Congr. Int. Ing. Electrón. Mem. Electro 2012, vol.34, pp. 318-323 Chihuahua, Chih. México
http://depi.itchihuahua.edu.mx/display/memorias_electro/MemoriaElectro2012.zip
ISSN 1405-2172

IMPLEMENTACIÓN DE ALGORITMOS PARA LA CONSTRUCCIÓN DE PROVEEDORES DE DATOS OAI

Oscar Beltrán Gómez, Norma J. Ortega Rodríguez, Milton J. Batres Márquez
Universidad Tecnológica de Chihuahua
Tecnologías de la Información y Comunicación
Av. Montes Americanos 9501 Sector 35. C.P. 31216 Chihuahua, Chih.
Tel. (614) 432-20-00
obeltran@utch.edu.mx, jortega@utch.edu.mx, mbatres@utch.edu.mx

RESUMEN.
La Iniciativa de Archivos Abiertos u OAI, por sus siglas en Ingles, aborda el problema de la dispersión de recursos y documentos en Internet implementando un protocolo de recolección [1]. En este trabajo se presenta una introducción y una serie de algoritmos para la implementación de nodos OAI en su carácter Protocolo Recolector de Metadatos o PMH. El patrón propuesto para su diseño es el Modelo-Vista-Controlador y el uso de lenguajes y tecnologías como: PHP, MySQL y el motor de plantillas Smarty. Finalmente se describe la validación del servicio en: *www.openarchives.org/data/registerasprovider.html*.

1. INTRODUCCIÓN

Con la tecnología e Internet avanzando día a día, es obvio el surgimiento de nuevos y mejores medios de divulgación. La Web 2.0 [2] en Internet brindando una plataforma para la creación y uso de nuevos y mejores modelos de publicación como alternativas a los modelos tradicionales.

En contraste a los modelos típicos e impresos, hoy en día tenemos alternativas para publicar y divulgar trabajos e investigaciones en sitios institucionales o personales, blogs, ebooks, memorias de congresos, revistas online e incluso redes sociales [2], sin embargo, esto también genera información hasta cierto punto aislada con poca visualización y sobre todo dispersa en Internet propinando un considerable esfuerzo de búsqueda para extraerla.

La OAI, viene a solucionar el problema de la dispersión de los documentos en múltiples depósitos temáticos y/o revistas individuales [3]; lo hace implementando un protocolo de recolección sobre cualquier material almacenado en forma electrónica para su posterior publicación en Internet.
Esta iniciativa, que data desde 1999 en la Convención de Santa Fe [1], [4], ya ha sido aceptada y acogida por varias universidades en el mundo, hoy podemos ver un lista de las universidades e instituciones que ya publican sus trabajos de esta forma en www.openarchives.org/Register/BrowseSites.

En México podemos citar al portal http://educonector.info [5] que también emplea la arquitectura y filosofía de OAI en su compromiso de crear un

"Buscador de Recursos Educativos Abiertos (REA) y Objetos de Aprendizaje (OA), para facilitar la educación a través de la distribución y el acceso abierto del conocimiento."[5]

En este sitio, de reciente creación (enero-octubre del 2011), participan: Tecnológico de Monterrey (ITESM), Universidad de Guadalajara (UdeG), Universidad de Montemorelos (UM), Instituto Tecnológico de Chihuahua (ITCH), Universidad Autónoma de Guadalajara (UAG) con la ayuda y patrocinio de CUDI (Corporación Universitaria para el Desarrollo de Internet) y del Consejo Nacional de Ciencia y Tecnología (CONACYT).

Lo anterior es muestra de la gran aceptación de esta iniciativa, por ello queremos exponer en este trabajo una serie de algoritmos y tecnologías para la implementación de este protocolo en su carácter de proveedor de datos u OAI-PMH.

2. ROLES Y PROTOCOLOS OAI

Ahora la OAI cuenta con dos vertientes:

- Proveedor de Datos (PD) u OAI-PMH.
- Proveedor de Servicio (PS) u OAI-ORE (Object Reuse and Exchange).

