特　集

簡易なOAI-PMHデータプロバイダの製作

A Simple Example of Implementation of OAI-PMH Data Provider

名古屋大学情報連携基盤センター
Nagoya University Information Technology Center

山本哲也
YAMAMOTO, Tetsuya

Abstract

This article shows an example of original implementation of OAI-PMH data provider. The script is written in PHP scripting language and can work on any ordinary web server, and support plain text metadata files. It can also be easily extended to support RDB and any data source.

1．はじめに

OAI-PMH（Open Archives Initiative Protocol for Metadata Harvesting）とは、メタデータの集まりをHTTP上で転送するための標準的な手続きのひとつで、Open Archives Initiativeがこれの開発と推進を行なっている[1)]。メタデータを渡す側（データプロバイダ）とそれを受け取る側（サービスプロバイダ、またはハーベスタ）の双方について通信手順が決められており、これに従うことでどのようなコンピュータ環境どうしでもメタデータのやりとりを実現できる。プロトコルの最新バージョンは2.0である。OAI-PMHの仕様はオープンなものであり、だれでもこれを利用することができる。仕様は英語で記述されているが、国立情報学研究所による日本語訳も存在する[2)]。

データプロバイダは、メタデータを一括提供することで、自らの組織では行なえなかったような高度な検索サービスなどがサービスプロバイダ上で実現されることを期待できる。複数のデータプロバイダから収集したメタデータをもとに検索サービスを構築しているサービスプロバイダの例として、国立情報学研究所のJuNii[3)]、ミシガン大学のOAIster[4)]などを挙げることができる。

大学図書館など、まとまった量の情報を扱う組織が、OAI-PMHプロトコルを使って外部サービスに自サービスのメタデータを一括提供したいという要求は、潜在的に少なくないと思われる。先にあげたJuNiiやOAIster、さらには、Googleにサイトマップ情報を提供するためのGoogle Webmaster Tools[5)]などがOAI-PMHを利用したメタデータの収集をサポートし、それらをもとに強力な検索サービスなどを提供しているからである。最近では、北海道大学が中心となって開発を進めているAIRway[6)]も注目に値する。AIRwayは、商用のリンクリゾルバの中間窓に機関リポジトリの該当アイテムへのリンクも表示させるという試みだが、この仕組みを担うサーバーが国内外の機関リポジトリからメタデータを収集するためのプロトコルにもOAI-PMHが採用された。

OAI-PMHの仕様に従ったデータ交換を実際に

行なうには、この仕様どおりの動作をするソフトウェアを作成（実装）しなければならない。この作業はプログラミングと通信技術について若干の知識を要するため、何か有用なサービスを立ち上げようとしてもこの障壁でつまづくことがあるだろう。もちろん、OAI-PMH の機能を実装した既成のソフトウェアはすでにいくつも存在するから、これらを活用することも可能である。しかし、次節以降で検討するように、筆者は、この機能を自作することにも価値があると考えている。

本稿で今回紹介するものは、データプロバイダ、すなわちメタデータをハーベスタに渡す側の簡易な実装例で、筆者オリジナルのものである。素人が作ったものなので完全な出来ばえとは言い難いが、AIRway にメタデータを渡すためにすでに稼動しているという実績があり、充分利用に堪えるものといえるだろう。

２．類似ソフトウェアの検討

OAI-PMH のデータプロバイダ側、ハーベスタ側とも、商用・非商用を含めて、すでに数多くの実装が存在する。データプロバイダ側としては、機関リポジトリを構築するためのソフトウェアとして知られている DSpace [7]、GNU EPrints [8]、E-repository [9]、XooNIps[10] といったものが、それぞれ独自の実装を行なっている。これらの専用ソフトを使ってデータベースを構築する場合は、それぞれのマニュアルに従ってインストール作業を行なえばそれだけで OAI-PMH を使ったメタデータ公開が可能になるという利点がある。その反面、純粋にデータプロバイダだけを構築したいような場合は、これ以外の機能、たとえば検索やブラウジングなどの機能は無駄なものとなってしまう。さらに、手元にあるデータの集まりをこれらのソフトウェア上に搭載するために、データ変換のためのツールを作るかカスタマイズするかといった何らかの手間を生じることはありうることで、その場合、技術上の障壁は必ずしも低くない。専用ソフトがサーバーマシンに高いスペックを要求する場合には、その全ての機能を必要とするわけでないのに、新しい機械を購入する必要が生じるかもしれない。オープンソースとして公開されているソフトウェアを直接利用する場合は、基本的には自助努力による保守やカスタマイズができることが導入の前提になる。総じて、データプロバイダの機能だけを実現するために大がかりな機関リポジトリ用ソフトウェアを導入することは、却ってコストを大きくしてしまうかもしれない。

OAI-PMH データプロバイダを構築するための手段として次に考えられるのは、公開されているソフトウェア部品を利用することである。OAI-PMH の機能を実現する部分だけが再利用可能な部品として公開されているものはいくつもあり[11]、例えば OCLC の OAICat[12]、バージニア工科大学 Digital Library Research Laboratory の OAI-PMH2 XMLFile File-based Data Provider[13]（以降、OAI-XMLFile と呼ぶ）などがある。任意のソフトウェアにこれらのソフトウェア部品を追加し、OAI-PMH の機能を組み入れることができるように作られているのがこれらの特徴である。先にあげた機関リポジトリのためのソフトウェアもこういったソフトウェア部品を利用していることがあり、たとえば DSpace では OAICat を内部的に組み込んで使っている。

OAICat は、Java サーブレットとして動作することを想定して書かれている。サービスを構築しようとするユーザー（開発者）は、まず Java サーブレットコンテナと呼ばれるソフトウェアをコンピュータ上に導入する。次に OAICat が提供しているいくつかの抽象クラス（AbstractCatalog、RecordFactory、各種 Crosswalk）を継承した派生クラスを開発し、これを組み込んでサーブレットコンテナを起動する。OAICat はデータの格納形式などに依存せず、柔軟な実装ができて完成度の高いものであるが、開発言語が Java に限定されており、クラスの継承やサーブレットといった概念にも充分親しんでいないと、これを利用した開発はやや難しい。

