

GWT でグラフ作成

Google Visualization API を利用すると、グラフを容易に作成することができます。

ここでは、GWT に Google Visualization API を組み合わせてグラフを作成する方法を解説します。

1 グラフのサンプル

山田ツールズにアクセスしてください。

<http://yamada-tools.appspot.com/>

図 1 山田ツールズ

開発者の味方！山田ツールズ powered by Google App Engine

開発者が楽をすれば、生産性も品質も向上します。
このツールたちは、「決まり切ったことをコンピューターにやらせよう！」という思想で作成しています。
このメニューも、ツールを関連情報とともに追加すると、自動追加されるように作っています。
なお、ツール関連情報をデータベースに入力するのは @kivistyle さん作の CNMV を使用しています。
[CNMVDemo ページヘジャンプ](#)

Slim3 も利用しています。
[Slim3 公式サイトヘジャンプ](#)

[山田ツールズの歴史](#)

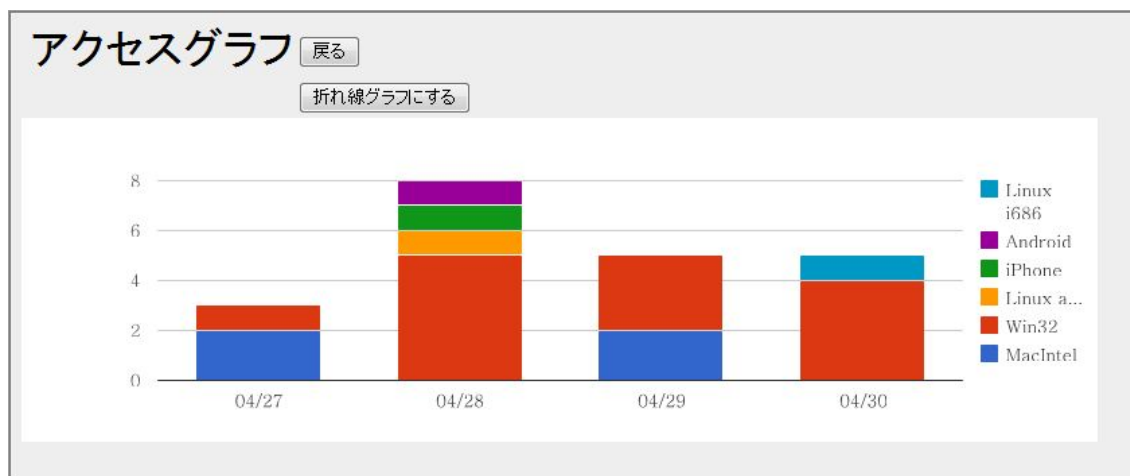
製作者: 山田健一 ツイッター: @yamada_ken1

ツール名	概要
UiField 生成	* ui.xml から @UiField Button buttongame; みたいに @UiField を生成するツールです。 内部では XML 解析ではなく、テキスト処理を使用しているため、<とか>の文字列を Text= に使っていると誤動作する可能性があります。許してやってください。
書籍サンプルのダウンロード	「オープンソース徹底活用 Slim3 による Web アプリケーション開発」のサンプルがダウンロードできます。 サンプルがバージョンアップした場合、バージョンアップ後のサンプルもダウンロードできます。
GWT Slim3 技術情報	GWT Slim3 の技術情報をまとめています。
GAE, GWT での日付利用の注意点	GAE, GWT で日付型プロパティを扱う上での注意点を記載しています。
GWT 用 Ui.xml 生成	GWT 用の Ui.xml を生成するツールです。 詳しくはこちら
アクセスグラフ	当サイトのアクセス状況を GWT の積み上げグラフで表します。 Android で見るときは、Opera Mobile のような SVG 対応ブラウザで見てください。
サービスメソッド生成 (GWT+Slim3 用)	GWT のサービス定義は、Service, ServiceAsync, ServiceImp と 3 つのクラスを作成する必要があります。 クラスの作成は Slim3 の build.xml で自動生成できますが、メソッドまでは作成してくれません。 この生成ツールでは、主キーを指定した読み取りのコードを生成します。 メソッド名は get[モデル名]ByKey となります。 読み込むキーは int, String が選択できます。 クライアントのひな型も生成します。

