

第 10 章

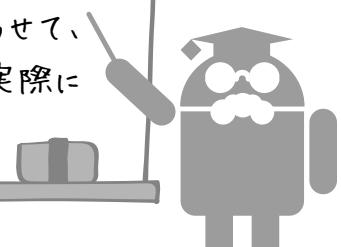
ユーティリティによる 実践

著：高橋憲一

10-1 ユーティリティによる実践(1)

著：高橋憲一

前章では、これまで「パート単位」で学んできた、Androidアプリを構成する様々な要素を組み合わせて、ゲームアプリを作つてみました。この章でも、同様にそれらの要素を組み合わせて、1つのアプリの形に仕上げることを実践します。講義で実際に役立つユーティリティアプリを作つてみましょう。



この節で学ぶこと

- ・これまで学んで来たことを組み合わせて1つのアプリを作成する
- ・複数のActivityを作ることによる画面遷移

この節で出てくるキーワード一覧

ユーティリティ
Activityの追加
明示的インテント
Activityにデータを渡す
Android SDKのクラス
ListView
AdapterView
TextView
View
ArrayAdapter
Intent
Activity
LayoutInflater



10-1-1 そもそもユーティリティって何だろう？

スマートフォンのアプリには、さまざまなジャンルのものがあります。今回のテーマとなっている「ユーティリティ」は、勉強でも仕事でも何かしら日常のことを使つてくれるようなもの、煩わしいことを快適にしてくれるものなどを指します。

今回は、この講座で学んでいる皆さんに便利に使えるものを考えてみました。普段疑問に思う「今日は何の講義だっけ?」「先生は誰だっけ?」ということをサッとアプリで見ることができる「シラバスアプリ」を作ることにしましょう。

シラバス (Syllabus) とは、日本では講義・授業の大まかな学習計画のこと。米国では、各回講義内容から教員連絡方法まで、個別講義の受講に関して必要な情報をすべて盛り込んだメモのこと。(出典:Wikipedia)



10-1-2 最小限のアプリを構成してみよう

図1のように日付と講義のタイトルを表示する「ListView」と、その要素をタップすると講義の詳細と講師の名前を表示する詳細画面が表示されます。この章の3回で全員ここまで完成を目指しましょう。



図1:最小限の構成

そして後半ではAction Barの活用など、より便利にするための実装例も解説します。早めに進んだ場合はチャレンジしてみてください。



10-1-3 新規プロジェクトの作成を開始！

Eclipseのメニューバーから「File」→「New」→「Android Application」とたどります。次のようなダイアログが表示されますので、「Application Name」欄に「Syllabus」と入力します。

「Project Name」欄には「Syllabus」が、「Package Name」欄には「com.example.syllabus」が自動で入力されていることを確認してください。ここでPackage Nameは「jp.techinstitute.syllabus」と変更します。

「com.example」の部分だけを範囲選択して「jp.techinstitute」と入力すると必要な部分のみの書き換えで済みます。

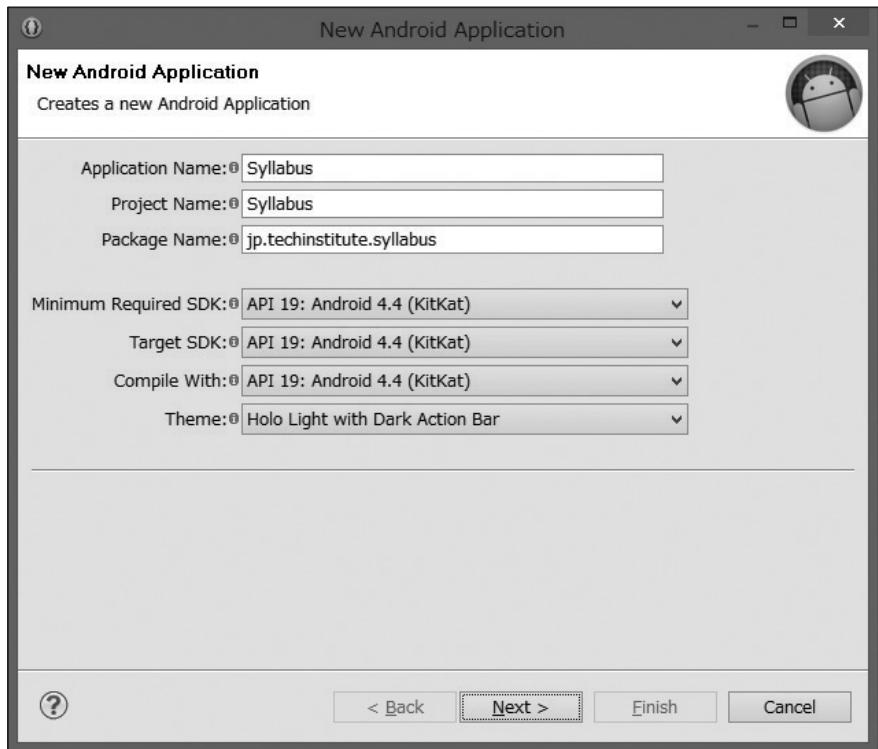


図2:New Android Applicationダイアログ

図2のように「Minimum Required SDK」「Target SDK」「Compile with」の3項目のすべてを「API 19: Android 4.4 (KitKat)」に設定します。「Theme」は「Holo Light with Dark Action Bar」を選択して「Next」ボタンを押します。

続いて「New Android Application」の「Configure Project」というダイアログに進んでいきます。

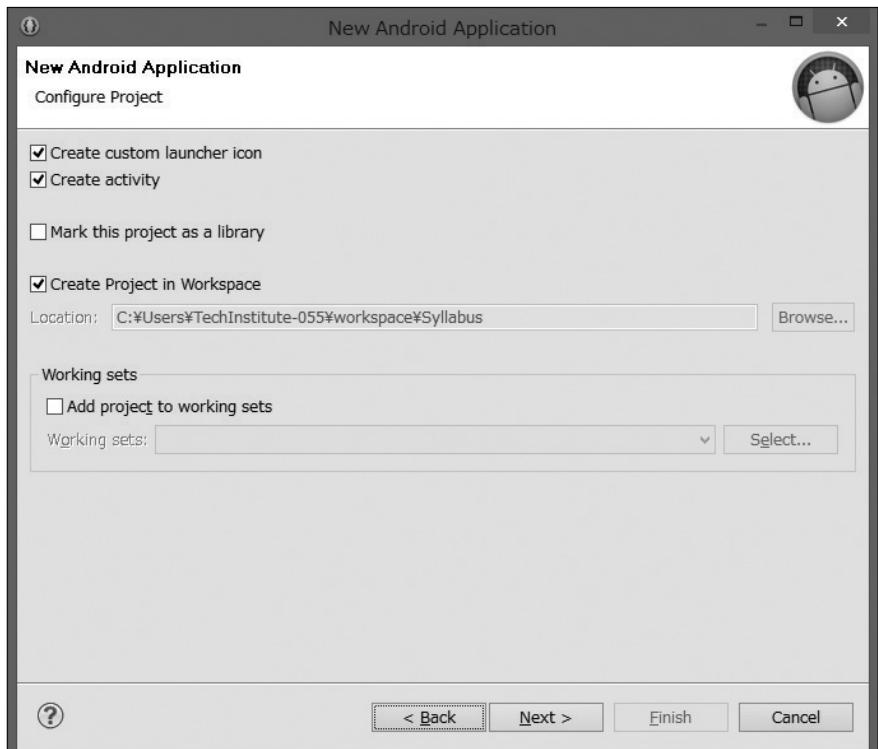


図3:「New Android Application」の「Configure Project」

「Create custom launcher icon」「Create activity」のチェックはオンに、

「Mark this project as a library」のチェックはオフになっていることを確認して「Next」ボタンを押します(図3)。

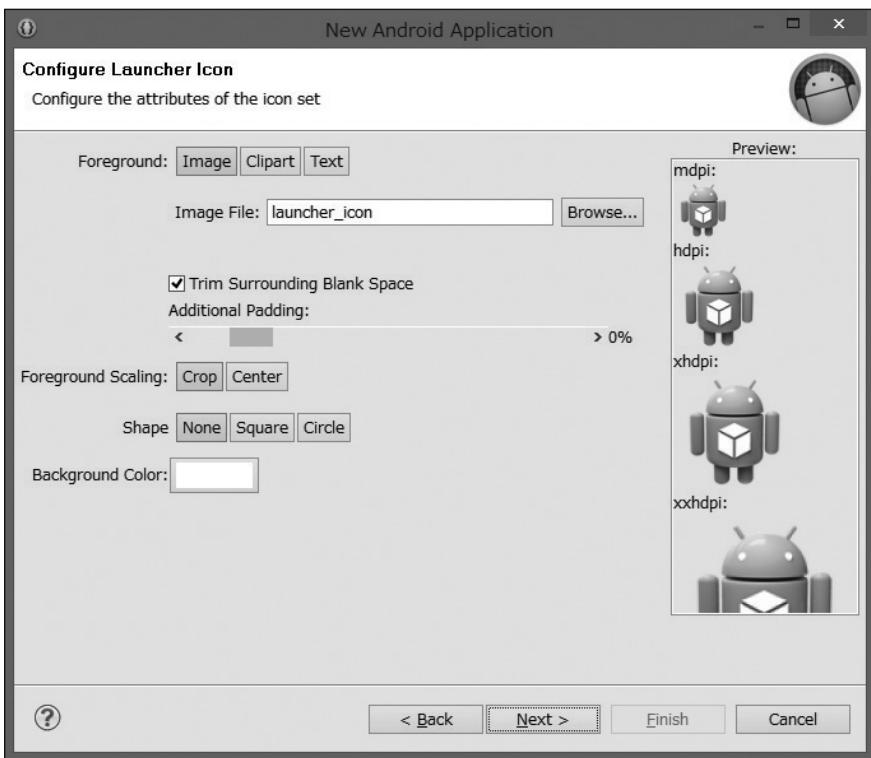


図4:「New Android Application」の「Configure Launcher Icon」

「Configure Launcher Icon」画面が開きます。特にアイコンを変更する必要が無ければそのまま「Next >」ボタンを押します。(図4)。

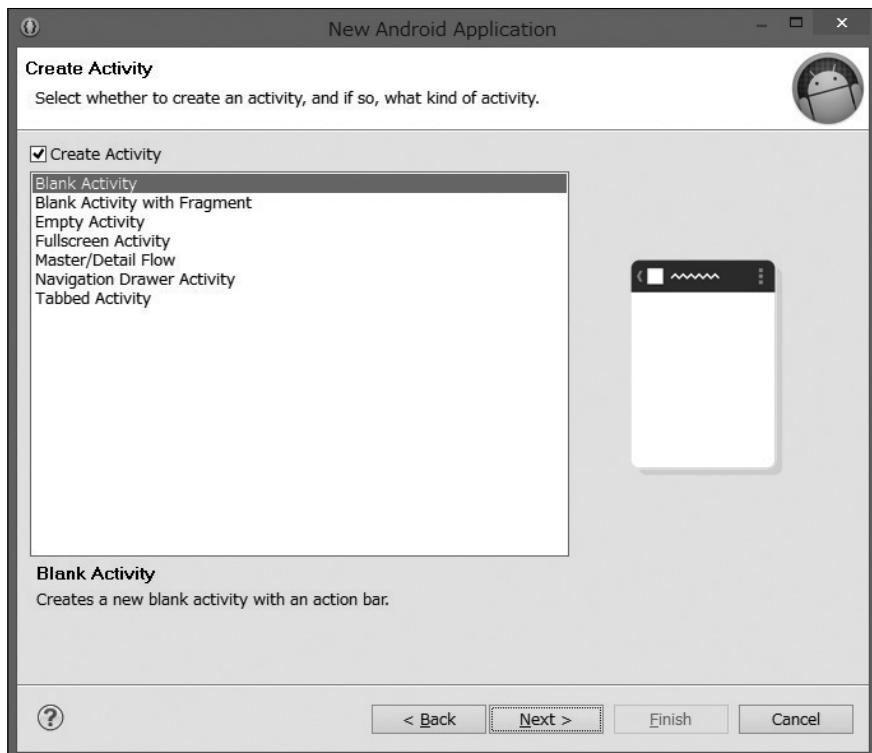


図5:「New Android Application」の「Create Activity」

Blank Activityには“Creates a new blank Activity with an action bar.”と説明が書いてあります。新しい空白のActivity (Fragmentは使わず) をaction bar付きで作成するテンプレートです。action barはあとで触れます。

「Create Activity」画面が開いたら、「Create Activity」にチェックが入っていることを確認し、「Blank Activity」を選択して「Next」ボタンを押します(図5)。

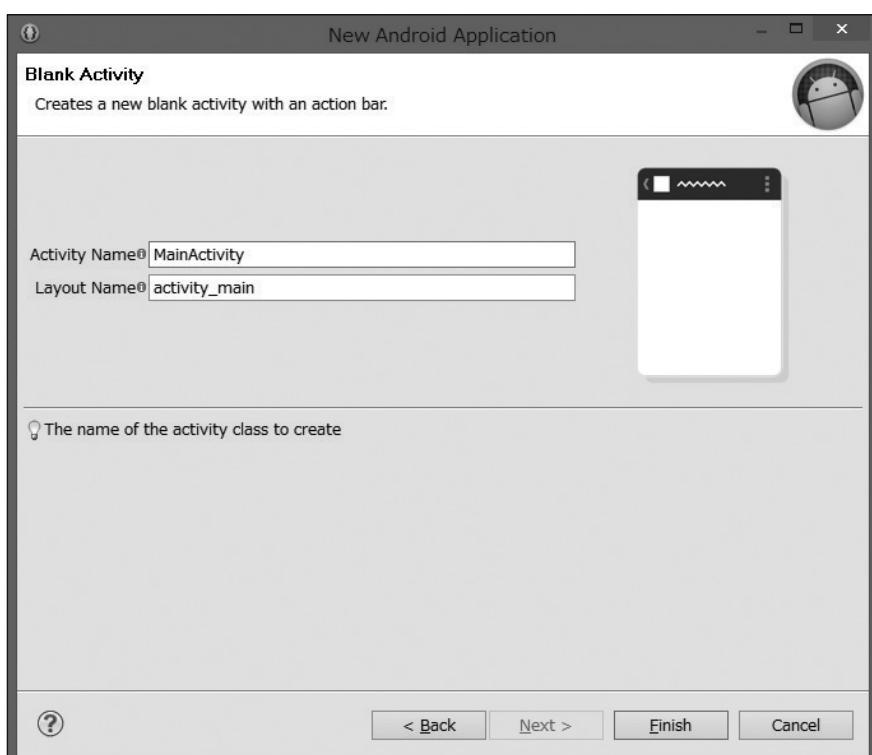


図6:「New Android Application」の「Blank Activity」

図6のように「Activity Name」には「MainActivity」が、「Layout Name」には「activity_main」が自動で入力されているはずです。この状態を確認して「Finish」ボタンを押すと完了します。