OAI-XMLFile は、あらかじめメタデータを XML ファイルとして格納しておいたものを必要に応じてサーバープロセスが読み取り、ハーベスタ側に情報を渡すという原理で動作するものである。Perl で書かれた cgi スクリプトとして提供されており、既存の環境に比較的簡単に導入ができる。一方、ファイルシステム上にプレーンテキストとして保存されたたくさんの XML ファイルを扱うため、多量のデータを扱う際にパフォーマンスを保ちにくくなることが予想される。また、

複数のデータフォーマットに対応させるためにXSLT（XMLの変換ルール）を準備する必要があり、導入のうえでやや複雑さを感じる要因になるように思われる。

他にもソフトウェア部品として公開されているものはあったが、まだ完成に至っていないものや、長期間メンテナンスがされていないものもあり、これらすべての評価は行なわなかった。

ここまで、いくつかの既存のソフトウェア、またはソフトウェア部品の検討を行い、それぞれ一長一短があることを見てきた。できるだけ簡易な方法でデータプロバイダを構築したいと考えていたのだが、この要求を満たすものを結局見つけることができなかったと感じた。そこで、OAI-PMHの元の仕様を頼りに、自分達が求めるものを新たに製作してみることにした。

３．作成方針

製作にあたっては、以下のような方針をとった。

まずは、OAI-PMHのデータプロバイダとして最小限必要な機能だけを実装することにした。たとえばWindows上で動作するようなグラフィカルな設定画面といったようなものは特に用意せず、ファイルなどの形で直接与えられたメタデータを使ってハーベスタとの通信だけを実現できればよいものとした。

次に、広く使われている環境にあわせて使用言語や開発環境を決めた。webサーバー上で動くスクリプト言語としてはPHPの人気が非常に高く、またインストールされている見込みも大きいため、これを採用した。開発時のPHPのバージョンは４とした。PHPの最新バージョンは５だが、バージョン４が安定稼動しているサーバーもまだ多く存在する。ここではバージョン５に固有の機能を使うことで適用範囲が狭くなってしまうことを避けた。もちろんバージョン４と５で問題なく動作することを確認した。UNIX系の環境やWindows上でも特に動作に違いはないようだった。また、依存するライブラリが極力少なくなるよう意識した。

次に、世に出ている他のソフトウェア部品同様に、再利用が容易になることを目指した。具体的には、OAI-PMHの基本機能を実現する部分をソフトウェアの雛形としてつくり、実際のデータを扱う部分だけをこれへのコード追加というスタイルで実装できるようにした。雛形部分のコード変更は基本的に不要である。以下、この雛形をフレームワークと呼ぶ。

４．フレームワークの使用方法

4.1　概略

今回製作したPHPスクリプト（フレームワーク部）は、付録資料１に掲載した。これの利用イメージを図１に示す。

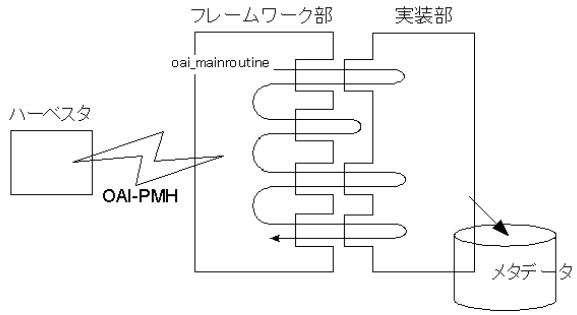

図１．フレームワークの利用イメージ

これを利用するためには、まずこのコード（フレームワーク部）を適当なファイル名で格納し、別のスクリプト（こちらがBaseURLに対応する。以降実装部と呼ぶ）からインクルードするといった手順を取る。フレームワーク部のファイル名は何でも構わないが、ここではtinyoai.inc.phpとする。拡張子はPHPの慣行に従った。

次に、フレームワークが要求する名前でいくつかの定数をセットし、サブルーチン（関数）を実装する。フレームワーク側は、決まった名前の関数が存在するものと仮定して処理を進めるので、注意する必要がある。関数ひとつひとつの説明は次節で詳しく行なう。いくつかは必須であるが、実現したいデータプロバイダの特長によっては省略してよいものもある。

これらすべてを実装部のスクリプトとして記述したら、このスクリプトの最後に、フレームワーク側で定義しているメインルーチンoai_mainroutineを呼び出すコードを書いて終了である。oai_mainroutineは、ハーベスタの要求に従っ

て、先に定義したサブルーチンなどを適切な順序で実行し、その結果得られた情報を XML 形式でハーベスタ側に返答するという役割を持っている。

4. 2　各種定数・関数

フレームワークを利用するために記述しなくてはならない、各種定数・関数について説明する。

・function fw_initialize() / fw_finalize()

一回のリクエストごとに何らかの初期化処理と終了処理が必要な場合、この名前をもつ関数を定義して、中にその処理を書く。扱うデータがプレーンテキストファイルで記述されている場合、そのファイルのオープン処理とクローズ処理を書くことになるかも知れない。また、MySQL などのようなリレーショナルデータベースである場合は、データベースへの接続や切断をここに書くことになるかも知れない。関数の返り値は必要ない。

・function fw_identify()

OAI-PMH プロトコルのうち、Identify 要求を受けたときの動作をここに定義する。ただし、フレームワーク側で大部分の処理を行なうため、ここに記述するべきことはあまり多くない。具体的には、OAI-PMH の仕様にある <description> 要素のみを、XML の一部になるような文字列として呼び出し元に返せばよい。<description> 要素が必要でない場合は、この関数の記述そのものを省略してよい。

