

# 実践 ANDROIDアプリ開発

@JX通信社

酒本伸也

2015/9/18

RETROFIT  
+  
PICASSO

# APIを叩いてよしなに画像取得

- part2の資料の補足と復習

# 準備するもの(NO CODE)

- APIのエンドポイント
- APIのリクエストパラメータ
- APIのレスポンス形式

- APIのエンドポイント
  - <https://qiita.com/api/v1/>
- APIのリクエストパラメータ
  - 特定タグの投稿取得
  - GET /api/v1/tags/:url\_name/items
    - :url\_name 「Android」 で試す



- APIのレスポンス形式

- <https://qiita.com/api/v1/api/v1/tags/Android/items>

- json

- xmlはコンバータが必要

- (<http://qiita.com/tsuyosh/items/7d38f41b2cfb4a6ae24b>)

# 準備するもの(CODE)

- モデルを作成
- インタフェースを作成
- クライアントを作成

PART2の資料で復習

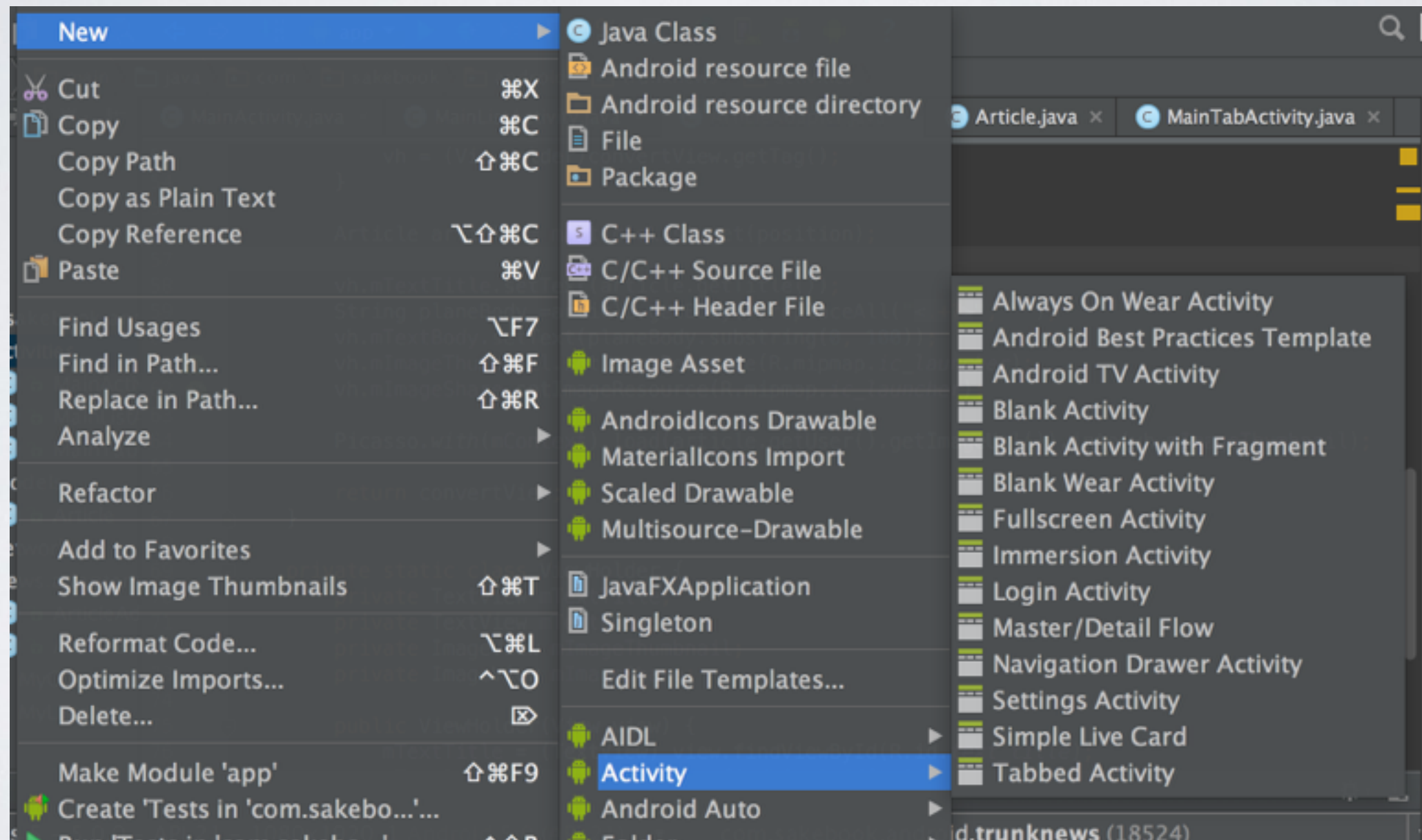


# PICASSO


- もっとも単純だと一行で済む
  - `Picasso.with(mContext).load(url).into(mImageView);`

# ACTIVITYの追加

# 機能を使うと楽



- AndroidManifest.xmlに追加しないと認識されない
  - <application>の中



```
<activity
    android:name=".activities.MainSampleActivity"
    android:label="@string/title_activity_main_sample" >
</activity>
</application>
```



# 呼び出すとき

- Intentを作成して、startActivity

```
@Override
public void onItemClick(AdapterView<?> parent,
                        View view, int position, long id) {
    Article article = (Article)parent.getItemAtPosition(position);
    Toast.makeText(MainListActivity.this, article.getTitle(),
                  Toast.LENGTH_SHORT).show();

    Intent intent = new Intent(MainListActivity.this, MainSampleActivity.class);
    startActivity(intent);
}
```



INTENT

# INTENT

- Activity(Component)同士のやりとりに用いる
- 別のアプリ間でもやりとり可能
- 様々な情報を詰められる(制限あり)

# どれも同じ

```
Intent intent = new Intent(MainListActivity.this, MainSampleActivity.class);  
startActivity(intent);
```

```
Intent intent = new Intent();  
intent.setClass(getBaseContext(), MainSampleActivity.class);  
startActivity(intent);
```

```
Intent intent = new Intent();  
intent.setClassName("com.sakebook.android.trunknews",  
    "com.sakebook.android.trunknews.activities.MainSampleActivity");  
startActivity(intent);
```

```
Intent intent = new Intent();  
ComponentName componentName = new ComponentName(getBaseContext(),  