作成されたプロジェクトの確認

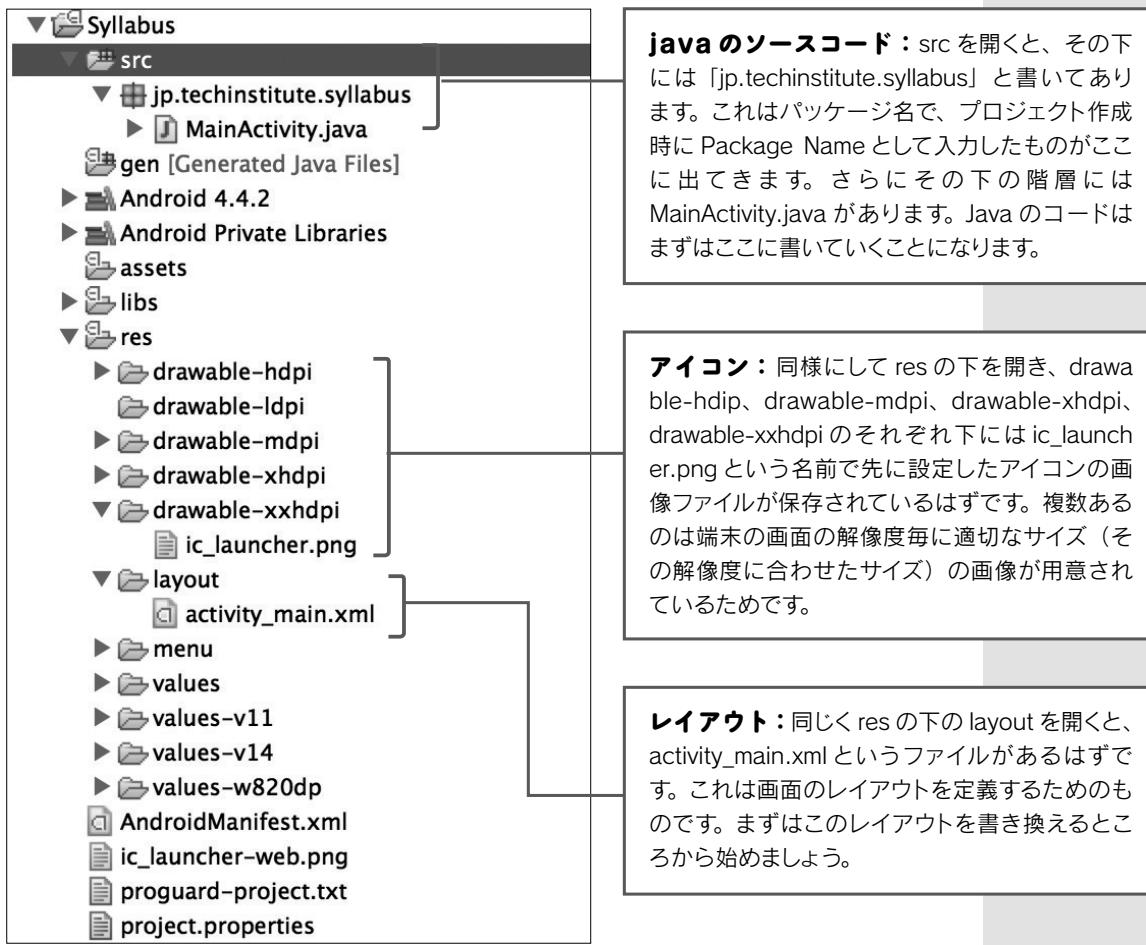


図7:EclipseのPackage Explorer

java のソースコード : src を開くと、その下には「jp.techinstitute.syllabus」と書いてあります。これはパッケージ名で、プロジェクト作成時に Package Name として入力したものがここに出てきます。さらにその下の階層には MainActivity.java があります。Java のコードはまずはここに書いていくことになります。

アイコン : 同様にして res の下を開き、drawable-hdpi, drawable-mdpi, drawable-xhdpi, drawable-xxhdpi のそれぞれ下には ic_launcher.png という名前で先に設定したアイコンの画像ファイルが保存されているはずです。複数あるのは端末の画面の解像度毎に適切なサイズ（その解像度に合わせたサイズ）の画像が用意されているためです。

レイアウト : 同じく res の下の layout を開くと、activity_main.xml というファイルがあるはずです。これは画面のレイアウトを定義するためのものです。まずはこのレイアウトを書き換えるところから始めましょう。

ここまででSyllabusというプロジェクトができているはずです。復習を兼ねて、作成したプロジェクトの中を確認してみましょう。「Syllabus」という名前のプロジェクトの下の階層が開かれていない場合は、フォルダーアイコン左の三角マークをクリックします。さらに「res」の中にある「layout」をクリックして、「activity_main.xml」を開きます。



10-1-4 リスト表示の作成

ListViewをレイアウトに追加する

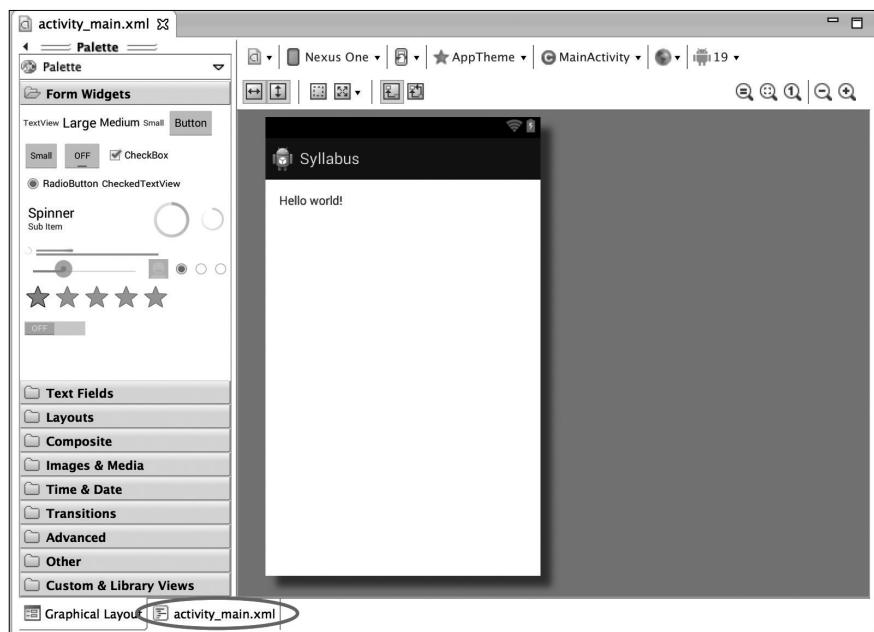


図8:Graphical Layout

「activity_main.xml」を開いたあと、図8のような状態になっている場合は「activity_main.xml」と書かれたタブをクリックします。するとXMLを直接編集する画面に切り替わります。

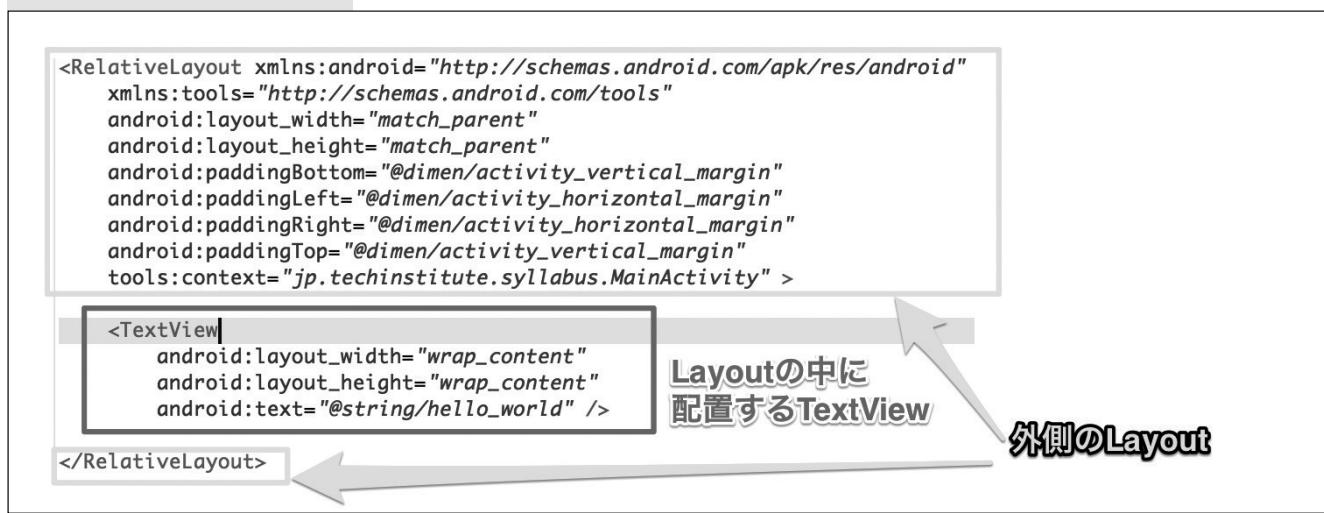


図9:activity_main.xml

「activity_main.xml」のファイルに記述してあるXMLの構造を見てみましょう。図9の「外側のLayout」という説明が付いている部分が、中の部品をどのように並べるかを示すレイアウト定義の部分です。ここではデフォルトで「RelativeLayout」が設定されています。次に「Layoutの中に配置するTextView」と説明を付けたコードが、実際に画面に表示する部品を定義している部分です。ここでは「TextView」と設定されています。

まずはこの「TextView」を次のように「ListView」へと変更します。

activity_main.xml

```
<ListView
    android:id="@+id/listview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    />
```

すべてを入力するのではなく「TextView」の「Text」の部分を「List」に書き換えます。「android:layout_width」と「android:layout_height」の行は「”(ダブルクオーテーション)」の中だけを「wrap_content」から「match_parent」に書き換えるとすればやく入力できます。

それぞれの意味について解説します。「ListView」は名前のとおり「ListView」を表示するためのもの。「android:layout_width」と「android:layout_height」に「match_parent」と指定しているのは、画面の上下左右方向全体にListViewを広げて表示するための設定となっています。入力し終えたら「Ctrl」+「S」キーを押して保存しておきましょう。

ListViewの表示要素の定義

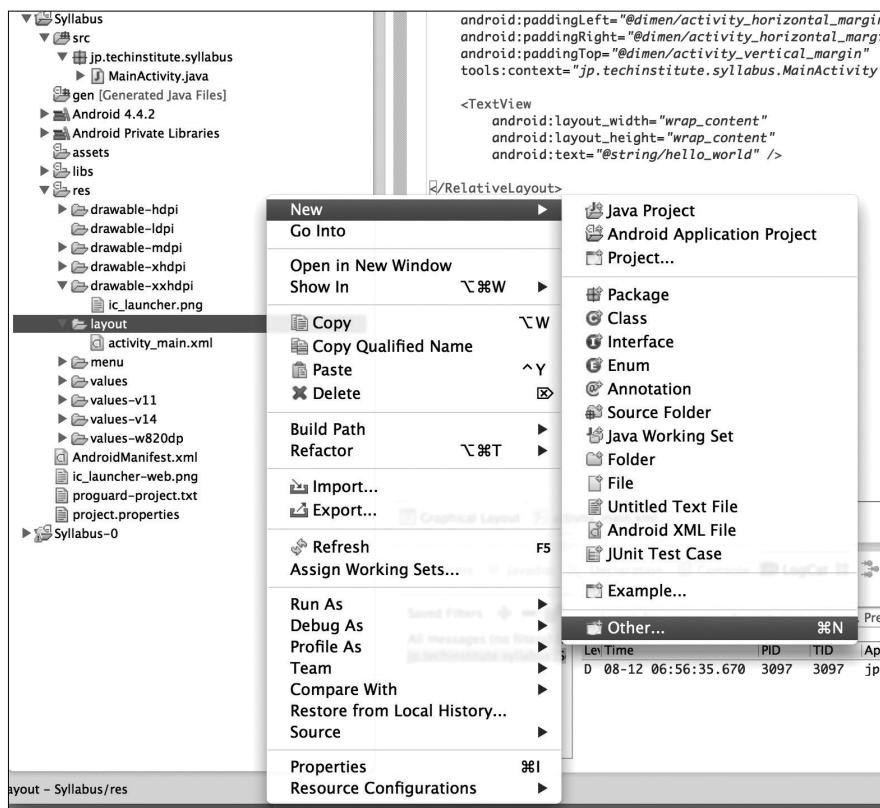


図10:項目を追加する

図10のように「layout」を選択後に右クリックして「New」→「Other」→「Android」→「Android XML Layout File」と選び「Next」ボタンを押します。そして次の画面では、図11のように「Android XML Layout File」を選択して「Next」ボタンを押します。

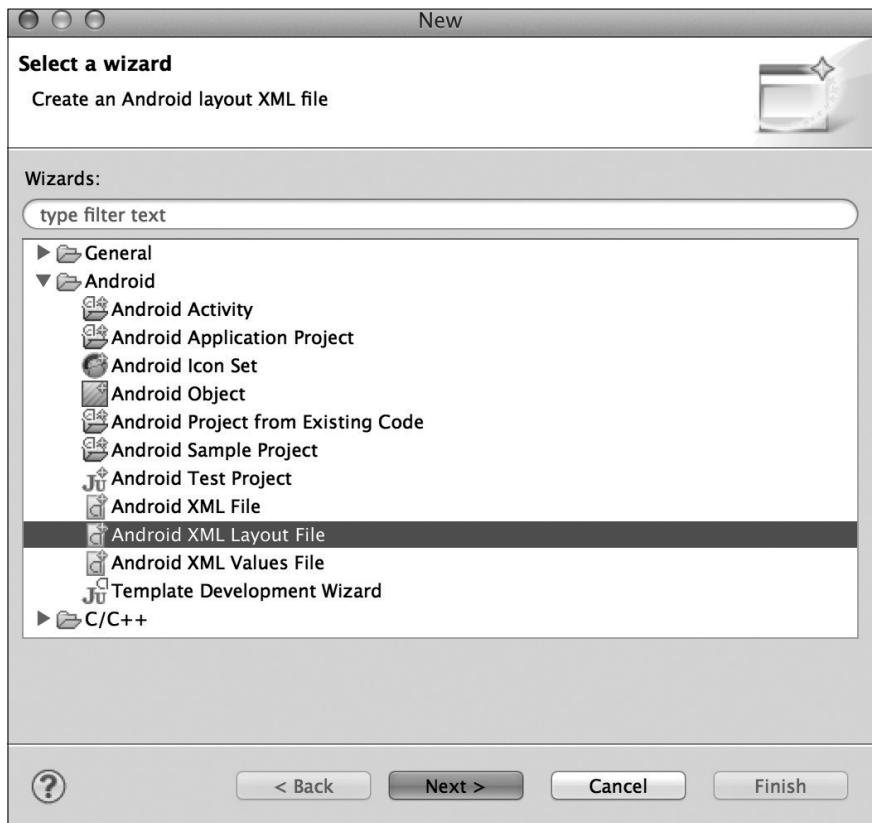


図11:追加する項目の選択

表示されるダイアログの「File」欄に「lecture_row.xml」と入力して、「RelativeLayout」を選択。「Next」ボタンを押します(図12)。

