

RDF アプリケーション構築のための フレームワーク SparqlEPCU

A WEB Application Development Framework SparqlEPCU for RDF Store

(XML/RDF/オントロジーを使った知識の構築及び交換のシステム)
(Knowledge Warehousing and Interchange with XML/RDF/Ontology)

年岡晃一 藤本椋也 安江翔平 鈴木裕利

中部大学 工学部 情報工学科

概要

Web 上の知識流通システムとして、RDF、LOD を用いたシステムが増え始めている。しかしより普及していくには、それを支える開発環境の基盤やアプリケーション構築支援の拡充が必要である。本稿では、クラウドとして機能する RDF データ格納基盤及びそれを利用するアプリケーションサイトの構築支援機能を持つフレームワーク SparqlEPCU について報告する。我々が構築している、この SparqlEPCU を使用すると、LOD アプリケーションやそれを擁する Web サイトを JavaScript のみで開発することが可能となる。SparqlEPCU は LOD Japan2012 を通じて一般公開されている。

Keyword: RDF、LOD、フレームワーク、クラウド、マッシュアップ、WebAPI、SparqlEPCU、セマンティック Web、

1. はじめに

近年オープンデータ或いはオープンデータガバメントと称せられる世界中の動きの中で、セマンティック Web の実現方式である Resource Description Framework(RDF)を用いた Linked Open Data(LOD)のシステムが増加している。今後更なる普及を図るにはより実用的な LOD サイトが、LOD の専門家でないがデータを持っているそれぞれの分野の人たち自身の手で作られていくことが必要と思われる。しかしそういった人たちにとって LOD サイトを作成するには技術的な負荷が伴う。その一つとして、使用するデータを RDF として保管しそれに対して SPARQL による検索を行うシステムの実装が必要な点がある。

通常は DBpedia[1] や LODAC[2] などの外部のエンドポイントを利用するか、もしくは Fuseki[3] などの RDFStore を用いて自身の管理下に SPARQL エンドポイントを実装する方法が取られる。前者の場合実装自体は比較的楽なものの、エンドポイント側で提供されているデータしか扱うことができない。ユーザーからのデータ登録も可能にすることが求められる。後者の場合自分自身で作成したデータを扱えるものの、実装にはコストが掛かりシステムの肥大化を招く。いずれの場合も Web ページを構成する HTML と JavaScript の他にエンドポイントとのデータのやり取りを PHP や Java、Python などを実装することが必要な場合が多く、LOD サイト立ち上げへの一つの障害と成っている。

そこで検索機能だけではなく、エンドユーザーからのデータ登録要求を受けられる RDF Store 機能を持つ RDF サーバーをクラウド環境として構築する。様々な分野や利用環境に対応するアプリケーションはこのクラウドと切り離されることでシステム構築の分業や相互運用性の向上を図る。アプリケーション サイト向けの API として JavaScript ライブラリを提供する。このライブラリ仕様が標準とされれば1つの RDF サーバーに対し様々なアプリケーションを作れるようになる。これには携帯電話や PC の OS の違いの吸収も含まれる。また逆に1つのアプリケーションが様々な RDF サーバーにもアクセス出来るようになる。このような環境をフレームワークとして提供する。詳細は後述されるが、以下

のものから成る。

- RDF サーバー
- ライブラリの提供
- RDF 登録・検索などのデータ処理 GUI の提供
- サンプルコードの提供

2. 知識管理と LOD 構想

2.1. RDF と LOD

後述する方法の説明を補うものとして、以下 RDF や LOD の方法論について簡単に述べる。

すべての知識及び情報を明示的に記述する対象とするために、一意に特定するための手段として Web の URI を使う。各知識・情報をそのまま処理対象とするのではなくそのメタデータについて関係づけることにして述語論理をベースにして機械による自動処理を図る。従って自然言語処理や統計的な扱いなどの曖昧な意味などを扱う処理等は取り除かれる。記述対象に対しタグ（プロパティ、或いは述語）を付けるようにして他記述対象と結びつけることで意味ネットワークを形成する。

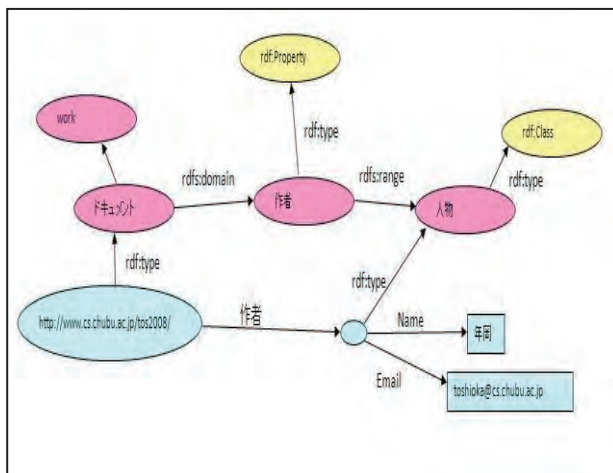


図 2.1 RDF による意味ネットワーク

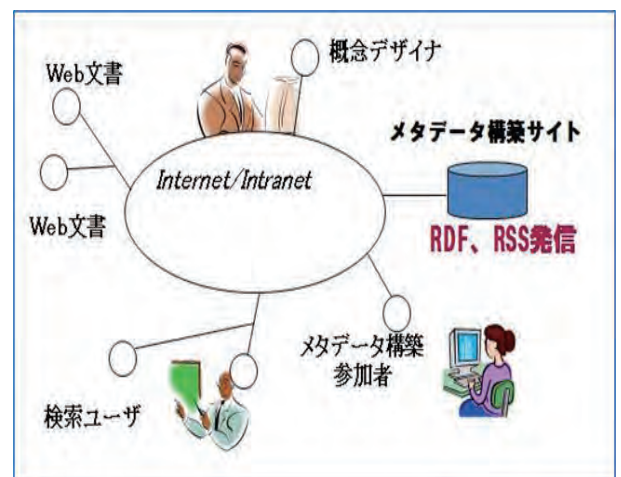


図 2.2 RDF による知識交換

この時、集合と要素或いはクラス、インスタンスを含めた意味の展開を可能なようにすることが多い。

クラスやプロパティという記述に用いるもの自身も記述対象として扱えるよう URI で特定出来るようにする。推論はプロパティなどの宣言により形式的に新たな RDF を生成することで行われる（意味論）。情報の取り出し（検索）は意味ネットワーク上のグラフ・パターンの探索となる。

