分散システムで FRBR モデルの書誌レコードを管理する: RESTful API の設計と利用

安藤 友晴
稚内北星学園大学情報メディア学部
〒097-0013 北海道稚内市若葉台 1 丁目 2290-28
Tel: 0162-32-7511, Fax: 0162-32-7500, E-Mail: tomoharu[at]wakhok.ac.jp

概要

分散システムで総合目録を開発するとき、書誌レコードの取得や登録を行うにはさまざまな手法があり、ソフトウェアが複雑になりがちである。また、総合目録上の書誌レコードと、各図書館の所蔵レコードを結びつける手法も確立していない。こうした問題を解決するため、書誌レコードと所蔵レコードを結びつけるしくみとして FRBR モデルを利用し、分散システムで FRBR モデルの書誌レコードを作成・取得・更新・削除する統一的な手法を提供する RESTful API を設計することにした。FRBR モデルの体現形と個別資料を RDF で記述したリソースと、リソース指向アーキテクチャによる RESTful API の設計をおこなった。API を用いて総合目録システムを試作したところ、従来の手法と比べても容易に総合目録を構築できることがわかった。

Managing bibliographic record based on FRBR model in a distributed system: Designing and using RESTful API

ANDOH Tomoharu
Wakkanai Hokusei Gakuen University, Faculty of Integrated Media
1-2290-28, Wakabadai, Wakkanai, Hokkaido, 097-0012, JAPAN
Tel: +81-162-32-7511, Fax: +81-162-32-7500, E-Mail: tomoharu[at]wakhok.ac.jp

Abstract

When we develop union catalog in a distributed system, there are various ways in getting and creating bibliographic record, so union catalog system tends to be complex. In addition, it has not been established how to combine bibliographic record on union catalog and holdings records which each library has. In order to solve these issues, I decided to use FRBR model to combine bibliographic record and holdings records, and design RESTful API in order to provide unified technique to create, get, update, and delete bibliographic record. I designed RDF resources which describe manifestation and item based on FRBR model and RESTful API based on Resource Oriented Architecture(ROA).

When I developed union catalog system experimentally, I found that it is easy to develop union catalog system rather than other techniques.

1 序論

複数の図書館の書誌レコードを統合する総合目録を作成しようとするとき、物理的もしくは論理的に分散されている書誌レコードを管理する必要がある。書誌レコードは、通常はそれぞれの図書館内のサーバで管理されているであろうが、今後はインターネット上の「クラウド」で管理されるケースもあることだろう。

総合目録に参加している A 図書館から、総合目録にアクセスすることを考えてみよう。A 図書館が新規購入した書籍データを登録するには、まず総合目録上で該当する書籍の書誌レコードを検索する。書誌レコードが存在すれば、そのレコードを取得して A 図書館の自館データを付加し、A 図書館の目録データベースで管理する。さらに、A 図書館の所蔵レコードを総合目録に登録する。

クラウドなどの分散システム上で書誌レコードの取得や登録を行うには、さまざまな手法が存在する。レコードの取得には HTTP をベースとした OpenSearch[3] や OAI-PMH[4]、SRW/SRU[5] といった手法が用いられる。レコードの登録には、FTP を用いる例が多い。また、国立情報学研究所の目録所在情報サービス [6] では、独自に CATP[1] というプロトコルを開発し、このプロトコルを用いてレコードの取得・登録などの処理を行っている。

このように、書誌レコードを扱うための手法が複数存在している。そしてレコードの取得には HTTP ベースの手法、レコードの登録には FTP を用いた手法と、それぞれ違うプロトコルが用いられている。これらの問題のため、書誌レコードを扱うソフトウェアは複雑な構造を持つことになり、ソフトウェアの開発は困難なものとなっている。それでは、総合目録上で書誌レコードを作成・取得・更新・削除するための統一的な手法を用意すれば良いのではないか？ そうすることによって、書誌レコードを管理するソフトウェアの作成が容易になるだろう。

次に、総合目録上の書誌レコードと、各図書館の所蔵レコードを結びつける手法も確立していない。このため、総合目録どうしで書誌レコードを統合的に扱うのに、その橋渡しをおこなうソフトウェアを必要とする。それでは、書誌レコードと所蔵レコードを結びつける手法を整理することで、総合目録間の書誌レコードの取扱いも容易になるだろう。

