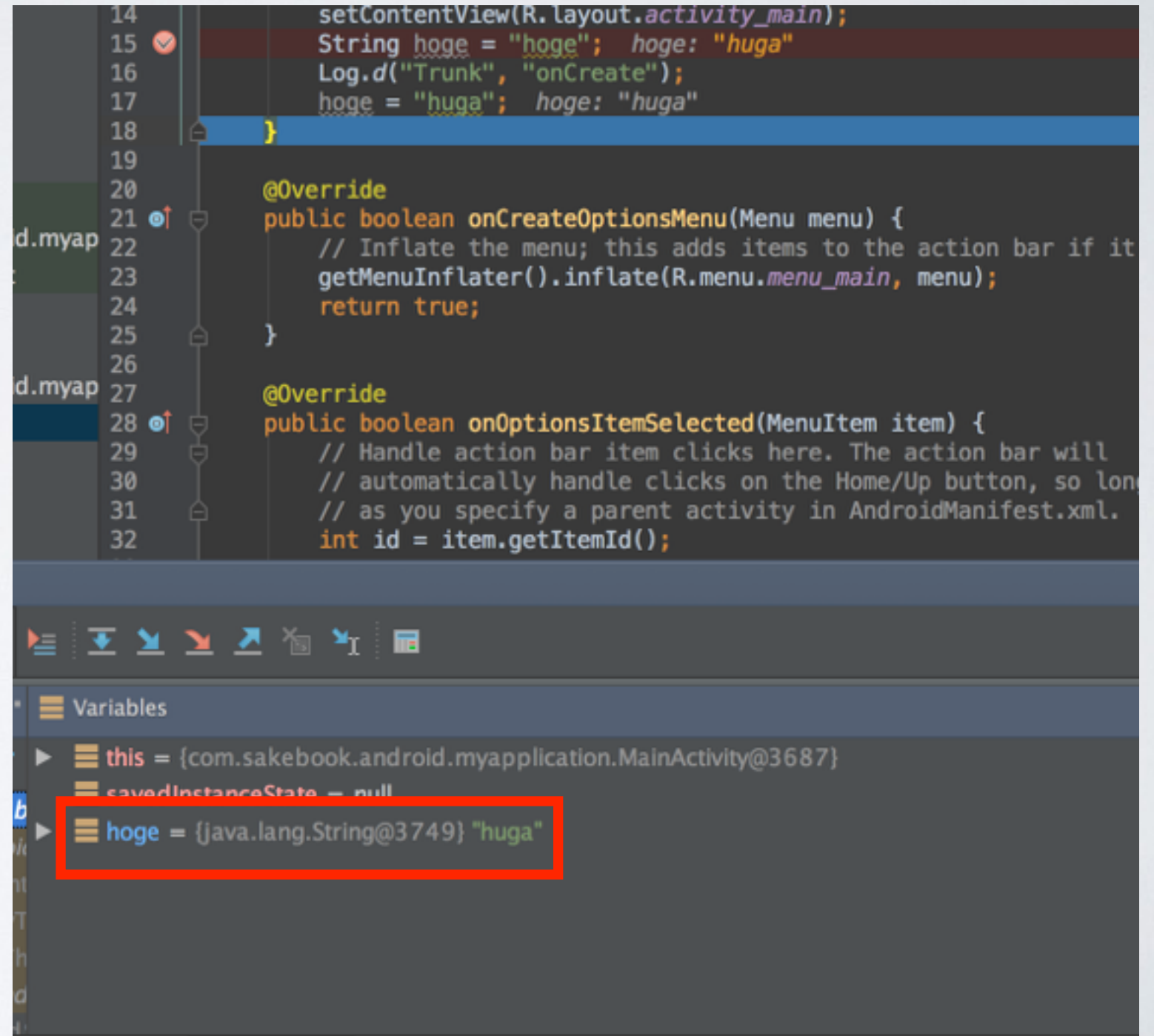
```
14 setContentView(R.layout.activity_main);
15 String hoge = "hoge"; hoge: "hoge"
16 Log.d("Trunk", "onCreate");
17 hoge = "huga";
18 }
19
20 @Override
21 public boolean onCreateOptionsMenu(Menu menu) {
22     // Inflate the menu; this adds items to the action bar if it
23     getMenuInflater().inflate(R.menu.menu_main, menu);
24     return true;
25 }
26
27 @Override
28 public boolean onOptionsItemSelected(MenuItem item) {
29     // Handle action bar item clicks here. The action bar will
30     // automatically handle clicks on the Home/Up button, so lon
```

