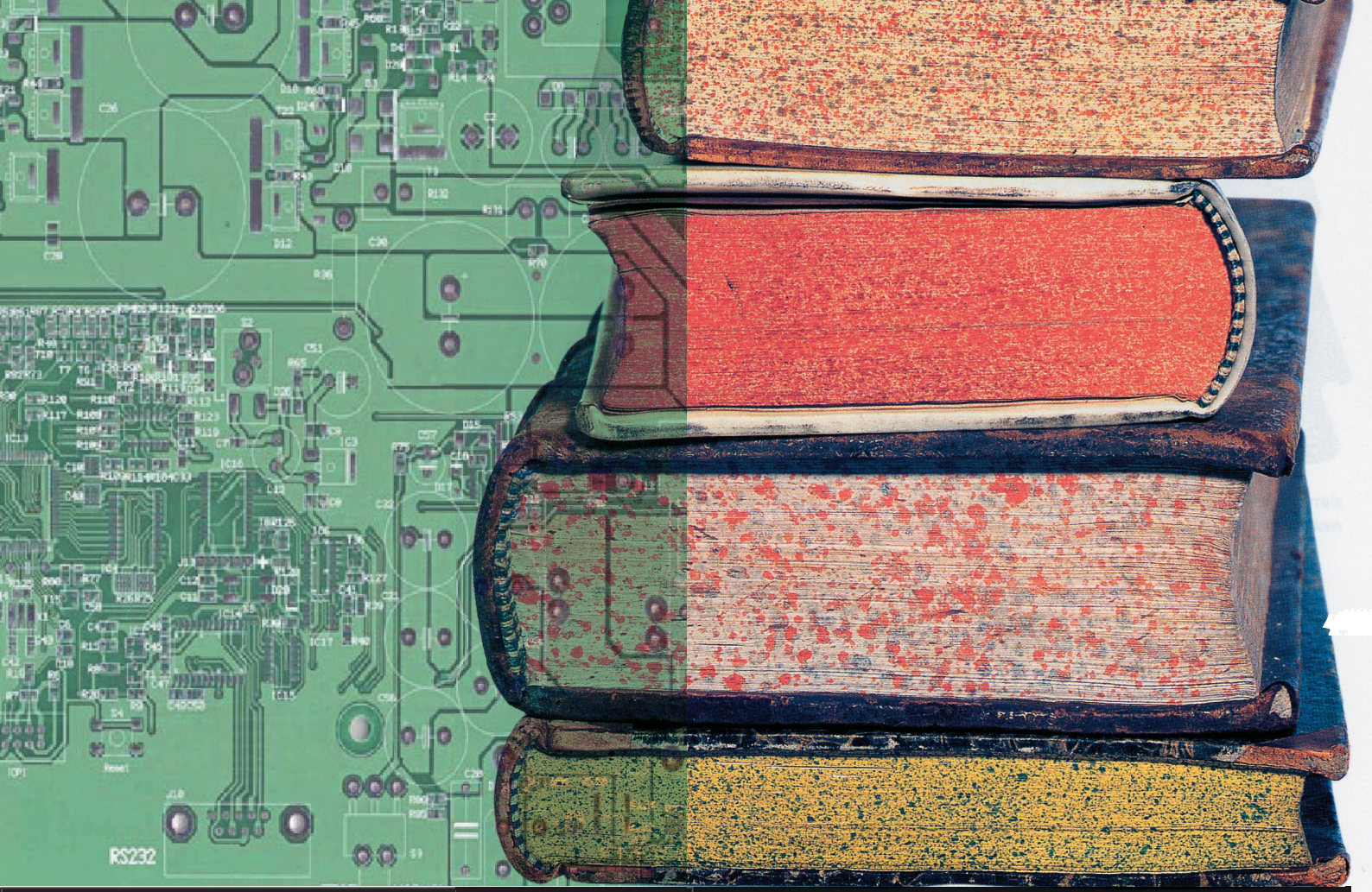
Digitale Bibliotheek Europese Minderheidstalen

van der Meer, C.; Albers, W.P.J.M.; van der Meer, P.L.G.; Versloot, A.P.; Boelhouwer, B.