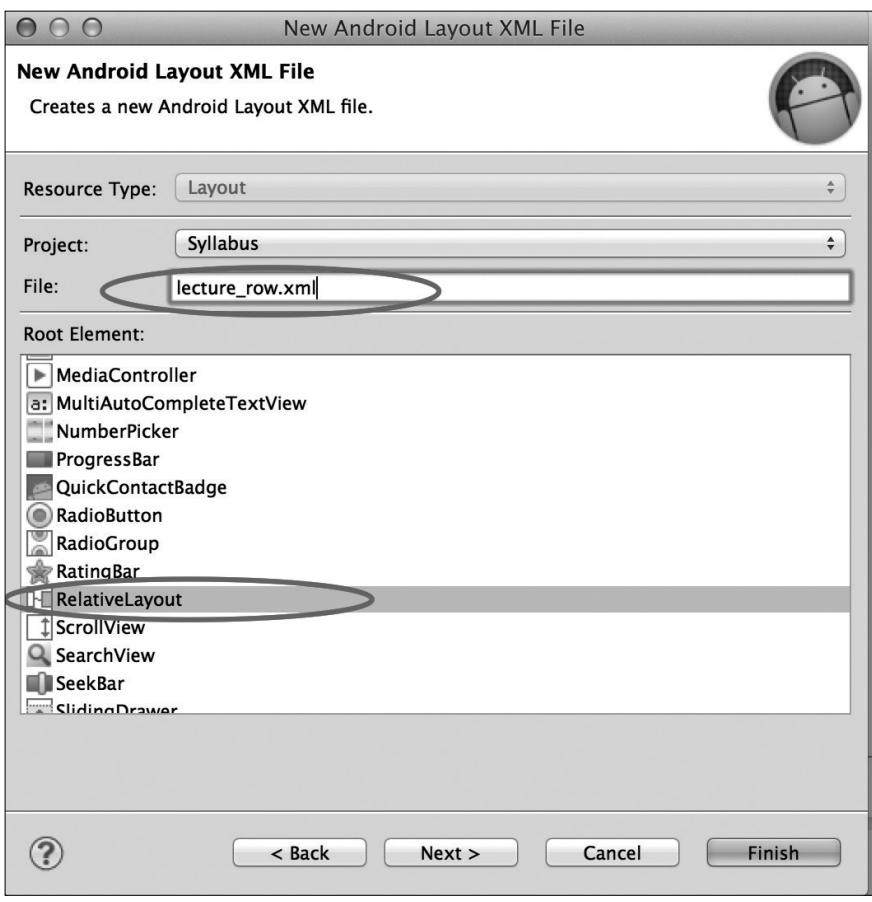


図12:New Android Layout XML File

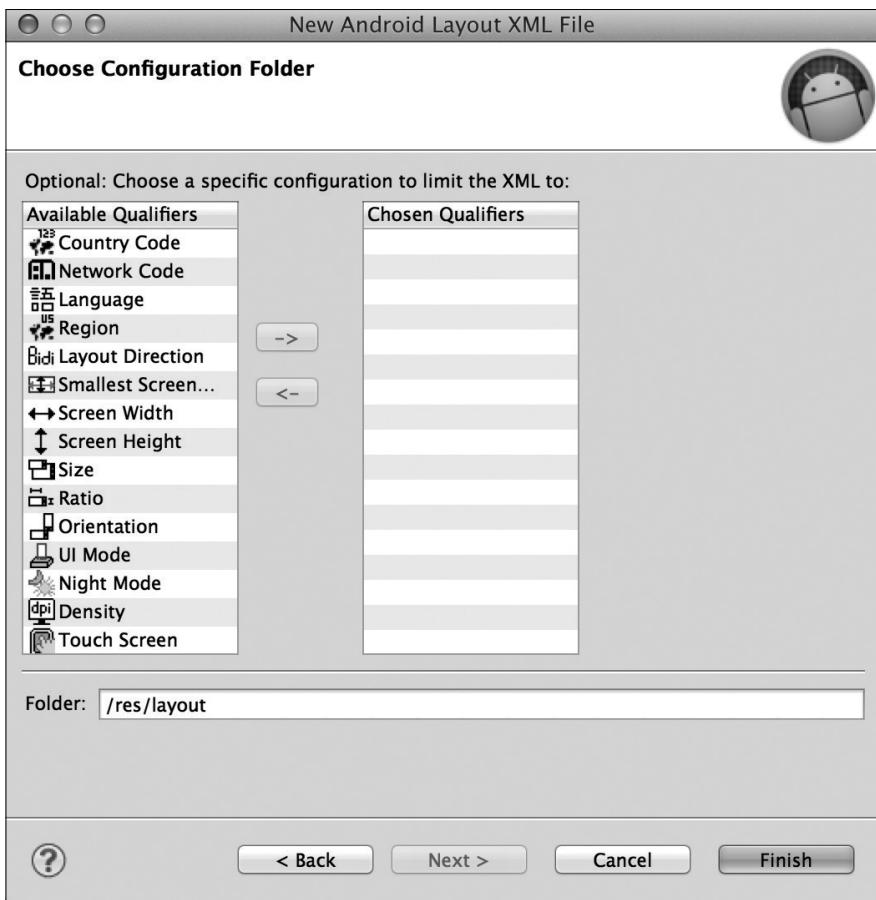


図13:Choose Configuration Folder

図13では何も選択せずに「Finish」ボタンを押します。ダイアログが閉じられて、グラフィカルレイアウト(画面イメージが表示されている)の状態になっている場合は「lecture_row.xml」と書かれたタブをクリックしてXMLを直接編集する画面に切り替えます。

TextViewの追加

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    | 
</RelativeLayout>
```

図14:lecture_row.xml

図14の四角で囲んだ部分に、表示要素である日付と講義タイトルを表示するためのTextViewを追加していきます。

```
lecture_row.xml
<TextView
    android:id="@+id/date"
    android:layout_width="56dp"
    android:layout_height="48dp"
    android:layout_alignParentLeft="true"
    android:textColor="@android:color/black" />
<TextView
    android:id="@+id/title"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:textColor="@android:color/black"
    android:layout_toRightOf="@+id/date" />
```

このレイアウト定義は図15のような並びにするためのものです。

8/28 ユーティリティによる実践(1)

日付

講義タイトル

図15:要素ごとのレイアウトイメージ

「Relative Layout」の中にあるので、相対関係として配置されます。1つ目の「TextView」は日付を表示するためものです。「android:layout_alignParentLeft」に「true」を指定しているので、左寄せの配置となります。

2つ目の「TextView」は講義タイトルを表示するためのもので、幅は「layout_width」で「wrap_content」を指定しているためにテキストの長さに応じたサイズとなります。「layout_toRightOf」で日付の「TextView」を示す「"@+id/date"」を指定しています。日付の右側に抗議タイトルが配置されます。高さはどちらも「android:layout_height」で「match_parent」を指定しているので、「ListView」の要素の上下全体の高さになります(図15)。



10-1-5 ListViewに表示するための実装

MainActivity.javaの中の構成

手を加える前に中がどのようにになっているか見てみましょう。

```

package jp.techinstitute.syllabus; パッケージ名

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem; 使用するクラスのインポート定義

public class MainActivity extends Activity { クラス定義

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
メソッド定義

```

図16:MainActivity.javaの中の構成

これから先、コードを追記していくにあたって、四角で囲んだそれぞれのブロックの階層、構造を常に意識してください。たとえば「メソッドを追加する」と指示がある場合にメソッド定義の中に追加するのではなく、メソッド定義の外でかつクラス定義の中、つまりメソッド定義とメソッド定義の間に追記することになります。

メソッド定義の「onCreate」はこのActivityが実行された時に最初に通過する部分です。

ほかの「onCreateOptionsMenu」と「onOptionsItemSelected」はあとで使うのでこのまま残しておいてください(図16)。

ListViewを使う実装

まずはローカルのデータで試してみましょう。

```
package jp.techinstitute.syllabus;
```

```
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
```

```
public class MainActivity extends Activity {
```

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

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
```

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

内部クラス定義、メンバ変数定義



図17:追加する位置

変数名、メソッド名、クラス名にはその意味、機能が分かれるような名前をつけることが望ましいです。

MainActivity.javaの図17の矢印で示されている「内部クラス定義、メンバ変数定義」の部分に次のコードを追加します。

追加する内容

```
private class CourseItem {
    String date;
    String title;
    String teacher;
    String detail;
}

private List<CourseItem> itemList;
private ItemAdapter adapter;
```

ここで追加したもののうち、「CourseItem」クラスは「ListView」に表示する各要素毎の日付、講義タイトル、講師の名前を保持するためのもの。「itemList」は保持したCourseItemクラスを複数用意して一つのまとまり(「List」)として保持するためのもの、「adapter」はその「itemList」と「ListView」を関連づけるためのクラス型の変数です。

この場所に定義を置くことによって「CourseItem」クラス、メンバ変数「itemList」および「adapter」ともに「MainActivity」の中全体からアクセス可能です。

「List」に赤い下線がついてビルドエラーとなる場合は、その部分にマウスポインターを置き(この時クリックはしない)暫く待つと図18のように黄色い小さなウィンドウが開くので、その中から「Import 'List'(Java.util)」を選択するとエラーが解決します。すると「MainActivity.java」のソースコード上の方に「import java.util.List;」と追加され、Listを使うための準備ができたのでエラーが解決されたということになります。

「ArrayList」「ArrayAdapter」などでエラーが出た際にも同様にして解決できます(図18)。

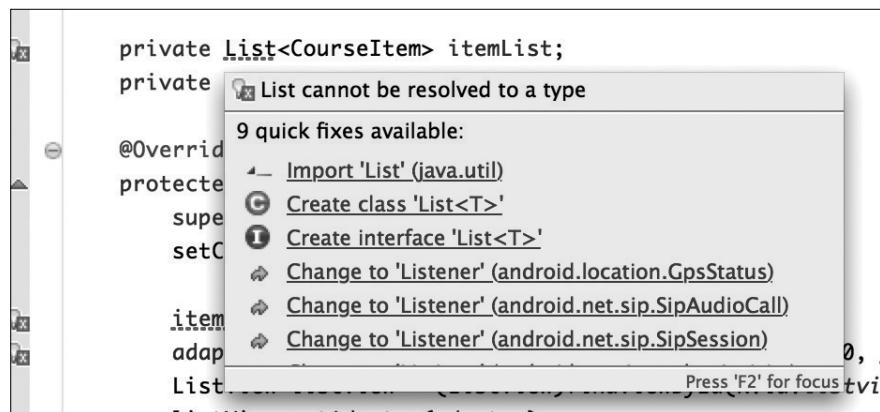


図18:ビルドエラー時の対処

データをセットするメソッドの追加

図19の「メソッド、内部クラス定義」の部分に、以下のコードを追加します。

ここではまず、ListView関連のコードが正しく動作するか確認するために、2件のデータを用意します。

メソッド、内部クラス定義に追加する内容

```
private void setCourseData() {  
    CourseItem item = new CourseItem();  
    item.date = "8/28";  
    item.title = "ユーティリティによる実践 (1)";  
    item.teacher = "高橋憲一";  
    item.detail = "この講義では一つのアプリとして仕上げることを目指します。";  
    itemList.add(item);  
  
    item = new CourseItem();  
    item.date = "9/2";  
    item.title = "ユーティリティによる実践 (2)";  
    item.teacher = "高橋憲一";  
    item.detail = "一つのアプリを仕上げることを目指す2回目。";  
    itemList.add(item);  
}
```

```
package jp.techinstitute.syllabus;
```

```
import android.app.Activity;  
import android.os.Bundle;  
import android.view.Menu;  
import android.view.MenuItem;
```

```
public class MainActivity extends Activity {
```

内部クラス定義、メンバ変数定義

メソッド、 内部クラス定義

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

```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    // Inflate the menu; this adds items to the action bar if it is present.  
    getMenuInflater().inflate(R.menu.main, menu);  
    return true;  
}
```

```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    // Handle action bar item clicks here. The action bar will  
    // automatically handle clicks on the Home/Up button, so long  
    // as you specify a parent activity in AndroidManifest.xml.  
    int id = item.getItemId();  
    if (id == R.id.action_settings) {  
        return true;  
    }  
    return super.onOptionsItemSelected(item);  
}
```

図19:定義を追加する場所

データとListViewの間を取り持つ内部クラスを追加

図19の「メソッド、内部クラス定義」の部分に、以下を追加します。

メソッド、内部クラス定義に追加する内容

```
private class ItemAdapter extends ArrayAdapter<CourseItem> {
    private LayoutInflater inflater;

    public ItemAdapter(Context context, int resource,
                       List<CourseItem> objects) {
        super(context, resource, objects);
        inflater = (LayoutInflater)context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View view = inflater.inflate(R.layout.lecture_row, null, false);
        TextView dateView = (TextView) view.findViewById(R.id.date);
        TextView titleView = (TextView) view.findViewById(R.id.title);
        CourseItem item = getItem(position);
        dateView.setText(item.date);
        titleView.setText(item.title);
        return view;
    }
}
```

ここまでできたら「ListView」に要素が2つ表示されるかどうか確認してみましょう。

正しく実行されると次のような画面になります。

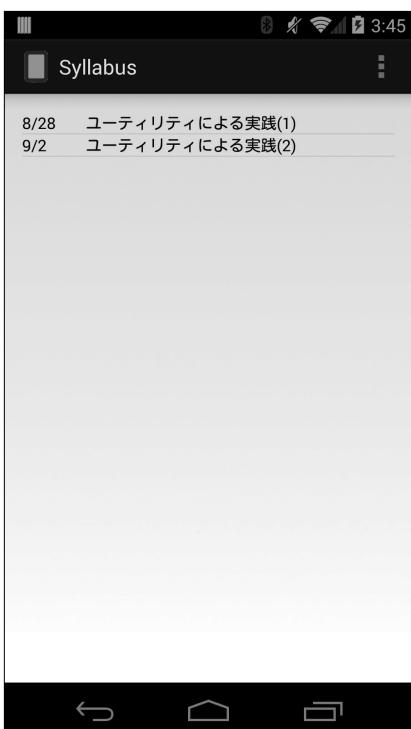
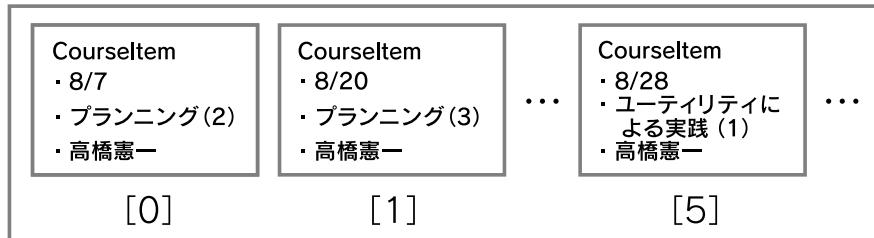


図20: 実行結果

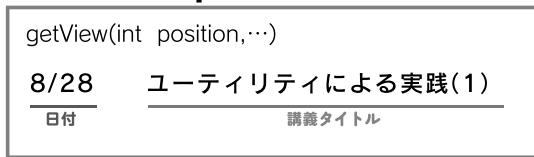
ListViewとデータの関係

ListViewとそこに表示するデータの関係は一見複雑なコードに見えますが、図示すると次のようにになります。

itemList



itemAdapter



ListView



図21:ListViewとデータの関係

「ListView」の各要素が画面に表示されるタイミングで、アダプターとして設定されたクラス(ここでは「ItemAdapter」)の「getView()」が呼び出されます。その際、引数の「position」にListの何番目かという情報が渡されます。

「getView」の中では渡された「position」をインデックスとして、アダプターに関連づけられている「List(データの集合、ここではCourseItem型の集合であるitemList)」を「getItem」で取得します。

「ListView」の要素用「Layout(ここではlecture_row.xml)」を展開して、2つの「TextView」を取り出します。

「getItem」で取得した「CourseItem」の日付と講演タイトルを取り出した「TextView」にそれぞれセットします。

「getView」はここまで構築された1要素分のView(二つの「TextView」が含まれている)を返します。

「ListView」は「getView」が返したViewをその要素として表示します。

JavaのArrayList

日付、講義タイトル、講師名をデータとして含むクラスである「CourseItem」の集合を定義するのに次のような書き方をしています。

```
itemList = new ArrayList<CourseItem>();
```

これはJavaのコレクションと呼ばれる機能のひとつで、<>で囲まれた中に任意のクラスを書くことでそのクラスの集合を表すことができます。



10-1-6 詳細情報を表示するための実装

新しいActivityの追加

詳細情報を表示するための新しいActivityを追加します。

図のようにsrcの階層を開いて「jp.techinstitute.syllabus」のパッケージの上にマウスポインターを合わせて右クリックして、「New」→「Other」の順に選択します(図22)。

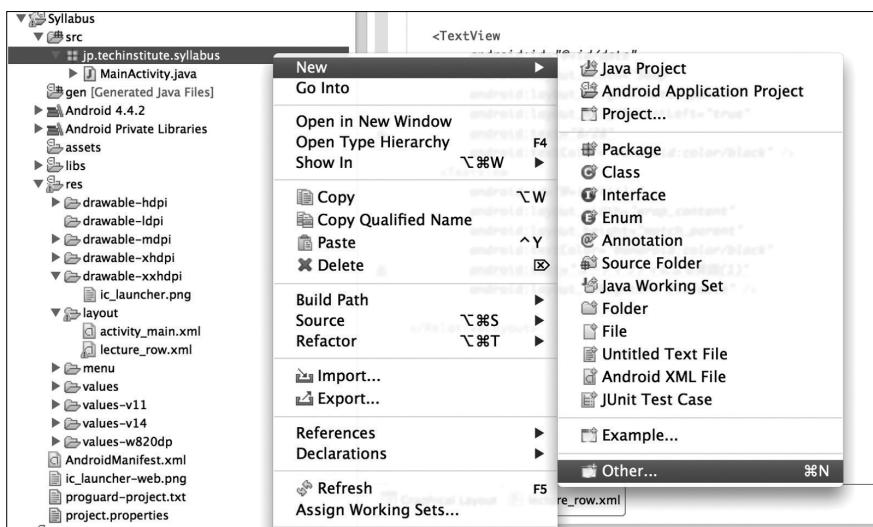


図22:Activityの追加

すると「New」ダイアログが開くので、Androidの階層を三角のボタンを押して開き、「Android Activity」を選択して「Next」ボタンを押します(図23)。