[GWT で作成した前年同月比グラフです。誰でも無料で利用できます。](#)
[Amazon で PC ノートを購入](#)
[Amazon で ノンコン・周辺機器を購入](#)

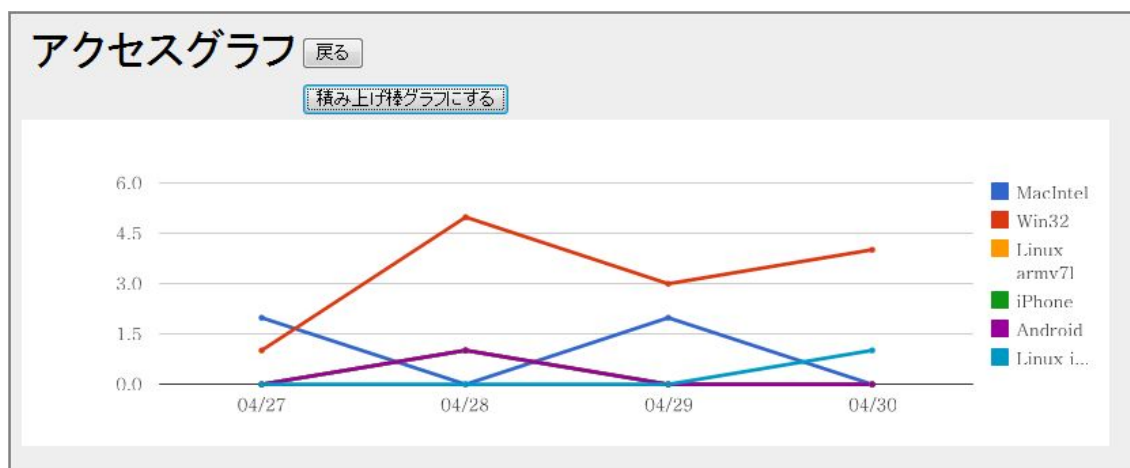
アクセスグラフを選択すると、次のようなグラフサンプルを見ることができます。

図 2 アクセスグラフ (積み上げ棒グラフ)



[折れ線グラフにする]ボタンをクリックすると、折れ線グラフに変わります。

図 3 アクセスグラフ (折れ線グラフ)