2005



DIGIBYB

Digital Library Project
European Minority Languages

Technical report

Fryske Akademy Mercator Education

Digibyb

Digital Library Project on
European Minority Languages

Technical report

Fryske Akademy / Mercator – Education
Leeuwarden / May 2005

Contents

1 Intended audience

1.1 Purpose

This document describes the technological concepts of a digital library. It is the result of a pilot project targeted at European Minority Languages. It is intended for those who want to start a digital library of their own, or those who wish to improve or extend existing digital libraries.
This report is intended to show how things can be done. Its focus, with regards to the audience, lies on potential participants of EMILL (http://www.emill.org), which provides a framework for knowledge exchange through a digital library and portal.
This document is part of the entire Digibyb documentation, and should be dealt with together with the part on Organisation and the part on Content.

1.2 Assumed knowledge

Large parts of this document can be read by anyone with an interest in the technological aspects of a digital library, ans with a minimum of prior knowledge. Some parts of this document are intended for people with some technical background: people with a basic understanding in the field of computer science, such as software engineers (someone who is responsible for the actual realisation of the technical requirements). To grasp the concepts mentioned in this document, a basic knowledge is assumed of:

- The Client/Server model
- Software Protocols in general
- The OSI reference model

The reader should know what the following terms mean or should at least have heard of them:

- TCP/IP (Transmission Control Protocol/Internet Protocol)
- HTTP (HyperText Transfer Protocol)
- HTML (HyperText Markup Language)
- XML (Extensible Markup Language)
- Unicode
- RPC (Remote Procedure Call) / RMI (Remote Method Invocation)
- SQL (Structured Query Language)

The reader should have some experience with:

- Building web sites
- General programming
- Web application programming
- Relational databases

Also useful to know or have experience with:

- The use of Open Source software
- Adhering to Open Standard

1.3 Recommended prior reading

To know what the pilot project is all about and to see the bigger picture, please read:

- Digibyb Starting Notes (NL)
- Digibyb Content Plan (NL)
- Digibyb Documentation, Part 1: Organisation (EN)
- Digibyb Documentation, Part 2: Content (EN)

If anything of the previous paragraph is mumbo jumbo to you, then Wikipedia (http://www.wikipedia.org) is an excellent place to start, or to brush up.

2 Introduction

2.1 Goals

In the digital library various types of media are collected, to be presented to users on the Internet. Users must be able to search in this collection, and retrieve contents from the library. Data in the library are structured in a uniform way to accomplish this.
A more elaborate description of requirements can be found in a previous part of the (non-technical) documentation.

2.2 Problems

In practice the material that is to be part of the digital library is contained in several potentially very different types of databases. This is particularly true for EMILL, for as these data are the property of, and maintained by, various organisations.
Only one user interface should be used to access data contained in these so-called repositories.

2.3 Solutions

The use of Open Standards greatly helps in realising this interoperability. These standards specify how information can be represented, exchanged and presented. The use of standards allow users easy future reading and exporting of data, and vendor lock-in is prevented. Also, the use of Open Standards increases the chances of being able to find and use good Open Source software that can work with these standards.

3 User interface

3.1 Client/Server

Users are allowed access to the digital library through a client/server model. Actually, several

intermediate client/server connections are made, not all synchronous, to achieve accessibility. Further details will be explained later.

3.2 Browsers

Users on the Internet make use of a standard modern web browser to access the user interface to the digital library. All parts of the GUI are presented entirely by means of HTML: no browser plugins are required.

3.3 Web page layout

At any time, the user interface shows a general search field, which can be used to search the entire web site - or part of it - depending on the interface context. Navigation is on the right and some useful text can be made visible in the centre.

3.4 Site structure

Using the navigation on the right-hand side of the screen, users can browse the hierarchical file-system-directory-like structure. The site is structurally split into different language parts, each having user-defined subcategories, possibly containing various kinds of objects.

3.5 Searching

Searching can be done, as mentioned, by using the general search field. More powerful search options can be used with an advanced search form, in which various criteria can be entered.
It's possible to use specialised search interfaces for separate repositories. Internally the search forms use a specific search query syntax, which can also be used directly in the general search field.

4 Site maintenance

4.1 Content management system

A practical means of realising the aforementioned user access can be a Content Management System, abbreviated CMS. We have used an Open Source product called i-Tor. i-Tor itself makes use of the Open Source Java Search Engine Software Lucene, for indexing, among other Open Source components. Future versions of i-Tor will be based on the Eclipse Plugin Framework. Search queries in i-Tor are expressed directly in Lucene syntax.
CMS jobs include:

- Easy site administration
- Presentation on the World Wide Web
- A database for site data

- Indexing module
- Search module
- Log-in accounts

i-Tor, Tools and Technology for Open Repositories, has these. Additionally, i-Tor has a feature for exhibiting Open Archives, which is one of the main reasons for choosing it, as this suits digital libraries very well.