    "com.sakebook.android.trunknews.activities.MainSampleActivity");  
intent.setComponent(componentName);  
startActivity(intent);
```

# 情報を渡す

- put
  - putExtra(key, value)

```
Intent intent = new Intent(MainListActivity.this, MainSampleActivity.class);  
intent.putExtra("title", article.getTitle());  
startActivity(intent);
```



# BUNDLE


- Bundle
  - データをまとめて保持できる
  - 中身はMap

```
Intent intent = new Intent(MainListActivity.this, MainSampleActivity.class);  
Bundle bundle = new Bundle();  
bundle.putString("title", article.getTitle());  
intent.putExtra("bundle", bundle);  
startActivity(intent);
```



# モデルを渡す

- そのままでは渡せない
  - Serializableを実装する

```
public class Article implements Serializable{  
    private String body;  
    private String title;  
     private String url;  
    private Date created_at;  
    private User user;  
}
```

# 情報の受け取り

- getIntent()
- Intentが取得できるので  
中身を展開して探す

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main_sample);

    Intent intent = getIntent();
    if (intent != null) {
        if (intent.hasExtra("article")) {
            Article article = (Article)intent.getSerializableExtra("article");
            ((TextView)findViewById(R.id.text_article_title)).setText(article.getTitle());
        }
    }
}
```

# 元の画面に戻るには

- バックボタンなどで、現在表示している Activityを終了させる
- 再度Intentを生成して呼び出したりはしない
  - LaunchMode, ActionFlagを設定し Intentを用いて戻る場合もある

WEBVIEW



# WEBVIEW

- URLを読み込んでhtmlをレンダリング
- htmlを直接読むことも可能
- 基本はメインスレッドで動く
- apkを変更せずに内容を変更できる



# URL読み込み

- loadUrl
- 不正なURLだった場合何も表示しない

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_web_view);

    mWebView = ((WebView) findViewById(R.id.web));
    mWebView.loadUrl("http://yahoo.co.jp");
}
```

# 基本設定

- 表示する縮尺をちょうどよく
- Javascriptを有効にする
  - XSSの危険性があるので注意

```
mWebView = ((WebView) findViewById(R.id.web));  
mWebView.setInitialScale(1);  
mWebView.getSettings().setUseWideViewPort(true);  
mWebView.getSettings().setLoadWithOverviewMode(true);  
mWebView.getSettings().setJavaScriptEnabled(true);
```