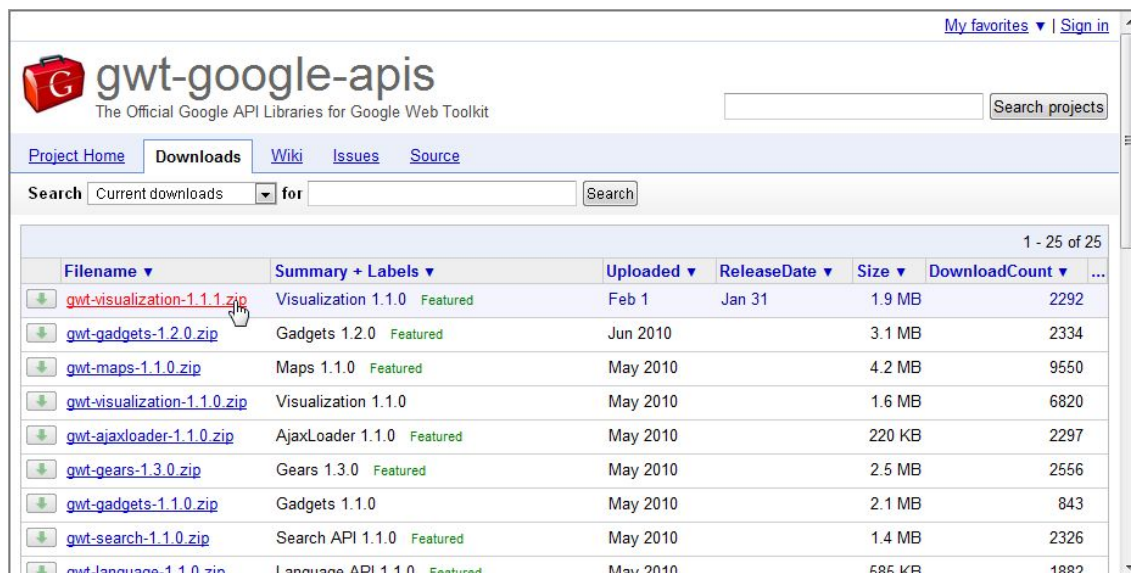


2 Google Visualization API のダウンロード

<http://code.google.com/p/gwt-google-apis/downloads/list>

にアクセスし、gwt-visualization-xxx.zip (xxx はバージョン) をダウンロードします。

図 4 ダウンロードページ



ダウンロードしたら、展開してください。

3 GWT プロジェクトへの配置と設定

Google Visualization API を展開したら、gwt-visualization.jar を GWT プロジェクトの war/WEB-INF/lib にコピーしてください。

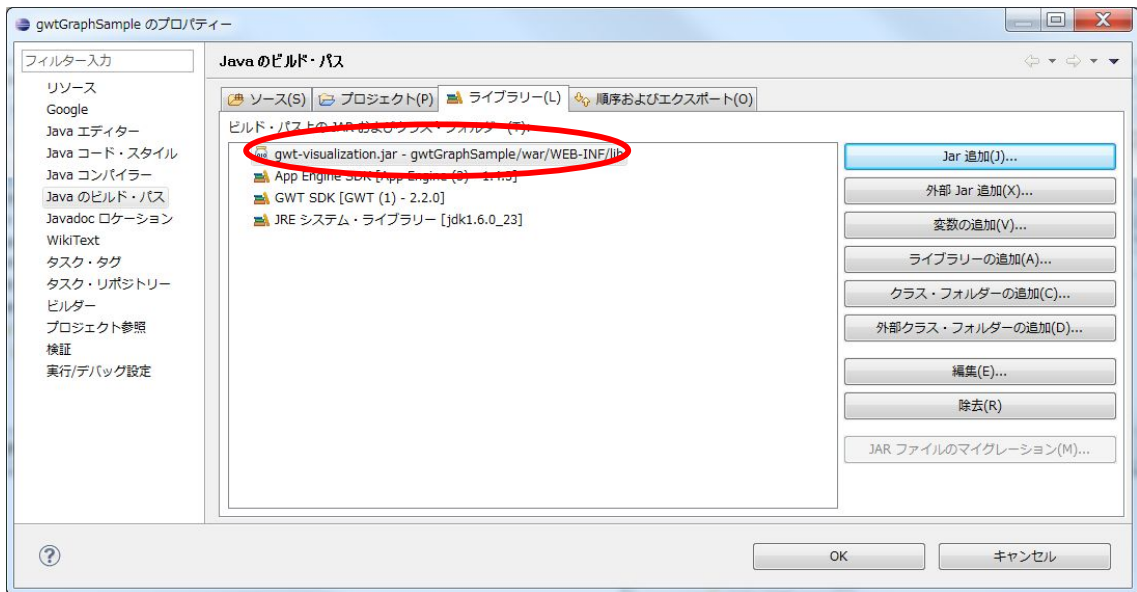
図 5 war/WEB-INF/lib に gwt-visualization.jar をコピー



コピーが終わったら、プロジェクトを右クリックし、「プロパティ」を選択し、プロパティページを開きます。

左側で「Java のビルド・パス」を選択し、「ライブラリー」タブを選択してください。次に、[Jar 追加] ボタンをクリックして gwt-visualization.jar を指定して追加してください。