まとめると LOD の実現方法として以下のような要素技術が必要となる。

1. 知識（RDF）の記述形式、つまり主語、述語、目的語という組み合わせ（いわゆる 3 つ組み）を基本構成要素とする。
2. すべての Web ページを基本的知識情報として Web 上のリソース（URI）としての扱い
3. 知識の永続化（RDF リポジトリ）
4. 知識（RDF）の発信・受信によるグローバル共有・構築と再利用
5. クラス・インスタンスのモデル

- 6.プロパティ
- 7.知識の検索
- 8.知識の推論

表 2.1 Semantic Web の要素機能と実現方法

要素機能	実現方法
流通方式（通信、メディア）	N3, RDF/XML、Turtle
格納する方式（永続化）	N-triples/RDB
取り出す方式（検索）	SPARQL
推論方式	対称律、遷移律などの推論
利用者へのインタフェース	キーワード検索+クラス/インスタンス統計など

これを実現するためのシステムの階層構成を図2に示す。本システムで扱うのは SPARQL、OWL、RDFS、RDF、URI、UNICODE である。UNICODE は、Web のデータを国際化するためにデータを格納する文字コードとして使用する。

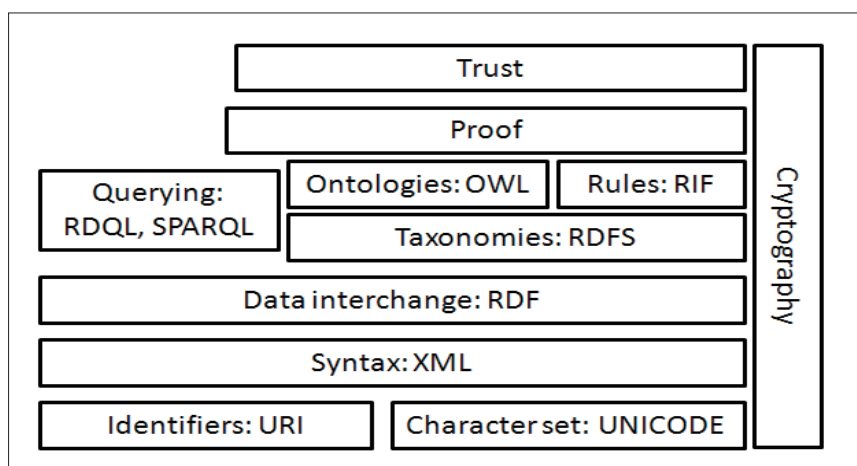


図 2.3 Semantic Web のレイヤーケーキ

3. フレームワーク SparqlEPCU

3.1. フレームワークの諸機能

フレームワークとして実現すべきことを以下に述べる。

RDF 格納をクラウドとして LOD アプリケーションサイトと別ドメインに置けるようにすること。つまりクロスドメイン通信が可能であること。そうであれば LOD アプリケーションは格納の仕方、場所などの制約から解放される。言い換えれば RDF サーバーと LOD アプリケーションは複数立ち上げて交換可能となる。

LOD アプリケーションは様々な分野・文化に渡り必ずしもプログラミング技術を深く・広く経験があるとは限らないので、アプリケーションを作るには JavaScript のみでよいようにすること。

RDF データの登録・検索に於いてはアクセス権制御が行えること。

LOD アプリケーション構築支援となるよう RDF データの登録・検索のテスト及び確認が容易に行えるような、例えば手動における SPARQL 実行などのユーザーサービスが提

供されること。

RDF データを CLOUD 上にあるものとして、GoogleMap のデータのように洗練された JavaScript オブジェクトの API を使って操作出来ること。