Roy Fielding 氏が提唱した REST[7] の考え方は、HTTP を用いてデータを作成・取得・更新・削除することを容易にしている。そのため、Amazon, Twitter, はてなブックマークなど大規模なインターネット上のサービスで REST が用いられている。また、前述の SRU でもレコードの取得に REST の考え方が採用されている。では書誌レコードの管理のために、REST の考え方を用いた RESTful API を設計・実装し、この API を通してレコードの作成・取得・更新・削除などの処理を行うことによって、書誌レコードを扱うソフトウェアの構造を単純にできるのではないだろうか？ そして、書誌レコードと所蔵レコードを結びつけるしくみとして、FRBR[9] が活用できる。既存の書誌レコードを FRBR モデルに変換するには、著作や表現形を同定するのが困難であるなどの問題がある。しかし、体現形と個別資料を結びつける作業は比較的容易なので、この作業によって書

誌レコードを統合的に扱うことが期待できる。

本研究では、分散システム上で FRBR モデルを扱える RESTful API を設計し、API を利用した書誌レコード管理システムを試作して、API の有用性を検証する。

2 RESTful API の設計

ここでは、「RESTful Web サービス [8]」にあるリソース指向アーキテクチャに則った設計技法に従い、RESTful API を設計する。

2.1 データセットの特定

先に述べたように、本研究では FRBR モデルのうち体現形と個別資料を扱うこととする。従って本 API では、体現形に対応する総合目録の書誌レコードと、個別資料に対応する各図書館の所蔵レコードを扱うこととする。

2.2 データセットをリソースに分ける

次に、データがどのように公開されるか検討した。本 API では、以下のデータを公開する。

- 総合目録の書誌レコード (体現形)
- 各図書館の所蔵レコード (個別資料)
- 総合目録の検索結果 (体現形と個別資料)
- 各図書館の検索結果 (個別資料)

2.3 リソースに URI をつける

上記のリソースにそれぞれ URI を割り振る。なお、ここでは総合目録と A 図書館の URI をそれぞれ次のように設定する。これらの URI は仮称である。

```
総合目録の URI  http://hogehoge.com/
A 図書館の URI   http://A.com/
```

総合目録の 1 つの書誌レコード (体現形) は次のようにする。なお、体現形の大部分は ISBN を持っているとみなし、ISBN を用いることもできる。

```
http://hogehoge.com/manifestation/(書誌レコード ID)
http://hogehoge.com/isbn/(書誌レコードが持つ ISBN)
```

A 図書館の所蔵レコードは次のようにする。A 図書館の URI を利用できるのはもちろん、総合目録上で対応する体現形の URI を利用できるようにする。

```
  http://A.com/item/(所蔵レコード ID)
  http://hogehoge.com/manifestation/(体現形の書誌レコード ID)/(個別資料のレコード ID)
```

総合目録の検索結果は次のようにする。

```
  http://hogehoge.com/search?q=(検索文字列)
```

A 図書館の所蔵レコードの検索結果は次のようにする。

```
  http://A.com/search?q=(検索文字列)
```

2.4 表現の設計

クライアントに提供するデータ形式を検討する。本 API では、XHTML/XML/JSON の 3 形式をサポートすることにした。それぞれの役割は次のとおりである。

XHTML　デフォルトの表現
XML　MODS を用いて書誌レコードを表現。URI の最後に".xml"を付加
JSON　JavaScript/Ajax に対応しやすい。書誌レコードを表現。URI の最後に".json"を付加

2.5 リソースの相互リンク

リソース間にリンクを張る。本 API では、体現形のレコードから個別資料のレコードへのリンクを貼る。また逆に、個別資料のレコードからは体現形のレコードへのリンクを貼るようにする。

2.6 API の一覧

図 2.6 で、本 API で利用できる操作と HTTP のメソッド、URI との対応を示す。なお、レコードの追加・更新・削除の処理には認証が必要となる。

表 1　RESTful API の一覧

操作	HTTP	対象となる URI
体現形のレコード追加	POST	http://hogehoge.com/manifestation/
		http://hogehoge.com/isbn/
体現形のレコード取得	GET	http://hogehoge.com/manifestation/(レコード ID)
		http://hogehoge.com/isbn/(レコード ID)
体現形のレコード更新	PUT	http://hogehoge.com/manifestation/(レコード ID)
		http://hogehoge.com/isbn/(レコード ID)
体現形のレコード削除	DELETE	http://hogehoge.com/manifestation/(レコード ID)
		http://hogehoge.com/isbn/(レコード ID)
個別資料のレコード追加	POST	http://A.com/item/
		http://hogehoge.com/manifestation/(体現形レコード ID)/
個別資料のレコード取得	GET	http://A.com/item/(レコード ID)
		http://hogehoge.com/manifestation/(体現形 ID)/(個別資料 ID)
個別資料のレコード更新	PUT	http://A.com/item/(レコード ID)
		http://hogehoge.com/manifestation/(体現形 ID)/(個別資料 ID)
個別資料のレコード削除	DELETE	http://A.com/item/(レコード ID)
		http://hogehoge.com/manifestation/(体現形 ID)/(個別資料 ID)
総合目録の検索結果一覧	GET	http://hogehoge.com/search?q=(検索文字列)
A 図書館の検索結果一覧	GET	http://A.com/search?q=(検索文字列)

