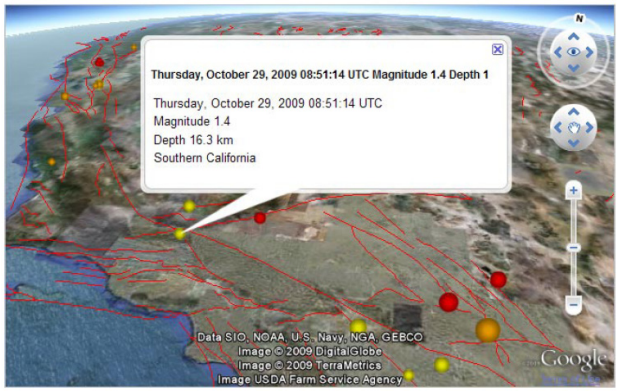
ドラフト
2009 年 12 月 30 日

動的な Web ジオマッシュアップの作成

マイクロイメージ社は、決められたスケジュールに従って定期的な処理を自動的に行うジオマッシュアップアプリケーションのデモを作成しました。このアプリケーションは、更新された地図データをインターネットからダウンロードし、GIS の処理をして他の地図データと組み合わせて、結果を Web ページにアップロードします。Google Earth のブラウザプラグインで表示することができます (テクニカルガイド「ジオメディアの公開：カリフォルニアおよびネバダ地震速報 (Geomedia Publishing: Today's Earthquakes in California and Nevada)」参照)。このアプリケーションで重要な要素は、TNT の地理空間スクリプト言語 (SML) で書かれたカスタム処理スクリプトが、TNTmips Pro のジョブ処理システムによって 1 時間毎に実行される点です。

ここで挙げる例で使用しているデータは、世界の地震の震央位置と関連する属性です。これらのデータは、米国地質調査所 (U.S. Geological Survey、以下 USGS) が地震災害プログラムでカンマ区切りのテキストファイルとして掲載、更新しているものです (http://earthquakes.usgs.gov/eqcenter/catalogs)。サンプル SML スクリプト (canvquakes.sml、2 ページ目に抜粋を掲載) は、次のような SML の機能を多く使って作られています。

- Web サーバへ接続してデータをダウンロードする HTTP_CLIENT クラス
- データインポート用の MIE クラス
- 多様な地理レイヤを含むレイアウトをレンダリングして KML ファイルを作成する KML クラス。Google Earth や Google Earth ブラウザプラグインを使って KML ファイルを表示できます。



Google Earth のプラグイン。Google の画像の上にサンプルスクリプトから作成した地震の KML ファイルを重ねて表示しています。地震の震央記号をクリックして表示されるバルーン情報は、TNT のデータティップで設定されているものです。

右上にスクリプトの処理手順をまとめました。テキストファイルをダウンロードして、地震のポイントデータや参照用の断層ラインにスタイルを付けたレイアウトを KML ファイルにレンダリングします。レイアウト内の地震の震央データは、KML ファイルのプレイスマークに変換されます。このスクリプトは、SML の KML レンダリング処理の自動処理機能を活用しています。最初に、

地震のデータを使って Google Earth のジオマッシュアップを作成するための SML スクリプトの処理手順 (canvquakes.sml)

1. USGS の Web サイトに接続し、過去 24 時間に全世界で起きた地震の震央に関するファイル (CSV ファイル) をダウンロードします。
2. テキストファイルのポイントデータを TNT の一時ベクタオブジェクトにインポートし、地震のマグニチュードや深度、その他のパラメータをポイント毎にデータベーステーブルに保存します。
3. 指定した範囲の緯度経度の震央ポイントと属性レコードを、新規の一時ベクタオブジェクトに抜き出します。
4. 文字処理フィールドをポイントデータベーステーブルに追加し、データティップとして使えるように各ポイントの地震属性から複数行にわたる文字リストを作成し、割り当てます。
5. 抜き出した震央ベクタオブジェクトを、主要な活断層線ベクタレイヤを含む既存のページレイアウトに追加します。
6. 地震の震央ベクタレイヤの設定を変更して、データティップ用のデータベースフィールドを指定します。
7. 地震の震央ポイントスタイルをスクリプト設定します。シンボルの大きさはマグニチュードよって、色は深度によって変わるようにします。
8. 変更したレイアウトを KML ファイルにレンダリングし、サンプルの Web ページから参照します。
9. KML ファイルが作成された日時のテキストファイルを作成します (Web ページ上で表示)。