The bottom pane shows the 'Variables' tab. It lists the current state of variables:

- `this` = {com.sakebook.android.myapplication.MainActivity@3687}
- `savedInstanceState` = null
- `hoge` = {java.lang.String@3728} "hoge"

The `hoge` variable is highlighted with a red rectangle.

- 「hoge」 の  
中身が""huga""に  
なっている



The screenshot shows the Android Studio IDE with a Java file named `d.myap`. The code is as follows:

```
14 setContentView(R.layout.activity_main);
15 String hoge = "hoge"; hoge: "huga"
16 Log.d("Trunk", "onCreate");
17 hoge = "huga"; hoge: "huga"
18 }
19
20 @Override
21 public boolean onCreateOptionsMenu(Menu menu) {
22     // Inflate the menu; this adds items to the action bar if it
23     getMenuInflater().inflate(R.menu.menu_main, menu);
24     return true;
25 }
26
27 @Override
28 public boolean onOptionsItemSelected(MenuItem item) {
29     // Handle action bar item clicks here. The action bar will
30     // automatically handle clicks on the Home/Up button, so long
31     // as you specify a parent activity in AndroidManifest.xml.
32     int id = item.getItemId();
```

Below the code editor, the **Variables** panel is visible. It shows the following variables:

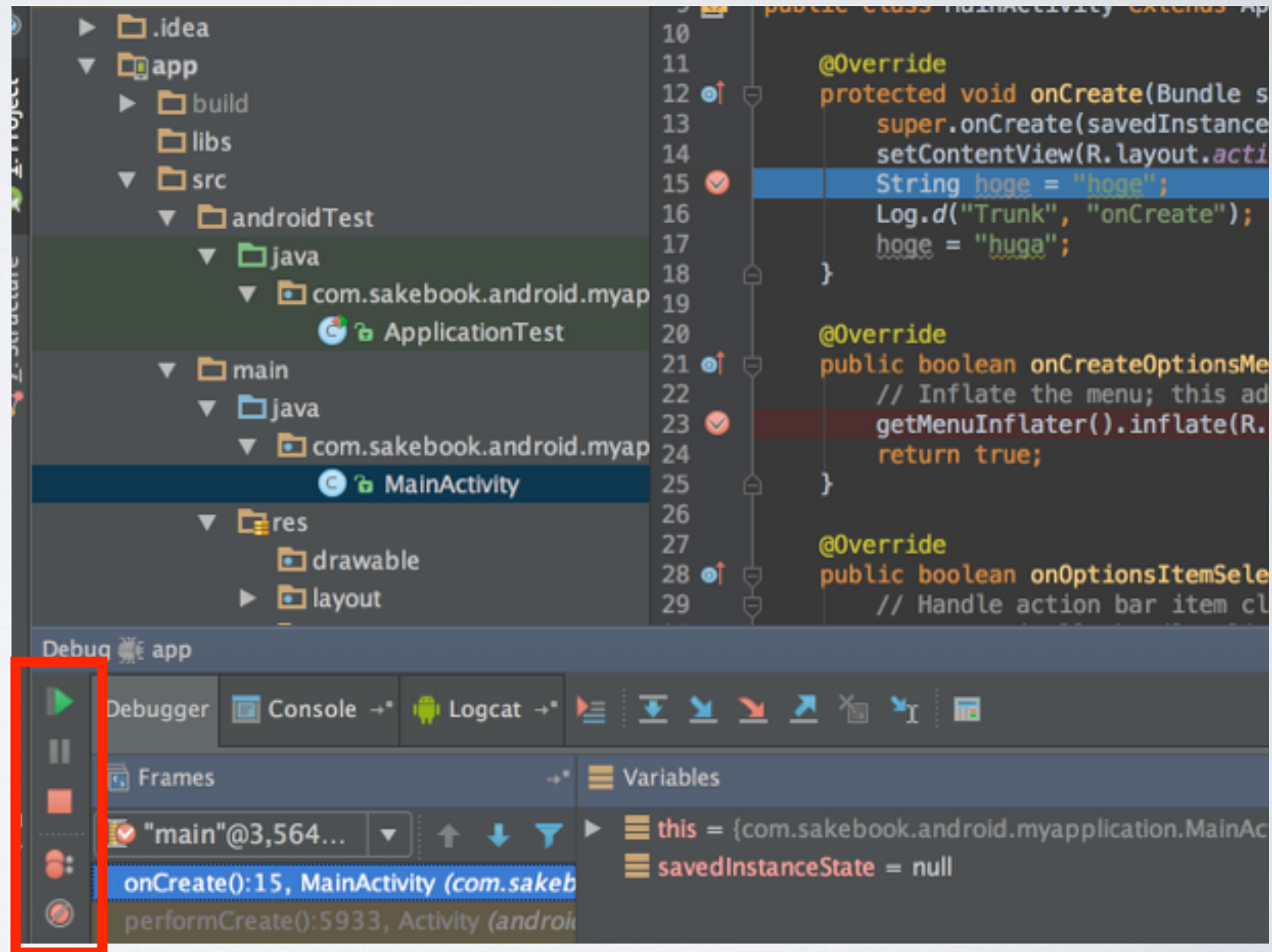
- `this` = {com.sakebook.android.myapplication.MainActivity@3687}
- `savedInstanceState` = null
- `hoge` = {java.lang.String@3749} "huga"