ISSN 1405-2172

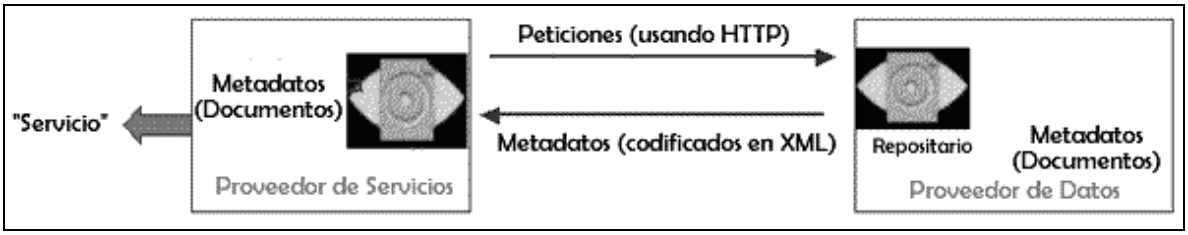

Figura 1. Diagrama de interacción HTTP entre los proveedores de datos y de servicios.

Los Proveedores de Datos u OAI-PMH, contienen los metadatos de los recursos a publicar en internet, es decir, los nombres, temas y grupos de los recursos. Los Proveedores de Servicio u OAI-ORE extraen esos metadatos y los usan para brindar y ofrecer un servicio de búsqueda en Internet [6].

Para el despliegue o publicación de los metadatos se pueden usar varios formatos como: MARC [7], Olac [8], [9] y Dublin Core [10], siendo este último el que se emplea en este proyecto.

2.1. El Protocolo HTTP.

El protocolo OAI en Internet, trae consigo el uso de otros protocolos como el HTTP (Hypertext Transfer Protocol), este servicio utiliza el mismo mecanismo que carga las páginas web en los navegadores empleando los métodos GET [11] y POST [11], (comandos propios del protocolo HTTP), sin embargo, las respuestas de este servicio no son devueltas en formato HTML (HyperText Markup Language) sino en XML (Extensible Markup Language). Los métodos GET y POST son los encargados de atender las solicitudes hechas por los proveedores de servicios hacia los proveedores de datos. (Figura 1).

Los métodos GET y POST establecen y asignan valores en forma de parámetros, sin embargo, la asignación entre un método GET y POST varía considerablemente. El método GET envía sus parámetros usando la barra de direcciones, el método POST los incluye en la solicitud del recurso y no en la barra de direcciones. La dirección web http://edoc.hu-berlin.de/OAI-2.0?verb=ListIdentifiers&metadataPrefix=oai_dc muestra una solicitud GET.

En la dirección anterior y en la figura 2 podemos observar una serie de llamadas HTTP usando el método GET, también vemos que a partir del símbolo "?" se establecen los parámetros de entrada con sus respectivos valores, separados por el símbolo "&".

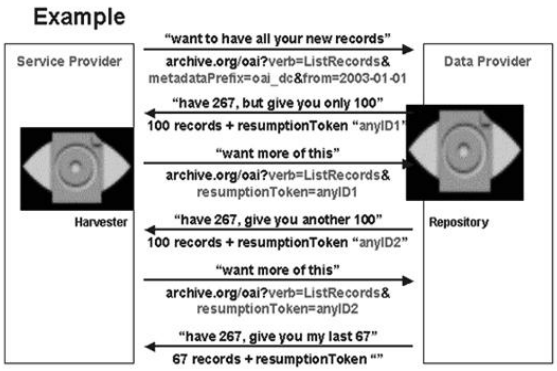

Figura 2. Diagrama de interacción PS-PD.

2.2. El Protocolo OAI-PMH.

El protocolo OAI-PMH, justo el tema de este articulo, establece seis acciones o como el protocolo los llama, seis verbos básicos [12] que son: Identify, GetRecord, ListIdentifiers, ListRecords, ListSets, ListMetadataFormats. Cada verbo tiene, además de una función, parámetros determinados para una identificación precisa de la información de los recursos expuestos.

El diagrama de la figura 2 muestra como los verbos propician la interacción entre los proveedores de datos y de servicios [15].

No hay que olvidar que el protocolo OAI expone metadatos de temas, nombres e información de cómo localizar recursos y material almacenado en forma electrónica, es necesario entonces, contar con esos recursos disponibles también en Internet.

3. LA IMPLEMENTACIÓN

La implementación es precisamente el desarrollo de cada uno de los verbos que define el protocolo; ya que los buscadores o proveedores de servicio se

comunican a través de esos verbos para la recolección de los metadatos.

3.1. El patrón de Diseño MVC

Para la codificación de los algoritmos encargados de cada verbo se propone el uso del patrón de diseño Modelo Vista Controlador (MVC) ya que éste separa el desarrollo de la aplicación en tres capas: el Modelo de la información, la Vista (la forma de cómo será desplegada esa información) y el Controlador (la lógica que maneja y trata esa información) [13].