データティップ用に設定されているベクタ要素の属性情報が KML ファイル内に作成されるプレイスマークの「説明 (description)」に変換されます。Google Earth でプレイスマークをクリックするとこの説明がポップアップ式の情報バルーンに表示されます。そのためにサンプルスクリプトでは、ベクタのポイントデータベーステーブル中に SML が作成する文字処理フィールドを使って、地震の各震央ポイントのデータティップを設定します。文字式を使って、テーブル内の複数のフィールドから地震に関係する属性の一覧を作成します。

また、SML スクリプトでの KML レンダリング処理は、Google Earth のマーカーライブラリの 3 次元マーカーシンボル (球または立方体) を予め定義したポイントシンボル (丸または四角) でスタイル設定されてベクタポイントに自動割当てします。そのためサンプルスクリプトは、参照用プロジェクトファイルにあるスタイルオブジェクトのスタイルを参照するクエリを使って震央のポイントシンボルを割り当てます。地震のマグニチュードや震源の深さによってマーカーの大きさや色を変えるように設定しています。

TNTmips Pro のジョブ処理システムでサンプルスクリプトをスケジュールに従って定期的に自動実行する方法については、テクニカルガイド「システム：ジョブを自動反復するスケジューリング (System: Scheduling Automatically Repeating Jobs)」をご覧下さい。

(2 ページ目へ)

www.microimages.com/downloads/scripts.htm にはダウンロード可能な多くのサンプルスクリプトがあり、TNT 製品のスクリプト言語をスクリプトやクエリでどのように利用するか解説しています。

canvquakes.sml(抜粋)

① 毎日の USGS 世界地震サマリーファイルをダウンロードします

```
class STRING address$ = "earthquakes.usgs.gov";
class STRING url$ =
  "http://earthquakes.usgs.gov/eqcenter/catalogs/eqs1day-M1.txt";
class STRING textfile$ = _context.ScriptDir + "/Download/eqs1day-M1.txt";
class HTTP_CLIENT http;
clear();

http.SetTimeOut(15);
err = http.Connect(address$, 80);

http.DownloadFile(url$, textfile$);
```

Web サーバに接続します

テキストファイルをダウンロードします

② 地震の震央に関する CSV ファイルをベクタオブジェクトのポイントにインポートします。

```
class RVC_OBJECT tempfile;
tempfile.MakeTempFile(1);
```

新規ベクタオブジェクトを tempfile にセットします

```
class RVC_OBJITEM fqObjItem;
class RVC_DESCRIPTOR fqDescript;
fqDescript.SetName("GlobalQuakes");
fqDescript.SetDescription("");

fqObjItem.CreateNew(tempfile.GetObjItem(), "VECTOR", fqDescript);
```

ベクタに対するオブジェクトアイテム

テキストから (へ) ベクタをインポート (エクスポート) するためのクラスを設定します。オブジェクトをインポートするクラスメソッドは、RVC_VECTOR クラスインスタンスではなく RVC_OBJITEM を使用します。

```
class MieTEXTVECTOR mieTV;
```

地震データ用の追加的なデータベースフィールドや座標参照系を含むインポート設定を全て記録する既存のフォーマットファイルを設定します。

```
class STRING formatFilename$ =
  _context.ScriptDir + "/resources/eqFormat.fmt";
mieTV.FormatFilename = formatFilename$;

err = mieTV.ImportObject(textfile$, fqObjItem);
printf("mie ImportObject returned %d\n\n", err);
```

③ 指定した緯度経度の範囲内にある震央ポイントを、新規の一時ベクタオブジェクトにコピーします。

```
class RVC_VECTOR CaNv_Quakes;

CreateTempVector(CaNv_Quakes, "Planar");
```

カリフォルニアーネバダ両州の震央ポイント用に新規一時ベクタオブジェクトを作成します。

目的の緯度経度範囲内にある震央ポイントを選択するためのクエリ式

```
class STRING ptQry;
ptQry = "if (CLASS.Lat > 32.5 && CLASS.Lat < 42.0 && CLASS.Lon > -
124.75 && CLASS.Lon < -114.0) return true;";
```