以上を踏まえて SparqlEPCU は設計、構築された。そのシステム構成を図 3. 1 に示す。

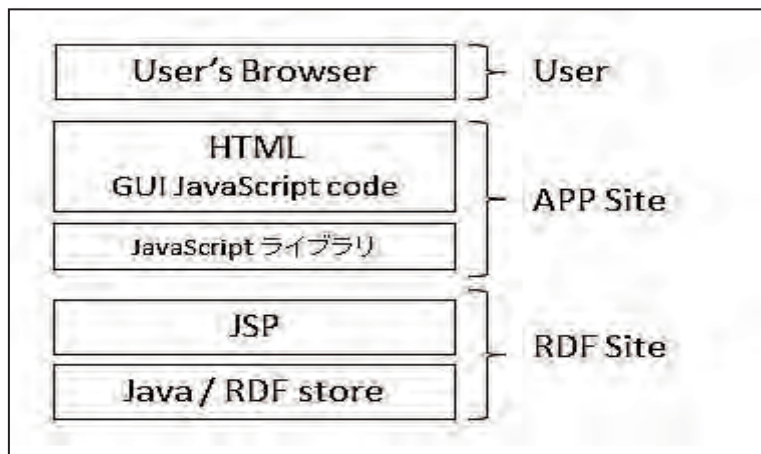


図 3.1 SparqlEPCU のシステム構成

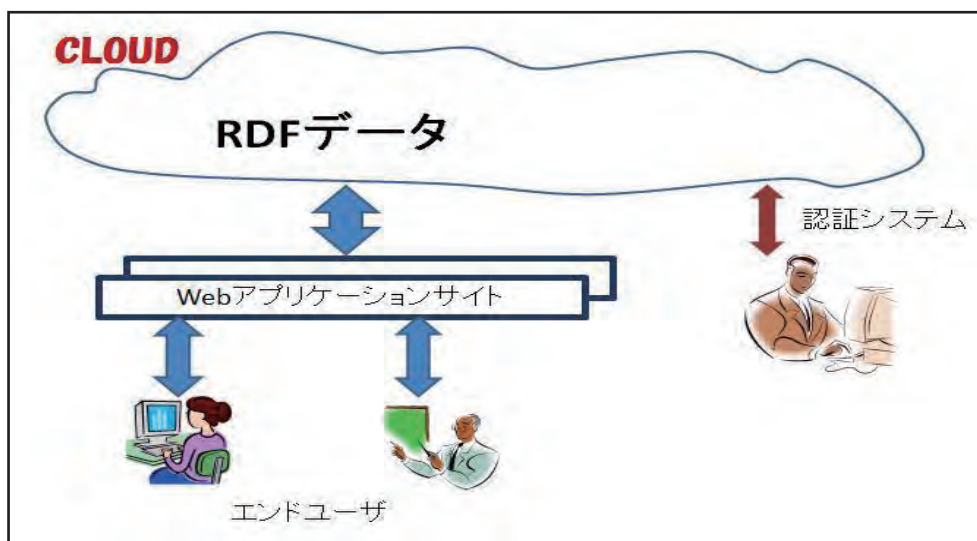


図 3.2 RDF CLOUD とユーザー

続く節に於いて、RDF サーバー、API 仕様、LOD アプリケーションの例、SparqlEPCU のサイトとしてのユーザーに提供される諸サービスについて述べて行く。

3.2. RDF サーバー

図 6 に示されるように SparqlEPCU の 3 層構造の内、最もバックエンドにあるサイトである。RDF データはここに関係データベースで永続化される。このレイヤーは CLOUD として振る舞うが、Java コードで作られており、AJAX で非同期に呼ぶことが出来る。現在は Apache の Jena プロジェクトを内部的に使用している。

このレイヤーは図 5 のように見かけ上アプリケーションレイヤーから呼ばれるように考

えることも出来るが、実はエンドユーザーのブラウザ JavaScript から AJAX で呼び出されているので注意が必要である。RDFserver は OAuth を通じてエンドユーザーの ID を取得し RDF データのアクセス権と照合した上で要求が許された範囲であればその処理を行う。Jena の SPARQL 検索結果などは JSON 形式に変換してアプリケーションに返される。

アクセス権

RDF のアプリケーションの扱いでは、Google の GMAP とは違って CLOUD のデータの改変も多くの場合要求されるのでユーザーの認証は必要である。OAuth を使って第3者サイトで認証されていればその認証を再利用できるようにしている。従って再ログインは必要無いし、認証トークンがネット上に流れるだけで、ID、PW そのものは NET を流れないので安全性は高い。

プロジェクト

SparqlEPCU の利用者は Web を通じて利用するので互いに独立にデータ登録し且つ管理運営を行うケースが多いと思われる。そこで RDF データはプロジェクトと称するデータグループに分け、アクセス権と所有者をそれぞれ持たせるようにする。

データ入出力

RDFserver への入力 は表形式と RDF 表現の N3 形式を用いる。詳細は後述。出力は JSON 形式とする。従って知識や情報という様々な形態はこれらの具体的データ形式とのマッピングを RDFstore の外のどこかで図る必要がある。