4.2 Publishing

Giving ordinary users (i.e. users with non-administrative privileges within the CMS) the ability to upload files is a desirable feature. Within the i-Tor powered web site, the root user (super user) has the ability to create accounts for other users. They can upload documents in a variety of formats. They can also change texts and other objects, provided that they are granted the rights to do so, which can be specified for each object individually.

All media uploaded to the site will be stored in the internal database of the CMS directly. It is not possible to update external repositories this way which applies to the vast majority of the digital library's material.

4.3 Updating

It is only possible to update data that are stored in the internal i-Tor database this way. A large part of what the digital library has to offer is linked from external resources. Updating data in these resources occurs locally, by methods defined locally.

5 Under the hood

5.1 Multiple databases

Library data are provided by already existing databases. These databases have not been grouped together to create a new database. There is no need for this, and it is even undesirable to do so.

The existing database systems used already have documented ways of maintaining and managing them, along with (Graphical) User Interfaces. Often, these databases are being used to provide applications which offer users useful service, their data. It would not be very economical if they had to be migrated to a new system.

There can be several databases within one organisation, or several databases spanning multiple organisations. What is important is that they all provide their data in the same way to the software providing a service to end users. What is required is an adapter that accepts multiple different inputs and generates one type of output. There can be multiple adapters (or adapter instances), but the output is required to be of the same type.

5.2 Meta-data and data

This, among other things, requires a mapping: a mapping of meta-data field names used in the original database to a general set of meta-data fields used for all data.
The data consist of electronic texts in various formats/encodings. There is also meta-data describing physical documents in the library. Meta-data is data about data, such as the author, title, etcetera of a book. One meta-data record describes attributes from one electronic/physical medium.

Protocols are used for connecting data to the application providing a service. Such a protocol defines everything needed or useful to interchange data in a generic way.

6 Standards

6.1 Why?

Wherever possible open standards are used. Because they have been thought out well, it is beneficial to use along with all best practices that come with them. This saves time and guarantees some quality.

Using software components that adhere to open standards makes it easier to substitute one for another if at some point in time this should become desirable.
Open standards will always remain open, guaranteeing that anyone looking at the software at a later point in time is able to understand the software and its concepts, and to read and export data from the system, without first having to find some antiquated piece of software that can read it.

6.2 Presentation

6.2.1 HTML

The standard used for presentation of the user interface and data is HTML version 4. Considering the situation of the World Wide Web, and the Internet in general, this is the obvious choice. HTML often works in combination with CSS[1] for styling. All modern web browser software supports this. The interface is still usable without CSS support, but it is not recommended.

6.2.2 Multimedia

Images can be in PNG or JPEG or GIF format. Other file formats for all kinds of media require some piece of software on the side of the client., such as Adobe Acrobat Reader (free, closed source), GNOME PDF Viewer (open source), Xpdf viewer (open source), GSView (open source, Ghostscript front end) to view PDF. Alternatively, a sound/music player can be used with with an MP3 codec for sound files distributed in the MP3 format.

6.2.3 Open Formats

In this paragraph, a few multimedia file formats are mentioned. Most of them are not used (yet) by our digital library, but are merely considered for future use. Open formats would be ideal, but sometimes they are not available for a certain goal, or not widely used. Therefore, there is also a section with propriety formats following this particylar paragraph. To make entire documents available to users - as file downloads - various formats can be used:

- HTML
  Like the entire website, individual documents could also be presented using HTML. This is not actually done in our case: only excerpts are displayed using HTML.
- Plain text (ASCII)
  Readable by all, but totally lacking layout and fonts and other presentation features.
- Open Document Architecture
  An initiative to define an open document format. Not in wide use.
- PostScript
  More or less the predecessor of PDF.
- PDF (Portable Document Format)
  The format that we actually use for all of our documents. PDF is an open standard defined by Adobe.

