合同会社 緑ＩＴ事務所

Midori IT Office, LLC

LODとSPARQL入門(3)

この記事は1年以上前に書かれました。
内容が古くなっている可能性がありますのでご注意下さい。

この連載では久々の記事となる今回は、LOD（Linked Open Data）のアクセスと結果の表示に、データビジュアライゼーションのJavaScriptライブラリD3.jsを使用します。

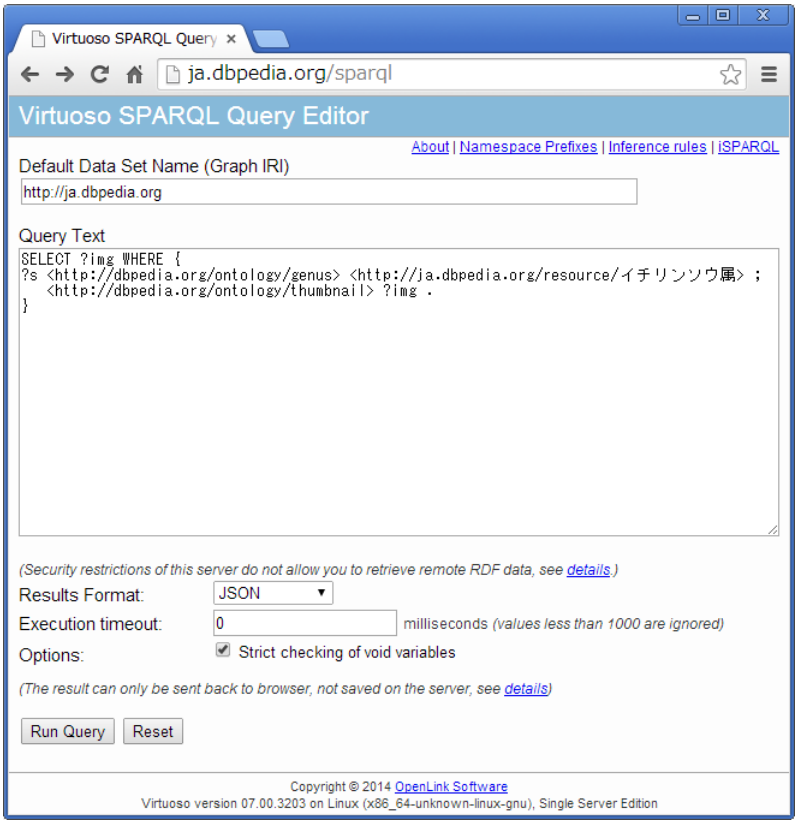
Wikipediaの情報をLODとして公開しているDBpedia JapaneseのSPARQLエンドポイントから、イチリンソウ属の植物のサムネイル画像のURLを取得し、ウィキメディア・コモンズにある画像を一覧表示することとします。
ここをクリックすると実行結果を見ることができます。

HTMLファイルの完全なリストは以下の通りです。

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>genus Anemone</title>
  <script type="text/javascript" src="d3.min.js"></script>
  <style type="text/css">
    img {
      margin: 5pt;
    }
  </style>
</head>
<body>
  <div></div>
  <script type="text/javascript">
    var server = 'http://ja.dbpedia.org/sparql';
    var graph = 'http://ja.dbpedia.org';
    var sparql = 'SELECT ?img WHERE {' +
      '?s <http://dbpedia.org/ontology/genus> <http://ja.dbpedia.org/resource/イチリンソウ属> ;' +
      ' <http://dbpedia.org/ontology/thumbnail> ?img . }';
    var query = server +
      '?default-graph-uri=' + encodeURIComponent(graph) +
      '&query=' + encodeURIComponent(sparql) +
      '&format=application%2Fsparql-results%2Bjson&timeout=0&debug=on';

    d3.json(query, function (error, json) {
      if(error) {
        console.log(error);
        d3.select("div")
          .append("h3")
          .text("Failed to get data.");
      } else {
        var imgs = json["results"]["bindings"];
        d3.select("div")
          .selectAll("img")
          .data(imgs)
          .enter()
          .append("img")
          .attr("src", function(d) {
            return d["img"]["value"];
          });
        }
      });
  </script>
</body>
</html>
```

スクリプトでは、はじめにJSONリクエストのパラメータを作成します。
DBpediaのSPARQLエンドポイントのWeb画面で、



この通り入力をした時と同じリクエストが送られるよう、パラメータ文字列を作成します。
次に、d3.json()でJSONリクエストを送ります。

```
d3.json(query, function (error, json) {
  if(error) {
    console.log(error);
    d3.select("div")
      .append("h3")
      .text("Failed to get data.");
```

JSONレスポンスが返るか、エラーが発生した場合、コールバック処理の無名関数が呼ばれます。エラーの場合には引数errorにオブジェクトが入りますので、errorがnullでないときにはエラーメッセージを出力します（実際、コネクションタイムアウトでエラーになることがあります）。
リクエストが成功すると、以下の形式でレスポンスが返ります。

```
{ "head": { "link": [], "vars": ["img"] },
  "results": { "distinct": false, "ordered": true, "bindings": [
    { "img": { "type": "uri", "value": "http://〜/〜.jpg" }},
    { "img": { "type": "uri", "value": "http://〜/〜.jpg" }},
    { "img": { "type": "uri", "value": "http://〜/〜.JPG" }},
    { "img": { "type": "uri", "value": "http://〜/〜.JPG" }},
    { "img": { "type": "uri", "value": "http://〜/〜.JPG" }},
    { "img": { "type": "uri", "value": "http://〜/〜.JPG" }},
    { "img": { "type": "uri", "value": "http://〜/〜.JPG" }},
    { "img": { "type": "uri", "value": "http://〜/〜.JPG" }},
    { "img": { "type": "uri", "value": "http://〜/〜.JPG" }},
    { "img": { "type": "uri", "value": "http://〜/〜.JPG" }} ] } }
```

従って、d3.json()でコールバック関数に返されたオブジェクトの["results"]["bindings"]で示される配列から["img"]["value"]で示される文字列を取り出せば良い事になります。

```
  var imgs = json["results"]["bindings"];
  d3.select("div")
    .selectAll("img")
    .data(imgs)
    .enter()
    .append("img")
    .attr("src", function(d) {
      return d["img"]["value"];
   });
```

まず、json["results"]["bindings"]をimgsオブジェクトに格納してから、div要素のすべての（とは言ってもこの時点ではまだ1つも存在しない）img要素を選択し、imgsオブジェクトとバインドします。enter()によって不足するimg要素のプレースホルダを作成し、append("img")でimg要素を追加し、attr()でimg要素のsrc属性を設定します。属性の値は、無名関数が返す、["img"]["value"]が示す文字列です。

実際の実行結果はこちらをご覧下さい。

カテゴリー: オープンデータ | タグ: D3.js, LOD, SPARQL | 投稿日: 2014年6月12日 [https://midoriit.com/2014/06/lod%e3%81%a8sparql%e5%85%a5%e9%96%803.html] | 投稿者: 小池隆