# カスタマイズ

- WebChromeClient
  - ブラウザのUIに影響を与えるものに関係する
- WebViewClient
  - コンテンツのレンダリングやURLの読み込みに関係する
- WebSettings
  - WebView自体の設定に関係する

# WEBCHROMECLIENT

- onJsAlert
  - WebでのAlertをキャッチして表示するかどうかを返す
  - 簡単なWeb→Androidのやりとりの方法



# WEBVIEWCLIENT

- onPageStarted / onPageFinished
  - 現在表示しようとしているURLの読み込み(開始時 / 終了時)に呼ばれる
- onLoadResource
  - 現在表示しようとしているURL内でimgやcss、 javascriptを読み込む際に都度呼ばれる



# WEBVIEWCLIENT

- shouldOverrideUrlLoading
  - 表示しようとするURLが変わる際に  
呼ばれる
  - trueでURLが変わり  
falseでURLが変わらない

# 確認しよう

```
mWebView.getSettings().setJavaScriptEnabled(true);
```

```
mWebView.setWebChromeClient(new WebChromeClient() {  
    Log.d(TAG, "onJsAlert: " + message);  
    return super.onJsAlert(view, url, message, result);  
});
```

```
mWebView.setWebViewClient(new WebViewClient() {  
    @Override  
    public boolean shouldOverrideUrlLoading(WebView view, String url) {  
        Log.d(TAG, "shouldOverrideUrlLoading: " + url);  
        return super.shouldOverrideUrlLoading(view, url);  
    }  
  
    @Override  
    public void onPageStarted(WebView view, String url, Bitmap favicon) {  
        Log.d(TAG, "onPageStarted: " + url);  
        super.onPageStarted(view, url, favicon);  
    }  
  
    @Override  
    public void onPageFinished(WebView view, String url) {  
        Log.d(TAG, "onPageFinished: " + url);  
        super.onPageFinished(view, url);  
    }  
  
    @Override  
    public void onLoadResource(WebView view, String url) {  
        Log.d(TAG, "onLoadResource: " + url);  
        super.onLoadResource(view, url);  
    }  
});
```

シェア

# アプリ内でシェアしたい

- アプリ内で記事を閲覧
- そのままシェアしたい
  - TwitterやLINEのアプリを呼び出したい
  - アプリ内で投稿は今回は扱わない(要実装)



# INTENT

- 他のActivityを呼び出すのはIntent
- 明確に相手がわかる場合
  - 明示的Intent
- 条件を満たす誰かの場合
  - 暗黙的Intent

# 暗黙的INTENT

- 条件設定
  - Action
  - Category
  - Data

# ACTION

- どのようなことをさせたいのか
  - Intent.ACTION\_VIEW
    - 見る
  - Intent.ACTION\_SEND
    - 渡す

# CATEGORY

- どんなカテゴリのコンポーネントに  
応答してもらうか
  - Intent.CATEGORY\_DEFAULT
    - 暗黙的Intentに対応している
  - Intent.CATEGORY\_LAUNCHER
    - ランチャーに表示している



# DATA

- コンポーネントに渡したいData
- `putExtra(String extra, Bundle data);`
  - `Intent.EXTRA_TEXT`
    - ACTION\_SENDで渡したいData(文字列)
  - `Intent.EXTRA_STREAM`
    - ACTION\_SENDで渡したいData(ファイルパス)

# DATA

- setType(String mimeType)
  - 実ファイルは渡せないなので教える必要がある
    - text/plain
      - テキスト
    - image/jpeg
      - 画像

# どうやって受け取ってるの

- 明示的に指定されないため  
受け取るための準備が必要

# 受け取る側

- 2箇所で記述
  - AndroidManifest.xml
  - 受け取るActivity



# ANDROIDMANIFEST.XML

- 受け取れるものを指定
  - 受け取れないものは書かない！

```
<activity
    android:name=".activities.WebIntentActivity"
    android:label="@string/title_activity_web_intent" >
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:scheme="http"/>
    </intent-filter>
</activity>
```

# 受け取るACTIVITY

- 明示的と同様にIntentを処理する
- 通常の画面遷移による呼び出しと混同しないようにする

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_web_intent);
    initWebViewSetting();