Raster/pixmap/bitmap images:

- PNG (Portable Network Graphics)
- JPEG (Joint Photographic Experts Group)

Vector images:

- SVG (Scalable Vector Graphics)

Sound files:

- Ogg/Vorbis audio codec

Video files:

- MPEG-2
- XviD codec (and other MPEG-4)

Streaming audio and video:

- Ogg
- VideoLAN

6.2.4 Propriety formats

Images:

- GIF (patent expired)

Sound files:

- MP3
  Despite its wide use this is not an open format.

Video files:

- DivX
  Its open source counterpart is XviD, which was based on OpenDivX.

Streaming audio and video

- RealNetworks - RealMedia (RealAudio, RealVideo)
- Microsoft - Windows Media
- Apple – QuickTime

6.3 Meta-data

6.3.1 Dublin Core

Data describing the actual file objects, called meta-data, can be represented in a lot of different ways. Various standards are available to our use, among which is one called Dublin Core. Dublin Core is a set of element and term definitions that are to be used for describing objects. Dublin Core is the recommended way to apply meta-data on objects when using the protocol for data exchange described later on in this chapter. It ensures interoperability with services implementing this protocol.

Mapping all meta-data from various databases/repositories, which are likely to have different internal representations of meta-data, to one target, makes it possible to search for, say, a specific word in a title in all repositories at once.

6.3.2 TEI Header

Another interesting meta-data format is that specified by the Text Encoding Initiative. TEI originally used SGML for its encoding scheme but can now also use the more modern XML standard.

Meta-data is described with the TEI header element. It is much more powerful than basic Dublin Core. It is also still bigger than Refined Dublin Core, though this does not necessarily mean that it is better. Dublin Core might be better suited because of its simplicity.

As mentioned before, Dublin Core is the presumed choice for use with the utilized data exchange protocol. Although this does not mean that it has to be used for the back-end representation (i.e. it does not have to be directly stored as DC), it could for example be realised/implemented using a generic on-the-fly mapping of TEI to DC. The TEI header would be embedded with the actual document, along with its body also encoded using TEI.

6.4 Textual data

6.4.1 TEI

Besides meta-data, TEI describes the actual contents of an object. It has element definitions for chapters, paragraphs, sentences and much, much more. Using TEI to markup texts helps efforts directed at semantic transparency.

It also uncouples content from presentation. Instead of specifying how parts of a document are to be displayed (as in HTML), it specifies what parts are (their meaning), therefore preserving the meaning of the part of the document concerned.

With a typesetting language or a mark-up language designed for presentation, it would be very

difficult to actually determine what part of the document represents for example a chapter header, or a footnote, or a footnote reference, or where the table of contents starts and ends.

Structuring all documents this way in the same manner also enables their uniform presentation. They will all have a similar look, which improves clarity.

6.4.2 TEI Lite

TEI is huge. TEI Lite is a simplification of TEI. It could have been ideal for use in our digital library, although it would take a fair amount of time to convert existing documents to this format. This cannot be done entirely automatically, since information has to be added manually.

6.4.3 PDF

Because the road to TEI is a long one, we have chosen a different approach. Though less elegant, all documents are/have been converted to PDF in the digital library. Separate documentation is available on experiences with this conversion process[2].
The reason for choosing PDF is that it can be fairly easily generated from other document types/formats. Furthermore, it is an open standard, which looks the same on every screen or printer.
The i-Tor indexing module for URLs only supports fulltext indexing of PDF files, as of this writing.
Please note that if we had used TEI/TEI Lite, documents would have been converted on-the-fly to PDF for presentation.

6.5 Data exchange

6.5.1 Protocols

The server and software that provide data are separate from the server and software providing a service using that data. In order for the service provider to access data from the data provider, some communication between the two is required.
Here is where the use of a protocol becomes important. Various protocols are available today that fit our purpose to some degree. To name a few:

- XML-RPC (remote procedure call protocol encoded in XML)
  Requesting the execution of a procedure on the server using XML to specify the request and to obtain some result back.
- SOAP (Simple Object Access Protocol)
  Derived from XML-RPC, SOAP is an extensible and decentralized framework that can work over multiple computer network protocol stacks.
- REST (Representational State Transfer)
  This is actually an architectural style to develop high-level protocols. See Resources for more information.

6.5.2 OAI-PMH

The Open Archives Initiative Protocol for Meta-data Harvesting falls in the REST category. It is the protocol we use - on the server side - and which i-Tor uses - on the client side - (no coincidence). It is a specialised protocol for digital libraries.
OAI-PMH is a stand-alone protocol on top of HTTP, it does not use SOAP for example, though this could be made possible in future specification releases.

