# PRESTO – Preservation Technologies for European Broadcast Archives

IST-1999-20013

# Deliverable 7.3

## Standardized Access to Broadcast Archive Catalogue Data

DOCUMENT IDENTIFIER PRESTO-T73-JRS-20020510 Standardized Metadata Access

DATE May, 10th, 2002

ABSTRACT

AUTHOR, COMPANY G. Kienast, H. Zeiner, H. Mayer (JRS)

INTERNAL REVIEWER

WORKPACKAGE / TASK Task 7.3

DOCUMENT HISTORY

| Release | Date | Reason of change | Status | Distribution |
|---|---|---|---|---|
| 0.1 | 2002-04-10 | Document created | Draft | Confidential |
| 0.2 | 2002-04-11 | Structure and initial input | Draft | Confidential |
| 1.0 | 2002-05-10 | Document finished | Final | Confidential |

# Table of Contents

# 1 Introduction

One of the objectives of the PRESTO project is to develop an end-to-end chain of technologies in audio-visual content archiving for indexing, search & retrieval and exchanging of digital audio-visual content. The targeted application domain for this chain of technologies is the current public audio-visual archiving industry (public broadcasters) and its professional end users.

In their current situation this industry faces the problem of not being able to integrate or to access the available catalogues with the demand for required content access or the exchange of relevant cultural heritage content in public audio-visual archiving industry. In order to bridge the gap between public content providers and professional end users standard access interfaces will be offered to enhance the access of relevant cultural heritage. As most archives are funded by public money, there will be a web interface for the broader public to access the archive holdings.

The problem obviously cannot be solved by tackling only one single technological aspect of the chain. The solutions require an impact on the complete workflow from content provision to content access and retrieval and are built on open standards (e.g. XML) to allow easy integration into up-to-date middle-ware solutions which are used currently in the content providing archives.

### Scope of this Document

This document describes the technical aspects of the Broadcast OPAC developed within PRESTO, but also contains a user's guide and administration and configuration instructions of the Broadcast OPAC.

### Related Documents

In Deliverable 3.2 – "Key Link System Specification" the Broadcast OPAC technology is described and specified.

# 2 System Architecture

## 2.1 Overview

Figure 1 provides an overview of the archive framework in which the Broadcast OPACs can be used.

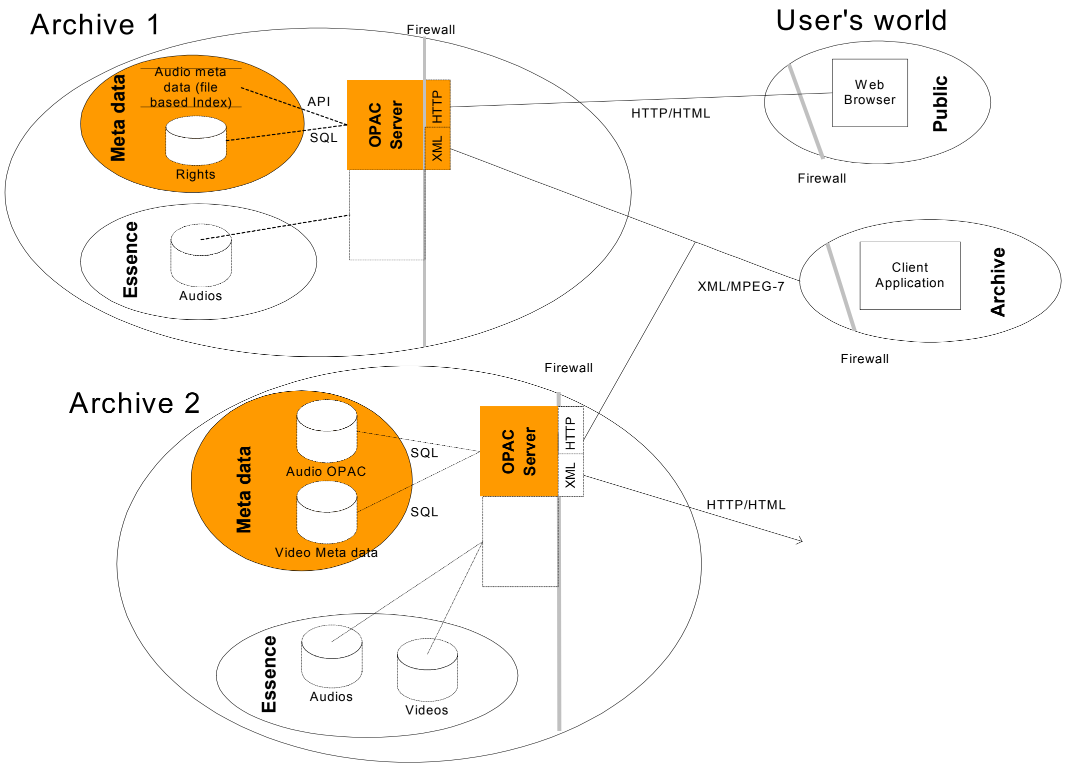


**Figure 1:** Overview of the system architecture

The basic requirements are that distributed archives (e.g. Archive 1 and Archive 2) provide a standard access interface to the archive via the Internet. In the archive different audio-visual content and descriptions are available. This can be audio, video, images and text.

The user may be outside of the professional archive (typical professional end user who works in the content production community). The goal of the end user is to search the interesting cultural heritage content in the archive in a simple way. This typically is done by a web-based interface by using a web browser.

Another type of end users are the archives themselves. The aim of the archives is to exchange the available content for reducing the production costs in the archives. In such a case the end user application is an application which should communicate via a standard interface to the archive. Moreover, technical limitations like firewalls are considered in this architecture, as well. The technical solutions have to be flexible and open to changes on the requirements of the search and retrieval clients, e.g. that the search attributes are be easily configurable. For the definition of the interface the XML standard is used.

## 2.2 Software Architecture

The implemented solution consists of three different layers, which are described below and are depicted in Figure 2.

- The **Access Layer:** This layer contains the public interface including the search & retrieval and browsing facilities. This comprises a web interface and a XML/SOAP interface
- The **Aggregation Layer** contains the core of the Broadcast OPAC, which includes facilities for information integration, administration, user registration, interfaces to the information resource, etc.
- The **Data Layer** contains the stored metadata. It provides the interface to the internal OPAC database as well as existing data sources and legacy databases.

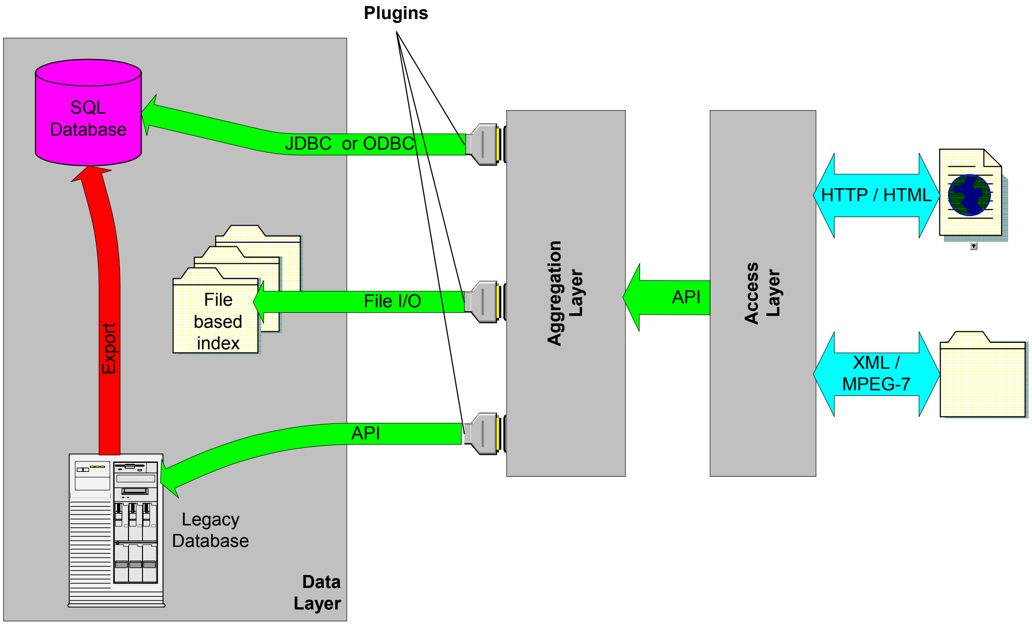


**Figure 2:** Software Layers in the Broadcast OPAC

In general the implemented solution is platform independent, however during development and testing phase only Windows 2000 and Linux were used. In the following table the supported and tested runtime environments are listed

## 2.3 Metadata

A main problem when integrating catalogue data from multiple archives is the different annotation methods in the individual archives. To overcome this problem in the implemented system the individual catalogue data has been mapped to a common subset. This subset is based on work of the Dublin Core metadata group.