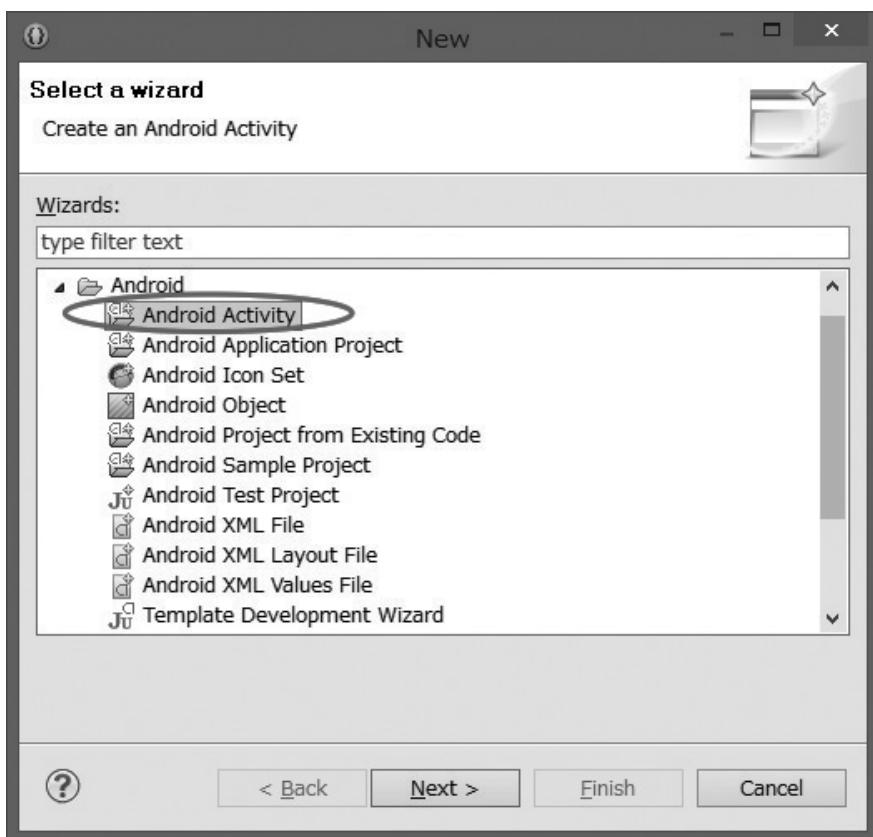


図23:追加する項目の選択

「New Activity」ダイアログが開くので「Blank Activity」を選択して「Next」ボタンを押します(図24)。



図24:Activityのテンプレート選択

「Activity」に名前を入力するためのダイアログが開きます。開いた直後は図25のような文字で埋められているはずです。

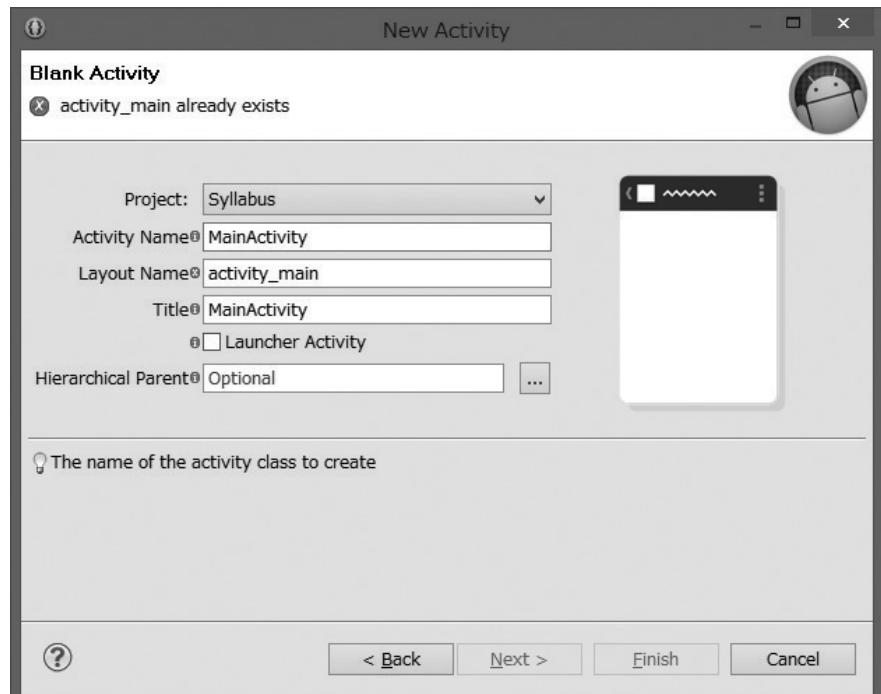


図25:Activityの設定

ここでActivity Nameを書き換えます。図26の「Activity Name」に「CourseDetail」と入力してください。それ以降の部分はそれに準じて自動的に変わりま

す。このとき、「Launcher Activity」はチェックされていないことを確認してください。

図26と同じようになっていることを確認できたら「Finish」ボタンで完了します。

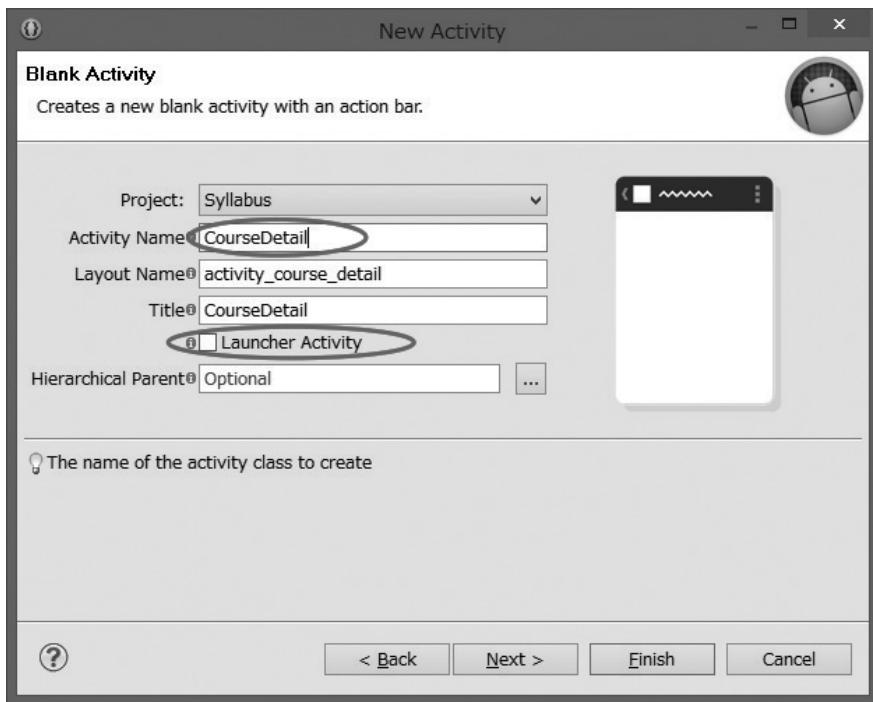


図26:Activityの名前入力

ここでもしLauncher Activityにチェックを入れると、ホーム画面のランチャーにこのアプリのアイコンが2つ(List表示画面と詳細画面)出てくることになります。List表示を経由せずに直接詳細画面を立ち上げられることになり、それは設計意図からは外れるのでそうならないようにします。

追加したActivityを呼び出すためのコードを追加

srcの下のjp.techinstitute.syllabusの下にあるMainActivity.javaをダブルクリックして開き、図27の四角で囲んだ部分、onCreate(Bundle savedInstanceState) の定義の下に次のコードを追加します。

```
public class MainActivity extends Activity {

    private class CourseItem {
        String date;
        String title;
        String teacher;
    }

    private List<CourseItem> itemList;
    private ItemAdapter adapter;

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

        itemList = new ArrayList<CourseItem>();
        adapter = new ItemAdapter(getApplicationContext(), 0, itemList);
        ListView listView = (ListView)findViewById(R.id.listView);
        listView.setAdapter(adapter);
        setCourseData();
    }
}
```

ここに追加

図27:追加する場所

図27で示した位置に次の1行を書き加えます。

```
listView.setOnItemClickListener(this);
```

この1行を追加すると、「setOnItemClickListener」の部分にビルドエラーを示す赤線がついた状態になります。赤線部分にマウスポインターを置くと(この時クリックはせずに上に置くだけ)、図28のようにエラーを解決するための手段が黄色いウインドウの中に表示されますので、その中から「Let MainActivity implement 'OnItemClickListener'」を選択します。見あたらないときは、黄色いウインドウを下にスクロールしてみてください。

The screenshot shows an Android Studio code editor with Java code. A warning message is displayed above the code:

```
itemList = new ArrayList<CourseItem>();
adapter = new ItemAdapter(getApplicationContext(), 0, itemList);
ListView listView = (ListView)findViewById(R.id.listview);
listView.setAdapter(adapter);
setCourseData();

listView.setOnItemClickListener(this);
```

The warning message is: "The method setOnItemClickListener(AdapterView.OnItemClickListener) in the type AdapterView<ListAdapter> is not applicable for the arguments (MainActivity)". Below this, a "11 quick fixes available" dropdown is open, with the option "Let 'MainActivity' implement 'OnItemClickListener'" highlighted with a yellow oval.

```
private void
CourseIt
item.date
item.titl
item.teac
itemList.
item = ne
item.date = "9/2";
item.title = "ユーティリティによる実践(2)";
item.teacher = "高橋憲一";
itemList.add(item);
```

図28:エラー解決の選択肢

すると「MainActivity」の宣言部分が図29のように自動で修正され、こんどは「MainActivity」の部分に赤線が付いている状態になります。同様にマウスポインターを赤線の上に置くと表示される黄色いウインドウの選択肢から「Add unimplemented methods」を選択します。

The screenshot shows an Android Studio code editor with Java code. A warning message is displayed above the code:

```
import android.widget.ListView;
import android.widget.TextView;

public class MainActivity extends Activity implements OnItemClickListener {
```

The warning message is: "The type MainActivity must implement the inherited abstract method AdapterView.OnItemClickListener.onItemClick(AdapterView<?>, View, int, long)". Below this, a "2 quick fixes available" dropdown is open, with the option "Add unimplemented methods" highlighted with a yellow oval.

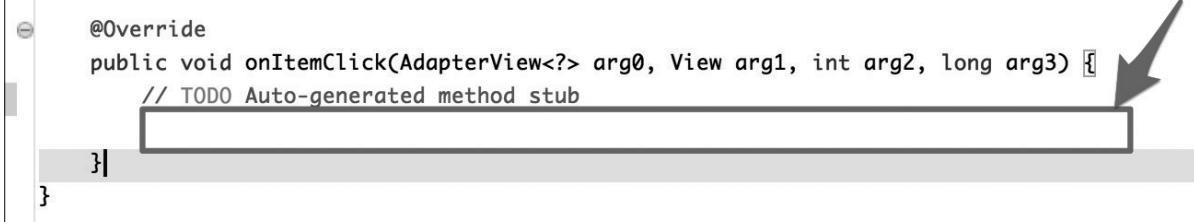
```
    private c
        String 2 quick fixes available:
        String
        String
    }

    private List<CourseItem> itemList;
    private ItemAdapter adapter;
```

At the bottom right of the code editor, there is a note: "Press 'F2' for focus".

図29:エラー解決の選択肢

それにより、MainActivity.javaの最後の方に自動的に図30のようなコードが追加されます。



```

@Override
public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
    // TODO Auto-generated method stub
}

```

The code editor shows an error under the line 'Intent intent = new Intent(this, CourseDetail.class);' with a red underline and a yellow warning icon. A large grey arrow points from the text '矢印で示した部分に次のコードを追加します。' to this error line.

図30:自動で追加されるコードと追記場所

これがListViewの要素をタップした際に呼び出される部分ですので、図30の矢印で示した部分に次のコードを追加します。

図30に追加する内容

```

Intent intent = new Intent(this, CourseDetail.class);
startActivity(intent);

```

Intentの部分についたエラーは、赤線の上にカーソルを置く→黄色いウインドウからImport 'Intent' (android.content) を選択して解決します。

ここで一度アプリを実行して、リストの要素をタップしたら別の画面(Activity)が表示されることを確認しましょう。この段階では「Hello World」と表示されるだけです。この後に詳細画面としての機能を実装していきます。

Intentで別のActivityを起動するしくみ

このプロジェクトにはMainActivityとCourseDetailという2つのActivityがあります。いま動かした「リストの要素をタップすると別の画面が表示される」機能は「明示的Intent(インテント)」というしくみが使われています。

MainActivity CourseDetail



図31:リスト画面から詳細画面を起動する

先ほど追加した2行のコードの中に「Intent intent = new Intent(this, CourseDetail.class);」とあります。「new Intent」をする際の1つめの引数に「this」を指定しているのは「MainActivity」自身です。2つめの引数に指定している「CourseDetail.class」は新たに起動したい画面(Activity)を意味しています。

この生成したintentを「`startActivity(intent);`」のようにして「`startActivity`」の引数にして呼び出すことで「CourseDetail.java」で実装している画面を起動することができます。

詳細画面のレイアウト作成

グラフィカルレイアウトの画面になっている場合は、`activity_course_detail.xml`と書いてあるタブをクリックしてXMLを直接編集する画面に切り替えます。

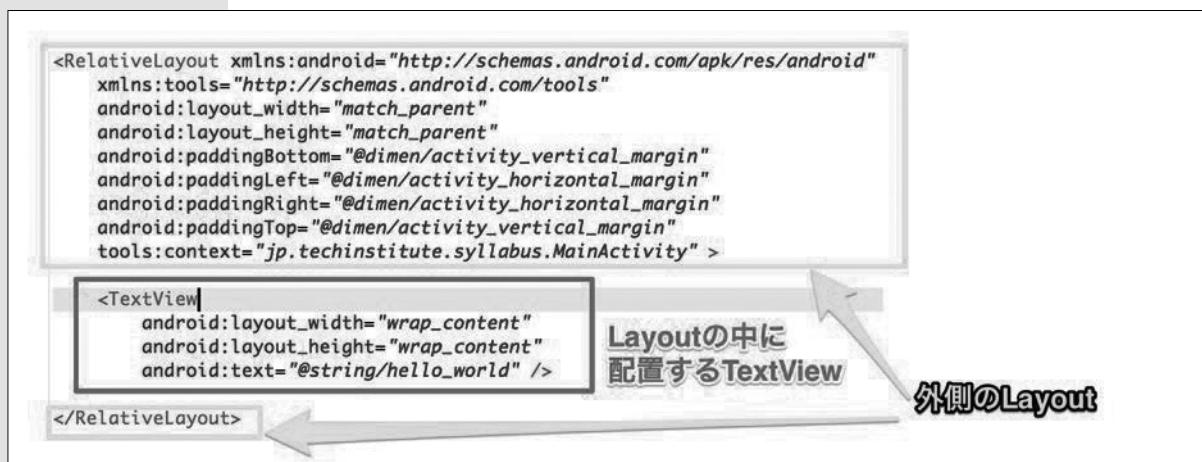


図32:activity_course_detail.xml

「ListView」を「activity_main.xml」に追加した時と同様ですが、このファイルに記述してあるXMLの構造を見てみましょう。図32の「外側のLayout」枠は中の部品をどのように並べるかを示すレイアウト定義の部分で、ここではデフォルトで「RelativeLayout」が設定されています。詳細画面でもレイアウトはこのまま「RelativeLayout」を使って部品(View)を配置していきます。次に「Layoutの中に配置するTextView」と説明が付いているコードが実際に画面に表示する部品を定義しているコードで、ここでは「TextView」が設定されています。まずはこの「TextView」に設定を追加して、日付を表示するためのものにしましょう。

idを付ける

「Layoutの中に配置するTextView」の1行目に、以下のように「`android:id="@+id/dateText"`」という行を追加します。Javaのプログラム中でこの「TextView」に対して文字をセットできるようにするために必要です。



文字を大きくする

続いて、同じTextViewに「`android:textAppearance="?android:attr/textAppearanceLarge"`」という行を追加します。詳細画面の一番上の部分で日付として見やすくするために文字を大きく表示する設定です。

文字サイズを変える

```
<TextView
    android:id="@+id/dateText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="@string/hello_world" />
```