It uses six so-called verbs, which can be considered remote procedure/function names. Parameters are passed using CGI-like[3] querystrings. All responses are valid XML.

The verbs:

- Identify
  Obtains some general information on the repository.
- ListSets
  Repositories can be logically divided into sets. This verb results in listing them.
  We support them, but do not use them directly with i-Tor, because its support for sets is limited. Instead we allow sets to be specified in the baseURL of the request, effectively splitting up the repository into multiple repositories. (i-Tor uses baseURLs as unique repository identifiers, which can be used in searches to limit a search to one or more specific repositories).
- ListMetadataFormats
  Lists the formats used by the repository for representing metadata records.
  We only use the standard oai_dc.
- GetRecord
  Retrieves a single record based on an id parameter passed along with the request.
- ListIdentifiers
  Lists all identifiers of records in the repository.
  We have no good use for this verb, i-Tor does not use it, and so it would be more or less be all right to not implement it.
- ListRecords
  Lists all records in the repository. Results of this verb and also of ListIdentifiers, are batches of a fixed number of records at a time, specifying a resumption token that the client (=harvester) can use to resume, getting the next number of records, until all records are retrieved.
  Grabbing these records can be done and is usually done (except for the first time harvesting) selectively by specifying from and until dates. Timestamps must be maintained by the OAI Data Provider for this, to be able for the OAI Service Provider to grab only new or modified document data.

Being able to participate in a digital library project using these principal concepts requires implementing this OAI-PMH protocol (version 2); for the specific situation of databases at one's site. This way, different parties can use a single service provider, which allows ordinary users acces to data of all parties involved.

6.5.3 Loose coupling by URL

OAI-PMH only specifies a protocol for meta-data harvesting, not the data objects themselves. It is easy, however, to provide access to these oblects via meta-data records by using a URL[4] field that contains a link to the actual object. This way, a Harvester can fulltext index these objects, too.

6.6 Data maintenance

OAI-PMH is a one-way read-only protocol. It provides a harvesting mechanism, but no update

mechanism. This cannot be considered a shortcoming of OAI, since it is not within its scope, but this could be an interesting feature. For that reason, OA-X deserves a mention here. OA-X is an experimental protocol designed by the i-Tordevelopers.

OA-X complements OAI-PMH in two ways. First, it allows not only for the exchange of meta-data, but also for the transfer of the object files themselves. Second, it will feature a standardised procedure for uploading meta-data and documents into an OAI compatible repository.

It is this author's opinion that the first aspect is hardly worthwhile, except maybe when a TEI-like structure is used for document data (not a binary format such as PDF). URL linking is really everything anyone needs. The second aspect could provide users with a uniform interface for uploading their material, which gives them the possibility to publish to a specific repository, provided they are granted this privilege.

7 Implementation

7.1 Platform

Since OAI-PMH is merely a protocol, it is not dependent on a specific platform. It can be implemented on any operating system desired. For the same reason, it is also programming language independent. Any logic can be realised using whatever programming language and environment is available for one's OS.

For obvious reasons, the server used to implement the protocol should be connected to a network, preferably the Internet (because of the desired openness of open archives), using the OS's support for the TCP/IP protocol. The TCP port used by the OAI data provider process for listening for incoming OAI request must be accessible from the outside world, where the OAI service provider is located. Therefore, any required configuration adjustments must be made, including opening up this TCP port in a corporate firewall.

The OAI-PMH specification does not specify a default TCP port to use. The choice is yours. In case no port is specified, since OAI works on top of HTTP, the default http port (tcp port 80) is assumed.

We chose Linux as our server platform (on Intel hardware) for our OAI data provider web service. Linux's open nature (it is entirely open source) fits our views well, although there is no immediate relation. Linux has proved itself to be a very reliable server solution, especially in (but definitely not limited to) the LAMP[5] configuration. The author of this document is a great admirer and advocate of Linux, but iti's not within the scope of this document to elaborate on this further.
As for now, the SourceForge.net release of i-Tor requires Linux, but since it has been written in Java, it should not be very hard to port it to other platforms (mainly adjusting the shell scripts). Please note that i-Tor is on a server separate from the OAI data provider, and that only one

instance can be found of an i-Tor server and multiple OAI data provider servers to which i-Tor actively connects.