Table 1 describes the Broadcast OPAC's metadata scheme and the list of searchable attributes.

| Name & Qualifiers | Definition | Explanation | Req. | Search-able | Example |
|---|---|---|---|---|---|
| **Title**<br>Title.Alternative | The name given to a resource | | Yes | Yes | Jurassic Park |
| **Creator** | An entity primarily responsible for the content | e.g. person, organisation | Yes | Yes | |
| **Subject** | Subject and Keywords | | | Yes | Dinosaurs |
| **Description** | | e.g. abstract, table of contents | | Yes (full text) | |
| **Publisher** | Entity responsible making the resource available | e.g. person, organisation | Yes | Yes | Universal Pictures |
| **Contributor**<br>Contributor.Role | | e.g. person, organisation | Yes | Yes | Director: Steven Spielberg, Actor: Sam Neill,... |
| **Date**<br>Date.Issued<br>Date.Created<br>Date.Modified | The date associated with an item | | Yes | Yes | 1993-06-29 |
| **Type** | Type of the resource | | Yes | | Movie |
| **Format**<br>Format.Extent<br>Format.Medium | The physical or digital format of the item | | Yes | | 35mm print |
| **Identifier** | A key that uniquely identifies the item (String or numeric value) | e.g. URI, UMID | Yes | | |
| **Source** | Reference to | | | Yes | |

| Name & Qualifiers | Definition | Explanation | Req. | Search-able | Example |
|---|---|---|---|---|---|
| | resource of which this item is (partly) derived | | | | |
| **Language** | Language of the item | 3 letter ISO-code | Yes | | ENG, GER |
| **Relation**<br>Relation.IsPartOf<br>Relation.IsVersionOf<br>Relation.Replaces<br>Relation.Requires<br>Relation.HasPart | Reference to related items | e.g. URI, UMID | | | |
| **Coverage** | | | | | |
| **Rights** | Information about rights held on the item | | Yes | | |

**Table 1:** Metadata scheme and search attributed (based on work by Scandinavian Audiovisual Metadata Group)

## 2.4 Access Layer

Collections are accessed either though a standard Web browser or a SOAP compatible client. The interface offers search facilities. Occasional users are provided a very simple query interface (e.g. a single line for full text) while expert or professional users need to make more sophisticated search queries, e.g. an advanced search form which includes attribute search (if supported by the media types). So the system supports both user groups by providing two different kinds of query interfaces.

### Type

For the user interface two modules have been implemented: An HTML/HTTP interface for access via a web client and an XML/SOAP interface for use with a dedicated client application.

### Function

In the following subsections the search user interface, the presentation user interface and the browse user interface are described.

### Search User Interface

Typical searches performed by a user can be divided into the following two groups:

- Full text searches like "Find everything related to the *PRESTO project*". The user only specifies the search terms "PRESTO" and "project" and the system scans all textual annotations in the database where these terms occur and reports all items that match the condition(s).
- Attribute searches like "Find all *documentaries* on *Bill Clinton* aired in *2000*". The user fills out a search form where he specifies a query like "genre == documentary" AND "subject == Bill Clinton" AND "date == 2000". This allows the user to specify a more detailed query than the first case

The user interfaces for simple searches contains one text field for entering the search terms.

For searches the user interface provides several elements:

- Areas for sub queries containing attribute fields, text fields
- Operators for combining sub queries. The default operator is "AND".
- A range selector for the number of returned results

**Attribute fields**

The attribute fields allow identifying the attributes to be searched. This can be done either in form of list boxes or in a hard wired way. The full text search can be either one "special" attribute or an additional field depending on the configuration of the search pages.

**Entering search terms**

Text fields are used for entering the search terms or a combination of terms. The simple search is attached to the full text search and in the advanced search each text field is attached to exactly one attribute field. Date related information can be handled in two text fields used for the beginning and the end of a period to be used in a search.

**Search grammar and syntax**

Within the text fields of the search user interface not only single words can be entered but this can be also structured according to a search grammar. Some rules apply to entered search terms respectively combinations of such terms

- Queries are automatically handled as "AND" queries (i.e. all terms must be found in one record to have that record included in the result). So no "AND" has to be included in a query.
- Search terms are not handled case sensitive (e.g. "video", "Video", "VIDEO" will return the same results).

Refining queries can be done by adding a new term means restricting query results to another term.Combination of queries

## 2.4.1 Interfaces

Two different types of interfaces to the PRESTO Broadcast OPAC are provided, namely an HTML/HTTP interface and an XML/SOAP interface.

The public access to broadcast archives could be seen as a special web service for searching, retrieval and access to meta data descriptions available in broadcast archives. For the purposes of this specification, the "public accessible interface" is a web service that performs a specific task, and conforms to a specific set of technical specifications which make it interoperable with compatible components. The key architectural principles are the service-oriented architecture as described in this section.

The PRESTO Broadcast OPAC design defines a clear publicly available interface to the archive data by using open standards.

- **HTML/HTTP:** This is a simple web interface to the OPAC database. It includes search pages which allow the user to specify queries by selecting search attributes and search terms, but also enable him to launch full text queries. Formatted search results are presented to the user.
- **XML:** Search results are provided in a standardised way by delivering XML-encoded data records (see [3] and [5]). The application of this standard enables system-to-system communication and can be used by rich client applications.
- **SOAP:** The transport protocol for exchanging messages is the RPC-style message protocol SOAP (see [11]). Existing RPC-style protocols such as DCOM and IIOP (CORBA) have not proven to be adaptable to the Internet. Both of these protocols require a non-trivial amount of dedicated runtime support in order to implement the complete set of services that both protocols have to offer. Finally, the existing Internet security infrastructure has embraced HTTP to the point that trying to communicate across organizations using anything else than HTTP requires an excessive amount of organizational and engineering resources. To support rich application clients which use the functionality of the OPAC framework in their applications they need the possibility to access the framework via XML/HTTP. One of the major advantages is that firewall systems do not require reconfigurations. Advanced features for automatic search and retrieval, download request processing can be integrated in several client tools.

### HTML/HTTP Interface

A simple web interface to the OPAC database is available. This web interface provides search pages which allow the user to specify queries through choosing search attributes and typing search terms. The user is also enabled to perform full text queries. Search results are presented to the user in a formatted way.

### XML & SOAP Interface

In this interface search results are provided in a standardised way by delivering XML-encoded data records. The application of these standards enables system-to-system communication and can be used by rich client applications.

The transport protocol for message exchange is the Remote Procedure Call (RPC)-style message protocol SOAP. SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. SOAP can be used in combination with a variety of protocols; however, the only bindings used in the Presto project is SOAP in combination with HTTP and HTTP Extension Framework.

SOAP consists of three parts:

- The SOAP envelope construct defines an overall framework for expressing the content of the message (e.g. definition of the search query).
- The SOAP encoding rules defines a serialization mechanism that can be used to exchange instances of application-defined data types.
- The SOAP RPC representation defines a convention that can be used to represent remote procedure calls and responses.

One advantage of SOAP is its text-based protocol nature by using XML. For example, it is easier to debug applications based on SOAP because it is much easier to read XML than a binary stream. The body of the request is in XML. A procedure executes on the server and the value it returns is also formatted in XML. The binary streams refer to the remote procedure call itself, not to the results set which can contain binary streams. Procedure parameters and returned values can be scalars, numbers, strings, dates, etc.; and can also be complex record, list structures and MIME types like e.g. images and Internet enabled video streams.

## 2.4.2 Design Details

**Data Flow**

Figure 3 shows the main parts of the Access Layer and the data flow between these parts. Queries can be submitted in two different formats (HTTP/HTML request ⇔ XML encoded requests). For every different query format there is a listener module which accepts incoming queries and passes it to a translator module. This module parses and translates the incoming queries into an internal data format which is a tree like data structure as outlined in Figure 4.

An instance of this data structure is then passed to the aggregation layer.

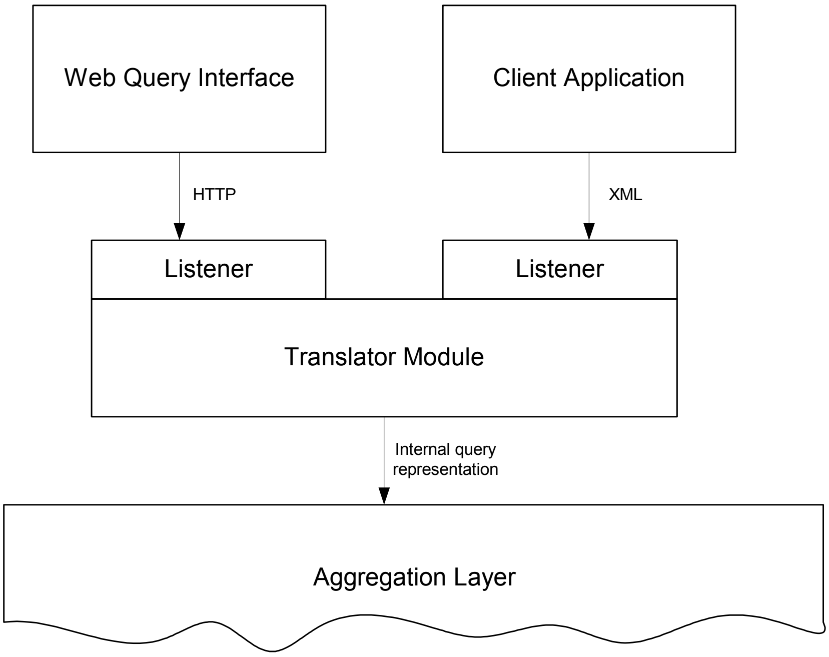


**Figure 3:** Data flow in the Access Layer of the Broadcast OPAC

### Translator module

The main task of the translator module is to parse the incoming queries to create the internal query data structure. Parsing the incoming query highly depends on the its format.