図 6 gwt-visualization.jar が追加されたところ

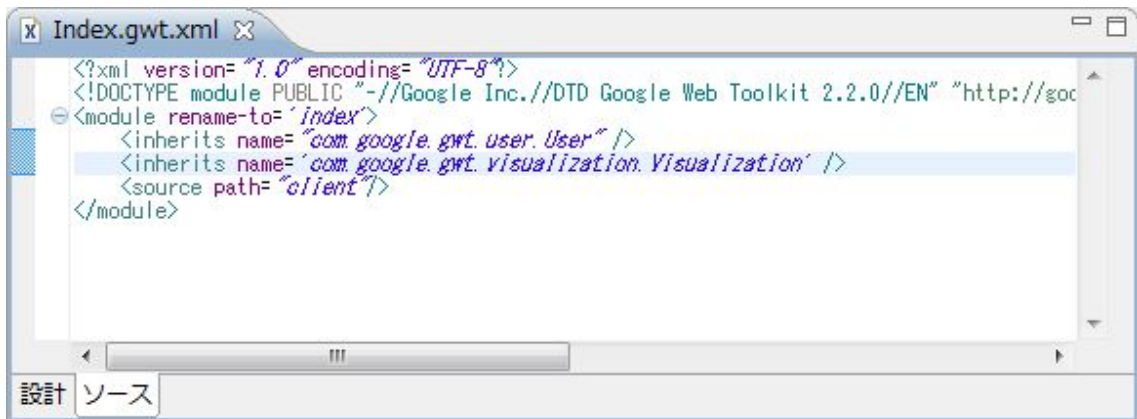


4 モジュールに inherits を追加

Google Visualization API を使用するモジュールに次の inherits を追加してください。

```
<inherits name='com.google.gwt.visualization.Visualization' />
```

図 7 inherits を追加したモジュール

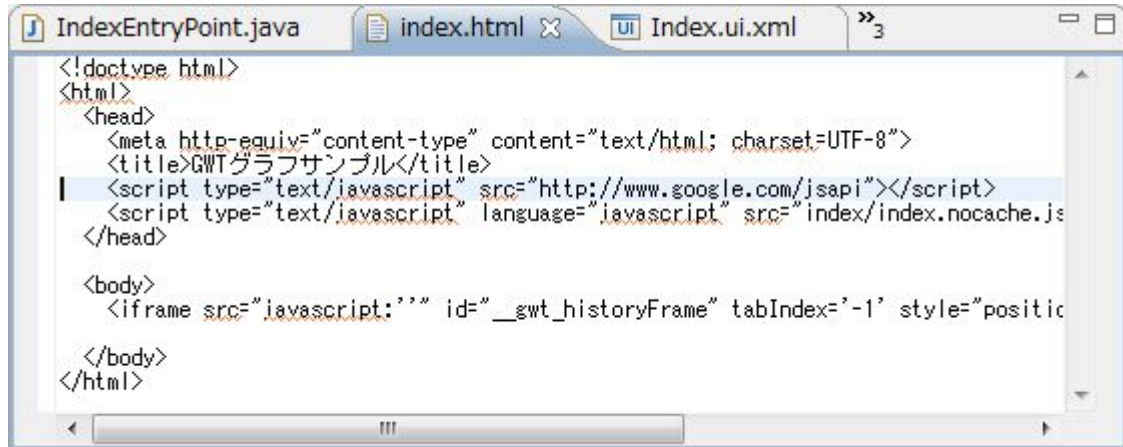


5 html に追加

Google Visualization API を使用する html に次の行を追加してください。

```
<script type="text/javascript" src="http://www.google.com/jsapi"></script>
```

図 8 html に追加



6 UiBinder

筆者が調査した限りでは、グラフを直接 UiBinder に記述することはできませんでした。

そこで、グラフを追加する HTMLPanel を UiBinder に追加します。

図 9 追加したグラフ用の HTMLPanel



7 積み上げ棒グラフ作成

積み上げ棒グラフ作成のコードは次のようになります。

コード 1 積み上げ棒グラフ作成

```
//積み上げ棒グラフ作成
private void crtColumnChart() {
    Runnable onLoadCallback = new Runnable() {
        public void run() {
            graphPanel.clear();

            ColumnChart column = new ColumnChart(createTable(), createOptions());

            graphPanel.add(column);
        }
    };

    VisualizationUtils.loadVisualizationApi(onLoadCallback, ColumnChart.PACKAGE);
}
```