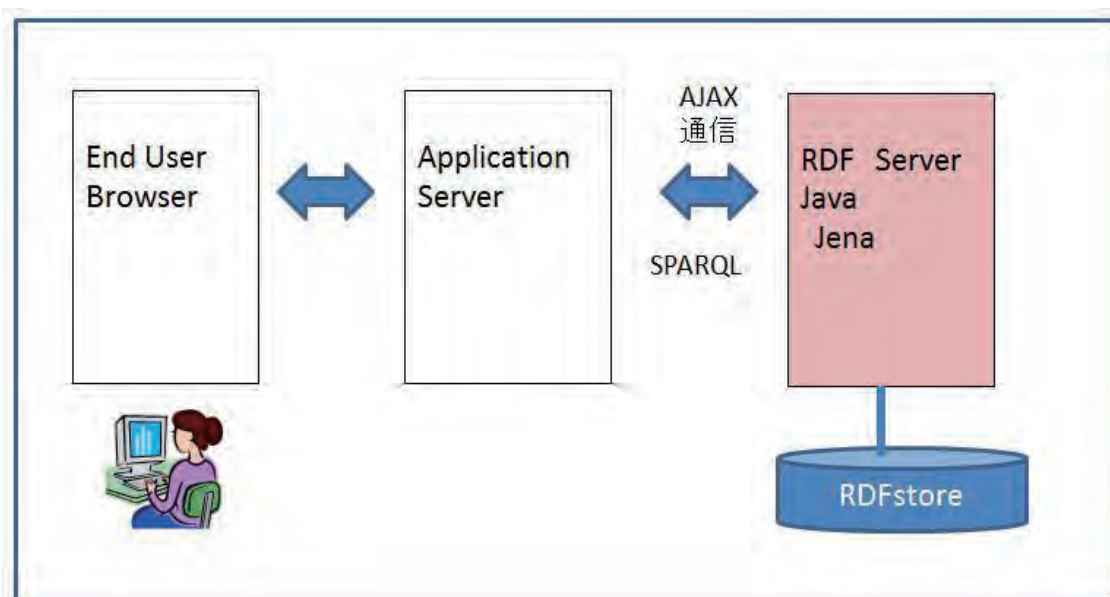


図 3.3 Cross Domain の 3 層構造

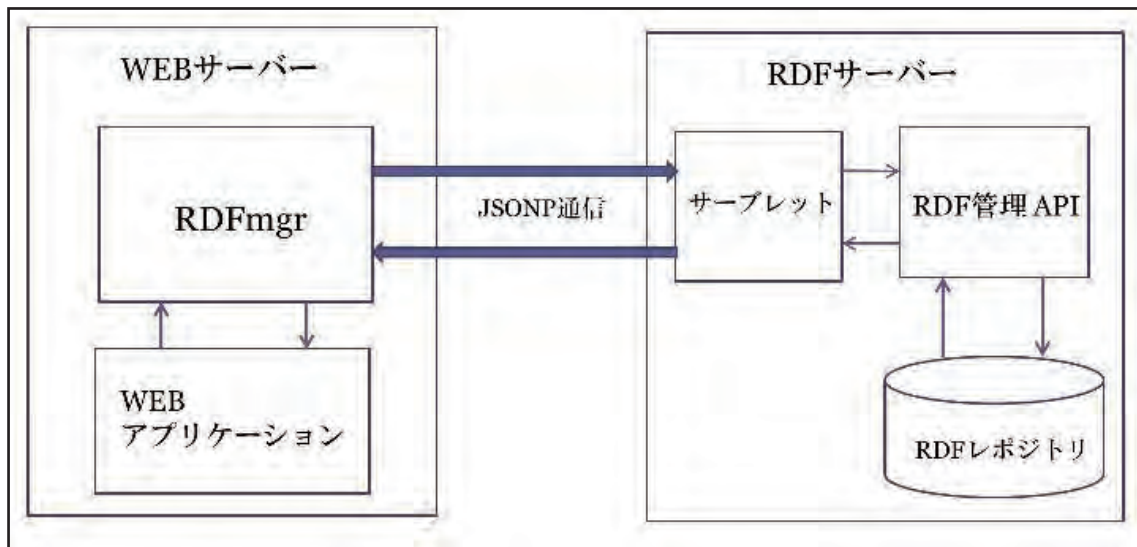


図 3.4 アプリケーションサイトと RDF サイト

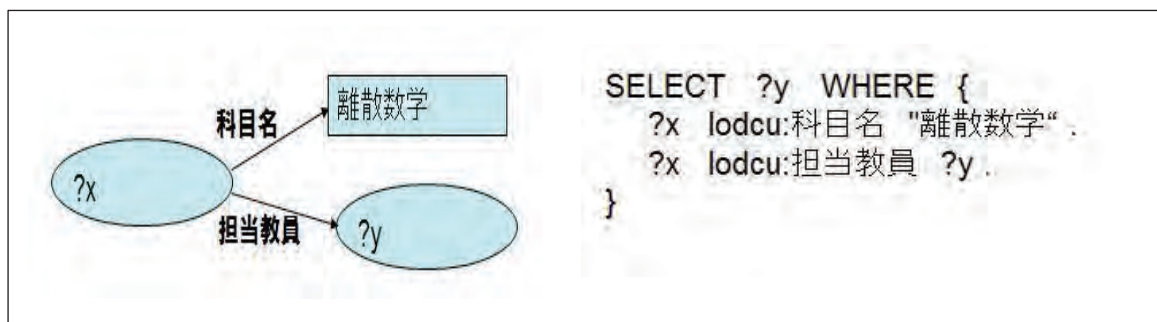


図 3.5 SPARQL 検索式

3.3. WebAPI

RDF サーバーへの要求及び通信はすべて RDFmgr という JavaScript オブジェクトを介して行われる。このライブラリは SparqlEPCU サイトから HTML ファイル内の link 宣言で持って来ることが出来る。RDFmgr.js が利用可能であれば先ず以下の宣言で JavaScript オブジェクトを生成し、それ以降はそれが持つメソッド呼び出しで行われる。この手順は Google Map や Facebook などと同じであり、様々な Web API を同時利用することで Web アプリケーションを作るマッシュアップ技術である。この形式は非同期でサーバーを呼び出すことが出来るのとコールバックのイベント処理の記述が容易になるということで GUI の大幅な動作改善が得られることに尽きる。

それに加えて RDF データをクラウドへ持って行くことになりデータ格納と様々な GUI 実現が必要なアプリケーションを分離することで開発の分業が可能となることである。

Web API の仕様は SparqlEPCU の Tutorial に詳細があるが、以下に特に有用な機能を紹介する。

先ず使用にあたって、最初に Javascript のオブジェクトを次の様にして生成する。

```
var rdfmgr = new RDFmgr("project_name", http://server_url/rdfServer");
```

次のこのオブジェクト rdfmgr が所有するメソッド群を呼び出していくことになる。以下にこのオブジェクトが持つメソッド一覧を示す。

表 3.1 RDFmgr のメソッドリスト

メソッド名	説明
executeSparql()	指定したプロジェクトに対して SPARQL を投げ、検索結果を受け取る
insertInstance()	指定したプロジェクトに CSV の表形式で RDF を登録する
updateInstance()	指定したインスタンスの RDF のオブジェクトを指定したものに書き換える
deleteInstance()	指定したインスタンスを削除する
insertStatement()	N3 などの形式で RDF を登録する
updateStatement()	プロジェクト内の RDF としてのステートメントを指定したものに書き換える
deleteStatement()	プロジェクト内の RDF としてのステートメントを削除する
clearProject()	指定したプロジェクトの内容を空に初期化する
projectDump()	プロジェクトの内容を、N3 など指定した形式で文字列として取得する
setURL()	通信先の RDF サーバーの URL を変更する
getURL()	通信先の RDF サーバーの URL を取得する
setDefaultProject()	デフォルトで使用する projectID を変更する
getDefaultProject()	設定されているデフォルトで使用する projectID を取得する

RDF サーバーは URL で指定できるようになっているため、アプリケーションサイトは、これを変更することで異なるデータセットを持つ RDF サーバーの間で切り替えることが出来る。

続くセクションでは代表的な2つのメソッドについて説明する。

検索要求を行う

指定したプロジェクトに対して SPARQL を投げ、検索結果を受け取るコードを以下に示す。

```

1.  rdfmgr.executeSparql({
2.      sparql: "select * where{?s ?p ?o}",
3.      success: function(result){
4.          $("#table1").empty();
5.          var str = new String("<tr>");
6.          for(var i=0; i<result.getKeyListLength();i++){
7.              str += "<td>"+result.getKey(i)+"</td>";
8.          }
9.          str += "</tr>";
10.         while(result.next()){
11.             for(var i=0; i < result.getLength();i++){
12.                 str += "<td><pre>"+result.getValue(i)+"</pre></td>";
13.             }
14.             str += "</tr>";
15.         }
16.         $("#table1").append(str);
17.     },
18.     error: getErrorMsg
19. });