    Intent intent = getIntent();
    if (intent != null) {
        String url = intent.getDataString();
        if (!TextUtils.isEmpty(url)) {
            Log.d(TAG, "URI: " + url);
            mWebView.loadUrl(url);
        } else {
            Toast.makeText(this, "data not set", Toast.LENGTH_SHORT).show();
        }
    } else {
        Toast.makeText(this, "intent is null", Toast.LENGTH_SHORT).show();
    }
}
```

# 渡す例

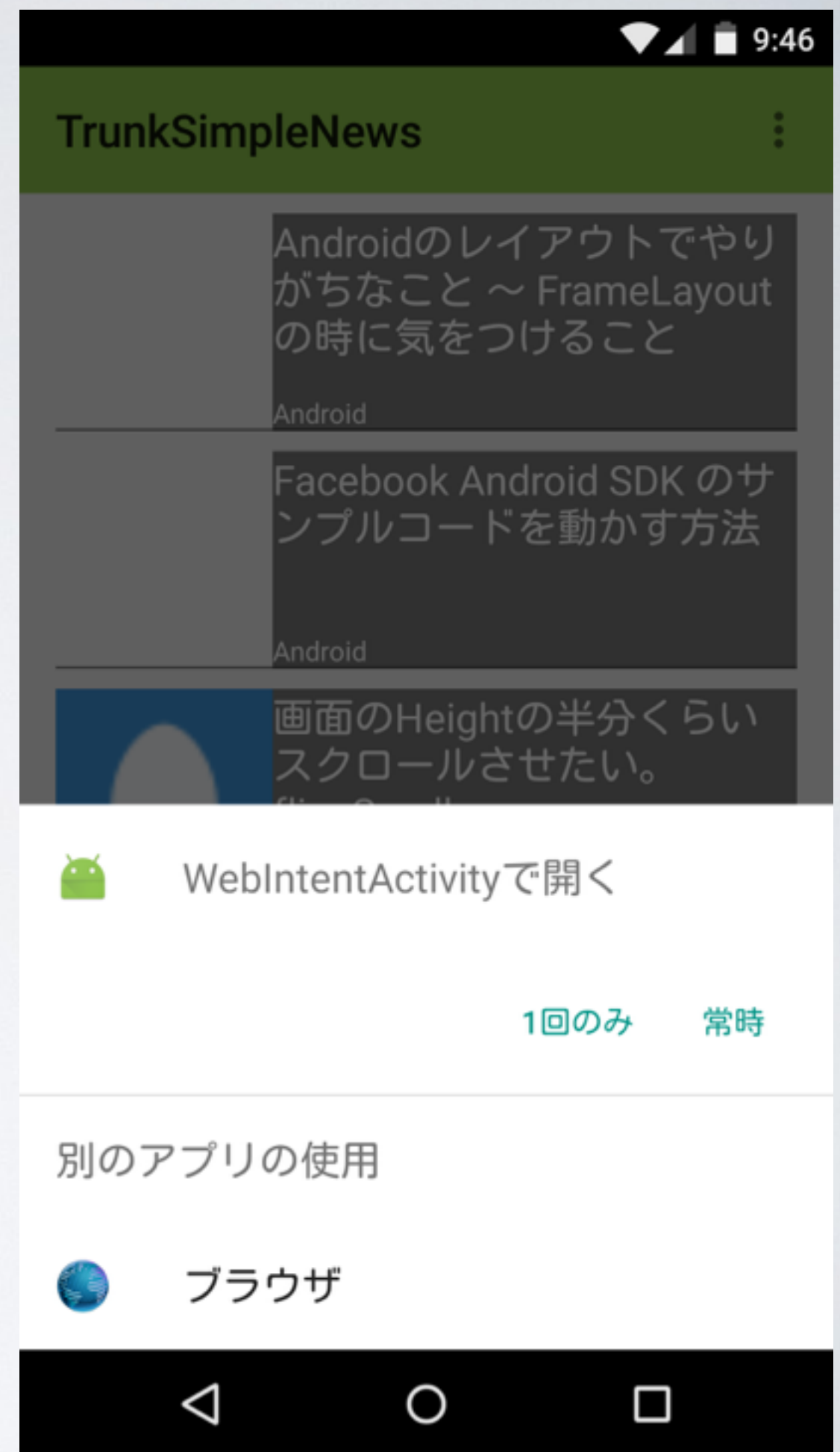
# URLを表示させたい

- setDataにURLを詰める
- 表示なのでACTION\_VIEW

```
private void shareArticle() {  
    Log.d("TAG", "shareArticle");  
    String url = "http://yahoo.co.jp";  
    Intent intent = new Intent();  
    intent.addCategory(Intent.CATEGORY_DEFAULT);  
    intent.setAction(Intent.ACTION_VIEW);  
    intent.setData(Uri.parse(url));  
    startActivity(intent);  
}
```



- ブラウザと同様にurlを受け取る



# SNSへシェアしたい

- 受け取りもわかるように定数を用いる
- putで渡すのでACTION\_SEND