<description> 要素以外にも、Identify 要求に対する返答には、データプロバイダ固有の値をセットしておくべき部分がある。これらの値は、ここの fw_identify 関数内でなく、あらかじめ決められた定数をセットしておくことで操作できる。Identify 要求に関係する定数は下の通り。

REPOSITORY_NAME … このデータプロバイダの名前。UTF-8 でエンコードされていれば、日本語などでの記述も可能。PHP スクリプト自体を UTF-8 で書いていれば、特にエンコードを意識する必要はない。これを明示的にセットしないときの省略値は、‘My Repository’。

BASEURL … このデータプロバイダの BaseURL を指定する。省略された場合は、環境変数などから自動的に推定される。大抵の場合にこの推定は正しいが、そうでないときは明示的にセットしておく。

ADMIN_EMAIL … 管理者のメールアドレス。必ず受け取りが可能なメールアドレスでなければならない。省略値は、‘anonymous@localhost’ であるが、この値のままで運用してはならない。

DELETED_RECORD … 削除レコードの扱い方針を記す。仕様上、persistent, no, transient のうちどれかを指定する。省略値は transient。この値はハーベスタ側に削除レコードの扱い方針（deletedRecord 値）を伝えるためだけのもので、データプロバイダの動作がこれによって変化するわけではない。

EARLIEST_DATE … 最も古いレコードのタイムスタンプか、それより充分古い値を指定する。省略値は、1900-01-01。

ところで、granularity（タイムスタンプの粒度）については、このフレームワークにおいては YYYY-MM-DD 固定としている。これより細かく時分秒までをタイムスタンプで表現する必要があるときはフレームワーク自体を若干カスタマイズしなくてはいけないが、ここではその必要はないものと仮定している。

・function fw_listsets()

OAI-PMH プロトコルのうち、ListSets 要求を受けたときの動作を定義する。この関数は、<set> 要素を必要なだけ列挙してできあがる単一の文字列を呼び出し元に返す。<set> 要素はさらにその中に <setSpec> 要素や <setName> 要素を入れ子に持つが、詳しい説明は OAI-PMH の仕様や、実際のレスポンスの例を参照のこと。

実装しようとするデータプロバイダがセットをサポートする必要がない場合、この関数の記述を完全に省略してもよい。その場合、ListSets 要求はエラーメッセージ（noSetsHierarchy）を発生させることになるが、仕様上これは許されている。

・function fw_listmetadataformats($identifier)

OAI-PMH プロトコルで、ListMetadataFormats 要求を受けたときの動作を定義する。この関数は、fw_listsets と同様に、XML の一部になるような単一の文字列を呼び出し元に返す。文字列は、<metadataFormat> 要素を必要なだけつなげたものになる。<metadataFormat> 要素の詳しい説明は、OAI-PMH の仕様などを参照のこと。

引数として $identifier が指定されている場合は、この ID を持つ特定のアイテムがどのフォーマットに対応しているかという問い合わせであることを意味する。特定のアイテムでなく、データプロバイダ全体としてサポートしているすべてのフォーマットを要求されているときは、$identifier 引数には NULL 値が指定される。アイテムごとに対応フォーマットを細かく指定したい場合は、この引数によって異なる動作をさせることが可能である。逆に、$identifier に関わらずフォーマット一覧が決まるときは、この引数を無視して処理を書けばよい。

・function fw_getrecord($identifier, $format)

OAI-PMH の GetRecord 要求に対応した動作をここに記述する。この関数はふたつの引数、すなわちアイテムの ID（$identifier）とフォーマット指定（$format）を渡すので、この値をもとに適切な値を返すようなコードを書くこと。ここで返すべき値は、これまでのような単一の文字列でなく、PHP の配列である。PHP の配列は他のプログラム言語でいう連想配列などに相当し、キーと値のペアが集まったものである。ここで返す配列が含むべきキー / 値は下の５つ。

id … アイテムの ID

datestamp … アイテムのタイムスタンプ。YYYY-MM-DD 形式の文字列で記述してあること（例：2007-01-11）。Identify 要求への返答にに含まれる granularity 値は、この表示形式を指定したものである。

deleted … アイテムが削除済みであることを明示的に表したいときに、このキーを設定する。値はなんでもよい。削除していないときはこのキーは必要ない。削除したレコードをわざわざこのように削除済レコードとして返答するか、それとも存在しないレコードとして返答から省いてしまうかは、Identify 要求への返答に含まれる deletedRecord 値がその方針を示すことになっている。deletedRecord が‘persistent’である場合は、かつて存在したことのある ID のレコードは必ず deleted の返答を返し、‘no’である場合は一切 deleted 情報を返さないという仕様になっている。‘transient’は、どちらとも保証できないことを意味する。

set … アイテムが所属しているセット（データベース内の部分集合）をここにセットする。ひとつのアイテムが同時に複数のセットに所属する場合は、この値部分にさらに配列をセットすること。所属セットがひとつの場合も、要素が一つの配列を設定すること。データプロバイダがセットをサポートしないときは、このキーを設定する必要はない。

metadata … アイテムのメタデータ本体を、<metadata> タグで始まって </metadata> タグで終わる文字列として生成し、このキーの値として格納する。ここは長い文字列になるかもしれない。deleted キーを設定したときは、ここは不要。

PHP では、配列を呼び出し元に返すときに参照返しをすることができるので、必要なら関数の定義の先頭に & 記号をつけることでこれを実現してもよい。

$identifier 引数に対応するアイテムがない場合は、関数の返り値には、配列の代わりに OAI_ERROR_NO_HIT 定数を返すこと。また、$format 引数に対応するフォーマットでの表示がサポートできないときは、OAI_ERROR_NO_FORMAT 定数を返すこと。この定数はフレームワーク部で定義されている。

・fw_listrecords($format, $from, $until, $set, $offset, $maxrecs)

OAI-PMH の ListRecords 要求に対応した動作をここに記述する。GetRecord が単数のアイテムを要求するのに対し、こちらは抽出条件に合致する