- A request from a web site is basically a list of parameters URL-encoded into a string like:
  operator=AND&attr1=genre&value1=Documentary&attr1=keyword&value1=Bush.
- A request from the client application is a well formed XML record, which can be analysed by any standard XML parser.

For every query format a dedicated translation routine has been implemented

### Internal Query Representation

A query like "Find all *documentaries* about one of the year 2000 presidential candidates *Al Gore* or *George W. Bush*" is internally represented as a tree like specified in Figure 4.

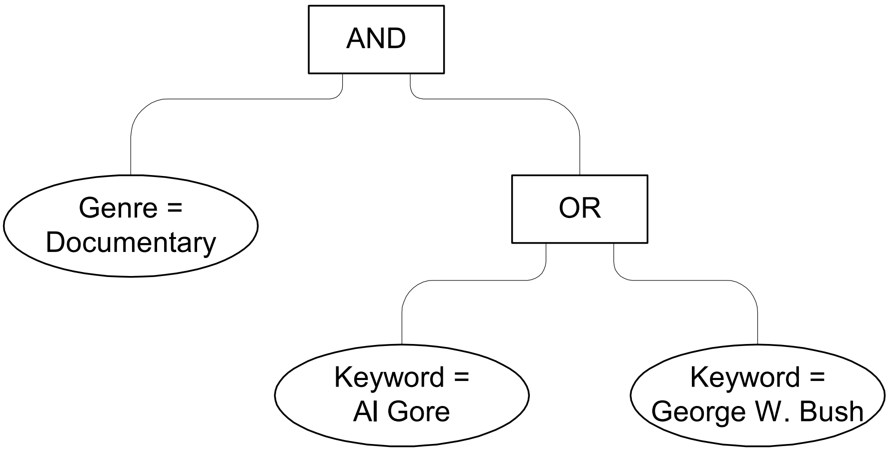


**Figure 4:** Internal query representation

The following classes can be composed to a query tree:

- **Description of a query:** This object describes all query parameters. It holds a *tree of the query* which specifies the query itself as well as a list of data sources on which the search should be performed.
- **Tree of a query:** This is an abstract base class for the classes *query value* and the *query operator*.
- **Query value :** Holds an (attribute, value)-pair. This object "terminates" a branch of the tree and has no further "sons". *Query value* objects are depicted as ellipses in Figure 4. It also holds a relation value (like EQUALS, LESS THAN, GREATER THAN) which is only valid for certain attributes (e.g. dates)
- **Query operator:** Specifies an operator (AND, OR, AND NOT) and a list of (two or more) subsequent *tree of a query* objects (which are again instances of qu*ery operator* or query tree*)*. Figure 4 shows instances of *query operator* as rectangles.

## 2.5 Aggregation Layer

The Aggregation Layer of the Broadcast OPAC consists of several different modules, namely:

- Translator module (query translation, extension and gather the result set)
- Administration modules
- Complementary features including the session manager, user registration, logging facilities etc.
- Multiple language module

The Broadcast OPAC is a lightweight middle-ware software programme (using JAVA technology and running within an application server)

### Function

The main functionality of the Aggregation Layer is the transformation of incoming query requests into valid data storage requests (e.g. a dedicated relational database or some legacy database system). After execution of the database queries and retrieval of results from the database the found results are converted into a form which can be returned to the requesting party. This means they are either XML-encoded or formatted as HTML for output via the web interface.

### Translator module

The translator module receives the search request from the access layer. The translator module extends the search request and translates it into the format of the underlying data resources (e.g. an SQL query), distributes it to the various data resources and delivers the results from the search targets to the access layer later on.

### Configuration assistance

The configuration database assists in two tasks:

- Translation of the query elements and entered terms from the user interface into the XML query presentation
- Translation of returned XML formatted results into a presentable format for the user interface

### Internal query representation

The internal query requests contains:

- identifiers for the databases to be queried (in case that more than one database exists)
- binary tree structures using operators like "AND", "OR" and "AND NOT" in the nodes of the tree and the operands (query terms) in the leafs of the tree
- operands being combinations of attributes and search terms

- number of records to be returned in the direct response to a search
- sort criteria (attribute to be used for sorting of results)

#### Record representation in XML

The XML record representation contains the elements as described in the section about search attributes. Additional elements may be added to that set of search attributes as needed by the archives. The software has been designed to allow a configurable set of metadata attributes and takes care of these configuration aspects.

#### Administration module

The application server is equipped with an administration possibility. This allows a systems administrator to configure and monitor the execution of the application server.

The Broadcast OPAC needs configuration data. This data is stored in a configuration file which is an XML-file.

This database contains following data:

- Data sources connected to the system,
- Information about these data sources (e.g. address, port number, database name)
- Search attributes and mapping rules
- Description of search interface (list and number of searchable attributes)

Facilitating the access and exchange of information between two audio-visual archives applications is a lot like translating between two people speaking different languages. In a complex environment like an audio-visual archive network, however, such differences must be resolved in a way that enables all participants to take part in the communication process. This is far more difficult challenge. However, the OPAC framework supports the user with a clear transformation tool to resolve the differences between the data models or the import/export formats.

### 2.5.1 Aggregation Layer Complementary Features

These modules are a set of functionalities which are additional value to the core search and retrieval functionality and guarantee a user-friendly framework. Each element of this framework is described in the following subsections.

#### User registration / management

The access to the PRESTO facilities can be restricted by a user registration and login procedure (for a user’s name and password). No user should be allowed to use the system without going through this authorization procedure. The registration module supports the definition of individuals or groups.

### Session management

The Broadcast OPAC stores temporary data like user session identifiers to keep track of the state of a connection. The session identifier is needed to identify the user who submitted a request. Because more than one request (from different users) may be sent to the Broadcast OPAC concurrently the results of each request have to be managed to the corresponding user session.

### Logging

Following logging services are possible:

- session tracking
- session logging of the underlying database access and index access
- query and result set
- performance and event logs

### Multilingual support

Within the Broadcast OPAC there are several areas which requires multilingual requirements.

- Thesaurus support
- Dates / times standards
- Character sets – Unicode support
- Multilingual web interface (search forms and presentations of results)

### Multilingual thesaurus:

A multilingual thesaurus support for query expansion with controlled vocabulary would be useful. In fact, within the PRESTO Broadcast archive no multilingual thesaurus was developed, but in a related EU-project AMICITIA [12] such a multilingual thesaurus tool was be developed by JOANNEUM RESEARCH and integrated into the Broadcast OPAC for evaluation and test purposes.

### Dates / times standard:

A standard for dates/times (ISO 8601) was used within the aggregation of the search results of several media archives.

### Character sets – Unicode support :

**Unicode (namely UTF-8)** is be used throughout the software, allowing different languages to be processed in a consistent manner. Several languages are be supported.

The Unicode standard is the universal character encoding standard used for representation of text for computer processing. Unicode provides a consistent way of encoding multilingual plain text and brings order to a chaotic state of affairs that has made it difficult to exchange text files internationally. Computer users who deal with multilingual texts (e.g. business people, archivists, linguists, researchers, scientists) will find that the Unicode standard greatly simplifies their work [6].

Unicode uses 2 bytes for each character. Therefore 65536 characters are possible (only 40000 are used in practice, see also Unicode web pages). The first 256 characters of Unicode are the same as in the ISO Latin-1 standard. This makes the parsing process of Unicode data simple. The first two bytes of a file are the first character. The next two bytes are the second character, and so on.

Within the Broadcast OPAC the character encoding standards based on UTF-8 is used. It follows a list of the Unicode support in the Broadcast OPAC software including browsers, protocols, databases and programming languages.

For improved search functionality on the server side the use of simple translation tables for frequently used terms is possible. These tables allow translation between common languages. As an alternative the set-up of a (multilingual) thesaurus may assist the search processes. The creation of translation tables or appropriate thesaurus structures is a task for the end user partners of the consortium or eventually the BAUG members due to their detailed knowledge of the specific domain.

The creation of properly working translation lists or structures is a very complex task and out of scope within the time frame of this project. Therefore the implementation and integration of such a component was not given very high priority at the moment and so it might not be included in the final software version within the project.

Nevertheless the attention of the development team was also turned to this topic during the design phase. This will allows an easier integration at a later stage (e.g. in follow up activities to the project). In a related R&D project (Amicitia) a thesaurus is currently under development. It is planned to integrate this thesaurus module into the Broadcast OPAC when available.

### Scalability

The application is scalable regarding two major aspects:

- Size of user group: The Broadcast OPAC supports a very large number of concurrent users accessing the database. This is achieved by using the scalability of the JRun application server, which handles load balancing issues. The application itself does not have to deal with it.
- Amount of accessible data: The data repositories can support very large amounts of data. It is essential in the development framework that the underlying background systems (indexing, multi-media database etc.) support the scalability of the solutions.

### Open Interfaces

The Broadcast OPAC’s Aggregation Layer has an internal interface to the access layer and includes several wrapper interfaces which can be added to the software via plug-in techniques.

The following interfaces have been implemented:

- SQL databases accessed via JDBC
- New Zealand digital library software [4]

- For testing purposes: other web based query systems

## 2.6 Data layer

Within the data layer the meta data are stored. There are several possibilities to store and access the meta data:

*Method 1:* Access to a copy of the metadata of the production system: The data is exported of the legacy (relational) database and the Broadcast OPAC directly accesses this database. This could be a possible solution, if the database schema of the legacy database is well designed to support fast retrieval of the data records.