`graphPanel.clear()` は必須ではありませんが、これを指定することで、グラフの書き換えができるようになります。

8 グラフオプションの指定

グラフオプションは次のように指定します。

コード 2 グラフオプション作成

```
//グラフオプション作成
private Options createOptions() {
    Options options = Options.create();
    // 描画するグラフ領域の幅
    options.setWidth(800);
    // 描画するグラフ領域の高さ
    options.setHeight(240);
    //ポイントサイズ
    options.setPointSize(2);
    //積み上げ
    options.setIsStacked(true);
    return options;
}
```

グラフの種類によっては無視されるオプションがあります。

積み上げ棒グラフではポイントサイズは無視されます。ポイントサイズは折れ線グラフのときに有効です。

`setIsStacked` で `false` を指定すると、通常の棒グラフを作成することができます。

9 折れ線グラフ作成

折れ線グラフ作成のコードは次のようになります。

コード 3 折れ線グラフ作成

```
//折れ線グラフ作成
private void crtLineChart() {
    Runnable onLoadCallback = new Runnable() {
        public void run() {
            graphPanel.clear();

            LineChart line = new LineChart(createTable(), createOptions());
            graphPanel.add(line);
        }
    };

    VisualizationUtils.loadVisualizationApi(onLoadCallback, LineChart.PACKAGE);
}
```

10 グラフデータ (AbstractDataTable) の構造

グラフデータは AbstractDataTable で渡しますが、その構造は次のようになります。

図 10 グラフデータの構造

↓ X軸	Android	iPhone	Linux armv7l	Linux i686	MacIntel	Win32	←データ系列
04/27					2	1	
04/28	1	1	1			5	
04/29					2	3	
04/30				1		3	

データ系列が列となり、X軸のデータが行になることに注目してください。

11 グラフデータの作成

グラフデータ作成のコードは次のようになります。

コード 4 グラフデータ作成

```
//データテーブル (グラフデータ) 作成
private AbstractDataTable createTable() {
    DataTable data = DataTable.create();
    data.addColumn(ColumnTypes.STRING); //日付分
    //データ系列作成
    data.addColumn(ColumnTypes.NUMBER, "Android");
    data.addColumn(ColumnTypes.NUMBER, "iPhone");
    data.addColumn(ColumnTypes.NUMBER, "Linux armv7l");
    data.addColumn(ColumnTypes.NUMBER, "Linux i686");
    data.addColumn(ColumnTypes.NUMBER, "MacIntel");
}
```

```
data.addColumn(ColumnTypes.NUMBER, "Win32");
//データの数だけ行を追加
data.addRow(4);
//1行目のセット
data.setValue(0, 0, "04/27");
data.setValue(0, 1, 0);
data.setValue(0, 2, 0);
data.setValue(0, 3, 0);
data.setValue(0, 4, 0);
data.setValue(0, 5, 2);
data.setValue(0, 6, 1);
//2行目のセット
data.setValue(1, 0, "04/28");
data.setValue(1, 1, 1);
data.setValue(1, 2, 1);
data.setValue(1, 3, 1);
data.setValue(1, 4, 0);
data.setValue(1, 5, 0);
data.setValue(1, 6, 5);
//3行目のセット
data.setValue(2, 0, "04/29");
data.setValue(2, 1, 0);
data.setValue(2, 2, 0);
data.setValue(2, 3, 0);
data.setValue(2, 4, 0);
data.setValue(2, 5, 2);
data.setValue(2, 6, 3);
//4行目のセット
data.setValue(3, 0, "04/30");
data.setValue(3, 1, 0);
data.setValue(3, 2, 0);
data.setValue(3, 3, 0);
data.setValue(3, 4, 1);
data.setValue(3, 5, 0);
data.setValue(3, 6, 3);
```



```
return data;  
}
```

図 10 と見比べてください。

実業務に応用するときは、クエリ結果の行数分、繰り返すようなコードになるはずですが。

以上