複数のレコードを要求しているという違いがある。抽出条件は引数が示すとおりで、$format で指定された表示フォーマットが適用可能なもののうち、タイムスタンプが $from から $until までの範囲のもの。$from か $until のどちらか、または両方が NULL 値であることもあるが、そのときは日付範囲の上限や下限がないか、または範囲指定はないことを意味する。$set 引数に NULL 以外の値が指定された場合は、このセットに含まれるアイテムのみが抽出対象となる。

たくさんのアイテムを扱う場合は、抽出条件に合致するアイテムが大量になる場合があるが、$offset 引数と $maxrecs 引数の指定に従って、抽出結果の一部だけを返答する。すなわち、抽出結果の先頭から $offset 番目（１から始まる）からはじまって、最大で $maxrecs 個を超えない数のアイテムを呼び出し元に返す。$maxrecs の数値は、LIST_MAX 定数に１を足したものが入ってくる。LIST_MAX の初期値は 50 だが、あらかじめ再定義しておいて、一度のリクエストで転送される情報の量を調整してよい。

関数の返り値は、先の fw_getrecord が返すような配列を、さらにアイテムの数だけ多重配列にしたものである。

抽出条件に合致するアイテムがひとつもない場合は、関数の返り値は OAI_ERROR_NO_HIT 定数とすること。

ところで、OAI-PMH の ListIdentifiers 要求を扱うための関数は明示的に書く必要がない。フレームワーク側からはこのときも fw_listrecords 関数を呼び出し、この返り値の一部を使って返答を生成するためである。

次節では、実際にこれら関数（の一部）を実装して、データプロバイダを構築した例を示す。サンプルの目的で掲載するもので、実際に稼動しているものとは少し異なるものである。

５．実装例

5. 1　データ記述の仕様

ここで行なってみる OAI-PMH データプロバイダの実装では、例として下のような仕様のデータを扱う。

・データの格納形式は、UTF-8エンコードで書かれた単純なテキストファイル。
・[　]で囲まれた行はアイテムのIDを表し、次の行からメタデータが始まる目印となる。
・メタデータ部分は、項目名と値がタブで区切られたもので、１行ずつからなる。
・空行が出現したら、ひとつのアイテムの記述が終了したことを示す。
・大きなテキストファイルになってくると処理効率が落ちるため、アイテムのIDとテキストファイル中でそのアイテムがはじまるシーク位置を別ファイルに分けて保存し、この情報も適宜使うことにする。ここではこれをカタログファイルと呼ぶことにする。
・なるべく単純にするため、ここではセットをサポートしない。

下は、この仕様に従って書いたデータの例である。項目名と値は、空白文字でなくタブ記号で区切られている。

```
[oai:mydomain.ac.jp:0001]
datestamp    2006-12-02
creator      YAMAMOTO, Tetsuya
creator      DARENO, Nanigashi
title        OAI-PMH Data Provider
publisher    some publisher
URI  http://www.mydomain.ac.jp/article/0001
description  This is an abstract of this article.

[oai:mydomain.ac.jp:0002]
datestamp    2006-12-07
creator      YAMAMOTO, Tetsuya
title        OAI-PMH Data Provider 2
publisher    some publisher
URI  http://www.mydomain.ac.jp/article/0002
description  This is an abstract of this article.

[oai:mydomain.ac.jp:0003]
datestamp    2006-12-11
...
```

5. 2　作成したコードの解説

スクリプトを２本掲載する。ひとつはテキストファイルからカタログファイルを作るための Perl スクリプト makecatalog.pl（付録資料２）、もうひとつがフレームワークが要求する機能を実装する

PHP スクリプト oai.php（付録資料３）である。

カタログファイルには、それぞれのアイテムについて、[ID，タイムスタンプ，シーク位置，メタデータの行数] を持たせる。カタログファイルは小さく保たれ、このスクリプトが動作するたびにすべてメモリ上にロードされることにした。テキストファイルに変更を加えたら、各アイテムのシーク位置を正しく保つために、必ず手動で Perl スクリプトを実行し、カタログファイルを作り直す必要がある。コマンドラインで、

```
perl makecatalog.pl data.txt data.txt.catalog
```

などと打ち込んで実行する（元のデータファイルを data.txt、カタログファイルを data.txt.catalog としたときの例。）

カタログファイルは、下のような見た目のものになるだろう。

```
oai:mydomain.ac.jp:0001,2006-12-02,27,7
oai:mydomain.ac.jp:0002,2006-12-07,297,6
oai:mydomain.ac.jp:0003,2006-12-17,548,6
oai:mydomain.ac.jp:0004,2006-12-18,799,6
oai:mydomain.ac.jp:0005,2007-01-04,1050,6
...
```

実装部のスクリプト oai.php について少し解説する。REPOSITORY_NAME 定数と ADMIN_EMAIL 定数を適切に設定した。カタログファイルとデータファイルの場所も配置にあわせて設定した。データ出力フォーマットは、OAI-PMH が標準でサポートする必要がある oai_dc と、もうひとつは例として testformat という架空のフォーマットをサポートした。それぞれのフォーマットに対応する関数はそれぞれ build_record_oai_dc と build_record_testformat である。ここで、生のメタデータをそれぞれの要素に対応付けて出力フォーマットを組み立てている。著者名については、生データ上で creator というフィールドになっているところを、oai_dc 形式では <dc:creator> タグで、testformat 形式では <author> タグで表現したりという差を設けている。

できたスクリプトは web サーバーからアクセスできる場所に置く。フレームワーク部のスクリプト、実装部のスクリプト、テキストファイル、カタログファイル、カタログ生成スクリプトの五つをすべて同じディレクトリに置いて、適当な web ブラウザから動作テストを行なう。（必要のない部分に外部からアクセスできないよう、実装部のスクリプト以外は web で公開しない場所に置くほうがよい。ここではテストのために簡易な方法を取っている。）