*Method 2:* Access to a copy of the metadata which transform the used database schema to a very fast search & retrieval enabled database schema. This method has been implemented using the metadata schema described in Table 1.

*Method 3:* Another possibility supported within the Broadcast OPAC is to access the content in a file based index (e.g. New Zealand Library via its web interface).

*Method 4:* Direct connection to production/legacy database. This has not been implemented for the prototype developed within PRESTO.

Within PRESTO method 2 and method 4 have been implemented for the demonstrator system.

### Type

The data layer is a database schema adapted and enhanced within the PRESTO project or uses already available database schemas.

When method 2 (Access to a transformed copy of the metadata) is used, scripts are provided which transform the available metadata into a new database schema which is used by the Broadcast OPAC. See data import (Chapter 4)

### Function

The software modules in this layer are based on databases or other retrieval software with a well defined client interface.

### Interfaces

There are two possible interfaces:

- The interfaces to the database are provided via JDBC / ODBC drivers.
- The underlying retrieval software provides a well defined API which can be used by the adapters.

# 2.7 Hardware and Software Requirements

## 2.7.1 Hardware Requirements

**Server hardware requirements**

The Hardware Requirements depend on the number of simultaneous users, the number of page impressions and the generated traffic.

- Minimal Installation: 1 simultaneous Users, very low traffic PII 750 Mhz, 256 MB RAM, approx. 100 MB free disk space
- Normal Installation: less than 10 simultaneous Users, less than 120 MB daily traffic 2 Double-PIII, each with: 1 Ghz, 1 GB RAM, SCSI HD
- High End Installation: more than 10 simultaneous Users, above 120 MB daily traffic 2+ Multiple-PIII, each with: 1 Ghz, 4 GB RAM, SCSI HD

**Client hardware requirements**

For the client software there a no specific hardware requirements. A PC with a web browser installed suffices to access the Broadcast OPAC's web interface.

## 2.7.2 Software Requirements

In general the implemented solution is platform independent, however during development and testing phase only Windows 2000 and Linux were used. In the following table the tested runtime environments is described.

| Environment | Description |
|---|---|
| Run time | • **Operation System:**<br>- Linux Kernel 2.2<br>- Microsoft Windows 2000<br>• **Web Server:**<br>- Apache 1.3.x or higher<br>- MS Internet Information Server 4.x or higher<br>• **Application Server**<br>- Allaire JRun with JDK 1.3.1<br>• **Middle-ware tools:**<br>SOAP Toolkit: WASP Lite 3.1<br>• **Database:**<br>- Oracle with XML-Extension |

**Table 2:** Run time environment used during development

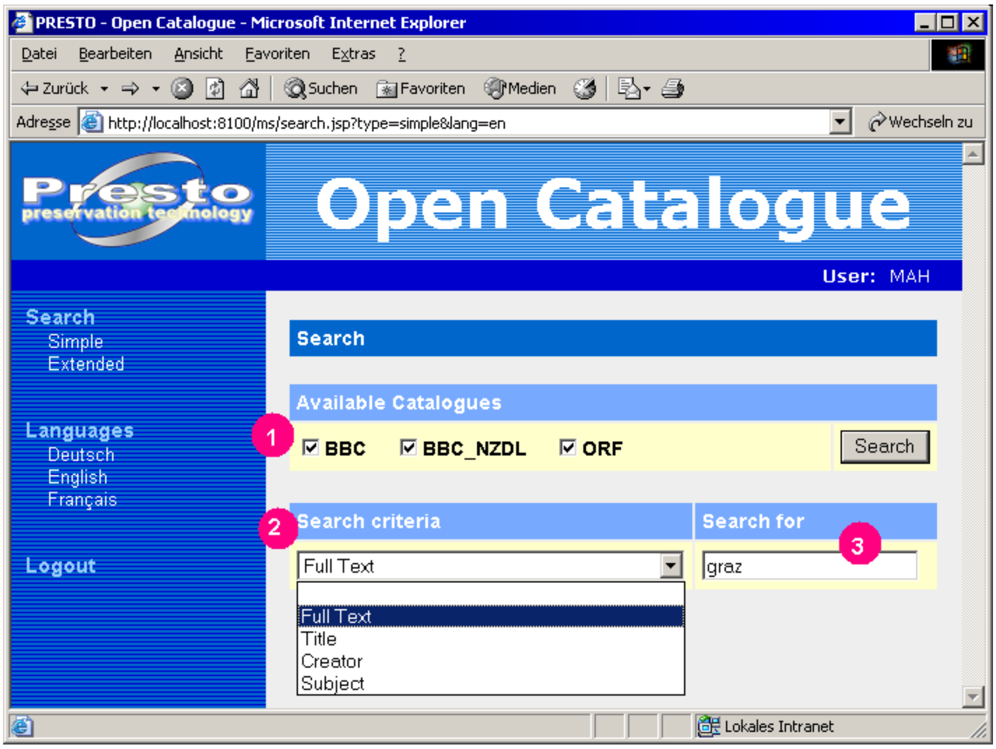
# 3 User's Guide

This section is intended to support the users of the Broadcast OPAC with using the demonstrator system. For a detailed description of how to install and maintain the system see chapter 4.

The functionality covered in this short overview is:

- Launching searches and retrieving records
- Usage of the different search pages
- Multilinguality – switching between language versions
- Logging in and out of the system

## 3.1 Web Interface



Adresse http://localhost:8100/ms/search.jsp?type=simple&lang=en

User: MAH

☑ BBC ☑ BBC_NZDL ☑ ORF

Search criteria: Full Text

Search for: graz

**Figure 5:** The web interface to specify simple queries

Figure 5 shows the entry page to the search interface. This is simplest search interface to the catalogue. The screen is divided into three parts.

### Title bar (top)

There is no functionality here, only the PRESTO logo is a link to the PRESTO project homepage (http://presto.joanneum.at).

The title area also indicates whether a user is currently logged in to the system or not. In case a user is logged in, the username is shown.

### Menu and navigation (lower left)

This allows to select a different language for the web interface or to select a search page that allows to specify more complicated queries.

- Change search pages: There can be a various number of different search pages. In the demonstrator system two different kinds of search pages are available. A simple page (the page shown above) and a more advanced page (Figure 6) allowing to composes more complicated queries.
- Change language: The web interface supports different language versions which can be selected by clicking on the desired language name.
- Login / logout: When no user is logged in a login button is shown here which allows the user to identify with username and password. After login the button is replaced with a log-out link.

### Query interface (lower right)

The query interface consists of three input areas for specifying a query (the numbers are references to Figure 5)

- ❶ Selection of catalogues: These are the checkboxes next to the catalogues' acronyms. The search is performed only on those catalogues where the boxes are checked.
- ❷ Selection of search attributes: The user can select different attributes for his search. The simplest way to search is to select "full text", in that case the search is performed on all attributes. To narrow the search down other attributes can be selected. The number of available attributes depends on the configuration and the type of search page that is active.
- ❸ Text field for entering search terms: Here the actual search term has to be entered.

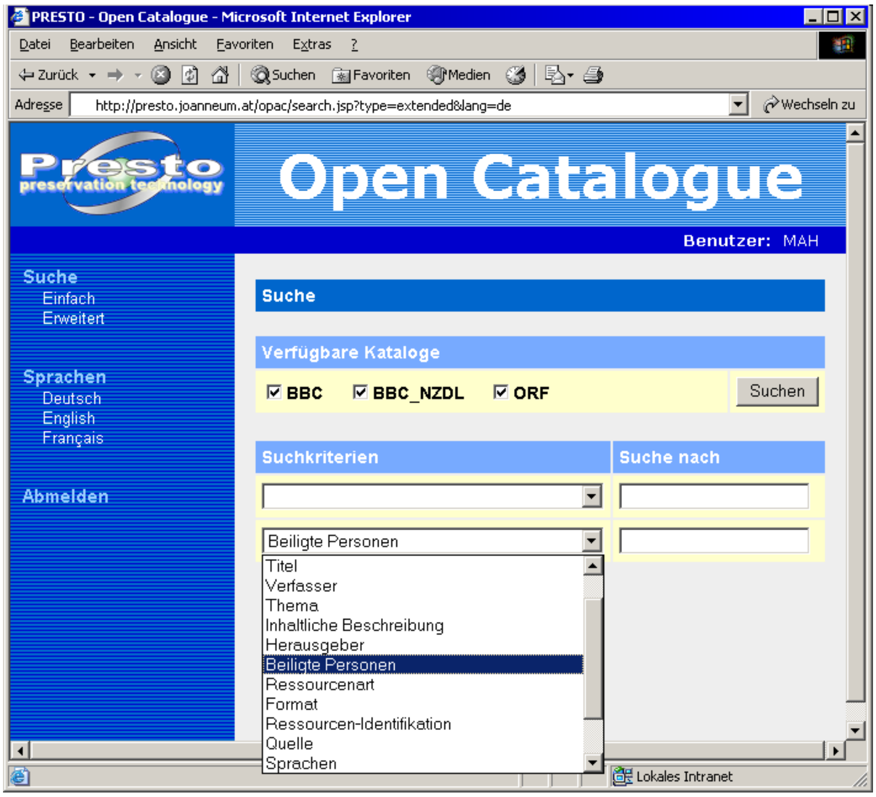
At least one catalogue has to be selected and a pair of search attribute and term has to be entered to successfully launch a query. If not an error page will be displayed.