そしてこの後は、講義タイトル、講師名、説明文の3つの表示要素を追加していきます。ここでひとつずつ「`<TextView～`」と入力していくには時間がかかるてしまいますね。今作成した日付表示用のTextViewの定義をコピー&ペーストして設定を書き換えていくといいでしょう。

「`<TextView`」で始まる行から「`/>`」で終わる行までの6行をコピーして、4回ペーストして合計5つのTextViewがある状態にします。

まずは図33の矢印で示した枠の中のTextViewを講義タイトル用に修正します。

```
<TextView
    android:id="@+id/dateText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="@string/hello_world" />
```



図33:5つのTextViewのうち、2つ目の書き換え

idを修正し、表示位置の設定を追加

図33で示した2つ目のTextViewにおいて、以下のように「`android:id="@+id/dateText"`」の「`dateText`」の部分を「`titleText`」に変更します。

続いて、日付表示と左端を揃えるために「`android:layout_alignLeft="@+id/dateText"`」を追加します。また、日付表示の下に位置するようにするために「`android:layout_below="@+id/dateText"`」と書き加えます。



以下、同様にして

3つ目のTextView

- ・「講師:」という固定文字を表示するためにidを「teacherItemName」に変更
- ・講義タイトルと左端を揃えて下に表示するように設定
- ・講義タイトルと少し感覚を空けるために、上方向に11dpのマージンを指定
- ・講義タイトルより小さく表示するために「textAppearanceMedium」を指定
- ・表示するテキストとして、「講師:」を設定

4つめのTextView

- ・講師名を表示するためにidを「teacherText」に変更
- ・「講師:」という表示の右側に表示されるように設定
- ・文字サイズは「「textAppearanceMedium」を指定

5つめのTextView

- ・詳細説明を表示するためにidを「detailText」
- ・講義タイトルと左端を揃えて、「講師:」という文字の下に表示するように設定
- ・少し感覚を空けるために上方向に11dpのマージンを指定
- ・講義タイトルより小さく表示するために「textAppearanceMedium」を指定

とそれぞれ修正すると、5つのTextViewはそれぞれ次のようにになります。

code activity_course_detail.xml

```
<TextView  
    android:id="@+id/dateText"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:textAppearance="?android:attr/textAppearanceLarge"  
    android:text="@string/hello_world" />  
  
<TextView  
    android:id="@+id/titleText"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignLeft="@+id/dateText"  
    android:layout_below="@+id/dateText"  
    android:textAppearance="?android:attr/textAppearanceLarge"  
    android:text="@string/hello_world" />  
  
<TextView  
    android:id="@+id/teacherItemName"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignLeft="@+id/titleText"  
    android:layout_below="@+id/titleText"  
    android:layout_marginTop="11dp"  
    android:textAppearance="?android:attr/textAppearanceMedium"  
    android:text="講師：" />  
  
<TextView  
    android:id="@+id/teacherText"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_toRightOf="@+id/teacherItemName"  
    android:layout_below="@+id/titleText"  
    android:layout_alignBaseline="@+id/teacherItemName"  
    android:textAppearance="?android:attr/textAppearanceMedium"  
    android:text="@string/hello_world" />  
  
<TextView  
    android:id="@+id/detailText"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignLeft="@+id/titleText"  
    android:layout_below="@+id/teacherText"  
    android:layout_marginTop="11dp"  
    android:textAppearance="?android:attr/textAppearanceMedium"  
    android:text="@string/hello_world" />
```

最終的な表示結果は図34のようになります。

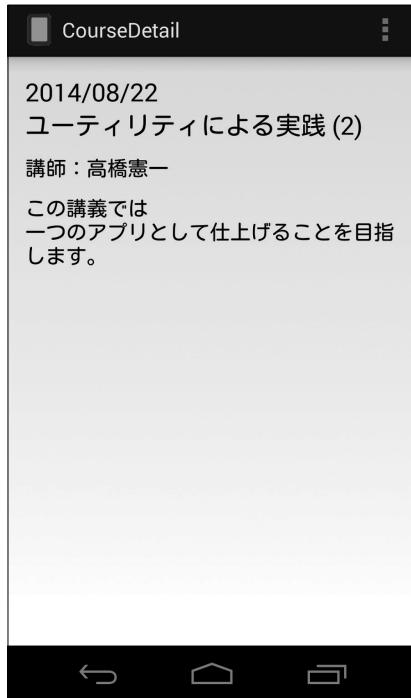


図34: 詳細画面のレイアウト

詳細画面にデータを渡す

実際に詳細画面に情報を表示するには、リストを表示している「MainActivity」から詳細画面を表示する「CourseDetail」のActivityにデータを渡す必要があります。ここで詳細画面を表示するには 前に解説したように「Intent」を設定して「startActivity」で起動するということを思い出してください。その際にデータを渡すように設定する方法について、以下で解説します。

まずは流れを見るために、講義タイトルをどのようにして渡すか見ていきましょう。

呼び出し側での設定

「MainActivity.java」で詳細画面を起動している部分を見てください。「MainActivity.java」の最後の方は、次のようにになっていると思います。

MainActivity.java

```
public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {  
    Intent intent = new Intent(this, CourseDetail.class);  
    startActivity(intent);
```

ここで出て来るintentに、次のようにして渡す情報を設定します。

MainActivity.java に追加する内容

```
public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {  
    Intent intent = new Intent(this, CourseDetail.class);  
    intent.putExtra("title", "ゲームによる実践");  
    startActivity(intent);
```

「`intent.putExtra(String name, String value)`」というメソッドを使って、1つ目の引数の「name」にはこのデータを読み出すためのキーとなる名前を設定し、2つ目の引数の「value」に渡したいデータ(ここでは文字列型)を設定します。

呼び出された側での設定

呼び出された側では「`intent`」を通じて渡されたデータを受け取ることができます。

「`CourseDetail.java`」を開いて「`onCreate`」の部分を見てみましょう。

```
package jp.techinstitute.syllabus;
import android.app.Activity;

public class CourseDetail extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_course_detail);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.course_detail, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

図35:CourseDetail.javaのonCreate

図35で示した枠の部分に次のような処理を追加します。

追加する内容

```
Intent intent = getIntent();
String titleStr = intent.getStringExtra("title");
```

まず1行目の「`getIntent()`」でこのActivityに関連付けられているIntentのオブジェクトを取得して、`intent`という名前の変数に保持します。その`intent`に対して「`getStringExtra(String name)`」というメソッドを、引数の「name」にデータを呼び出すためのキーを入れて呼び出します。そうすることによって、渡されたデータ(ここでは文字列型)を取得できます。この2行の処理により「`titleStr`」というString(文字列)型の変数に、呼び出し側から渡された "ゲームによる実践" という文字列が入ることになります。

ここで重要なのは、次の図36のように呼び出し元で設定した側のキーとなる文字列と、呼び出された側で受け取るキーの文字列が一致している必要があるということです。

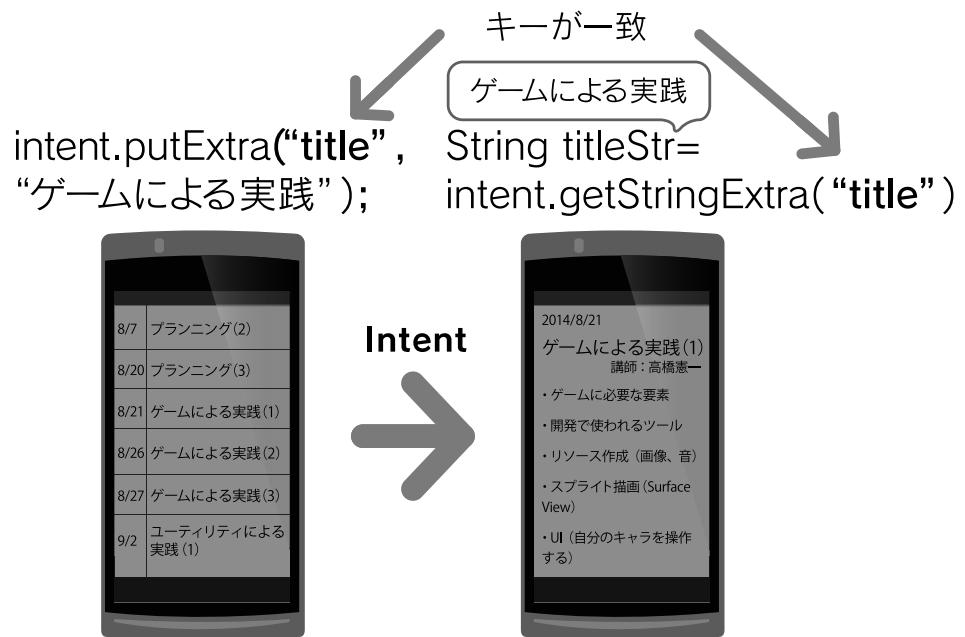


図36:intentによる値の受け渡し

渡されたデータの表示

渡された文字列を画面にレイアウトした「TextView」に表示するには次のようにします。この2行を先ほど書き足したコードの下に追加します。

追加する内容

```
TextView titleText = (TextView)findViewById(R.id.titleText);
titleText.setText(titleStr);
```

IntentとTextViewに赤線がついてエラーとなる場合は、マウスポインターを赤線部分に置いて出て来る黄色いウィンドウでImportで始まる選択肢をクリックして解決します。

「findViewById」に講義タイトルを表示する「TextView」のIDである「R.id.titleText」を指定して「TextView」のオブジェクトを取得します。そのオブジェクトに引数として先ほど「intent」から取得した文字列が入っている「titleStr」を指定し、「setText」というメソッドを呼び出します。

こうした処理はActivityが起動された時に最初に通る部分であるonCreateの中でやる必要があります。

ここでアプリをビルド、起動してみてデータが正しく渡されて表示されることを確認してみましょう。正しく渡されていれば講義タイトルの部分に表示されているはずです。

タップした要素のデータ

ここでもうひとつ、呼び出す側の「MainActivity.java」でやっておくことがあります。先ほどは流れを見るために直接「ゲームによる実践」という文字列を設定していましたが、リストビューの要素をタップして詳細画面を呼び出すということは、そのタップされた要素に応じたデータを渡す必要があるということです。

もう一度「MainActivity.java」の一番下の方にある「onItemClick」の部分を見てみましょう。

```
@Override
public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
    Intent intent = new Intent(this, CourseDetail.class);
    intent.putExtra("title", "ゲームによる実践");
    startActivityForResult(intent);
}
```

図37:onItemClick

ここで、タップされた要素のデータを取得する以下のコードを①の部分に追加して、その結果を②の部分にセットします。

```
CourseItem item = (CourseItem)arg0.getItemAtPosition(arg2);
```

このコードの「arg0」は「onItemClick」の1つ目の引数で、ListView自身のオブジェクトが入っています。これに対し「getItemAtPosition(int position)」というメソッドを呼ぶと、タップされた要素に対応するデータを取得できます。ここで引数の「position」には「onItemClick」の3つ目の引数である「arg2」を渡すことで、タップされた要素の何番目なのかを指定できます。

②の部分には「item.title」をセットして、最終的に「onItemClick」は次のようにになります。

前に開設したListViewとデータの関係の図を思い出してみてください。

最終的な内容

```
public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
    CourseItem item = (CourseItem)arg0.getItemAtPosition(arg2);
    Intent intent = new Intent(this, CourseDetail.class);
    intent.putExtra("title", item.title);
    startActivityForResult(intent);
}
```

残りのデータの追加

同様にして残りのデータ(日付、講師名、詳細説明)についてもデータの受け渡しと表示するためのコードを追加します。「MainActivity.java」の「onItemClick」でデータを渡す部分は合計4つの「putExtra」を行うことになります。

MainActivity.java

```
public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {  
    CourseItem item = (CourseItem)arg0.getItemAtPosition(arg2);  
    Intent intent = new Intent(this, CourseDetail.class);  
    intent.putExtra("date", item.date);  
    intent.putExtra("title", item.title);  
    intent.putExtra("teacher", item.teacher);  
    intent.putExtra("detail", item.detail);  
    startActivity(intent);  
}
```

「CourseDetail.java」の「onCreate」の方は次のようにになります。同じく合計4つの「getStringExtra」を行う部分と、「TextView」を取得して「setText」する部分があります。

CourseDetail.java

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_course_detail);  
  
    Intent intent = getIntent();  
    String dateStr = intent.getStringExtra("date");  
    String titleStr = intent.getStringExtra("title");  
    String teacherStr = intent.getStringExtra("teacher");  
    String detailStr = intent.getStringExtra("detail");  
  
    TextView dateText = (TextView)findViewById(R.id.dateText);  
    dateText.setText(dateStr);  
    TextView titleText = (TextView)findViewById(R.id.titleText);  
    titleText.setText(titleStr);  
    TextView teacherText = (TextView)findViewById(R.id.teacherText);  
    teacherText.setText(teacherStr);  
    TextView detailText = (TextView)findViewById(R.id.detailText);  
    detailText.setText(detailStr);  
}
```

ここまでで、ソースコード上で設定した2件のみですが、最小限の構成の流れを完成することができました。動作を確認してみましょう。



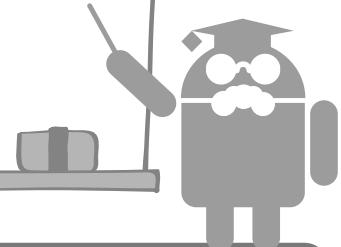
演習問題

ListViewや詳細画面のレイアウトを工夫してみましょう。文字の大きさやバックグラウンドの色指定、右寄せ、左寄せ、センタリング等、より見やすく使いやすくなるよう試してみてください。

10-2 ユーティリティによる実践(2)

著：高橋憲一

10-1ではJavaのソースコード内で「設定したデータを表示しただけ」でしたが、ここではネットワーク上にあるデータを取得して全講義の情報を表示できるようにします。



この節で学ぶこと

- ・これまで学んで来たことを組合わせて一つのアプリを作成する
- ・加えてネットワーク経由でデータを取得して表示する

この節で出てくるキーワード、Android SDKのクラス、外部ライブラリの一覧

ネットワーク	AdapterView
JSON	TextView
日付のフォーマット	View
ListViewの要素の再利用	ArrayAdapter
Android SDKのクラス	Intent
JSONObject	Activity
JSONArray	LayoutInflater
Date	外部ライブラリ
SimpleDateFormat	Volley
ProgressBar	LayoutInflater
ListView	

10-2-1 ネットワーク経由でデータを取得する

ネットワークを経由したデータの取得に、今回は「Volley(ボリー)」というライブラリを使います。グーグルにより開発されているもので、ネットワークにアクセスする際に気を配る必要があるさまざまな部分の実装を軽減してくれます。

たとえば、ネットワーク経由でデータをダウンロードする時はメインスレッドで行ってはいけません。UIが止まってしまい、動作していないアプリとして中断させられることになりますし、今のバージョンのAndroidではそのような実装がそもそもできないようになっています。そういった部分もこのVolleyのしくみに則って実装することで比較的楽にできます。

Volleyの取得と追加

Eclipseで「Gitパースペクティブ」の設定をします。図1の矢印で示す部分「Open Perspective」を押します。

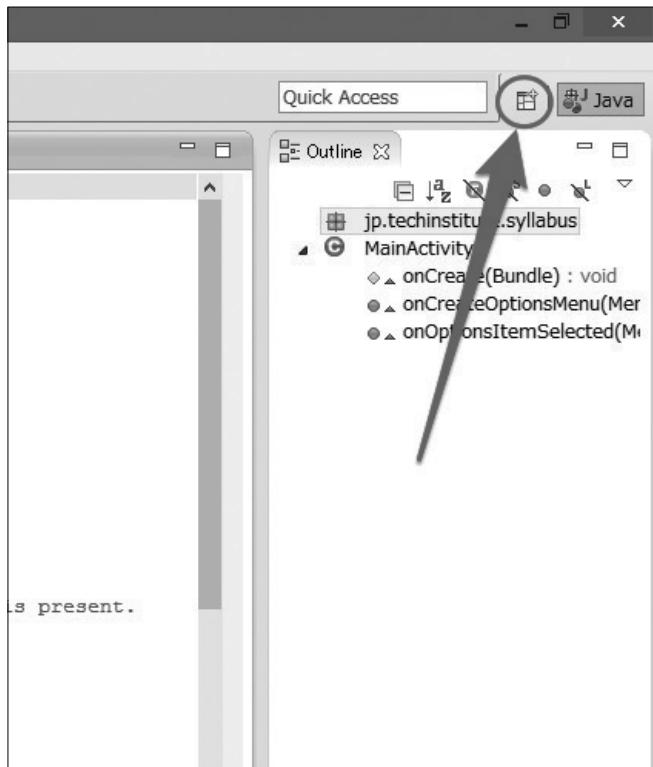


図1:Open Perspective

図2のような「Open Perspective」ダイアログが表示されます。図2と同様に「Git Repository Exploring」を選択して「OK」ボタンを押します。