一通りの動作が仕様に従った完全なものであるか、妥当性をチェックするサービス[14] を使って確認することができる。チェックの結果は ADMIN_EMAIL で設定したメールアドレスに返ってくるので、これが有効なものであるか注意すること。

5. 3　他の実装の可能性

今回の例では単純なテキストファイル上のメタデータを扱うものを製作したが、これ以外の方法で格納されたようなものも、同じフレームワークを使って作ることができる。実装部のスクリプトでテキストファイルを読み込んでいるような部分を、データベースに SQL を発行するなどして結果を取得するようなコードに書き換えれば、本格的な RDB 上でデータを管理するようなものに拡張していくことも難しくはないだろう。PHP は RDB との連携にも優れたスクリプト言語である。

今回行なわなかったセットのサポートも、これを扱うためのコードを必要に応じて実装部に書き足していくことで、出来るようになるだろう。もちろん､小規模なデータを扱えればよい場合には、セットをしばらくサポートしないという選択も引き続き可能である。

6．おわりに

ここまで、OAI-PMH のデータプロバイダ機能を自作するという題に従って稿を進めてきた。完成したスクリプトはとても短いものなので、Perl や PHP の知識が少しあれば完全に理解することができるだろう。筆者は、このスクリプトを書いてみて、OAI-PMH は思ったよりも易しく実装ができるように設計されたプロトコルだったのだという実感を持つことができた。この例をヒントにして、埋もれがちになってしまった情報資源を有効に再発信してみようという試みがよりたくさん現れてくれれば幸いである。

なお、これらの開発にはPHPなどを始めとしてたくさんの無償ソフトウェアを活用させていただいた。最後にそれぞれの作者に感謝の意を示したい。

注記：

ここに掲載したスクリプトはすべて著者オリジナルのもので、自由に改変･再利用していただいて構いません。また、これを利用したことで何らかの損害が発生しても、責任を負うことはできませんので、あらかじめご理解の上ご利用ください。

スクリプト等は、
http://info.nul.nagoya-u.ac.jp/pubwiki/index.php?tinyoai
からもダウンロードできます。

参考文献・URL

1）OAI-PMH
http://www.openarchives.org/OAI/openarchivesprotocol.html
2）OAI-PMH 邦訳
http://www.nii.ac.jp/metadata/oai-pmh2.0/OpenArchivesProtocol.htm
3）JuNii
http://ju.nii.ac.jp/
4）OAIster
http://oaister.umdl.umich.edu/o/oaister/
5）Google Webmaster Tools / Google ウェブマスター ツール
http://www.google.com/webmasters/sitemaps
6）AIRway
http://airway.lib.hokudai.ac.jp/
7）DSpace
http://www.dspace.org/
8）GNU EPrints
http://www.eprints.org/software/
9）E-repository
http://www.cmsc.co.jp/E-repository/
10）XooNIps
http://xoonips.sourceforge.jp/
11）OAI-PMH tools
http://www.openarchives.org/tools/tools.html
12）OAICat
http://www.oclc.org/research/software/oai/cat.htm
13）OAI-PMH2 XMLFile File-based Data Provider
http://www.dlib.vt.edu/projects/OAI/software/xmlfile/xmlfile.html
14）Registering as a Data Provider OAI-PMH version 2.0
http://www.openarchives.org/data/registerasprovider.html
このURLはOAI-PMHプロバイダの「登録」のためのフォームであるが、Validate onlyを選ぶことで、登録なしに機能チェックだけを行なうサービスとしても機能する。