The `hoge` variable is highlighted with a red rectangle, indicating its current value is "huga".



# 複数設置した時

- 現在の  
Breakpointから  
抜けたり接続を  
切ることも  
できる



# ソースコード管理

# GIT

- 分散型バージョン管理システム
- GitHub
  - Gitを用いたWebサービス

# よく使うコマンド

- `git status`
- `git diff` ファイル
- `git checkout` ファイル
- `git add` ファイル
- `git commit -m “メッセージ”`
- `git pull “リモートリポジトリのURL” “ローカルのブランチ名”`
- `git push “リモートリポジトリのURL” “ローカルのブランチ名”`



- git status

- 現在のブランチで  
変更があるかどうかを確認

- git diff ファイル

- 対象のファイルの  
どこに変更があるのかを確認

- git checkout ファイル
  - 対象のファイルのローカルリポジトリでの変更を戻す
- git add ファイル
  - 対象のファイルをcommit対象にする

- `git commit -m “メッセージ”`
  - `commit`対象にしたファイルたちを  
`push`対象にする
- `git pull “リモートリポジトリのURL” “ローカルの  
ブランチ名”`
  - リモートリポジトリの変更をローカルに反  
映させる

- git push “リモートリポジトリのURL” “ローカルのブランチ名”
  - push対象にしたファイルたちをリモートリポジトリへpushする



# ~/.GITCONFIG

[user]

name = Shinya Sakemoto

email = sakebook@gmail.com

[color]

ui = auto

[alias]

st = status

br = branch -a

ch = checkout

cm = commit

stt = status -uno

difff = diff --word-diff

- よく使うものは  
登録しておく と便利

詳しくは

- <http://blog.nanapi.co.jp/tech/2014/04/23/git-love/>

# GITHUBに登録しよう

- 先ほど作成したプロジェクトを登録
- GitHubで適当なリポジトリ作成
- `$git remote add origin リポジトリ`

ADOBE CREATIVE CLOUD



# SDK

- Software Development Kit
- 今回だとAdobeが提供している機能を、アプリ内に組み込むことができる。
- <https://creativesdk.adobe.com/docs/android/#/articles/gettingstarted/index.html>
  - 一緒に設定しましょう