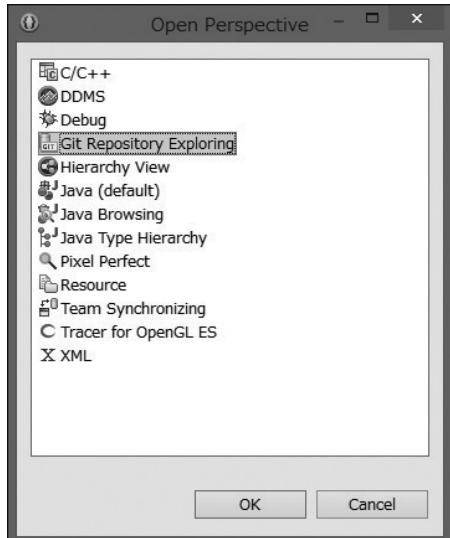


図2:Open Perspectiveダイアログ

ネットワークに関する詳細な解説は次のVolume.5で行います。ここでは必要な部分のみの解説に留めます。volleyについてはこちらを参照してください。
<http://developer.android.com/training/volley/index.html>

「Gitパースペクティブ」とは、ソースコードのバージョン管理システムである「Git」(ギット)をEclipseから使うためのものです。ここで使うVolleyも含め、オープンソースで提供されているライブラリーはGitを使用して管理／提供されているものが多く、そのソースコードを取得する際はGitのコマンドを用いることになります。ここでは、Gitパースペクティブを使用することにより、複雑なGitのコマンドを入力することなく利用できるようにしています。

Eclipseは図3のような「Git Repository Exploring」の状態になるので「Clone a git repository」を押します。

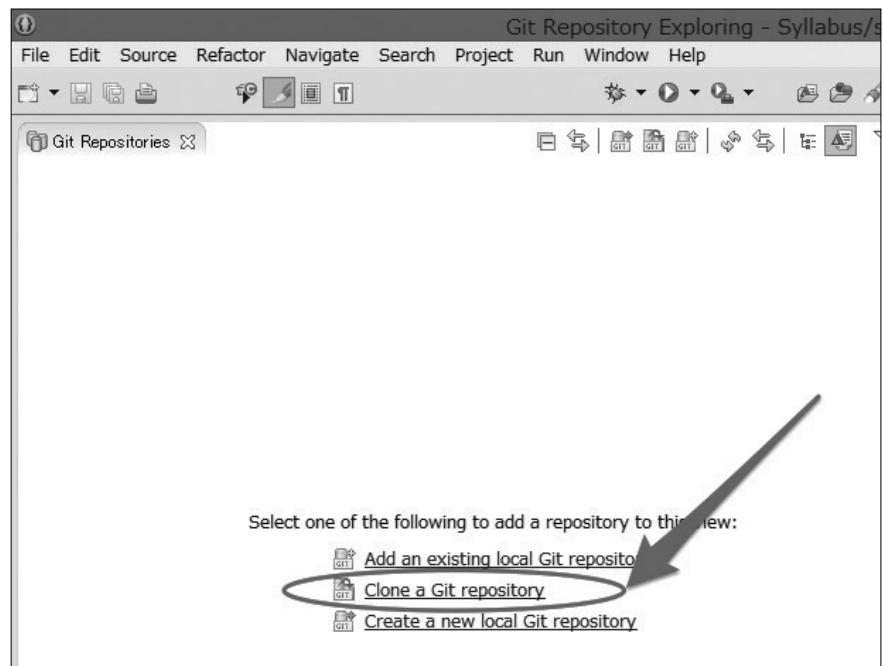


図3:Git Repository Exploring

「Clone Git Repository」ダイアログが表示されます(図4)。矢印で示したURIの部分に「<https://android.googlesource.com/platform/frameworks/volley>」と入力して「Next」ボタンを押します。

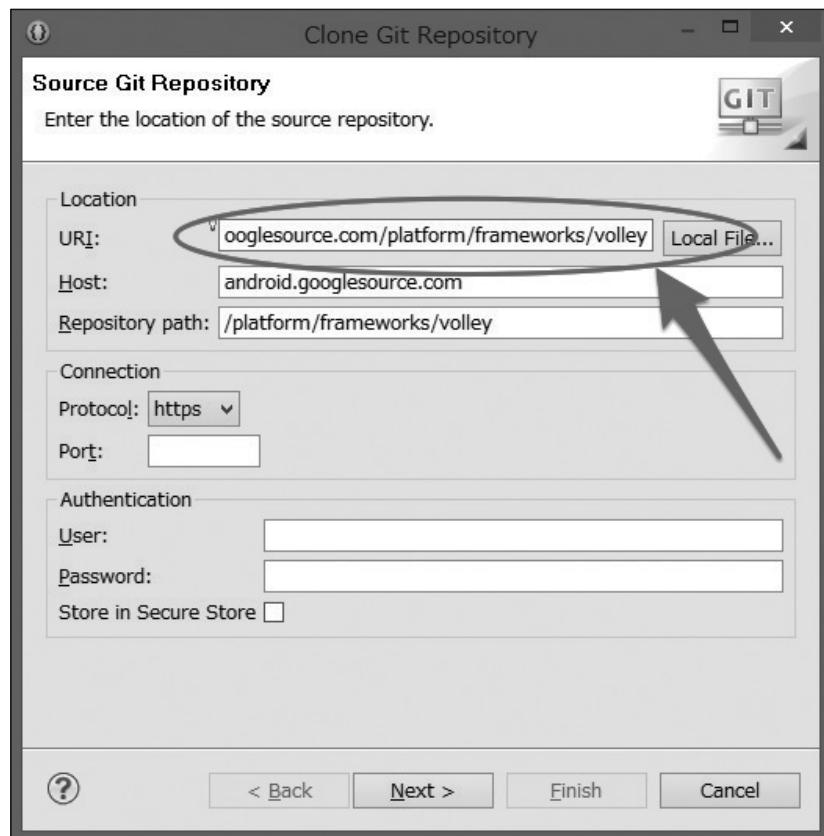


図4:Clone Git Repositoryダイアログ

「Branch」を選択するダイアログが開きますが(図5)、すべての項目(プランチ)が選択された状態になっています。ここで一旦「Deselect All」を押してすべてを解除します。

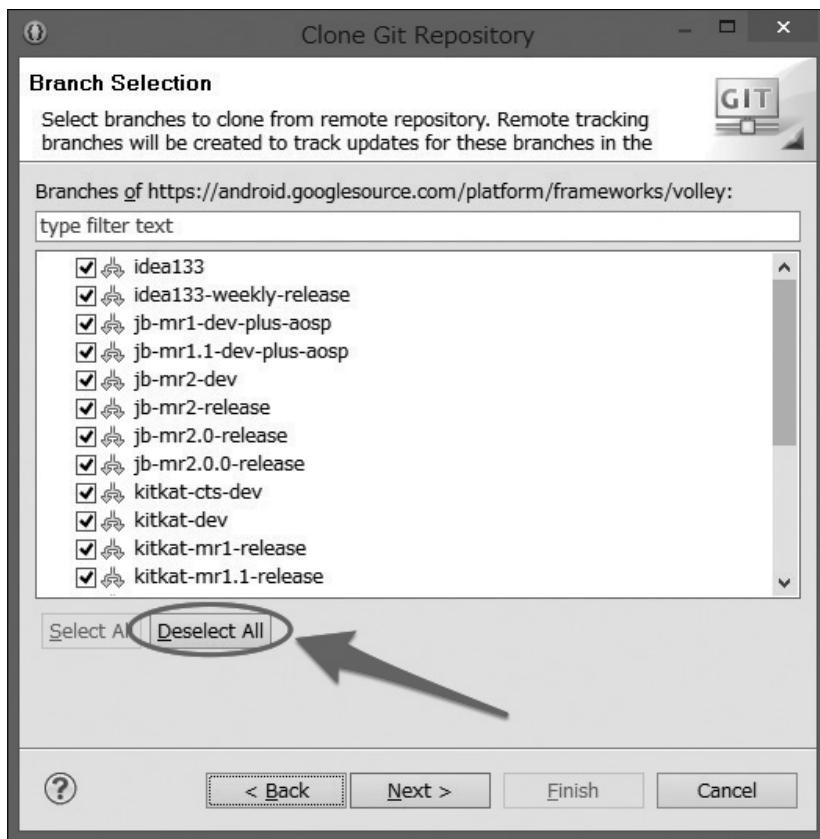


図5:Branch Selectionダイアログ

その後、図6のように「master」プランチのみを選択して「Next」ボタンを押します。

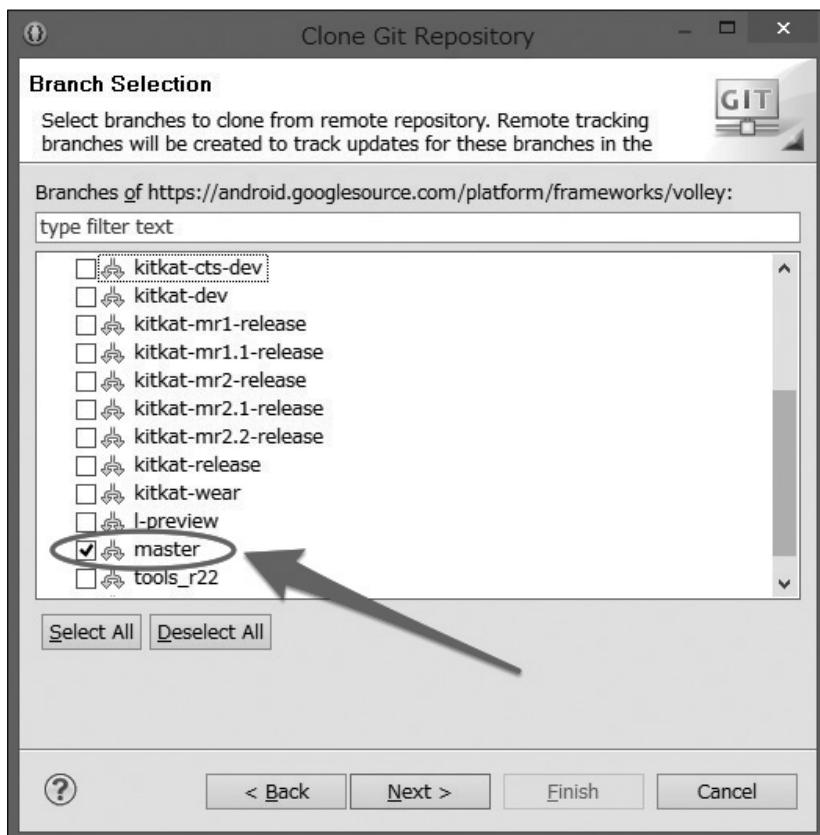


図6:masterプランチの選択

「Local Destination(ローカルでの保存先)」ダイアログが開いたら、Directory等の設定はデフォルトのままにしておきます(図7)。自動でJavaのパースペクティブにインポートされるようにするために、「Projects」の「Import all projects after clone finishes」にチェックを入れてから「Finish」ボタンを押します。

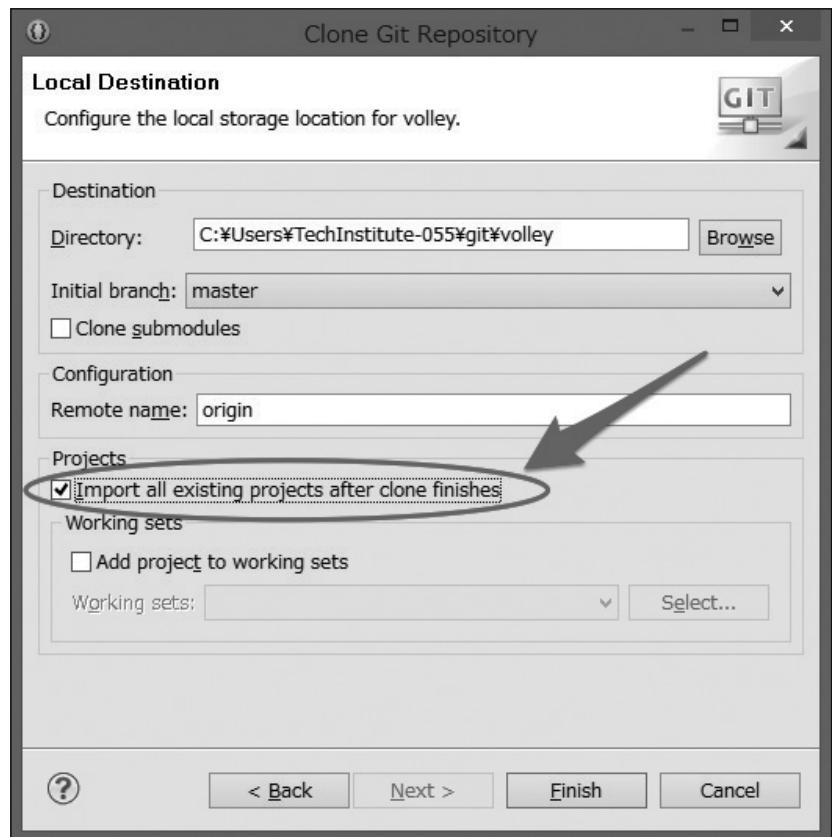


図7:Local Destination

図8のように「Git Repositories」に「volley」と表示されていれば完了です。

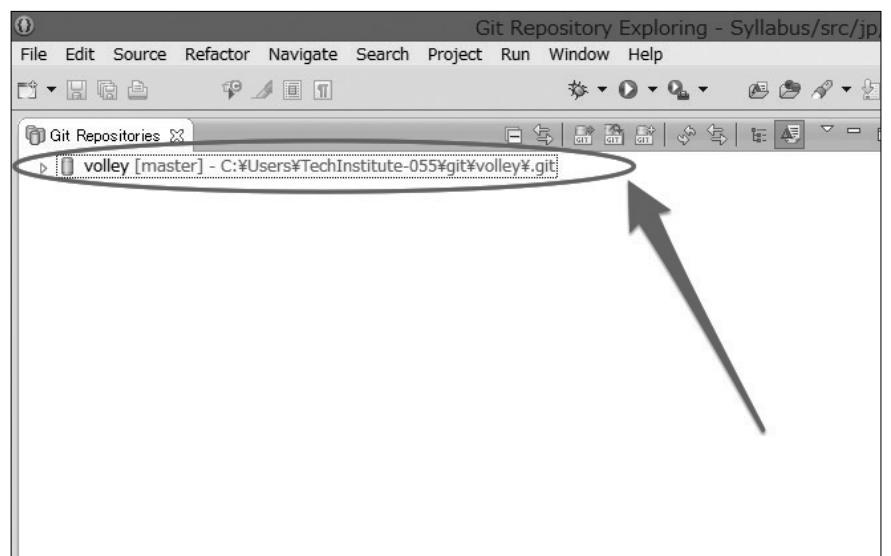


図8:Git RepositoriesにVolleyがある状態

Javaベースペクティブに戻ります。図9のようにworkspaceの中にVolleyが追加されていることを確認します。

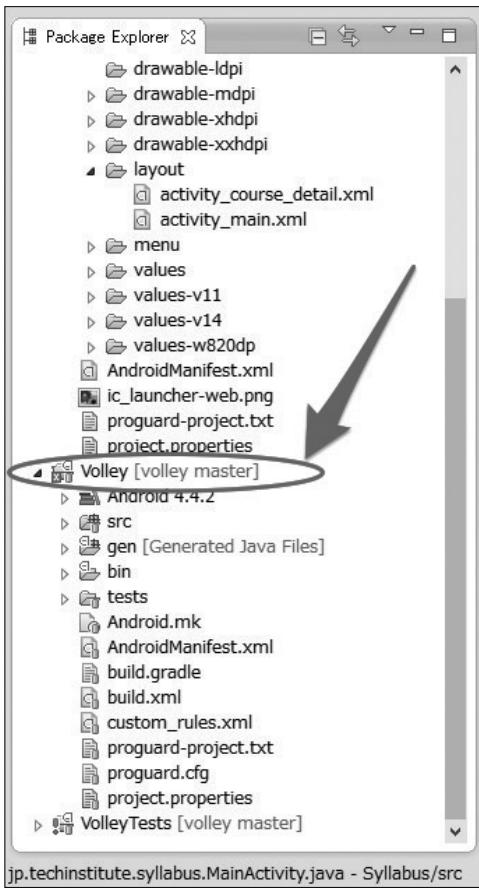


図9:Javaベースペクティブ

Eclipseのプロジェクトとしてビルドできるようにするための必須の設定が2つあります。1つ目は、VolleyプロジェクトのプロパティのAndroidを開いて"Is Library"にチェックが入っているかどうかを確認します。入っていないければ、チェックを付けて「OK」を押します(図10)。