3　RESTful API の実装

RESTful API を用いた総合目録の利用例を以下に示す。A 図書館で総合目録から書誌レコードを取得し、所蔵レコードを加えて総合目録に登録する例である。なお、実装にあたっては Ruby on Rails 2.3.3 を用いた。

1. A 図書館で、登録したい書籍の書誌データを総合目録で検索する
2. 総合目録の検索結果一覧が表示される
3. 目的の書誌データをクリックする
4. 登録したい書誌データを取得する
5. A 図書館独自の所蔵レコードを入力する
6. 所蔵データを A 図書館に登録する
7. 所蔵データを総合目録に登録する。これにより体現形と個別資料が結びつく

以上の処理は、すべて RESTful API を用いることで可能になる。なお、7 番の処理をおこなう HTTP の要求を以下に示した。ここでは、FRBR の RDF 語彙 [2] を用い、A 図書館の所蔵データ (http://A.com/books/00001 で指定) を総合目録の書誌レコード (体現形: ID は 00000001) に結びつけている。

```
POST /manifestation/00000001 HTTP/1.1
(中略)

<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:frbr="http://purl.org/vocab/frbr/core#">
    <frbr:Item rdf:about="http://A.com/books/00001">
        <frbr:exemplarOf rdf:resource="http://hogehoge.com/manifestation/00000001"/>
    </frbr:Manifestation>
</rdf:RDF>
```

次の例は、ある書籍をどこの図書館で所有しているか調べる例である。

1. 総合目録を使い、ある書籍の書誌データを検索する
2. 総合目録の検索結果一覧が表示される
3. 個別資料一覧をクリックする

体現形のレコードから個別資料のレコードへのリンクを貼ることにより、こうした処理が可能になる。この例も、すべて RESTful API を用いている。

4 まとめ

本研究では、分散システム上で FRBR モデルを扱える RESTful API を設計し、API を利用した書誌レコード管理システムを試作した。その結果、書誌レコード管理システムの構造は単純になり、アプリケーション開発の負担が軽減された。また、体現形と個別資料を結びつけることも比較的容易にできるようになった。

本研究では、FRBR の構成要素のうち体現形と個別資料のみを扱ったが、今後の研究では著作や表現形、さらには第 2 グループの個人や団体、第 3 グループの概念・物・出来事・場所についても考慮していきたい。

また、書誌レコードを RESTful API で提供することにより、他の RESTful API を用いたサービスとの連携も容易になると考えられる。こちらについても今後の課題としていきたい。

参考文献

[1] Catp/1.1.
URL: http://www.nii.ac.jp/CAT-ILL/INFO/newcat/catp1.1/hyoushi.html

アクセス日時:Sat Oct 16 18:27:51 2009.

[2] Expression of core frbr concepts in rdf.
URL: http://vocab.org/frbr/core.html
アクセス日時:Sat Oct 16 16:35:49 2009.

[3] Home - opensearch.
URL: http://www.opensearch.org/Home
アクセス日時:Sat Oct 16 16:36:57 2009.

[4] Open archives initiative - protocol for metadata harvesting - v.2.0.
URL: http://www.openarchives.org/OAI/openarchivesprotocol.html
アクセス日時:Sat Oct 17 16:44:50 2009.

[5] Sru: Search/retrieval via url – sru, cql and zeerex (standards, library of congress).
URL: http://www.loc.gov/standards/sru/
アクセス日時:Sat Oct 16 16:47:03 2009.

[6] 目録所在情報サービス.
URL: http://www.nii.ac.jp/CAT-ILL/
アクセス日時:Sat Oct 16 18:24:25 2009.

[7] Roy Thomas Fielding. Fielding dissertation: Chapter 5: Representational state transfer (rest), 2000.
URL: http://www.ics.uci.edu/˜fielding/pubs/dissertation/rest_arch_style.htm
アクセス日時:Sat Oct 16 18:32:12 2009.

[8] Leonard Richardson and Sam Ruby. RESTful Web サービス. オライリー・ジャパン, 12 2007.

[9] 和中幹雄, 古川肇, 永田治樹（編）. 書誌レコードの機能要件 - IFLA 書誌レコード機能要件研究グループ最終報告. 日本図書館協会, 3 2004.