3.2. El Front Controller

En este punto es evidente la necesidad de identificar que exista una solicitud a un verbo, para ello se implementa un **Front Controller** o **Controlador Frontal** el cual valida dicha solicitud al preguntar por el parámetro "verb" y su contenido "no nulo", en caso contrario la acción por defecto será un error (figura 3).

El parámetro "*verb*" es parte básica de la interacción OAI y el controlador frontal lo utiliza para asegurar que la solicitud corresponda a uno de los verbos en cuestión [14].

En la figura 3, línea 9 se pregunta si la solicitud http://nombre-dominio/oai.php?verb=Identify corresponde a:

- Un archivo que implemente un verbo "*is_file ($ruta_controlador)*" para después requerirlo con la sentencia "*require_once $ruta_controlador*" y además

- que contenga la función adecuada para accionar el verbo con la llamada a "*is_callable($controlador.= '_action')*".

Solo después de este proceso la sentencia "*if*" del archivo llamará la función "*controlador*();", ésta realmente hace referencia a la función "*identify_action*()".

```
           // archivo oai.conf.php
 1: <?php
 2:        $carpetaVerbos = "verbs/";
 3:        require_once ('./libs/Smarty.class.php'); // Instalación Smarty
 4:
 5:        function getParam( $name )
 6:        {
 4:          return ($_SERVER["REQUEST_METHOD"]=="GET"? $_GET[$name] :
                                                              $_POST[$name]);
 7:        }
 8: ?>

           // archivo oai.php
 1: <?php
 2: require_once './conf/oai.conf.php'; //Cargamos la Configuración
 3: $controlador = getParam('verb');
 4:
 5: //Formamos la ruta del archivo del verbo a accionar
 6: $ruta_controlador = $carpetaVerbos . $controlador . '.php';
 7:
 8: //Incluimos el controlador y llamamos la accion
 9: if(  is_file ($ruta_controlador) && (require_once $ruta_controlador) &&
        is_callable ($controlador.= "_action")
      )
10:      $controlador();
11: else
12: {
13:   require_once "./errors/badVerb_error.php";
14:   error_action();
15: }
16: ?>
```

Figura 3. Código fuente del Front Controller.

ISSN 1405-2172

```
 1: <?php
 2: function identify_action()
 3: {
 4:     $smarty = new Smarty;
 5:
 6:     $smarty->assign("repositoryName", "Repositorio de Objetos OAI", true);
 7:     $smarty->assign("protocolVersion", "2.0", true);
 8:     $smarty->assign("adminEmail", "obeltran@utch.edu.mx", true);
 9:     $smarty->assign("deletedRecord", "persistent", true);
10:     $smarty->assign("granularity", "YYYY-MM-DDThh:mm:ssZ", true);
11:     $smarty->assign("compression1", "no", true);
12:     $smarty->assign("compression2", "deflate", true);
13:
14:     $smarty->display('../templates/verbs/identify.tpl');
15: }
```

Figura 4. Código fuente del controlador Identify (Identify.php).

El verbo Identify es válido al encontrarse un archivo con ese nombre y cuando ese archivo contiene una función controlador que genere una respuesta (figura 4). Si cualquiera de los puntos anteriores no se cumplieran, se procedería a cargar el controlador de error (figura 3, línea 13). El error badArgument se refiere a que el verbo solicitado no existe y solo se presenta cuando la solicitud no corresponde a un verbo.

Para que el Controlador Frontal pueda encontrar y cargar los verbos es necesaria una estructura de directorios y archivos como la que se muestra en la figura 5.

```
\oai\
   |
   |_oai.php
   |
   |_verbs\
         |
         |_GetRecord.php
         |_Identify.php
         |_ListIdentifiers.php
         |_ListMetadataFormats.php
         |_ListRecords.php
         |_ListSets.php
```

Figura 5. Estructura de directorios y archivos.

3.3. La Implementación de los Verbos

La implementación de los verbos se puede dividir en dos categorías [17]. La primera define a los verbos que exponen información de la aplicación y la segunda se encarga de exponer la información pertinente de los recursos a compartir.

En la primer categoría se incluye información de la aplicación como nombre del servidor, versión del protocolo usado y tipos de formato así como los verbos Identify, ListSets y ListMetadataFormats. En la segunda se incluyen los verbos GetRecord, ListIdentifiers, y ListRecords.