```

図 8. 検索実行のメソッド呼出しのコード

このメソッド実行は、JavaScript オブジェクトを引数として渡す形式となっている。以下に引き渡すオブジェクトのプロパティのリストを示す。

表 3.2 検索実行メソッドがサーバーに送るオブジェクトの持つプロパティ

JS プロパティ	Type	説明
success	function	通信成功時に実行されるコールバック関数
error	function	RDF サーバー内でエラーが発生した場合に実行されるコールバック関数. 引数としてエラータイプとエラーメッセージ、エラーに関する情報を取得できる.
connectionError	function	RDF サーバーとの通信に失敗した場合に実行されるコールバック関数
projectID	string	使用する project の ID を指定
sparql	string	SPARQL 構文を指定
inference	boolean	SPARQL 検索を行うとき、true が指定された場合は推論を実行し、false の場合は推論を実行しません。デフォルトでは true.
repalcePrefix	boolean	SPARQL 検索を行うとき、true が指定された場合は URI を別途登録された prefix を参照して置換し、検索結果を簡潔に表示する。false の場合は検索結果が URI でそのまま表示される。デフォルトでは true.

検索結果の詳細を取り出すにはイテレータを利用する。JSON 形式で返された内容を、イテレータを利用して簡単に取り出すことができる。その内容は、SPARQL で?マークで指定した変数名が格納される KeyList と、それに対応する値が格納される ValueList に分かれている。

イテレータメソッド一覧を以下に示す。

表 3.3 イテレータメソッド

メソッド名	説明
getJson()	検索結果を JSON 形式で取得する.
getKeyList()	KeyList を配列として取得する.
getKeyListLength()	KeyList の長さを数値として取得する.
getKey(i)	KeyList の i 番目の値を取得する.
getValueList()	ValueList を配列として取得する.
getValueListLength()	ValueList の長さを数値として取得する.
next()	ValueList を 1 行分進める. ValueList の長さを超えた場合 false を返し、そうでない場合は true を返す.
getList()	ValueList の現在いる行の内容を配列として取得する.
getLength()	ValueList の現在いる行の配列長を数値として取得する.
getValue(keyname)	ValueList の現在いる行から変数名を指定して値を取得する. 数値を指定した場合 KeyList に準じた順に値を取り出す.
getType(keyname)	getValue()と同じ方法で値の型を取り出す(例: uri, literal).
reset()	next()で進められた行数をリセットする.