［付録資料1　tinyoai.inc.php］

```
<?php

/*
 * tiny OAI-PMH data provider framework for php4(or higher)
 * Yamamoto.T
 *
 * 2006.9.7  ver. 0.85
 * 2006.9.15 ver. 0.86 bugfix
 * 2006.9.28 ver. 0.87 add comment
 *           ver. 0.88 add initialize/finalize
 * 2007.2.15 ver. 0.89 getrecords -> listrecords
 *
 * this script cannot run by itself. write your script that 'include' this.
 *   example:
 *     <?php
 *       include 'tinyoai.php.inc';
 *         ... define your functions ...
 *       oai_mainroutine();
 *     ?>
 *
 * in your script, you must implement your original functions below:
 *   * fw_identify (optional)
 *   * fw_listsets (optional)
```

```
 *    * fw_listmetadataformats
 *    * fw_getrecord
 *    * fw_listrecords
 *    * fw_initialize / fw_finalize (if you need)
 *
 *  read README for details.
 */

define ('OAI_ERROR_NO_SET', 1);
define ('OAI_ERROR_NO_FORMAT', 2);
define ('OAI_ERROR_NO_HIT', 3);

// set default constants if not defined
function oai_check_constants() {

  $default = array(
    'REPOSITORY_NAME' => 'My Repository',
    'ADMIN_EMAIL' => 'anonymous@localhost',
    'DELETED_RECORD' => 'transient',
    'EARLIEST_DATE' => '1900-01-01',
    'LIST_MAX' =>50,
  );
  $default['BASEURL'] = "http://" . $_SERVER['SERVER_NAME'] . $_SERVER['SCRIPT_NAME'];

  foreach($default as $key => $value) {
    if (!defined($key)) { define ($key, $value); }
  }
}

// build XML segment (header element)
function oai_array2recordheader(&$rec) {

  $buf = ($rec['deleted'] ? '<header status="deleted">' : '<header>');
  $buf .= "<identifier>" . $rec['id'] . "</identifier>";
  $buf .= "<datestamp>" . $rec['datestamp'] . "</datestamp>";
  if (is_array($rec['set'])) {
    foreach ($rec['set'] as $i) { $buf .= "<setSpec>".$i."</setSpec>"; }
  }
  $buf .= "</header>";
  return $buf;
}

// build XML segment (record element)
function oai_array2fullrecord(&$rec) {

  $buf = "<record>";
  $buf .= oai_array2recordheader(&$rec);
  if (!$rec['deleted']) {
    $buf .= "<metadata>" . $rec['metadata'] . "</metadata>";
  }
  $buf .= "</record>";
```

```
  return $buf;
}

// validate date string (granurality YYYY-MM-DD only)
function oai_validate_date($value) {
  if (! $value) {
    return TRUE;
  }
  if (strlen($value) != 10) {
    return FALSE;
  }
  $a = sscanf($value, "%04d-%02d-%02d");
  if (! checkdate($a[1], $a[2], $a[0])) {
    return FALSE;
  }
  return TRUE;
}

function oai_do_identify(&$r) {

  if ($r['identifier'] or $r['from'] or $r['until'] or $r['set'] or $r['resumptionToken']) {
    return array('badArgument', 'illegal argument for Identify');
  }

  $r['oai_body'] =
      "<repositoryName>".REPOSITORY_NAME."</repositoryName>" .
      "<baseURL>".BASEURL."</baseURL>" .
      "<protocolVersion>2.0</protocolVersion>" .
      "<earliestDatestamp>".EARLIEST_DATE."</earliestDatestamp>" .
      "<deletedRecord>".DELETED_RECORD."</deletedRecord>" .
      "<granularity>YYYY-MM-DD</granularity>" .
      "<adminEmail>".ADMIN_EMAIL."</adminEmail>";

  if (function_exists("fw_identify")) {
    $r['oai_body'] .= fw_identify();
  }

  return NULL;
}

function oai_do_listmetadataformats(&$r) {

  if ($r['from'] or $r['until'] or $r['set'] or $r['resumptionToken']) {
    return array('badArgument', 'illegal argument for ListMetadataFormats');
  }

  if (!function_exists("fw_listmetadataformats")) {
    return array('noMetadataFormats', 'ListMetadataFormats not implemented');
  }

  $r['oai_body'] = fw_listmetadataformats($r['identifier']);
```

```
  if ($r['oai_body'] == OAI_ERROR_NO_FORMAT) {
    return array('noMetadataFormats', 'no format available');
  } elseif ($r['oai_body'] == OAI_ERROR_NO_HIT) {
    return array('idDoesNotExist', 'no such identifier available');
  }

  return NULL;
}

function oai_do_listsets(&$r) {

  if ($r['identifier'] or $r['from'] or $r['until'] or $r['set'] or $r['resumptionToken']) {
    return array('badArgument', 'illegal argument for ListSets');
  }

  if (!function_exists("fw_listsets")) {
    return array('noSetHierarchy', 'ListSets not implemented');
  }

  $r['oai_body'] = fw_listsets();

  if ($r['oai_body'] == OAI_ERROR_NO_SET) {
    return array('noSetHierarchy', 'no set');
  }

  return NULL;
}

function oai_do_getrecord(&$r) {

  if (!$r['identifier'] or !$r['metadataPrefix']) {
    return array('badArgument', 'some argument missing');
  }

  if ($r['from'] or $r['until'] or $r['set'] or $r['resumptionToken']) {
    return array('badArgument', 'illegal argument for GetRecord');
  }

  if (strstr($r['identifier'], "¥"")) {
    return array('badArgument', 'malformed identifier');
  }