El verbo Identify es usado, entre otras cosas, para validar y registrar la aplicación como proveedor de datos en el sitio oficial. El registro toma el correo electrónico descrito en este verbo para pasar la validación inicial. El verbo Identify es el primero que se debe codificar.

3.4. La Vista

Para desplegar la información manejada por cada verbo, hemos empleado un manejador de plantillas llamado Smarty. A las plantillas creadas por el Smarty se les pasa un "Modelo de datos" en forma de arreglo asociativo para cubrir las variables definidas "rellenando" de esta forma la plantilla para posteriormente desplegarla. (Figura7, líneas 4 a 14).

Los verbos Identify y ListMetadataFormats manejan información, hasta cierto punto, estática por tanto no se ocupa un modelo, la información se asigna directamente. (figura 4, líneas 4 a 14)

El uso de plantillas agrega funcionalidades deseables como: mostrar la hora en el formato requerido (figura 6, línea 4) o resolver el nombre y ruta del sitio (figura 6, línea 6).

Cuando los datos están "colocados" en la plantilla es llamado el método display (figuras 4 y 6, línea 14) para ser desplegada.

3.5. El Modelo

El verbo ListIdentifiers perteneciente la segunda categoría. Se centra en la recuperación de los recursos a publicar por tanto ocupa un modelo que alimente a la plantilla, Se maneja una "variable llamada ListIdentifiers" que es la que contendrá el arreglo asociativo (modelo) regresado por la función "fnc_assoc". (Figura 7, línea 10).

```
 1: <?xml version="1.0" encoding="UTF-8"?>
 2:: <OAI-PMH xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
 3:                                 http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
 4:     <responseDate>{$smarty.now|date_format:"%Y-%m-%dT%H:%M:%SZ"}</responseDate>
 5:     <request verb="Identify">
 6:                 http://{$smarty.server.SERVER_NAME}{$SCRIPT_NAME}
 7:        </request>
 8:
 9:     <Identify>
10:         <repositoryName>
11:             {$repositoryName}
12:         </repositoryName>
13:         <baseURL>http://{$smarty.server.SERVER_NAME}{$SCRIPT_NAME}</baseURL>
14:         <protocolVersion>{$protocolVersion}</protocolVersion>
15:         <adminEmail>{$adminEmail}</adminEmail>
16:         <earliestDatestamp>
17:                 {$smarty.now|date_format:"%Y-%m-%dT%H:%M:%SZ"}
18:           </earliestDatestamp>
19:         <deletedRecord>{$deletedRecord}</deletedRecord>
20:         <granularity>{$granularity}</granularity>
21:         <compression>{$compression1}</compression>
22:         <compression>{$compression2}</compression>
23:     </Identify>
24: </OAI-PMH>
```

Figura 6. Plantilla del verbo Identify (identify.tpl).

La función fnc_assoc retorna un modelo (nombre_variable/ valor), a partir de una cadena en SQL (Structured Query Language), esto hace que el manejador Smarty entienda en donde asignar cada valor y cuantas veces hacerlo (una vez por registro encontrado). El código que genera este arreglo asociativo al igual que el resto de los algoritmos se puede descargar de: https://github.com/oscarbego/oai_pmh.

```
$smarty->assign(
     'ListIdentifiers',
      fnc_assoc(
        "SELECT * FROM
           listidentofiers;" ));
```
(1)

Como en este proyecto se emplea el motor MySQL para el almacenamiento de la información de cada recurso, se hace necesaria una serie de tablas y vistas capaces de almacenar y exponer los metadatos de los recursos a publicar.

Se sugiere congruencia en los nombres de las columnas entre vistas y plantillas ya que de esos nombres se generará el arreglo asociativo del modelo y también se recomienda el uso de vistas en lugar de una interacción directa con las tablas.

Las vistas agregan performance a las búsquedas, a demás la interacción con el servicio PD no involucra ningún tipo de modificación.

4. VALIDACIÓN OAI

La OAI cuenta con una herramienta para validar y registrar a los PD. En la dirección http://www.openarchives.org/data/registerasprovider.html podemos encontrar los pasos para registrar o validar nuestra aplicación PD.

Una vez que se escriba la dirección en internet de nuestra PD y al presionar el botón de “submit” la aplicación “OAI Register As Provider” (OAI-RAP) validará el verbo Identify buscando entre otras cosas la descripción de la etiqueta “adminEmail”. El OAI-RAP identificará entre los datos la dirección de correo electrónico y enviará un mensaje con los pasos para terminar la validación. En caso de errores se tendrá que ingresar de nuevo al “OAI-RAP” para reiniciar la validación.