In the example shown in Figure 5 a full-text query in all three available catalogues will be performed. The search term is the word "Graz" (a town in Austria).

### Automatic query translation

For certain common search terms multilingual term lists are used for automatic query translation.

The catalogue data itself is in the archive's original language. Hence retrieved result records are displayed in the original language and not in the language selected for the user interface.



Adresse http://presto.joanneum.at/opac/search.jsp?type=extended&lang=de

Benutzer: MAH

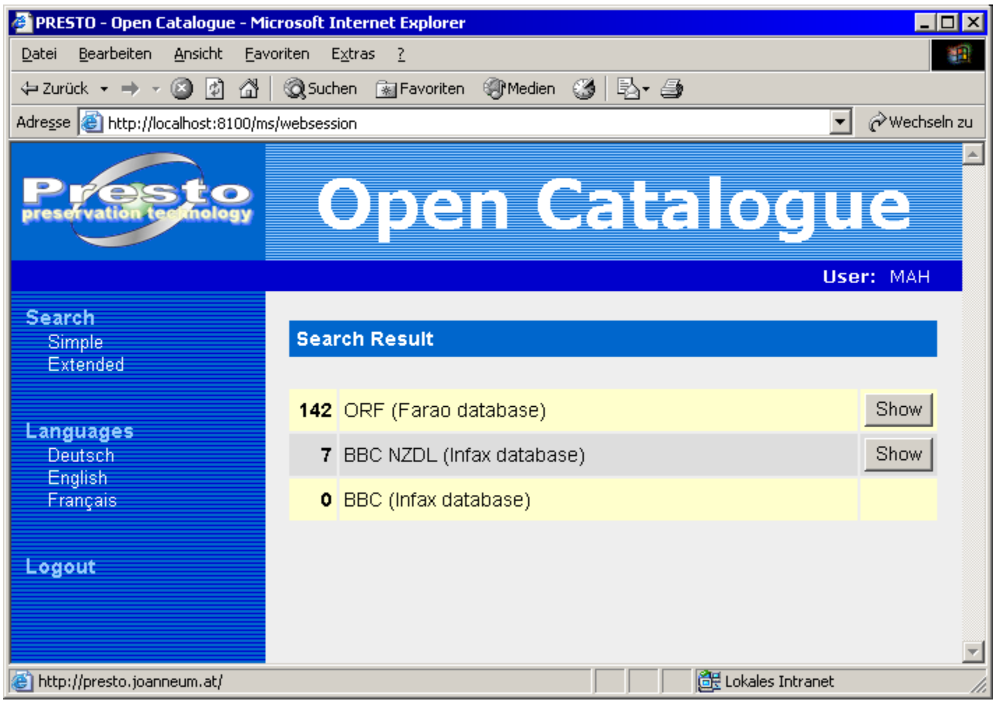
☑ BBC ☑ BBC_NZDL ☑ ORF

Beiligte Personen

**Figure 6:** The German version of a more advanced search page

Figure 6 shows the more advanced search page. More attributes are available and also the language has been changed to German.

In the demonstrator system three languages have been provided: English, German and French.


PRESTO - Open Catalogue - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten Medien
Wechseln zu
Lokales Intranet
Presto preservation technology
Open Catalogue
Search
Simple
Extended
Languages
Deutsch
English
Français
Logout
Search Result

Adresse http://localhost:8100/ms/websession

User: MAH

| | | |
|---|---|---|
| 142 | ORF (Farao database) | Show |
| 7 | BBC NZDL (Infax database) | Show |
| 0 | BBC (Infax database) | |

http://presto.joanneum.at/

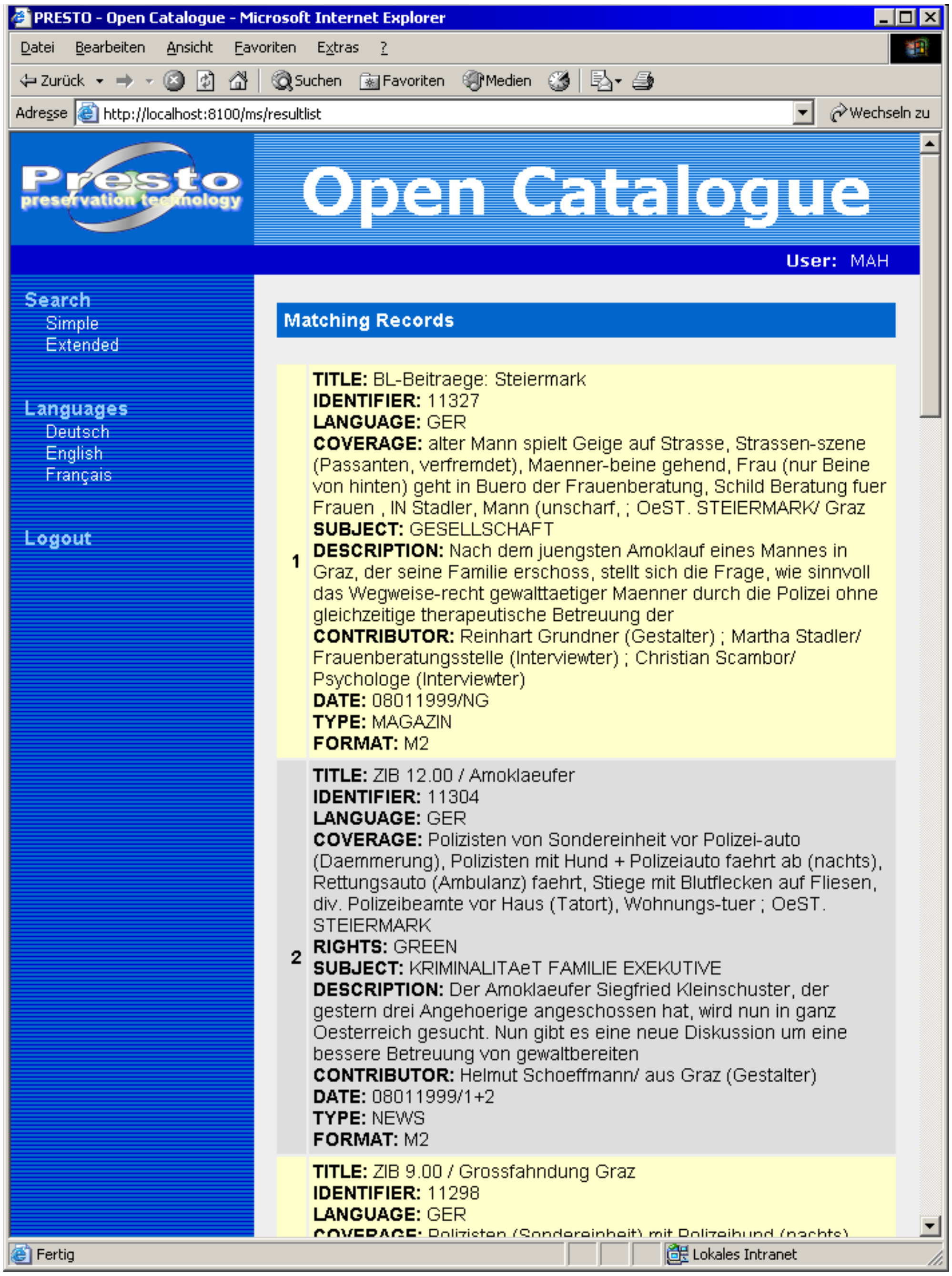
**Figure 7:** First result page that gives an overview on the number of results

found in the individual catalogues

For every catalogue that was selected on the search page the number of matching records is shown.

If the number of results for a catalogue is greater than zero a button for displaying the matching records is displayed. By clicking one of these buttons the user can view all records matching the query.

After clicking the "Show" button a list of up to 10 (this value is a configurable default) records is displayed (Figure 8). If there are more than 10 results a list of links to the next pages is displayed on the bottom of the page.