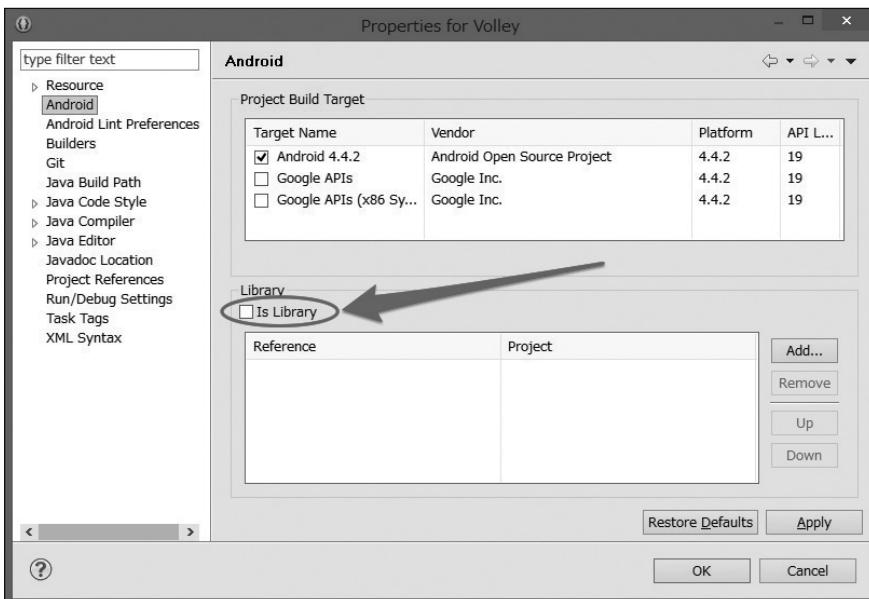


図10:Libraryプロジェクトの設定

Volleyプロジェクトで右クリックして図11のように「New」→「Folder」を選択します。

ここでworkspace中にVolleyが無い場合は、Clone Git Repository の "Import all projects after clone finishes" の部分にチェックを入れ忘れていた可能性があります。その場合でも救済方法があります。gitベースペクティブに戻ってVolleyのところで右クリック →import projectsを行います。

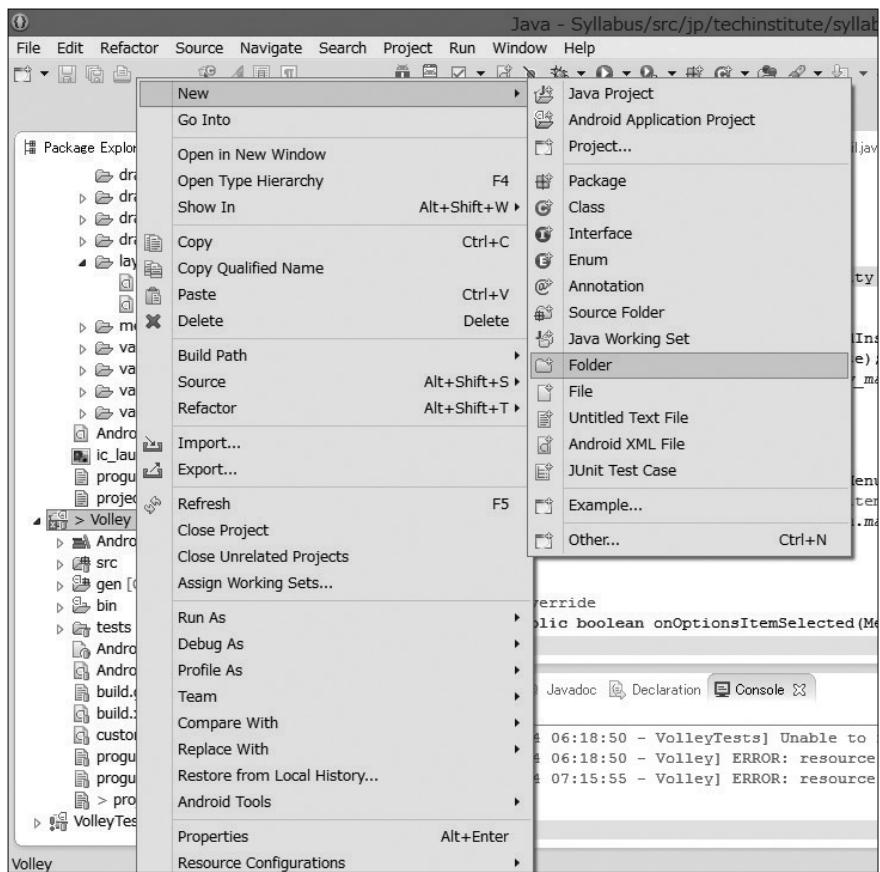


図11:Folder作成

図12のように「New Folder」ダイアログが開くので、Volleyが選択されていることを確認してから「Folder name」に「res」と入力してからFinishを押します。

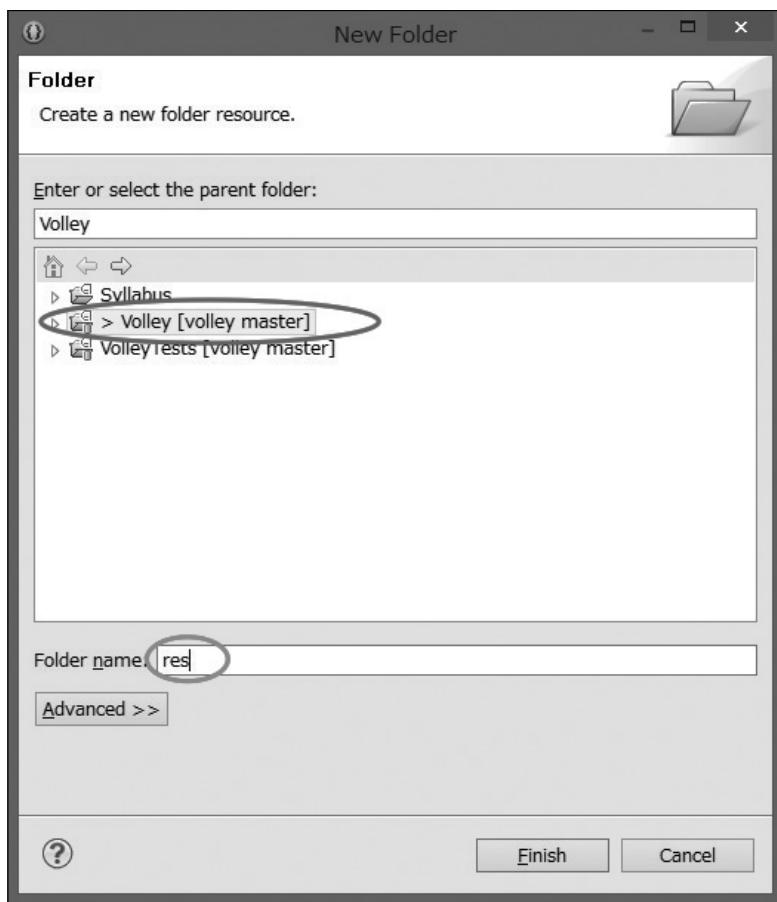


図12:New Folderダイアログ

Volleyのプロジェクトに次のようにresフォルダが追加されていればOKです。

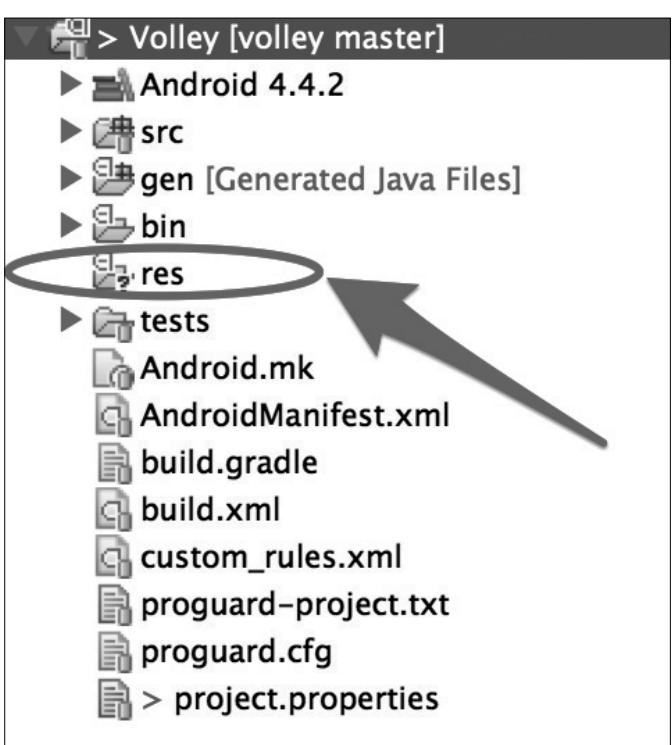


図13:resが追加されたVolley

つぎに「Syllabus」のプロジェクトに「Volley」を追加します。Syllabusプロジェクト名のところで右クリックして「Properties」を開きます。「Properties for Syllabus」ダイアログでAndroidを選択し、Libraryの枠の中の「Add」ボタンを押します。

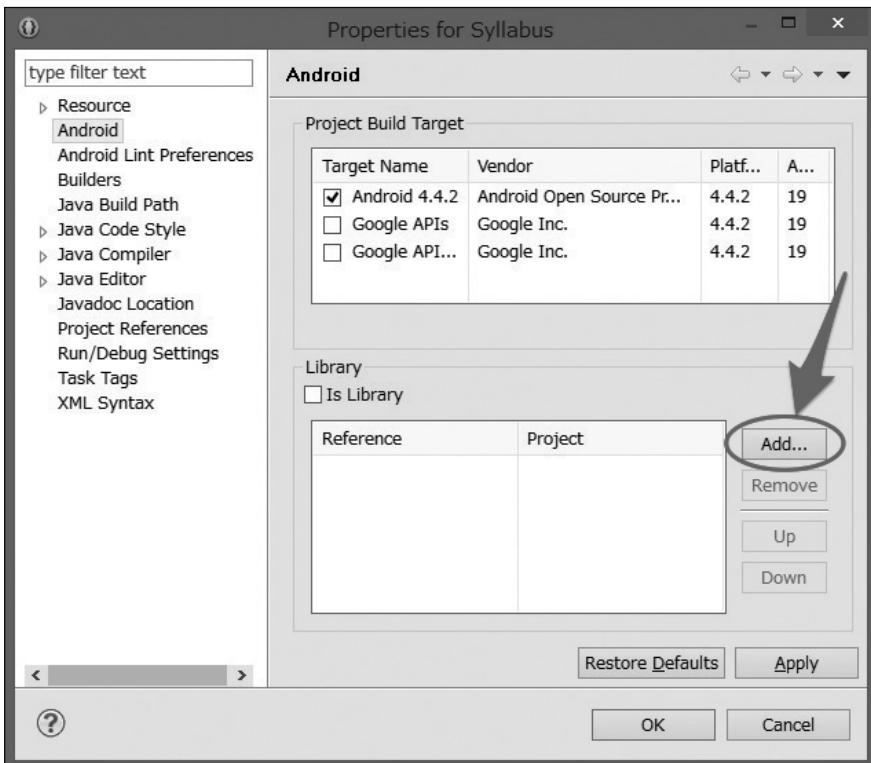


図14:Androidプロパティ

図15のようにLibraryプロジェクトを選択するダイアログが出てきますので「Volley」を選択します。

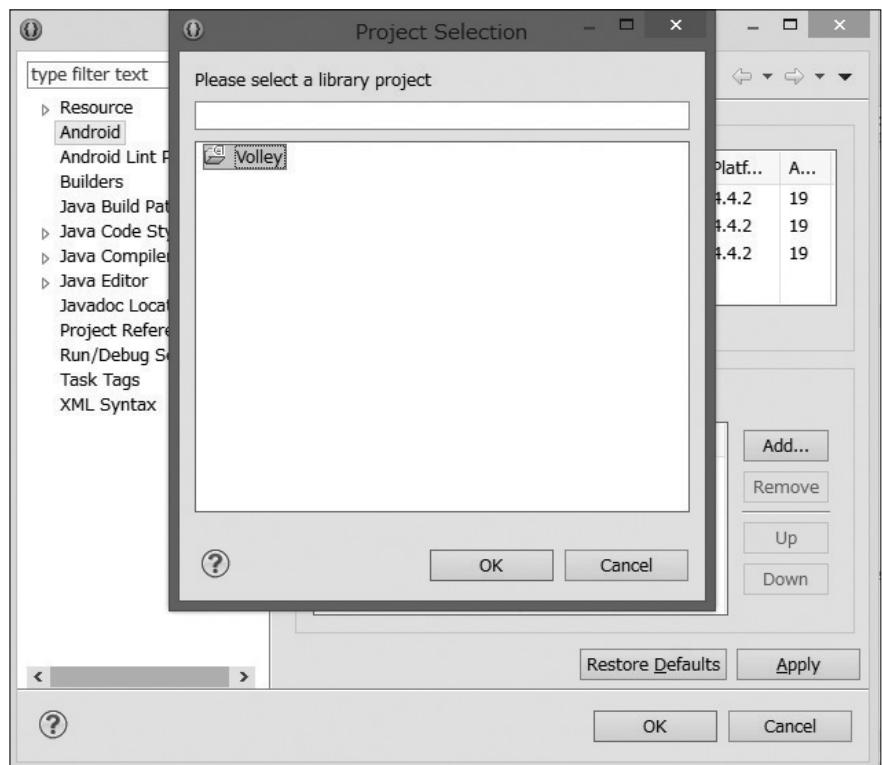


図15:Project Selection

「OK」を押してAndroidプロパティを閉じます。

データの取得

これはGoogle Driveのスプレッドシートの内容をJSON形式で取得できるようにするGoogle Apps Scriptを用いて作成したものを保存しています。

本来なら直接そのスクリプトをウェブAPIとして参照したかったのですが、ユーザー認証の実装が必要になり、現段階では処理が複雑になりすぎるから一旦データを別の場所に置くことにしました。こちらのデータを適宜アップデートするようにします。

データは次のURLでアクセスできるようになっています。

https://dl.dropboxusercontent.com/u/1088314/tech_institute/2014/syllabus.json

シラバスアプリからこのURLにアクセスするにはMainActivity.javaに次のような実装を追加します。

```

public class MainActivity extends Activity implements OnItemClickListener {

    private class CourseItem {
        String date; // ①
        String title;
        String teacher;
        String detail;
    }

    private List<CourseItem> itemList;
    private ItemAdapter adapter;

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

        itemList = new ArrayList<CourseItem>();
        adapter = new ItemAdapter(getApplicationContext(), 0, itemList);
        ListView listView = (ListView) findViewById(R.id.listview);
        listView.setAdapter(adapter);
        setCourseData(); // ②

        listView.setOnItemClickListener(this);
    }

    private void setCourseData() {
        CourseItem item = new CourseItem();
        item.date = "8/28";
        item.title = "ユーティリティによる実践(1)";
        item.teacher = "吉澤 実". // ③
    }
}