登録要求を行う

ここでは表形式になった情報を RDF 登録するメソッド呼び出しを説明する。複数の要素を持つ集合を登録すると考えてよい。図 7 を参照。

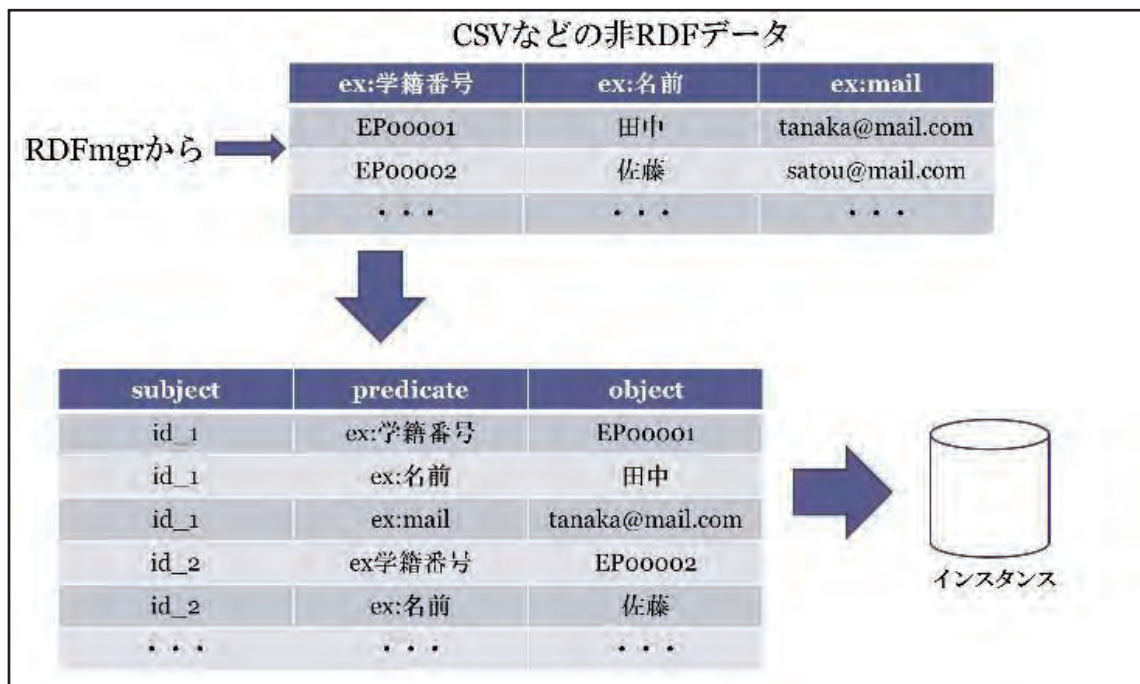


図 3.6 CSV を RDF データとして登録する

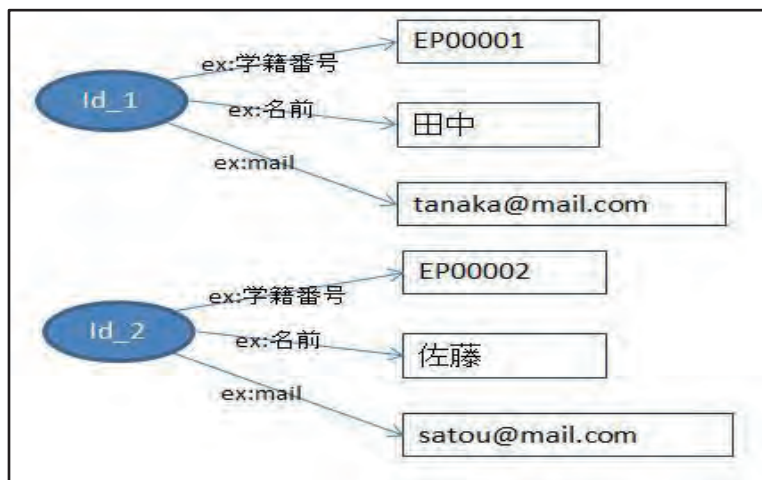


図 3.7 登録される RDF グラフ

指定したプロジェクトに CSV の表形式で RDF を登録するコードを以下に示す。

```

1.  rdfmgr.insertInstance({
2.      rdfdata: csvText,
3.      reset: false,
4.      success: func,
5.      projectName: "project1"
6.  });
  
```

図 3.8 CSV を RDF データとして登録するコード

ここでは `rdldata` プロパティに `csvText`(文字列)データを割り当てている。`csvText` のデータはブラウザの `textarea` などから持って来る単なる文字列であるが以下にその例を示す。

```

1. @学生,ex:学籍,名前,mail;
2. EP09038,藤木 晋太郎,fujiki_s@mail.com;
3. EP09000,青山 哲也,aoyama_t@mail.com;
4. EP09001,伊藤 信士,ito_s@mail.com;
5. EP09002,遠藤 京子,endo_k@mail.com;

```

図 3.9 RDF として登録する CSV データ

表の内容は列名の先頭に@をつけて指定された名前のクラスのインスタンスとして扱われる。表の 1 行分を 1 つのインスタンスとし、`id_5` 等の行番号を `subject` として、列名を `predicate`、各値を `object` とした RDF の形式で登録される。

表 3.4 登録メソッドで引数として渡される JavaScript オブジェクトのプロパティ

JS プロパティ名	Type	説明
Success	function	通信成功時に実行されるコールバック関数
Error	function	RDF サーバー内でエラーが発生した場合に実行されるコールバック関数。引数としてエラータイプとエラーメッセージ、エラーに関する情報を取得できる。
connectionError	function	RDF サーバーとの通信に失敗した場合に実行されるコールバック関数
projectID	string	使用する project の ID を指定
Rdldata	string	CSV の表形式データをテキストで指定する。もしここに配列を指定した場合、 <code>columnList</code> に指定された配列を各列名、 <code>rdldata</code> の配列を各値として、1 行分の CSV を整形しサーバーに送信する。
columnList	array	<code>rdldata</code> を参照。
overwrite	boolean	<code>true</code> が指定された場合プロジェクトに対して上書きを行う。元の内容は初期化されてしまうため注意。デフォルトは <code>false</code> 。

3.4. LOD アプリケーションのサンプル

SparqlEPCU のフレームワークを使った LOD アプリケーションの例をいくつか以下に示す。いずれも SparqlEPCU サイトの Demo タブの中でサンプルとして紹介されているもので、例えば或る地域の公共施設の情報を RDF に格納しそれをブラウザにより Google API とのマッシュアップで作られているなどである。各場所の属性情報がインタラクティブに表示される。

これらの情報は階層構造化された RDF クラスのインスタンスである。エンドユーザーは地図上の情報を様々な条件検索で取り出すことも可能である。情報の格納と検索の機能がクラウドの RDFserver でやってくれているので、LOD アプリケーションはエンドユーザーの入力や出力画面の構成などに専念できる。

鯖江市公共施設

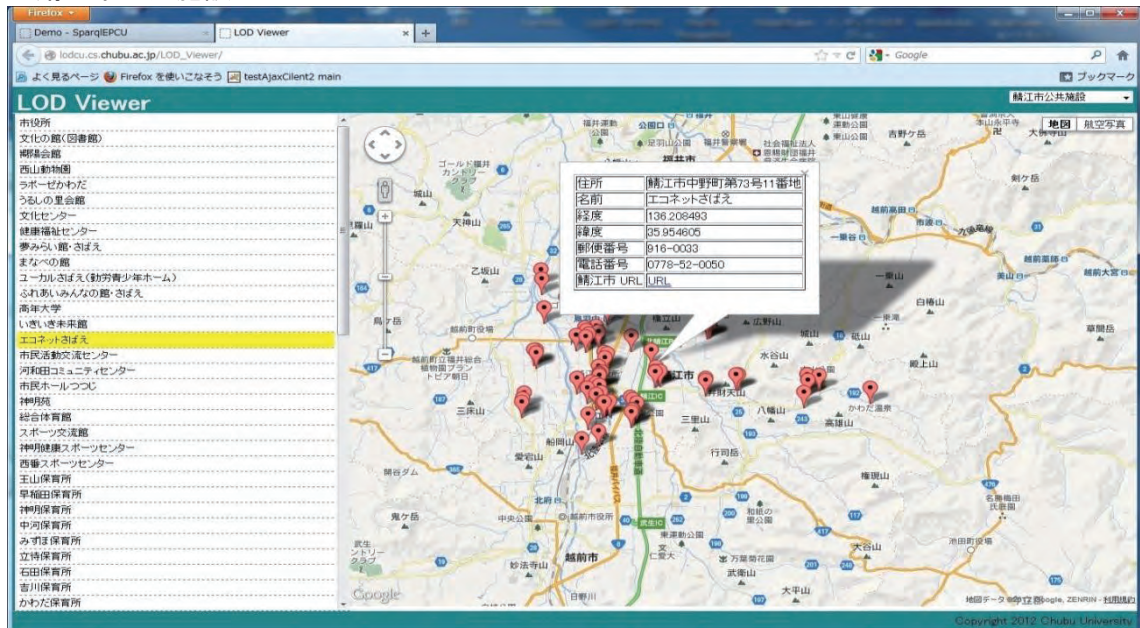


図 3.10 LOD viewer クラス：公共施設

ここで紹介しているアプリケーション LOD Viewer のコードでは、検索結果に緯度・経度の位置情報があれば自動的に Google Map を生成し指定位置を表示できるようにしている。もし位置情報が無ければ表形式で表示する。