PRESTO - Open Catalogue - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten Medien
Adresse http://localhost:8100/ms/resultlist
Wechseln zu
Presto preservation technology
Open Catalogue
User: MAH
Search
Simple
Extended
Languages
Deutsch
English
Français
Logout
Fertig
Lokales Intranet

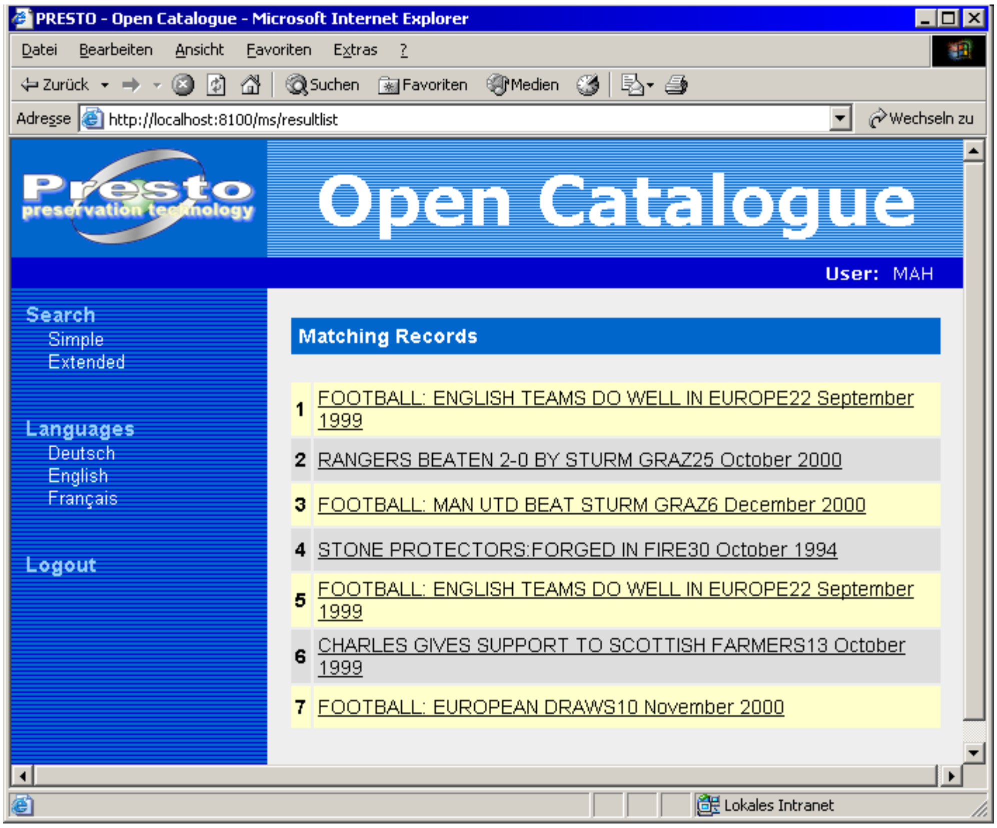
Matching Records

1
**TITLE:** BL-Beitraege: Steiermark
**IDENTIFIER:** 11327
**LANGUAGE:** GER
**COVERAGE:** alter Mann spielt Geige auf Strasse, Strassen-szene (Passanten, verfremdet), Maenner-beine gehend, Frau (nur Beine von hinten) geht in Buero der Frauenberatung, Schild Beratung fuer Frauen , IN Stadler, Mann (unscharf, ; OeST. STEIERMARK/ Graz
**SUBJECT:** GESELLSCHAFT
**DESCRIPTION:** Nach dem juengsten Amoklauf eines Mannes in Graz, der seine Familie erschoss, stellt sich die Frage, wie sinnvoll das Wegweise-recht gewalttaetiger Maenner durch die Polizei ohne gleichzeitige therapeutische Betreuung der
**CONTRIBUTOR:** Reinhart Grundner (Gestalter) ; Martha Stadler/ Frauenberatungsstelle (Interviewter) ; Christian Scambor/ Psychologe (Interviewter)
**DATE:** 08011999/NG
**TYPE:** MAGAZIN
**FORMAT:** M2

2
**TITLE:** ZIB 12.00 / Amoklaeufer
**IDENTIFIER:** 11304
**LANGUAGE:** GER
**COVERAGE:** Polizisten von Sondereinheit vor Polizei-auto (Daemmerung), Polizisten mit Hund + Polizeiauto faehrt ab (nachts), Rettungsauto (Ambulanz) faehrt, Stiege mit Blutflecken auf Fliesen, div. Polizeibeamte vor Haus (Tatort), Wohnungs-tuer ; OeST. STEIERMARK
**RIGHTS:** GREEN
**SUBJECT:** KRIMINALITAeT FAMILIE EXEKUTIVE
**DESCRIPTION:** Der Amoklaeufer Siegfried Kleinschuster, der gestern drei Angehoerige angeschossen hat, wird nun in ganz Oesterreich gesucht. Nun gibt es eine neue Diskussion um eine bessere Betreuung von gewaltbereiten
**CONTRIBUTOR:** Helmut Schoeffmann/ aus Graz (Gestalter)
**DATE:** 08011999/1+2
**TYPE:** NEWS
**FORMAT:** M2

**TITLE:** ZIB 9.00 / Grossfahndung Graz
**IDENTIFIER:** 11298
**LANGUAGE:** GER
**COVERAGE:** Polizisten (Sondereinheit) mit Polizeihund (nachts)

**Figure 8:** Overview on retrieved records (ORF catalogue)


PRESTO - Open Catalogue - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten Medien
Adresse http://localhost:8100/ms/resultlist
Wechseln zu
Presto preservation technology
Open Catalogue
User: MAH
Search
Simple
Extended
Languages
Deutsch
English
Français
Logout
Lokales Intranet

Matching Records

1 FOOTBALL: ENGLISH TEAMS DO WELL IN EUROPE22 September 1999
2 RANGERS BEATEN 2-0 BY STURM GRAZ25 October 2000
3 FOOTBALL: MAN UTD BEAT STURM GRAZ6 December 2000
4 STONE PROTECTORS:FORGED IN FIRE30 October 1994
5 FOOTBALL: ENGLISH TEAMS DO WELL IN EUROPE22 September 1999
6 CHARLES GIVES SUPPORT TO SCOTTISH FARMERS13 October 1999
7 FOOTBALL: EUROPEAN DRAWS10 November 2000

**Figure 9:** The records retrieved form an external data source (here BBC data is accessed via the Greenstone New Zealand Digital Library system). These records may contain HTML-links to resources outside the PRESTO open catalogue system.

# 4 Installation Guide

The installation guide gives a short overview about the installation of the PRESTO.

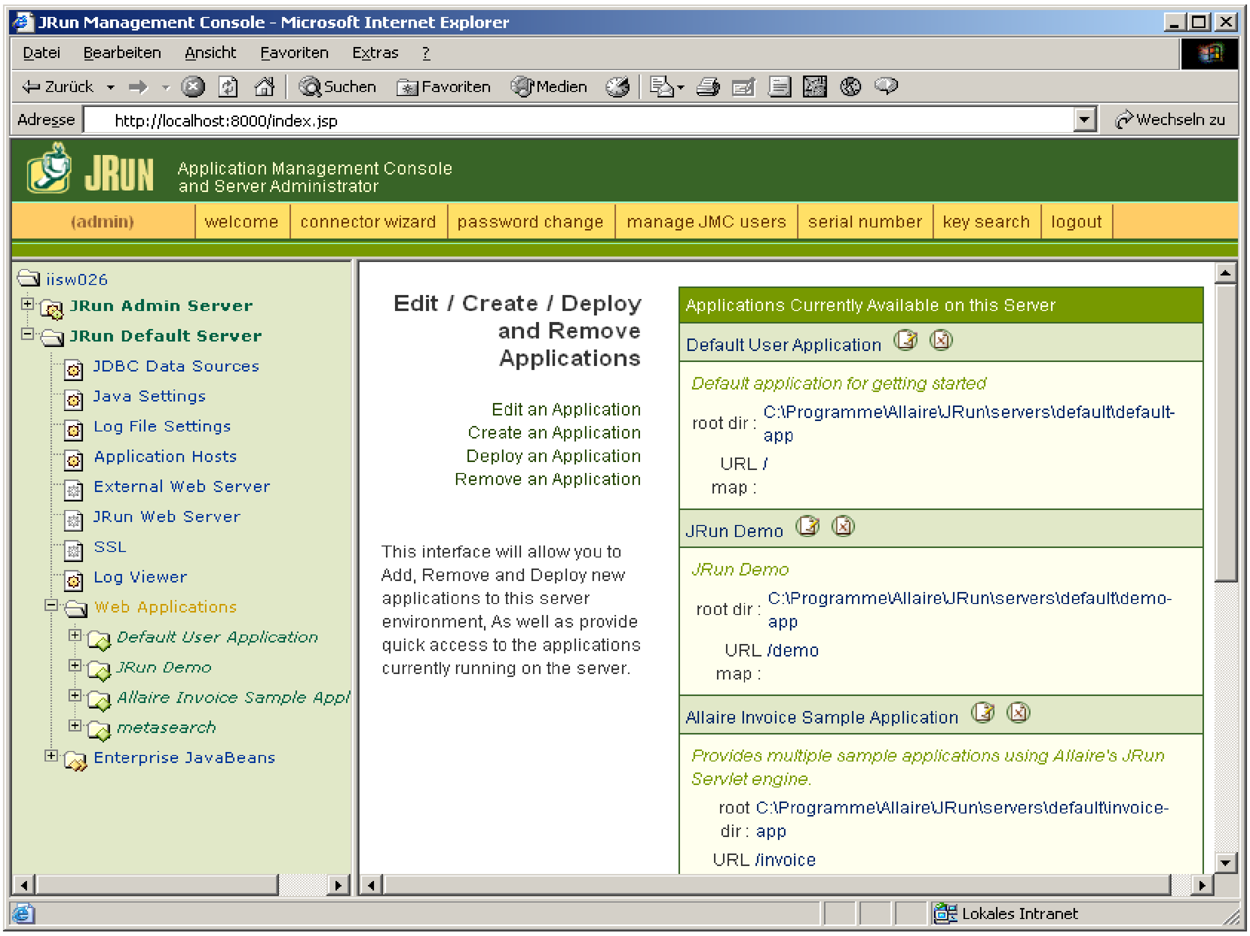
## 4.1 Introduction

This section is intended to support the administrator of an installation of the Broadcast OPAC with setting up the system.

The installation of the JRun application server itself is not covered here, please refer to Allaire's documentation of JRun. In this chapter it is assumed that JRun has been previously installed.