2.2 Web server software

The next decision to take is what web server software to use. The OAI-PMH protocol was designed to work on top of HTTP, so the smart thing to do is to use software that already implements the specifics of this protocol, instead of building it from scratch. The web service does not require a dedicated server, so one possibility is to use the web server that is already in use to host the website.

The web server needs support for dynamic execution of code on the server side. This is needed because the server application needs to respond to request of the OAI protocol. Providing only static responses is theoretically possible, but is hopelessly limited, and against the nature of any protocol.

We chose the Apache HTTP server. Apache is open source, available for many platforms, and has lots of pluggable modules to provide a wide range of functions. ("The February 2005 Netcraft Web Server Survey found that more than 68% of the web sites on the Internet are using Apache.")

Alternatively, a well-known web serving solution is Microsoft's IIS (Internet Information Services), as is used by Tresoar[6]. A pure Java solution could be used, applying a Jakarta Tomcat (open source, Apache Software Foundation) servlet container. This is the method used by Galicia[7].

7.3 Database software

There is a chance that database servers with data and meta-data required to be made available via OAI have already been set up. Thus, choosing database software is normally no longer an issue.

The question what type of database is used is not important. It could be a relational database, an object-oriented database, a native XML database, btree files, or whatever. All logic dealing with database access is described using a programming language of choice.

There is, however, an implementation design choice to be made, regarding the question whether to implement the OAI services directly on top of the original database(s), or to use separate processes to gather data from these databases, and represent them in a uniform way, before further processing them to OAI.
We have several databases that need to make their data available to our digital library service, by means of repositories. They include a BRS[8] database, MySQL database and an MS Access database file. The latter does not have a database server interface, so it is placed on a central filesystem directory share on the server, and converted with MDB Tools (open source). Tresoar uses MySQL, Galicia uses MS SQL Server.

7.4 Programming language

Realising the OAI web service requires some programming effort. The programming language to use for this is arbitrary, though it has to be usable in a web server context (usage with CGI is always an option, but is not efficient). Databases are typically accessed with the programming language using an API[9]. Alternatively command line tools that accompany the Database System concerned could be used (either locally or by first telnetting/ssh-ing to the database server) to generate dumps, and process these from within your programme.

We use the Python programming language. The Python implementation is freely usable and distributable. Python is about as simple as a general purpose programming language can get, yet very powerful.
Tresoar uses PHP (PHP: Hypertext Preprocessor, recursive acronym). Galicia uses JSP (Java Server Pages).
Support for these programming languages in the web server is acquired by modules. Mod_python is the Apache module for Python support. Likewise there's a PHP extension for IIS. Tomcat has built-in support for JSP, which is translated under the covers to Java Servlets.

7.5 Design principles

Mixing logic with presentation- by generating HTML - or in our case XML, is does not present a very readable or elegant solution. Therefore, we use XSLT from within Python to generate XML. This way, a lot of OAI specifics are abstracted from the Python logic. (For more information on this concept see Model View Controller Design Pattern.) (Standard use of PHP actually does mix logic with presentation.)
It could be beneficial, in case there are multiple databases to deal with, to encapsulate logic for the different databases in separate objects as well as on the data layer. This could make the programme implementing the actual OAI quirks, a lot simpler, while letting separate processes deal with conversion of the backing databases.

We selected the widely used Berkeley DB to implement this gathering principle, and to add information such as timestamps and identifiers required by the OAI protocol.

7.6 Considerations

7.6.1 Dublin Core mapping

The mapping of record fields in a database to Dublin Core is usually not a major problem. It is a one-to-one mapping that can be generated on the fly. Sometimes DC lacks a field, in which case one is allowed to use a non-dc field in the OAI meta-data record, or just leave it out completely.

7.6.2 Timestamps and identifiers

It might be the case that a database does not have a convenient way of accessing from within one's own programmes/scripts: for example if the database system lacks an API, or because it is a propriety-closed system. Another problem might be that the database system used does not

have a mechanism to track (or log) changes made to individual database records. This impedes the provision of timestamps for the OAI service.
When a database system does have timestamp support (as MySQL has), then that could be used. An update to a record then automatically changes the timestamp. The database also has to have unique identifiers for each meta-data record. These ids uniquely identify a record and should not change when updates are made. These requirements may require changes to the database schema.