5. CONCLUCIONES

Como hemos visto, el OAI es un protocolo de recopilación que implementa búsquedas distribuidas de metadatos para, posteriormente, brindar un servicio al exponerlos en Internet.

Los investigadores e instituciones educativas pueden considerar a la OAI como la plataforma de entrada para exponer sus contenidos en acceso abierto, sin embargo, se debe de contar primero con los mecanismos para que los PS puedan buscar y exponer esos recursos.

```
1: <?php
2: require_once('./models/fnc_asociativo.php');
3:
4:  function listidentifiers_action()
5:  {
6:        $smarty = new Smarty;
7:
8:     if ($_GET['metadataPrefix'] == 'oai_dc')
9:     {
10:       $smarty->assign('ListIdentifiers', fnc_assoc(
                                              "SELECT * FROM listidentofiers;" ) );
11:        $smarty->display('../templates/verbs/ListIdentifiers.tpl');
12:     }
13:     else
14:                $smarty->display('../templates/errors/ErrorListIdentifiers.tpl');
15: }
16:?>
```

Figura 7. Código fuente del controlador ListIdentiers (ListIdentifiers.php).

Aunque los algoritmos presentados en este trabajo son capaces de validar un sitio proveedor de datos, éstos deberán usarse solo para explicar el funcionamiento del protocolo OAI-PMH y como base para implementaciones mas robustas.

La OAI es un buen instrumento para demostrar a los autores la utilidad de poner sus trabajos en abierto, sin embargo, es necesario revisar sus objetivos y audiencias de acuerdo a la filosofía del acceso abierto para "maximizar el impacto al maximizar la difusión".

6. BIBLIOGRAFIA

[1] Open Archives Initiative. Home Page. http://www.openarchives.org/.

[2] D. Torres y E. Delgado, "Estrategia para mejorar la difusión de los resultados de investigación con la Web 2.0". El profesional de la información, 2009, sep-oct, v. 18, n. 5, pp.534-539.

[3] La iniciativa de ficheros abiertos (OAI): protocolo OAI-PMH, proveedores de datos, proveedores de servicios, Sitio Web Asociación Española de Documentación e Información (SEDIC).http://www.sedic.es/autoformacion/acceso_abierto/4-iniciativa-ficheros-abiertos.html.

[4] Van de Sompel, H; Lagoze, C; The Santa Fe Convention of the OAI. D-Lib Magazine Feb 2000 Volume 6 N.2 ISSN 1082-9873.

[5] Buscador de Recursos Educativos Abiertos (REA) y Objetos de Aprendizaje (OA), Home Page. http://educonector.info.

[6] Maslov, A. et al. Adding OAI-ORE Support to Repository Platforms. Journal of Digital Information, Vol 11, No 1 (2010) .

[7] MARC STANDARDS, Librarian of Congress, HomePage. http://www.loc.gov/marc/marcspa.html

[8] OLAC, Home Page, http://www.language-archives.org/

[9] S, Bird; G, Simons; The OLAC Metadata Set and Controlled Vocabularies

[10] D, Hillmann; Using Dublin Core, in Web Site http://dublincore.org/documents/usageguide/

[11] RFC 2616 Fielding, et al. The World Wide Web Consortium; in Web Site http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html.

[12] J, Barrueco; OAI-PMH: Protocolo para la transmisión de contenidos en Internet; Biblioteca de Ciències Socials. Universitat de València, in http://eprints.rclis.org/bitstream/10760/4093/1/cardedeu.pdf.

[13] Zend Framework & MVC Introduction; Programmer's Reference Guide; in Web Site http://framework.zend.com/manual/en/learning.quickstart.intro.html.

[14] Implementing OAI-PMH; Open Archives Forum; in Web Site http://www.oaforum.org/tutorial/english/page4.htm

[15] J. Smith; M. Nelson; Creating Preservation-Ready Web Resources; D-Lib Magazine January/February 2008 Volume 14 Number 1/2 ISSN:10829873.http://www.dlib.org/dlib/january08/smith/01smith.html.

[16] Error and Exception Conditions, The Open Archives Initiative Protocol for Metadata Harvesting;http://www.openarchives.org/OAI/openarchivesprotocol.html#ErrorConditions.

[17] M, Nelson; et al. mod_oai: An Apache Module for Metadata Harvesting; http://arxiv.org/ftp/cs/papers/0503/0503069.pdf.