観光施設

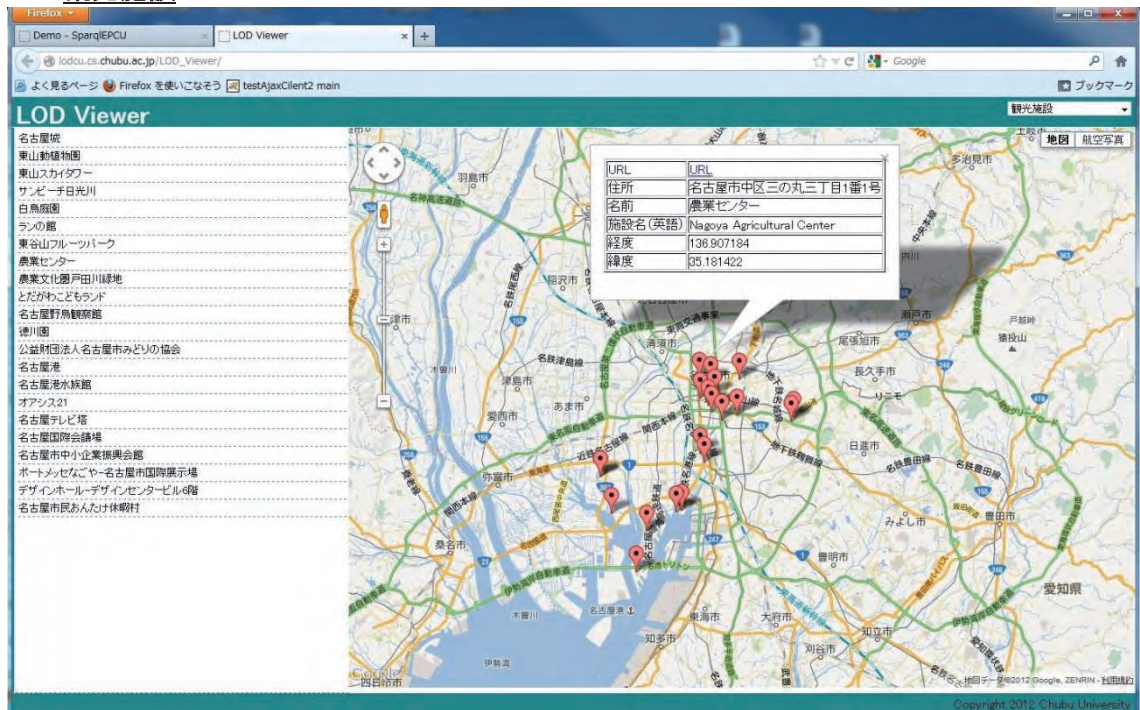


図 3.10 LOD viewer クラス観光施設

鯖江百景

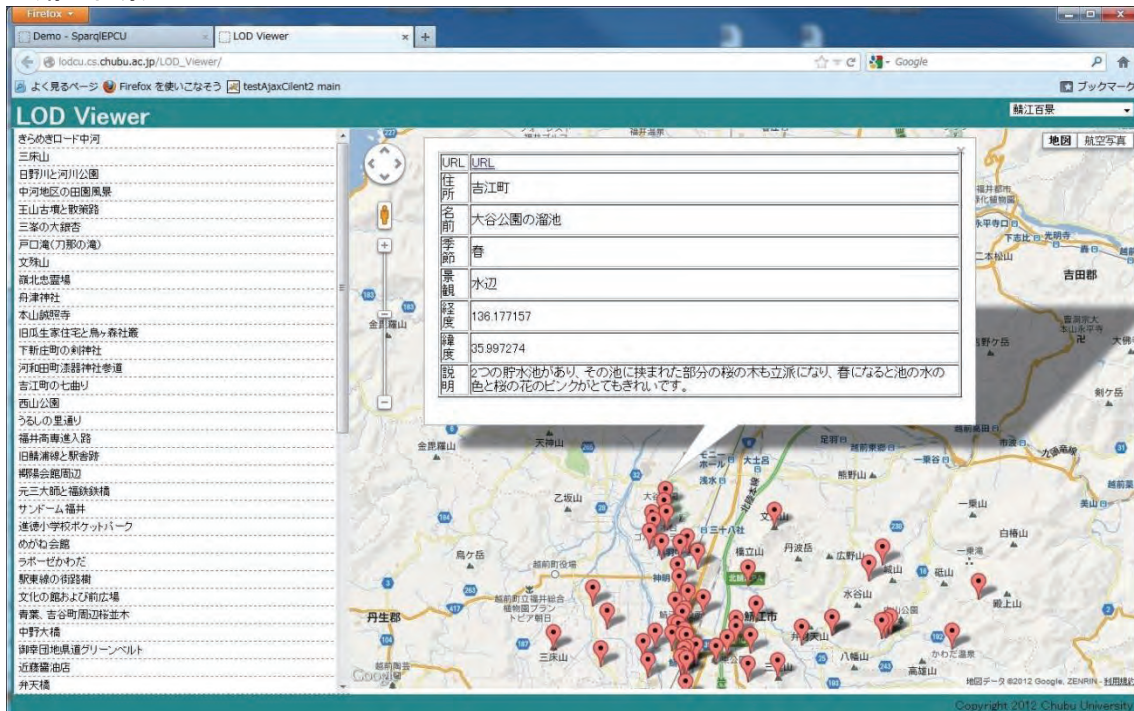


図 3.11 LOD viewer クラス:名所

Web ページ・ブックマーク

図 3. 1 3に Web ページのブックマークをカテゴリツリーとして管理するアプリケーションを示す。カテゴリや Web ページは Drag&Drop で移動出来、選択 Web ページの切り替え表示もマウス選択が非同期で処理されるので非常に使いやすい。データカタログのツールとしても有用と考える。