## 4.2 Broadcast OPAC Installation on JRun

The PRESTO OPAC can be installed with the JRUN Management Console (shown in Figure 10).



Edit / Create / Deploy and Remove Applications

Edit an Application
Create an Application
Deploy an Application
Remove an Application

This interface will allow you to Add, Remove and Deploy new applications to this server environment, As well as provide quick access to the applications currently running on the server.

Applications Currently Available on this Server

Default User Application

*Default application for getting started*

root dir : C:\Programme\Allaire\JRun\servers\default\default-app
URL /
map :

JRun Demo

*JRun Demo*

root dir : C:\Programme\Allaire\JRun\servers\default\demo-app
URL /demo
map :

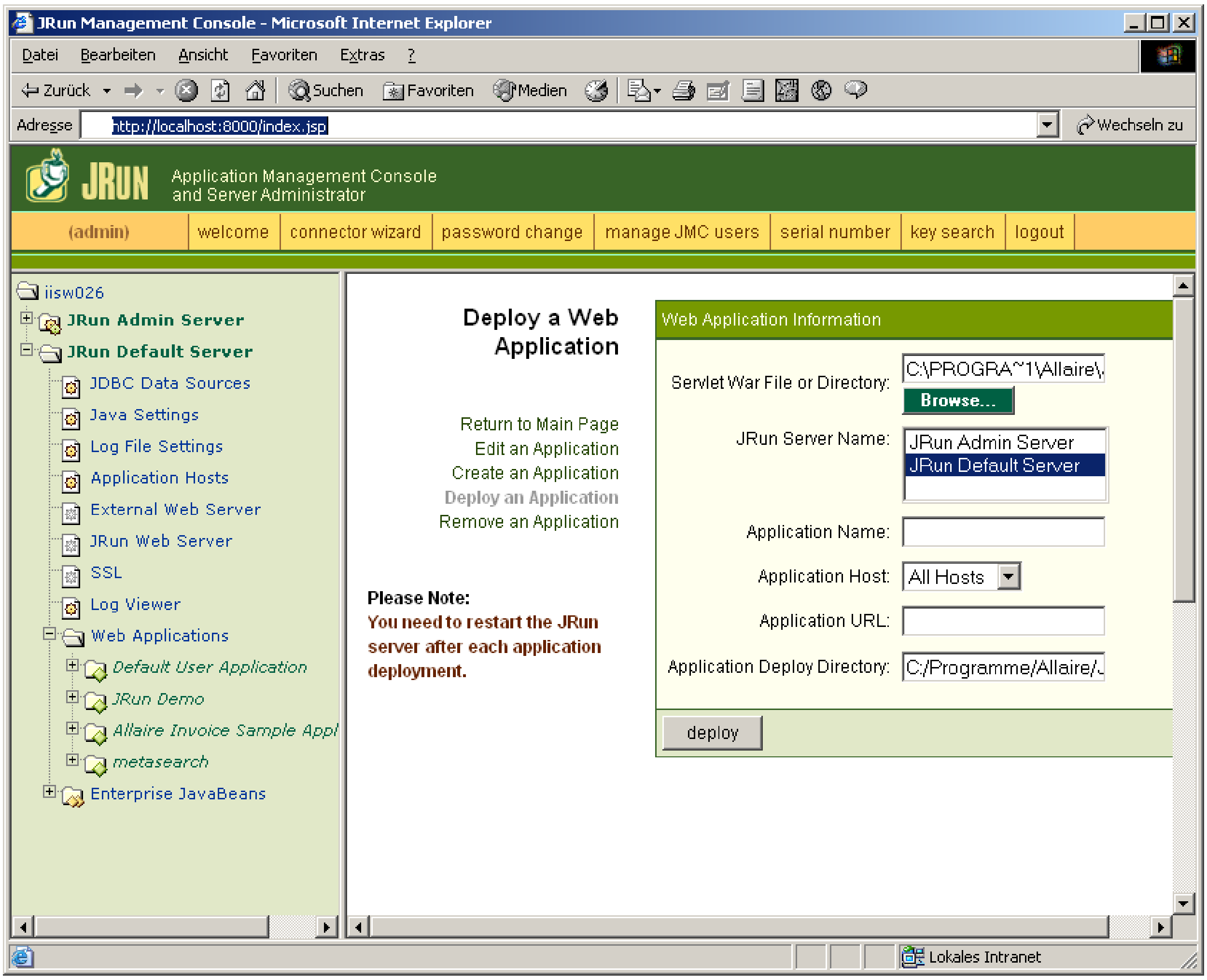
Allaire Invoice Sample Application

*Provides multiple sample applications using Allaire's JRun Servlet engine.*

root dir : C:\Programme\Allaire\JRun\servers\default\invoice-app
URL /invoice

**Figure 10:** JRUN Management Console

The application can be installed with the following steps by using the select "Deploy an Application" and go through the wizard provided by the JRUN management console shown in Figure 11.



Deploy a Web Application

Please Note:
You need to restart the JRun server after each application deployment.

Web Application Information

Servlet War File or Directory: C:\PROGRA~1\Allaire\

JRun Server Name: JRun Admin Server / JRun Default Server

Application Name:

Application Host: All Hosts

Application URL:

Application Deploy Directory: C:/Programme/Allaire/J

**Figure 11:** Deploy a web application

An application can be deployed by selecting the *PRESTO Opac.war* file, edit the *Application Name* e.g. presto, editing the *Application URL* e.g. /presto and clicking on the *deploy button*.

# 4.3 Broadcast OPAC Configuration

The PRESTO OPAC can be configured via several XML files. File structure and naming convention are:

**Configuration (directory)**

Metasearch.xml

**Searchengines (directory)**

Target1.xml

Target2.xml

The following general section contains the user languages, different search forms of the application.

**General Section**

```
<general>
   <name>Meta Search Engine</name>
   <version>1.0</version>
  <languages>
     <name xml:lang="en">English</name>
     <name xml:lang="de">Deutsch</name>
     <name xml:lang="fr">Français</name>
   </languages>
 </general>
<search>
  <forms>
    <form>
       <formid>simple</formid>
       <name xml:lang="de">Einfach</name>
       <name xml:lang="en">Simple</name>
       <name xml:lang="fr">Simple</name>
       <itemid>simpleFields</itemid>
     </form>
    <form>
       <formid>extended</formid>
       <name xml:lang="de">Erweitert</name>
       <name xml:lang="en">Extended</name>
       <name xml:lang="fr">Etendue</name>
       <itemid>simpleFields</itemid>
       <itemid>extendedFields</itemid>
     </form>
   </forms>
```

**Table 3:** General section of the configuration file

Each form contains several items. The items are defined in the search sections. An item can be e.g. a combo box of search attribute list as shown in the following section. A search attribute e.g. dc_title contains the different names in the required search languages as shown in .

**Search section (small part)**

```
<itemid>simpleFields</itemid>
       <fields>
         <field>
         <id>dc_text</id>
         <name xml:lang="en">Full Text</name>
         <name xml:lang="de">Volltext</name>
         <name xml:lang="fr">Texte intégral</name>
      </field>
    <field>
         <id>dc_title</id>
         <name xml:lang="en">Title</name>
         <name xml:lang="de">Titel</name>
         <name xml:lang="fr">Titre</name>
      </field>
    <field>
         <id>dc_creator</id>
         <name xml:lang="en">Creator</name>
         <name xml:lang="de">Verfasser</name>
         <name xml:lang="fr">Créateur</name>
      </field>
    <field>
         <id>dc_subject</id>
         <name xml:lang="en">Subject</name>
         <name xml:lang="de">Thema</name>
         <name xml:lang="fr">Suject</name>
      </field>
   </fields>
</item>
```

**Table 4:** Configuration - search attributes

The result list section (as shown in Table 5: Configuration – result list section):

- *number*: This parameter defines how many items of information will be shown on the results detail page. The default value is 10.
- *cachenumber*: defines how many items are requested from an OPAC after performing a search. T
- *pagenumber*: If there are more results available than fit the *Number*, a list of page numbers is displayed at the bottom of the results detail page. The default value is 10, this means that starting from the displayed page 10 numbers to the front and 10 to the back are displayed. If the user is on page 1, a list from 1 to 10 is displayed.