  if (!function_exists("fw_getrecord")) {
    return array('idDoesNotExist ', 'GetRecord not implemented');
  }

  $rec = fw_getrecord($r['identifier'], $r['metadataPrefix']);

  if ($rec == OAI_ERROR_NO_FORMAT) {
    return array('noMetadataFormats', 'no format available');
```

```
 } elseif ($rec == OAI_ERROR_NO_HIT) {
   return array('idDoesNotExist', 'no such identifier available');
 }

 $r['oai_body'] = oai_array2fullrecord(&$rec);
 return NULL;
}

function oai_do_listrecords(&$r, $mode) {

 if ($r['identifier']) {
   return array('badArgument', 'illegal argument for ListRecords(ListIdentifiers)');
 }

 if ($r['resumptionToken']) {
   if ($r['metadataPrefix'] or $r['from'] or $r['until'] or $r['set']) {
     return array('badArgument', 'illegal argument (query redundant)');
   } else {
     list($r['metadataPrefix'], $r['offset'], $r['from'], $r['to'], $r['set']) = explode("!", $r['resumptionToken']);
     if (! $r['offset']) {
       return array('badResumptionToken', 'malformed resumptionToken');
     }
     $r['resumptionToken'] = NULL;
   }
 } else {
   if (!$r['metadataPrefix']) {
     return array('badArgument', 'metadataPrefix missing');
   }
   $r['offset'] = 1;
 }

 if (! oai_validate_date($r['from'])) {
    return array('badArgument', 'malformed date');
 }

 if (! oai_validate_date($r['until'])) {
    return array('badArgument', 'malformed date');
 }

 if ($r['until']) {
   if ($r['until'] < EARLIEST_DATE) {
     return array('noRecordsMatch', 'too old record requested');
   }
 }

 if (!function_exists("fw_listrecords")) {
   return array('noRecordsMatch', 'ListRecords(ListIdentifiers) not implemented');
 }

 $recs = fw_listrecords($r['metadataPrefix'], $r['from'], $r['until'], $r['set'], $r['offset'], LIST_MAX+1);
```

```
  if ($recs == OAI_ERROR_NO_SET) {
    return array('noSetHierarchy', 'no set');
  } elseif ($recs == OAI_ERROR_NO_HIT) {
    return array('noRecordsMatch', 'no records hit');
  } elseif ($recs == OAI_ERROR_NO_FORMAT) {
    return array('noMetadataFormats', 'no such format available');
  }

  if (count($recs) == 0) {
    return array('noRecordsMatch', 'no records hit');
  }

  $max = (count($recs) > LIST_MAX ? LIST_MAX : count($recs));

  for ($i = 0; $i < $max; $i++) {
    if ($mode == 'i') {
      $r['oai_body'] .= oai_array2recordheader(&$recs[$i]);
    } else { // mode == 'r'
      $r['oai_body'] .= oai_array2fullrecord(&$recs[$i]);
    }
  }

  if (count($recs) > LIST_MAX) {
    $r['oai_body'] .= "<resumptionToken>"
        . rtrim(join("!", array($r['metadataPrefix'], $r['offset'] + LIST_MAX, $r['from'], $r['to'], $r['set'])),"!")
        . "</resumptionToken>";
  } elseif ($r['offset'] > 1) {
    $r['oai_body'] .= "<resumptionToken/>";
  }

  return NULL;
}

// MAIN ROUTINE
function oai_mainroutine() {

  if (function_exists("fw_initialize")) {
    fw_initialize();
  }

  oai_check_constants();

  $r = array();

  foreach (array('verb', 'set', 'identifier', 'from', 'until', 'metadataPrefix', 'resumptionToken') as $i) {
    if ($_POST[$i]) { $_GET[$i] = $_POST[$i]; }
    $r[$i] = $_GET[$i];
  }

  $r['oai_body'] = '';
```

```
  switch($r['verb']) {
  case 'Identify':
    $err = oai_do_identify(&$r);
    break;
  case 'ListMetadataFormats':
    $err = oai_do_listmetadataformats(&$r);
    break;
  case 'ListSets':
    $err = oai_do_listsets(&$r);
    break;
  case 'ListIdentifiers':
    $err = oai_do_listrecords(&$r, "i");
    break;
  case 'GetRecord':
    $err = oai_do_getrecord(&$r);
    break;
  case 'ListRecords':
    $err = oai_do_listrecords(&$r, "r");
    break;
  default:
    $r['verb'] = NULL;
    $err = array('badVerb', 'illegal OAI verb');
  }

  if (function_exists("fw_finalize")) {
    fw_finalize();
  }

  header("Content-type: text/xml");

  echo <<<END_HEREDOC
<?xml version="1.0" encoding="utf-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
         http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
END_HEREDOC;

  echo "<responseDate>" . gmdate("Y-m-d¥TH:i:s¥Z", time()) . "</responseDate>";

  if ($r['verb']) {
    echo "<request verb=¥"" . $r['verb'] . "¥"";
    if (! $err) {
      foreach (array('set', 'identifier', 'from', 'until', 'metadataPrefix', 'resumptionToken') as $i) {
        if ($_GET[$i]) { echo " $i=¥"" . htmlspecialchars($_GET[$i]) . "¥""; }
      }
    }
    echo ">".BASEURL."</request>";
  } else {
    echo "<request>".BASEURL."</request>";
  }
```

```
 if ($err) {
  echo "<error code=¥"" . $err[0] . "¥">" . $err[1] . "</error>";
 } else {
  echo "<" . $r['verb'] . ">" . $r['oai_body'] . "</" . $r['verb'] . ">";
 }

 echo "</OAI-PMH>";
}

?>
```