VectorCopy Element 関数に必要な、全世界地震ベクタオブジェクト用のクラスインスタンス

```
class RVC_VECTOR GlobalQuakes;
```

全世界地震ベクタオブジェクトを開きます

```
GlobalQuakes.OpenByName(fqObjItem.GetFilePath(),
                        fqObjItem.GetObjectPath(), "Read");
```

クエリを使って、全世界地震ベクタオブジェクトからカリフォルニアーネバダ両州の一時ベクタオブジェクトに震央をコピーします。

```
err = VectorCopyElements(GlobalQuakes, CaNv_Quakes,
                         "RemExRecords", ptQry);
printf("VectorCopyElements returned %d\n\n", err);

GlobalQuakes.Close();
```

全世界地震ベクタオブジェクトを閉じます

⑤ 主要断層線を持つページレイアウトを読み込み、それに地震の震央ベクタを追加します

```
class STRING layoutfile$ = _context.ScriptDir +
                "/resources/QuakeResources.rvc";
class GRE_LAYOUT layout;
layout.Read(layoutfile$, "FaultLayout");

class GRE_GROUP group;
group = layout.FirstGroup;
```

レイアウトのファイル名

レイアウト内の 1 つのグループ用のハンドルを入手します

レイアウト内のグループに地震の震央ベクタを追加します

```
class GRE_LAYER_VECTOR quakeLayer;
quakeLayer = GroupQuickAddVectorVar(group, CaNv_Quakes);
```

⑦ 以前に保存したスタイルスクリプトファイルを使って、震央ポイントのスタイルをスクリプト設定します。このファイルは、既に作成され、参照プロジェクトファイルのスタイルオブジェクトに保存されたポイントスタイルを使います。これらのスタイルは、地震の深度に応じて異なる色が定義された塗りつぶしの円のシンボルを使っています。KML へのレンダリングを行うと、これらの円のシンボルを Google Earth の陰影の付いた球シンボルに自動的に変換します。

使用するポイントスタイルがある作成済みのスタイルオブジェクトを設定します

```
class STRING styleFile$ = _context.ScriptDir +
      "/resources/QuakeResources.rvc";
class FILEPATH styleFilepath(styleFile$);
class RVC_STYLE ptStyles;
ptStyles.OpenByName(styleFilepath, "PtStyles.STYLE", "Read");
quakeLayer.StyleObject = ptStyles;
```

地震ポイントのスタイルをスクリプトで設定します。保存したファイルからスタイルスクリプトを読み込み、クエリ内のテーブルとフィールドの参照にエラーがないか文法チェックします。

```
class STRING ptStyleQry;
ptStyleQry = TextFileReadFull( sprintf("%s/resources/QuakeStyle.qry",
                                _context.ScriptDir) );
quakeLayer.Point.StyleMode = "ByScript";
quakeLayer.Point.Script = ptStyleQry;
```

⑧ レイアウトを KML ファイルにレンダリングします

```
print("Rendering updated layout to KML.");
class DATETIME dt;
dt.SetCurrent();
dt.ConvertToUTC();
```

現在の日時を入手し、UTC に変換します

```
class STRING datetime$;
datetime$ = dt.GetString();
datetime$ += " UTC";
printf("update datetime: \n", datetime$);
```

Web ページ用の日時を表わす文字式

テキストファイルを開き、日時用の式を書きこみます (上書き)

```
class STRING dtFilename$ = _context.ScriptDir + "/datetime.txt";
class FILE dtFile = fopen(dtFilename$, "w");
fwritestring(dtFile, datetime$);
fclose(dtFile);

class STRING kmlName$ = _context.ScriptDir + "/eqs1day_canv.kml";
class FILEPATH kmlPath(kmlName$);
```

KML ファイルの情報

```
class KML kml;
kml.SetPath(kmlPath);
kml.SetLayout(layout);
kml.SetResolution(450);
kml.Write();
```

変更したレイアウトを KML にレンダリングします

対象の解像度 450m

```
CaNv_Quakes.Close();
tempfile.Close();
```

一時オブジェクトを閉じます