7.6.3 Periodic synchronisation scenario

Alternatively, when timestamp support is not available or the database schema may not be altered, there is another solution. One can track changes made to the database, for exemple using a command line client to generate a dump of the entire database, and compare that dump to a previous version of a dump, and do this periodically. The standard Unix diff tool is an example of how differences can be analysed.

For changes in file objects themselves one could make use of the timestamps of the filesystem on which the files are stored.
Please note that timestamps must always be in UTC/GMT time zone, not local time. This is obvious because they are part of a protocol that can be communicated spanning different time zones.

7.6.4 Character encoding

When characters are used in data that are not within the range of the 128 ASCII characters, i.e. non-English characters, then an encoding must be specified. For OAI only UTF-8 is valid. No other encodings, such as for exemple iso-8859-1, may be used. Please see Resources for the authoritative UTF-8 specification. Alternatively, one could specify all non-ASCII characters as XML character entities.

7.6.5 Common mistakes

- Incorrect Content-Type. The Content-Type of all responses must be text/xml.
- Typos in namespaces. Namespaces are literal so the trailing slash in xmlns="...OAI/2.0/" matters.
- Missing key/value pairs in responses. Attributes for the request element are the keys of the key=value pairs.
- Whitespace before <?xml
- Wrong character encoding, or not using entities. When using entities encoding can be made irrelevant.
- Wrong baseURL - it must be really base, nothing more, and a working URL
- Not using UTC/GMT times
- Missing verbs. There are six verbs, and error handling. Implement them all.
- Dates not conform with ISO 8601.
- Missing from and until support or incorrect from and until date syntax
- Incorrect error handling. Should always respond with valid XML, no HTTP level error codes.
- Using non-dc element in a dc record.

- Invalid oai-identifiers. Conforming the URI spec, in oai-identifiers this, for example, needs to be escaped: «Afiioho1 --> %22Afiioho1.
- Invalid resumptionToken use. Either you use it or you do not, but do not specify one if flow control by resuming is not supported.
- Invalid URL for fulltext indexing. Must be absolute URL, may not be relative.

8 Resources

8.1 Suggested Further Reading

- Rest of the Digibyb project documentation
- OAI specification (see next paragraph)
- News on the EMILL/DBFrysk[10] websites, for the latest developments

8.2 Internet hyperlinks

OAI
- http://www.openarchives.org
- http://www.openarchives.org/OAI/openarchivesprotocol.html
- http://www.openarchives.org/OAI/2.0/guidelines.htm

Dublin Core
- http://dublincore.org
- http://dublincore.org/documents/dcmi-terms/
- http://dublincore.org/documents/dc-xml-guidelines/

TEI
- http://www.tei-c.org
- http://www.tei-c.org/Lite/

XML
- http://www.w3.org/XML/
- http://www.w3.org/TR/xslt

RFCs
- http://www.ietf.org/rfc/rfc3629.txt (UTF-8)
- http://www.w3.org/Protocols/rfc2616/rfc2616.html (HTTP)

PDF
- http://partners.adobe.com/public/developer/pdf/topic.html
- http://partners.adobe.com/public/developer/pdf/index_reference.html

Other interesting links:
- Linux Online - http://www.linux.org
- The Apache HTTP Server Project - http://httpd.apache.org
- The world's most popular open source database - http://www.mysql.com
- Makers of Berkeley DB - http://www.sleepycat.com
- Python Programming Language - http://www.python.org

- Representational State Transfer - http://en.wikipedia.org/wiki/REST
- Model-View-Controller - http://en.wikipedia.org/wiki/Model_view_controller

9 About this document

- This document was generated using Docutils reStructuredText. [11]
- PDF versions of this document were created using LaTeX (pdfTeX). [12]
- Images in this document were adjusted using the GIMP. [13]
- The text editor used was jEdit. [14]

.

(Footnotes)

[1] CSS: Cascading Style Sheets

[2] Lolkes docs/notes

[3] CGI: Common Gateway Interface

[4] URL: Uniform Resource Locator

[5] LAMP: Linux, Apache, MySQL, PHP/Perl/Python

[6] Tresoar: http://www.tresoar.nl

[7] Galicia: http://bvg.udc.es

[8] BRS: see http://www.textinfo.nl

[9] API: Application Programming Interface

[10] http://www.dbfrysk.org

[11] http://docutils.sourceforge.net/rst.html

[12] http://www.tug.org/teTeX/

[13] http://www.gimp.org

[14] http://www.jedit.org