［付録資料２　makecatalog.pl］

```
#!/usr/bin/perl

#
# usage: makecatalog.pl [infile] [outfile]
#

if (@ARGV != 2) {
 print "usage: makecatalog.pl [infile] [outfile]¥n";
 exit;
}

open TXT, $ARGV[0] or die "no $ARGV[0]";
open OTXT, ">$ARGV[1]" or die "can't write $ARGV[1]";

$currentpos = 0;
$bkey = '';
$datestamp = '';
$lines = 0;

while(<TXT>){
 chomp;
 if ($_ =~ /^¥[(.*?)¥]/) {
  if ($bkey) {
   print OTXT "$bkey,$datestamp,$currentpos,$lines¥n";
   $lines = 0;
  }
  $datestamp = '';
  $bkey = $1;
  $currentpos = tell TXT;
  next;
 }

 next unless $_;

 $lines++;
 @a = split(/¥t/, $_, 2);
 if ($a[0] eq 'datestamp') { $datestamp = $a[1]; }
}
```

```
print OTXT "$bkey,$datestamp,$currentpos,$lines¥n" if $bkey;
close OTXT;
close TXT;
```

［付録資料３　oai.php］

```
<?php

// must be stored in UTF8 encoding

// tinyoai フレームワークを使ったデータプロバイダ実装の例。
//
//  セット構造の実装はなし。
//  カタログファイルとメタデータデータファイルをあらかじめ準備すること。
//

require 'tinyoai.php.inc';

// フレームワーク内定数のオーバーライド
define ('REPOSITORY_NAME', 'A simple Data provider');
define ('ADMIN_EMAIL', 'yamamoto@nul.nagoya-u.ac.jp');
define ('LIST_MAX', 50);

// カタログファイルの場所
define ('CATALOGUE_FILE' , './data.txt.catalog');
// メタデータファイルの場所
define ('METADATA_FILE' , './data.txt');

// フォーマット一覧。グローバル変数
$formats['oai_dc'] = array(
  'http://www.openarchives.org/OAI/2.0/oai_dc/',
  'http://www.openarchives.org/OAI/2.0/oai_dc.xsd'
);
$formats['testformat'] = array(
  'http://testformat.ac.jp/testformat',
  'http://testformat.ac.jp/testformat/dummy.xsd'
);

// カタログファイル読み込み用
$catalog = array();

// メタデータファイル読み込み用
$datafilehandle = null;

// 初期化。フレームワークが呼び出す
function fw_initialize() {
  global $datafilehandle;
  $datafilehandle = fopen(METADATA_FILE,'r');
}

// 終了処理。フレームワークが呼び出す
```

```
function fw_finalize() {
  global $datafilehandle;
  fclose($datafilehandle);
}

// 使える metadataPrefix 一覧を返す。フレームワークが呼び出す
function fw_listmetadataformats($identifier) {

  global $formats;

  while (list($key, $val) = each($formats)) {
  $b .= "<metadataFormat>".
        "<metadataPrefix>$key</metadataPrefix>".
        "<metadataNamespace>$val[0]</metadataNamespace>".
        "<schema>$val[1]</schema>".
        "</metadataFormat>";
  }
  return $b;
}

// カタログファイルを読み込む。2 回目は何もしない
function ensure_catalog() {

  global $catalog;

  if (count($catalog) > 0) { return; }

  $f = fopen(CATALOGUE_FILE,'r');
  while ($line = fgets($f)) {
    $a = explode(",",rtrim($line));
    $catalog[$a[0]] = $a;
  }
  fclose($f);
}

// ID に対応する生データを得る。配列のリファレンスとして。
// 連想配列のそれぞれの値が単純配列という感じのものになる。
function & get_rawdata($identifier) {

  global $catalog;
  global $datafilehandle;

  if ($catalog[$identifier] == null) { return null; }

  $raw = array();
  $raw['id'] = $identifier;
  $raw['datestamp'] = $catalog[$identifier][1];

  fseek($datafilehandle, $catalog[$identifier][2]);

  for ($i = 0; $i < $catalog[$identifier][3]; $i++) {
```

```
    $line = fgets($datafilehandle);
    $a = explode("¥t", $line, 3);
    $key = trim($a[0]);
    $val = trim($a[1]);
    if ($key == 'datestamp') { continue; }
    if (array_key_exists($key, $raw)) {
      $raw[$key][] = $val;
    } else {
      $raw[$key] = array($val);
    }
  }
  return $raw;
}

// レコード組み上げ。引数は raw 構造体。oai_dc 版
function & build_record_oai_dc(&$raw) {

  $rec = array();
  $rec['id'] = $raw['id'];
  $rec['datestamp'] = $raw['datestamp'];
  $rec['set'] = $raw['set'];
  $rec['deleted'] = $raw['deleted'];

  $b = '<oai_dc:dc xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/" ' .
       'xmlns:dc="http://purl.org/dc/elements/1.1/" ' .
       'xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ' .
       'xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/ ' .
       'http://www.openarchives.org/OAI/2.0/oai_dc.xsd">';

  if ($raw['title']) {
    $b .= '<dc:title>' . htmlspecialchars($raw['title'][0]) . '</dc:title>';
  } else {
    $b .= '<dc:title>(no title)</dc:title>';
  }

  if ($raw['creator']) {
    foreach ($raw['creator'] as $i) {
      $b .= '<dc:creator>' . htmlspecialchars($i) . '</dc:creator>';
    }
  }

  if ($raw['publisher']) {
    foreach ($raw['publisher'] as $i) {
      $b .= '<dc:publisher>' . htmlspecialchars($i) . '</dc:publisher>';
    }
  }

  if ($raw['URI']) {

    $b .= '<dc:identifier>' . htmlspecialchars($raw['URI'][0]) . '</dc:identifier>';
```

```
  }

  if ($raw['description']) {
    foreach ($raw['description'] as $i) {
      $b .= '<dc:description>' . htmlspecialchars($i) . '</dc:description>';
    }
  }

  $b .= '</oai_dc:dc>';
  $rec['metadata'] = $b;

  return $rec;
}

// レコード組み上げ。引数は raw 構造体。testformat 版
function & build_record_testformat(&$raw) {

  $rec = array();
  $rec['id'] = $raw['id'];
  $rec['datestamp'] = $raw['datestamp'];
  $rec['set'] = $raw['set'];
  $rec['deleted'] = $raw['deleted'];

  $b = '<meta xmlns="http://testformat.ac.jp/testformat">';

  if ($raw['title']) {
    $b .= '<title>' . htmlspecialchars($raw['title'][0]) . '</title>';
  } else {
    $b .= '<title>(no title)</title>';
  }

  if ($raw['creator']) {
    foreach ($raw['creator'] as $i) {
      $b .= '<author>' . htmlspecialchars($i) . '</author>';
    }
  }

  if ($raw['URI']) {
    $b .= '<URI>' . htmlspecialchars($raw['URI'][0]) . '</URI>';
  }

  $b .= '</meta>';
  $rec['metadata'] = $b;

  return $rec;
}

// レコード一件分のメタデータを整形して返す。フレームワークが呼び出す
function & fw_getrecord($identifier, $format) {

  global $formats;
```

```
  if (!array_key_exists($format, $formats)) { return OAI_ERROR_NO_FORMAT; }

  ensure_catalog();

  $raw = get_rawdata($identifier);
  if (!$raw) { return OAI_ERROR_NO_HIT; }

  if ($format == 'oai_dc') {
    return build_record_oai_dc(&$raw);
  } elseif ($format == 'testformat') {
    return build_record_testformat(&$raw);
  }
}

// 絞込み条件に一致するレコード一覧を返す。フレームワークが呼び出す。
// カタログファイルを読んで選ぶ。
function & fw_listrecords($format, $from, $until, $set, $offset, $maxrecs) {

  global $formats;
  global $catalog;

  // no support for sets
  if ($set) { return OAI_ERROR_NO_SETS; }
  if (!array_key_exists($format, $formats)) { return OAI_ERROR_NO_FORMAT; }
  if (!file_exists(CATALOGUE_FILE)) { return OAI_ERROR_NO_HIT; }

  ensure_catalog();

  $hit_buf = array();
  $hit = 0;

  foreach ($catalog as $a) {

    if ($from and ($a[1] < $from))   { continue; }
    if ($until and ($until < $a[1])) { continue; }

    $hit++;
    if ($hit < $offset) { continue; } // skip BEFORE offset

    $hit_buf[] = $a[0];

    if ($hit >= $offset + $maxrecs - 1) { break; }  // end loop AFTER maxrecs
  }

  $recs = array();
  foreach ($hit_buf as $i) {
    $recs[] = fw_getrecord($i, $format);
  }
  return $recs;
}

// メインルーチン
oai_mainroutine();
?>
```