```
private void shareArticle() {  
    Log.d("TAG", "shareArticle");  
    String url = "http://jxpress.net/";  
    String shareText = "ホームページイカす。 / " + url;  
  
    Intent intent = new Intent();  
    intent.addCategory(Intent.CATEGORY_DEFAULT);  
    intent.setAction(Intent.ACTION_SEND);  
    intent.setType("text/plain");  
    intent.putExtra(Intent.EXTRA_TEXT, shareText);  
    startActivity(intent);  
}
```

- Twitterが対応してくれる



# 処理の完了をハンドリング

- 暗黙的Intentで起動した  
Componentが終了すると、自分アプリに戻る



# STARTACTIVITYFORRESULT

- startActivityForResult(Intent intent, int requestCode)
  - startActivityの代わりに使う
  - 遷移したComponentから結果を送り返してもらう
  - onActivityResultがコールバックされる

- intentは先ほどもでと同じ
- REQUEST\_CODEには、自身でユニークとなる数字を入れる
- 結果受け取り時に判別するため

```
final int REQUEST_CODE = 1000;
try{
    startActivityForResult(intent, REQUEST_CODE);
} catch (ActivityNotFoundException exception) {
    Toast.makeText(this, "対応するアプリがありません",
        Toast.LENGTH_SHORT).show();
}
```

- ActivityNotFoundException
  - 暗黙的Intentで呼び出せる  
Componentがない場合に呼ばれる
  - ユーザにフィードバックを与えるために用いる

```
final int REQUEST_CODE = 1000;
try{
    startActivityForResult(intent, REQUEST_CODE);
} catch (ActivityNotFoundException exception) {
    Toast.makeText(this, "対応するアプリがありません",
        Toast.LENGTH_SHORT).show();
}
```

# ONACTIVITYRESULT

- `onActivityResult(int requestCode, int resultCode, Intent data)`
  - `startActivityForResult`で  
他のComponentへ遷移し  
処理が完了した際に呼ばれる
  - `requestCode`: 呼び出し元で定義したcode
  - `resultCode`: 処理の結果
  - `data`: 遷移先から送り返されるdata



- requestCodeで自アプリが対応すべきものの判定
- resultCodeの比較には定数を用いる