図 3.12 LOD アプリケーション ツリー表示のデータカタログ

4. SparqlEPCU のサイトとしての機能

フレームワーク SparqlEPCU は RDFstore や Web API のライブラリ配布を行う以外にも、LOD アプリケーションサイトとしても機能している。ここではアプリケーションサイトとしてエンドユーザーに提供している GUI サービスを紹介する。

4.1. プロジェクト管理

RDF サーバーにあるプロジェクトを各プロジェクト所有者がアクセス権の変更などの設定を変更したりする管理機能のユーザーインターフェイスを提供する。

4.2. ログイン機能

Twitter 経由か Facebook 経由かのログイン認証手続きの選択をエンドユーザーに委ねる。

4.3. RDF の登録画面

ユーザーからの RDF 登録要求を受けて、RDF を関係データベースへ格納する。テキスト入力エリアに TDF を直接書くかファイルアップロードを取るかの選択をすることが出来る。

4.4. RDF の検索画面

RDF サーバーへつながる 1 つのアプリケーションサイトでもある。エンドユーザーは標準的な SPARQL 検索を公開プロジェクトに対し行える。これによりエンドユーザーは手動により様々な SPARQL の検索式を試すことが出来る。

4.5. ライブラリ API 仕様の閲覧画面

JavaScript の API 解説を公開している。

4.6. サンプルコードのダウンロード

JavaScript ライブラリ rdfmgr を使った LOD アプリケーションをユーザーが簡単に作れるようサンプルコードをダウンロード出来るようにしている。

5. ライブラリ及びサイトの評価

本フレームワークは、2012 年 12 月より LOD2012 Japan の Web サイトを通じて広く一般に公開されている。<http://lodcu.cs.chubu.ac.jp/SparqlEPCU>

チャレンジ・デイと称して LOD Japan は年に 10 回程度のプロモーション活動を行っている。本稿の SparqlEPCU の概要が本学名古屋キャンパスに於いて紹介された。また人工知能学会研究会、LOD Japan のプロモーション活動として 9 月での LOD チャレンジ・デイ at 名古屋、1 月の横浜オープンデータハッカソンで紹介及びハンズオンで著者等により一般ユーザーに説明されている。これらの催しでの主催者及び参加者からはかなりよい評価を得ている。しかし総数が各回 30 名程度なので今後はハンズオンのやり方やオープンデータなどでの講演も検討している。

RDF を登録・検索する処理をシステム側でカプセル化し、ユーザーの手続きをパターン化することでユーザー、ここでは LOD アプリケーションサーバーを作る人、の負担をかなり軽減することが出来たと思われる。しかし RDF データの構成自身はまだ多様で、データの概念化、ビジネスモデルなどパターンの抽出でさらにシステム構築負担軽減の要素は大きいと考える。

6. まとめ

LOD アプリケーション開発支援システムとしての SparqlEPCU フレームワークについて述べた。先行する LinkData.JP や Yokohama Art、DBpedia Japan に比べ SPARQL 検索実行やより使いやすい公開 Web API で本稿フレームワークの特長があると考ええる。

今後の課題としてはエンドユーザーの声を聞きながら諸機能の改善を図って行きたいと考える。特に RDF サーバーとしてのデータハブの在り方、セキュリティ機能の拡張、推論機能の拡充、クラス、プロパティで使われる語彙の再利用や概念の集約、SPARQL 検索の並列処理など取り組むべき課題は多い。中でもクラス、プロパティのスキーマ管理を含む概念ハブの構築が重要と考えている。

参考文献、参考 URL

- [1] DBpedia. <http://dbpedia.org/> (参照: 2012/10/01)
- [2] LODAC. <http://lod.ac/> (参照: 2012/10/01)
- [3] Apache Jena. http://jena.apache.org/documentation/serving_data/ (参照: 2012/10/01)
- [4] 年岡晃一、鈴木裕利、”オントロジー構築と利用を目指す BLOG 形式グループウェア”、情報科学リサーチジャーナル、Vol. 13、2006 3
- [5] 年岡晃一、鈴木裕利、”Semantic Web 実現への課題”、情報科学リサーチジャーナル、Vol. 14、2007. 3
- [6] 年岡晃一、鈴木裕利、”構造化文書の知識表現と語彙の統計情報を併用した高度情報検索手法の構築の検討”、情報科学リサーチジャーナル、Vol. 15、2008. 3
- [7] 年岡晃一、鈴木裕利、”文書管理システム ONDOC でのオントロジー・メタデータの利用”、第 7 回情報科学技術フォーラム (FIT2008) L-026 Vol4 pp145-146、2008 年 9 月.
- [8] 年岡晃一、鈴木裕利、”XML/RDF/オントロジーを使った知識の構築及び交換のシステム”、情報科学リサーチジャーナル、Vol. 16、2009. 3
- [9] “Jena” <http://jena.sourceforge.net/>
- [10] 小出誠二、武田英明、”OWL における明示的閉世界と局所閉世界仮説”、The 22nd Annual Conference of the Japanese Society for Artificial Intelligence, 2008
- [11] <http://semweb.salzburgresearch.at/apps/rdf-gravity/index.html>
- [12] 竹内隆文、宇佐美智哉、年岡晃一:”RDF/オントロジーを利用した知識の構築及び交換システム”、平成 18 年度電気関係学会東海支部連合大会 講演論文集、2010 年 9 月



年岡晃一
中部大学工学部情報工学科教授.



鈴木裕利
中部大学工学部情報工学科
准教授



藤本 椋也
中部大学工学部情報工学科



安江翔平
中部大学工学部情報工学科