```

図16:メソッド追加位置

Dateにビルドエラーを示す赤線が付くので、マウスポインターをその上に置いて表示される黄色いウィンドウ中のimport'Date'(java.util.Date)を選択して解決します。その際にimport 'Date'の選択肢が2つ出てくるので、(java.sql.Date)の方を選ばないよう注意しましょう。

まず、図16の①で示した「String date」を「Date date;」に書き換えます。

つぎに図の②の矢印で示した枠の部分に次の変数定義を追加します。

MainActivity.java の②に追加する内容

```

private RequestQueue reqQueue;
private static final String syllabusUrl = "https://dl.dropboxusercontent.com/u/1088314/
tech_institute/2014/syllabus.json";

```

つぎに図の③の矢印で示した枠の部分に次のメソッドを追加します。

RequestQueueにビルドエラーを示す赤線が付くので、マウスポインターをその上に置いて表示される黄色いウィンドウ中のImport'RequestQueue'(com.android.volley)を選択して解決します。

MainActivity.java の③に追加する内容

```
private void getCourseData() {
    Response.Listener<JSONObject> listener = new Response.Listener<JSONObject>() {

        @Override
        public void onResponse(JSONObject response) {
            try {
                JSONArray array = response.getJSONArray("course");
                setCourseArray(array);
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    };

    Response.ErrorListener errorListener = new Response.ErrorListener() {

        @Override
        public void onErrorResponse(VolleyError error) {
            Log.e("onResponse", "error=" + error);
        }
    };

    JsonObjectRequest jsonReq = new JsonObjectRequest(syllabusUrl, null, listener,
    errorListener);
    reqQueue.add(jsonReq);
}
```

一見入力量が多く見えますが、自動補完の機能を使うとかなりの部分を自動で入力できます。「Response.Listener<JSONObject>」まで入力すると「Response」の下に赤線が付きます。ここにマウスポインターを置くと出てくる黄色いウインドウ中の「Import ‘Response’(com.android.volley)」を選択して解決します。

その後に「Response.Listener<JSONObject> listener = new Response.」と「.」まで入力すれば、自動補完の選択肢が表示されます。ここから「Response.Listener()」を選択してください。すると、「Response.Listener<JSONObject> listener=new Response.Listener<JSONObject>(){ };」となります。ここで「Response.Listener<JSONObject>()」の下に赤線が付いてるはずです。同様に黄色いウインドウを開き、エラー解決の選択肢から「Add unimplemented methods」をクリックしてください。結果は次のようになります。

MainActivity.java

```
Response.Listener<JSONObject> listener = new Response.Listener<JSONObject>() {

    @Override
    public void onResponse(JSONObject response) {
        // TODO Auto-generated method stub

    }

};
```

この中の「//TODO Auto-generated method stub」と書かれている部分に書いたコードがデータを正常に受信できた場合の処理になります。

もう1つ「Response.ErrorListener」も同様にして自動補完で途中までのコードを入力できます。

MainActivity.java

```
Response.ErrorListener errorListener = new Response.ErrorListener() {

    @Override
    public void onErrorResponse(VolleyError error) {
        // TODO Auto-generated method stub

    }

};
```

この「onErrorResponse」にある「//TODO Auto-generated method stub」以下のコードが、データ受信時に何らかのエラーがあった場合の処理になります。さらにその下にもう1つメソッドを追加します。

追加するメソッド

```
private void setCourseArray(JSONArray array) throws JSONException {
    int num = array.length();
    SimpleDateFormat inputDateFormat = new SimpleDateFormat("yyyy-MM-dd");
    for(int i = 0; i < num; i++) {
        CourseItem item = new CourseItem();
        JSONObject obj = array.getJSONObject(i);
        String dateStr = obj.getString("date");
        Date date = null;
        try {
            date = inputDateFormat.parse(dateStr);
            item.date = date;
            item.title = obj.getString("title");
            item.teacher = obj.getString("teacher");
            item.detail = obj.getString("detail");
            itemList.add(item);
        } catch (ParseException e) {
            e.printStackTrace();
        }
    }
    adapter.notifyDataSetChanged();
}
```

データのフォーマット

ネットワーク経由で取得したデータのフォーマットは「JSON」と呼ばれる形式で、
{}で囲まれている中に全講義分のデータが配列となって入っています。その配列の
1要素、つまり1講義あたりのデータは次のようにになっています。

データの内容

```
{"No":22,"date":"2014-08-28T07:00:00.000Z","day":"木","time":"19:00-21:00","chapter":"10章","title":"ユーティリティによる実践(1)","teacher":"高橋憲一","supporter-1":"武藤繁夫","supporter-2":"中澤慧","supporter-3":"金澤創平","supporter-4":"","supporter-5":"宮川大輔","detail":"この講義では一つのアプリとして仕上げることを目指します。"}
```

「"キー":値」という組み合わせで各項目が定義されており、例えば講義タイトル
は「"title":"ユーティリティによる実践(1)"」。講師名は「"teacher":"高橋憲
一"」というようになっています。

図16の④矢印が示すsetCourseData()の呼び出し部分は削除し、その部分
に次の2行を追加します。

追加する内容

```
reqQueue = Volley.newRequestQueue(this);  
getCourseData();
```

Volleyにビルドエラーを示す赤線が
付くので、マウスポインターをその上
に置いて表示される黄色いウインドウ
中のImport'Volley'(com.android.
volley.toolbox)を選択して解決しま
す。

テスト用のデータをセットしていたsetCourseData()は削除します。次の図17の
削除という矢印が示す枠の部分が削除の対象です。

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    itemList = new ArrayList<CourseItem>();  
    adapter = new ItemAdapter(getApplicationContext(), 0, itemList);  
    ListView listView = (ListView)findViewById(R.id.listView);  
    listView.setAdapter(adapter);  
    setCourseData();  
    listView.setOnItemClickListener(this);  
}  
  
private void setCourseData() {  
    CourseItem item = new CourseItem();  
    item.date = "8/28";  
    item.title = "ユーティリティによる実践(1)";  
    item.teacher = "高橋憲一";  
    item.detail = "この講義では一つのアプリとして仕上げることを目指します。";  
    itemList.add(item);  
  
    item = new CourseItem();  
    item.date = "9/2";  
    item.title = "ユーティリティによる実践(2)";  
    item.teacher = "高橋憲一";  
    item.detail = "一つのアプリを仕上げることを目指す2回目。";  
    itemList.add(item);  
}
```

図17:削除対象

```

private class ItemAdapter extends ArrayAdapter<CourseItem> {
    private LayoutInflater inflater;
    public ItemAdapter(Context context, int resource,
                      List<CourseItem> objects) {
        super(context, resource, objects);
        inflater = (LayoutInflater)context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View view = inflater.inflate(R.layout.lecture_row, null, false);
        TextView dateView = (TextView) view.findViewById(R.id.date);
        TextView titleView = (TextView) view.findViewById(R.id.title);
        CourseItem item = getItem(position);
        dateView.setText(item.date);
        titleView.setText(item.title);
        return view;
    }
}

```

図18:ItemAdapterの修正箇所

図18の①が示す部分に次のコードを追加します。

ItemAdapter の修正 1

```
SimpleDateFormat dateFormat = new SimpleDateFormat("MM/dd");
```

②が示す部分の、()の中を書き換えます。item.dateを直接渡すのではなく、
①に追加したdateFormatを使用して"08/22"のように"月/日"のみ表示するよう
にします。

ItemAdapter の修正 2

```
dateView.setText(dateFormat.format(item.date));
```

詳細に渡す日付の処理を修正するため、「onItemClick」も修正します。

```

@Override
public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
    CourseItem item = (CourseItem)arg0.getItemAtPosition(arg2);
    Intent intent = new Intent(this, CourseDetail.class);
    intent.putExtra("date", item.date);
    intent.putExtra("title", item.title);
    intent.putExtra("teacher", item.teacher);
    intent.putExtra("detail", item.detail);
    startActivity(intent);
}

```

図19:onItemClick修正箇所

図19の矢印が示す部分に「"2014/08/22"」といった西暦を追加した形式で
フォーマットするための1行を追加します。「item.date」を直接「putExtra」に渡す
のではなく「dateFormat」で変換してから渡すように修正します。

oneItemClick の修正

```
SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd");
intent.putExtra("date", dateFormat.format(item.date));
```



10-2-2 データ受信中の表示

データを受信している間は、その処理の最中であることを表示するのがユーザーにとって親切なアプリでしょう。ここでは画面の中心で回る「ProgressBar」を表示するようにしてみましょう。

レイアウトの修正

「activity_main.xml」の「ListView」定義の下に次のようにして「Progressbar」の定義を追加します。

activity_main.xml の修正

```
<ProgressBar  
    android:id="@+id/progressBar1"  
    style="?android:attr/progressBarStyleLarge"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_centerInParent="true"  
    android:visibility="invisible" />
```

android:layout_centerInParent="true"を指定することで画面の中心に表示して、初期状態では表示されないようにするためにandroid:visibility="invisible"を指定しています。

MainActivity.java の修正

```
public class MainActivity extends Activity implements OnItemClickListener {  
  
    private class CourseItem {  
        Date date;  
        String title;  
        String teacher;  
        String detail;  
    }  
  
    private List<CourseItem> itemList;  
    private ItemAdapter adapter;  
      
    private RequestQueue reqQueue;  
    private static final String syllabusUrl = "https://dl.dropboxusercontent.com/u/1088314/tech_institu-  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        itemList = new ArrayList<CourseItem>();  
        adapter = new ItemAdapter(getApplicationContext(), 0, itemList);  
        ListView listView = (ListView) findViewById(R.id.listview);  
        listView.setAdapter(adapter);  
          
        reqQueue = Volley.newRequestQueue(this);  
        getCourseData();  
  
        listView.setOnItemClickListener(this);  
    }
```

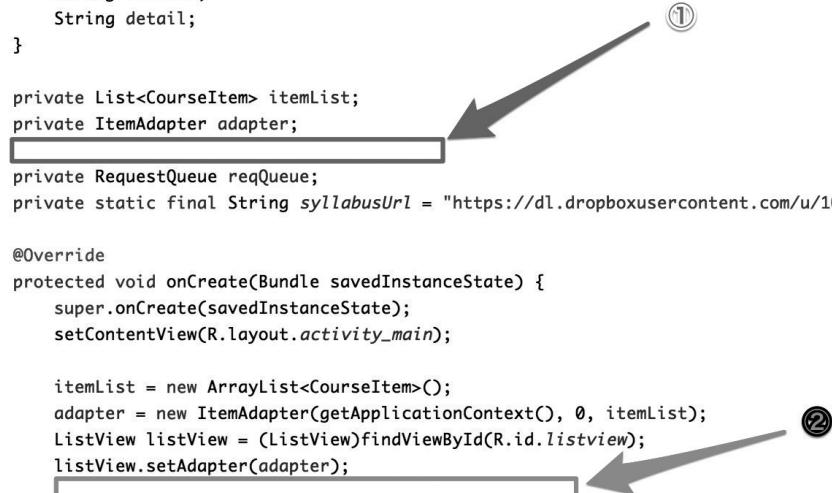


図20:ProgressBarの追加

図20の①で示される枠の部分に

```
MainActivity.java
```

```
private ProgressBar progressBar;
```

さらに、②で示される枠の部分に

```
MainActivity.java
```

```
progressBar = (ProgressBar) findViewById(R.id.progressBar1);
```

以上をそれぞれに追加します。さらに、データの受信処理に入る前に表示し、受信処理が終わったところで非表示にします。

```
private void getCourseData() {
    Response.Listener<JSONObject> listener = new Response.Listener<JSONObject>() {
        @Override
        public void onResponse(JSONObject response) {
            try {
                JSONArray array = response.getJSONArray("course");
                setCourseArray(array);
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    };

    Response.ErrorListener errorListener = new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Log.e("onResponse", "error=" + error);
        }
    };

    JsonObjectRequest jsonReq = new JsonObjectRequest(syllabusUrl, null, listener, errorListener);
    reqQueue.add(jsonReq);
}
```

図21:ProgressBarのON/OFF

図21の①で示される枠の部分(データ受信処理開始)に表示するコードを

```
MainActivity.java
```

```
progressBar.setVisibility(View.VISIBLE);
```

②で示される枠の部分に(データ受信処理正常終了)に非表示にするコードを

```
MainActivity.java
```

```
progressBar.setVisibility(View.INVISIBLE);
```

それぞれ追加します。これらによりデータ受信最中は画面の中心で回るProgressbarが表示されるようになります。



10-2-3 ListView のパフォーマンスを良くする

Listの要素の再利用

「ListView」には、画面の外に出て見えなくなった要素のオブジェクトは再利用できるようなしくみが用意されています。このしくみを利用することで、毎回要素ごとのレイアウトを展開しなくても良くなり、メモリ使用効率と動作パフォーマンスの向上に繋がります。とくに、今回は65件とまだそれほどでもない要素数ですが、数百件にも及ぶ要素をつかうようなケースでは、顕著にこのしくみが効いてきます。

ここまで実装は？

ここまで実装した「ListView」の、各要素が表示されるタイミングで毎回呼び出される「ItemAdapter」の「getView」では、実装量をシンプルにするためにあえて再利用の処理を省いていました。そのため、毎回「inflater.inflate」で「lecture_row.xml」のレイアウトを展開しており、決して効率的ではありません。ここではこの「getView」をメモリ使用効率、処理パフォーマンスの観点からも効率的なやり方に修正していきます。

```
private class ItemAdapter extends ArrayAdapter<CourseItem> {
    private LayoutInflater inflater;
    SimpleDateFormat dateFormat = new SimpleDateFormat("MM/dd");

    public ItemAdapter(Context context, int resource,
                      List<CourseItem> objects) {
        super(context, resource, objects);
        inflater = (LayoutInflater)context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View view = inflater.inflate(R.layout.lecture_row, null, false);
        TextView dateView = (TextView) view.findViewById(R.id.date);
        TextView titleView = (TextView) view.findViewById(R.id.title);
        CourseItem item = getItem(position);
        dateView.setText(dateFormat.format(item.date));
        titleView.setText(item.title);
        return view;
    }
}
```

図22:修正前のgetView

図22の①の部分に次の定義を追加します。

①に追加する内容

```
private class ViewHolder {
    TextView date;
    TextView title;
}
```

日付と講義タイトルのTextViewをメンバ変数として持つておき、これが要素の再利用のために保持するオブジェクトの型になります。

図22の②の部分は次のように書き換えます。

②を修正する内容

```
ViewHolder holder;
if(convertView == null) {
    convertView = inflater.inflate(R.layout.lecture_row, null, false);
    holder = new ViewHolder();
    holder.date = (TextView) convertView.findViewById(R.id.date);
    holder.title = (TextView) convertView.findViewById(R.id.title);
    convertView.setTag(holder);
} else {
    holder = (ViewHolder)convertView.getTag();
}
```

「 getView」の引数である「 convertView」(★印を付けた矢印)に、再利用できるオブジェクトがある場合はnull以外の値、再利用できないために新たにレイアウト(lecture_row.xml)から生成する必要がある場合はnullが入ってきます。これを利用するために「 convertView」がnullかどうかをifで判断してから処理を分けています。

新規に生成する場合は「ViewHolder」に日付と講義タイトルの「 TextView」をセットして、「 View」の「 setTag」のしくみを使って「 View」のオブジェクトで保持できるようにしています。再利用可能な場合は「 View」の「 getTag」を使って保持している「 ViewHolder」を取り出します。この変更に対応して③の部分も次のように修正します。

③を修正する内容

```
holder.date.setText(dateFormat.format(item.date));
holder.title.setText(item.title);
return convertView;
```

再利用の可否にかかわらず「 ViewHolder」 経由で日付と講義タイトルの「 TextView」にアクセスするようにしてあります。

ここまで修正で正しくListViewの要素を再利用できるようになりました。今回の65件分のデータではそれほど違いが感じられないかもしれません、ListViewを使う際はここでやった要素の再利用を心がけて実装していってください。



演習問題

詳細画面でサポート講師も表示できるようにしてみましょう。

サポート講師は日によって人数が変わるので表示領域は可変にできると良いのですが、まずは5人分のTextViewを定義してそこに表示するだけでも構いません。