```
@Override
protected void onActivityResult(int requestCode, int resultCode,
                                Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    final int REQUEST_CODE = 1000;
    if (requestCode == REQUEST_CODE) {
        if (resultCode == Activity.RESULT_OK) {
            Toast.makeText(this, "成功",
                           Toast.LENGTH_SHORT).show();
        } else if (resultCode == Activity.RESULT_CANCELED) {
            Toast.makeText(this, "キャンセル",
                           Toast.LENGTH_SHORT).show();
        }
    } else {
        Log.d("TAG", "知らない結果");
    }
}
```

URLスキーム

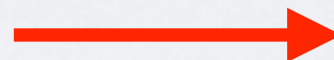
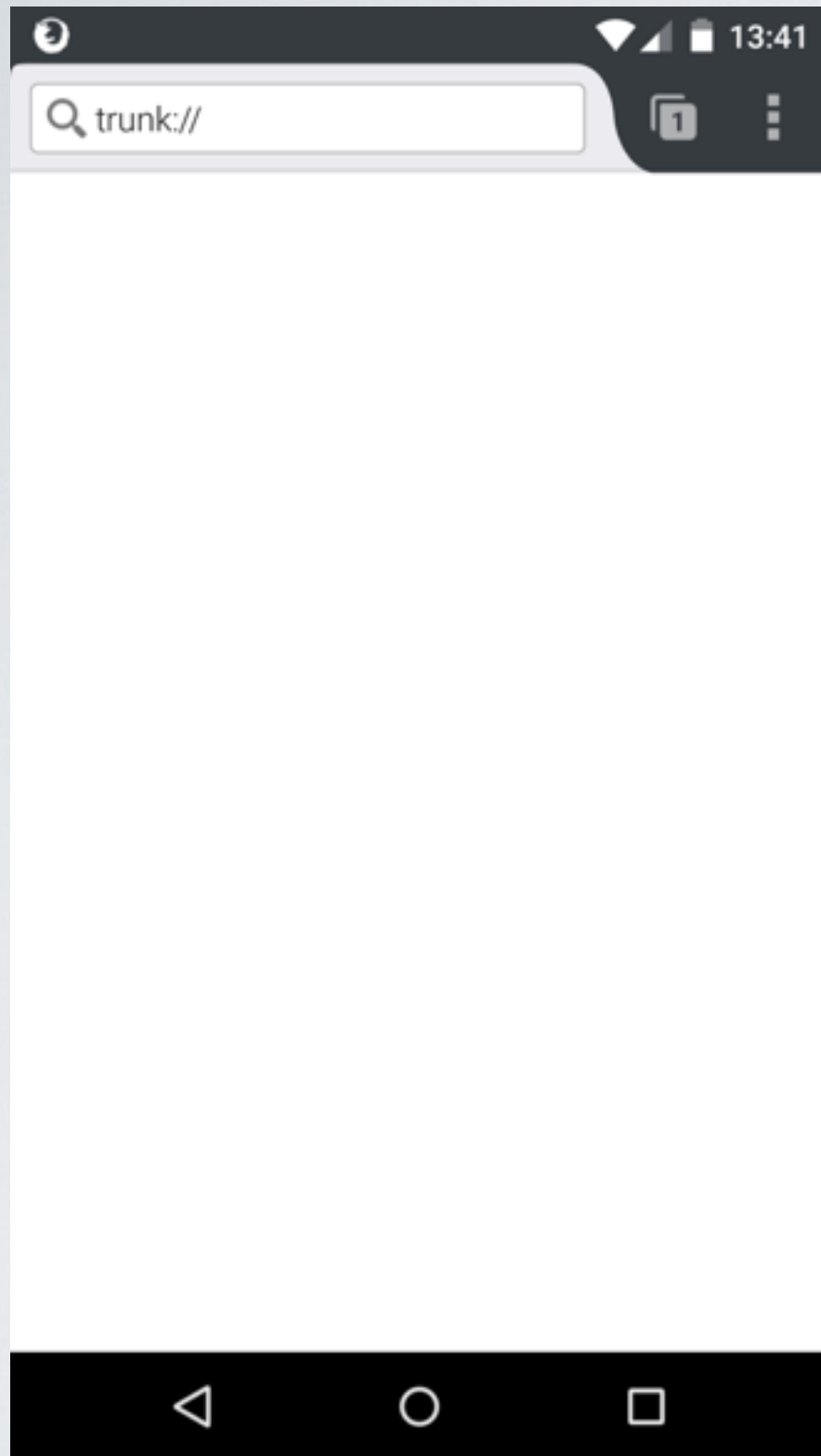
# URLスキーム

- 特定のURLからアプリを起動させる
- 暗黙的Intent
- 対応するComponentをユニークにすることで  
シームレスに見せる

- trunk://~
- これに反応するようにする
- BROWSABLEでWebブラウザからも遷移

```
<activity
    android:name=".activities.WebIntentActivity"
    android:label="WebIntentActivity" >
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:scheme="http"/>
    </intent-filter>
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="android.intent.category.BROWSABLE"/>
        <data android:scheme="trunk"/>
    </intent-filter>
</activity>
```





# 受け取った後

- 従来と同じく Intent の処理

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_web_intent);
    initWebViewSetting();

    Intent intent = getIntent();
    if (intent != null) {
        String url = intent.getDataString();
        if (!TextUtils.isEmpty(url)) {
            Log.d(TAG, "URI: " + url);
            mWebView.loadUrl(url);
        } else {
            Toast.makeText(this, "data not set", Toast.LENGTH_SHORT).show();
        }
    } else {
        Toast.makeText(this, "intent is null", Toast.LENGTH_SHORT).show();
    }
}
```

# 限定させたい

- trunk://
- trunk://hoge.com
- trunk://hoge.com/huga

- trunk://hoge.com

```
<data  
    android:scheme="trunk"  
    android:host="hoge.com"/>
```

- 「trunk://」 だけでは反応しない

- trunk://hoge.com/huga

```
<data  
    android:scheme="trunk"  
    android:host="hoge.com"  
    android:path="/huga"/>
```

- 「trunk://hoge.com/」 だけでは反応しない

- trunk://○○/huga

```
<data  
    android:scheme="trunk"  
    android:path="/huga"/>
```

- 「trunk://」 からはじまり

「/huga」 があるだけで反応



# 反応範囲は限定させる

- 想定外の挙動を引き起こす
- いろんな暗黙的Intentに対応してきて  
鬱陶しく感じさせる(ユーザ)