**Resultlist section (small part)**

```
<resultlist>
    <number>10</number>
    <cachnumber>20</cachnumber>
    <pagenumber>10</pagenumber>
  </resultlist>
```

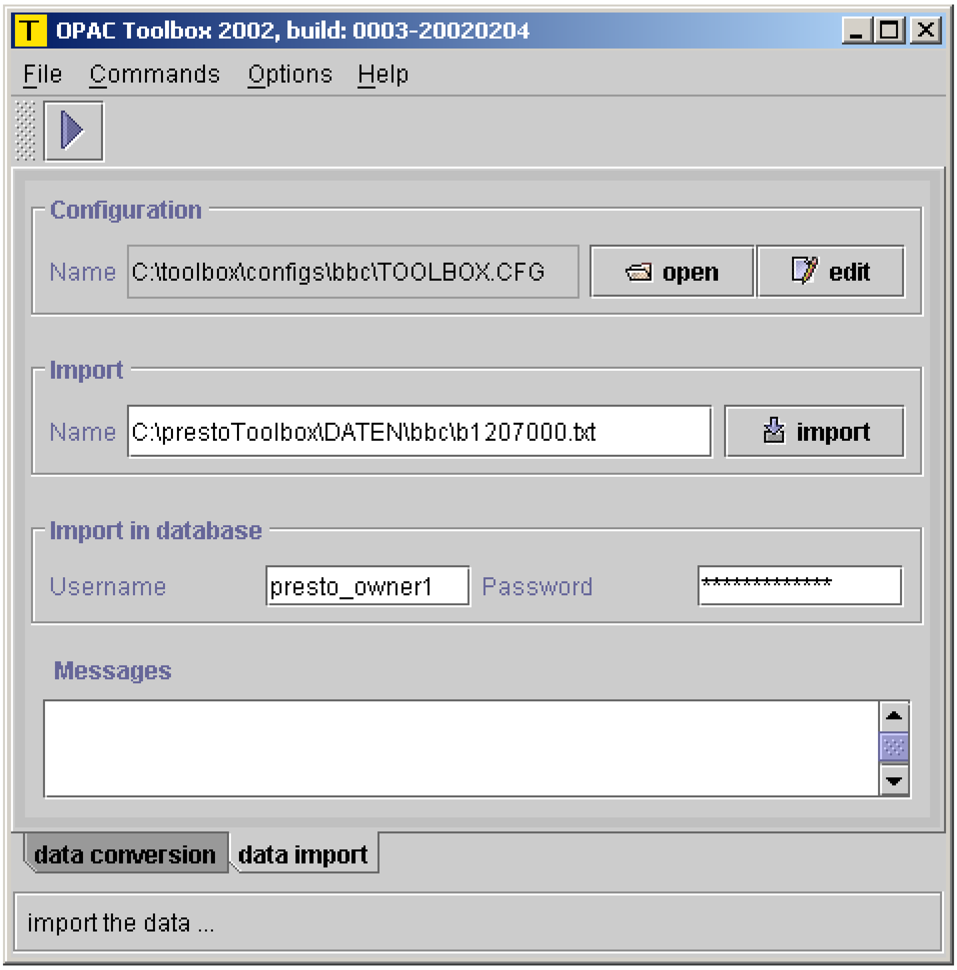
**Table 5:** Configuration – result list section

# 5 Data Import Tool

## 5.1 Description

This OPAC Toolbox 2002 utility is a tool to import data in the PRESTO OPAC. The mapping of the tag/value elements to Dublin Core can be configured externally.

The following screen shot shows the import tool.



Name C:\toolbox\configs\bbc\TOOLBOX.CFG

Name C:\prestoToolbox\DATEN\bbc\b1207000.txt

Username presto_owner1 Password *************

import the data ...

For the following description we suppose that you have installed the OPAC toolbox 2002 on a MS Windows machine in the default directory c:\toolbox. To import the data with the following steps has to be performed:

1. Load a project file.
2. Change in the directory of the data provider e.g. c:\toolbox\config\bbc and select the file c:\toolbox\config\bbc\toolbox.cfg.
3. Load the input file.

4. Change in the directory of the data provider e.g. c:\toolbox\data\bbc and select the file c:\toolbox\config\bbc\*data.txt*.
5. Type in the user name and the password of the database.
6. Click the start button in the toolbar or select it in the menu

Wait until the data has been important. After a few minutes you can see a progress bar which shows you the already imported data records. If you wish, you can click the cancel button an the conversion process will be stopped.

## 5.2 Data Sets Used During Development

**BBC data set**

This test data set contains around 1.2 million single data entries from the BBC Infax database. The data sets where delivered in two different versions.

1. These data were delivered in NZDL (Greenstone New Zealand Digital Library) format and were used to test access from the Broadcast OPAC to NZDL's web interface.
2. The data sets were delivered also as text files.

To evaluate it, 300 000 data records were imported in the PRESTO OPAC. During the evaluation this will be extended to all data records.

**ORF data set**

500.000 data sets. They were exported from the ORF internal ORFEUS data base.

The data were delivered as Microsoft Excel files. The data has been exported to plain text files (comma separated values). Excel can be used for the task. The exported files can be import in the PRESTO OPAC.

To evaluate it, 100 000 data records were imported in the PRESTO OPAC. During the evaluation this will be extended to all data records.

**INA data set**

Around 3.000 data sets from German/French television station ARTE can be used in the test scenario. The data set contains data records in French and German. Therefore, it is a very small, but extreme valuable scenario which shows the multilingualism impact of such repositories.

To evaluate it, all data records were imported in the PRESTO OPAC.

**Acknowledgements**

JOANNEUM RESEARCH wishes to express its explicit thanks to the three data providers for their efforts to support the development processing by making available an excerpt of their data.

# 6 Annex I: Implementation Details

## 6.1 Overview

The Broadcast OPAC was implemented in the JAVA programming language and is running within a JRun application server.

## 6.2 Class Documentation

During the development process source code documentation was consistently done in JAVAdoc format. From this added documentation source library documentation can be generated automatically which is included in this part of the annex.

### Package Structure

JAVA classes are grouped to packages. For each package a list of included classes is given with a short description of the function of the class.

| **Packages** | |
|---|---|
| **metasearch.aggregation.data** | |
| **metasearch.aggregation.data.provider** | |
| **metasearch.aggregation.data.request** | Structures for storing search requests |
| **metasearch.aggregation.data.result** | Structures for storing search results |
| **metasearch.aggregation.data.servletwrapper** | |
| **metasearch.aggregation.searchtarget** | Encapsulation of different types of search engines |
| **metasearch.main** | Main classes |
| **metasearch.presentation** | Presentation of results |
| **metasearch.util** | Utility classes |
| **metasearch.util.config** | Classes for configuration |

## Package metasearch.aggregation.data

| Class Summary | |
|---|---|
| **MetaSearchAttributeType** | Implements a data type for date attribute types in queries. |
| **MetaSearchOperator** | Implements a data type for operators in queries. |
| **MetaSearchRelation** | Implements a data type for relations in queries. |
| **MetaSearchRequest** | This class contains the implementation of a generic search request. |
| **MetaSearchRequestItem** | A request item is an attribute value pair with additional parameters the attribute is the search tag the value the search term. |
| **MetaSearchRequestObject** | Implements an empty class needed for type-safety reasons. |
| **MetaSearchRequestTree** | This class contains the implementation of a generic request tree. |
| **MetaSearchRequestWebGUI** | Translates an incoming request from the web interface to the internal request format. |

## Package metasearch.aggregation.data.provider

| Class Summary | |
|---|---|
| **DBXClient** | The client for a database search target (DBX). |
| **DBXRequest** | |
| **DBXSearch** | Queries a database and post-processes the XML content. |
| **W3XClient** | The client for a web search target (W3X). |
| **W3XPresent** | This class actually retrieves the search results. |
| **W3XRequest** | |
| **W3XSearch** | This class loads an HTML page from the web search target and evaluates its content. |

## Package metasearch.aggregation.data.request

| Class Summary | |
|---|---|
| **RequestParameters** | Contains either a TestRequest or a HttpServletRequest object, depending on which constructor is called |
| **RequestParametersHttp** | Contains either a HttpServletRequest object, depending on which constructor is called. |
| **RequestType** | Implements a type for operator in queries. |

## Package metasearch.aggregation.data.result

| Class Summary | |
|---|---|
| **Region** | Defines a region by start and count. |
| **ResultRec** | Is used to hold data records which are sent as present responses form a search target. |
| **ResultRecFactory** | Creates a different ResultRecXXXXX objects, depending on the parameter recordFormat. |
| **ResultRecSUTRS** | Used to read and hold SUTRS (Simple Unstructured Text Record Syntax) records. |
| **ResultRecXML** | Used to read and hold XML records. |
| **ResultSet** | Holds a result set. |
| **ResultSetList** | Holds several result sets (needed by MetaSearchRequest) the result sets are identified by a search target. |
| **TagValueItem** | This is used in the ResultRec. |

## Package metasearch.aggregation.data.servletwrapper

| Class Summary | |
|---|---|
| **DBXClientWrapper** | This class wraps the DBXClient as a HTTP servlet. |
| **DBXSearchWrapper** | DBX main search class. |
| **W3XClientWrapper** | This class wraps the W3XClient as a HTTP servlet. |
| **W3XPresentWrapper** | DBX main search class. |
| **W3XSearchWrapper** | DBX main search class. |

## Package metasearch.aggregation.searchtarget

| Class Summary | |
|---|---|
| **DBXSearchTarget** | This class represents the DBX search target. |
| **SearchTarget** | Abstract class for generic search engines. |
| **SearchTargetFactory** | This class creates different search engines. |
| **SearchTargetList** | This class holds all available search targets. |
| **W3XSearchTarget** | This class represents the W3X search target. |

### Package metasearch.main

| **Class Summary** | |
|---|---|
| **MetaSearchWebService** | this servlet is waiting for search requests. it is started by the application server at OPAC's start up time |
| **MetaSearchWebSession** | This class is called directly from the application server whenever a search request comes from the WEB GUI. |

### Package metasearch.presentation

| **Class Summary** | |
|---|---|
| **ErrorBean** | If an error occurred this Bean hold the error message which is then displayed in the error.jsp page. |
| **SearchResultBean** | This bean stores information about the request and the result of the request. |
| **ViewAll** | This servlet handles the request for specific records in a result set. |

### Package metasearch.util

| **Class Summary** | |
|---|---|
| **ErrorHandler** | Class responsible for error handling |
| **Messages** | Holds all error messages. |
| **MSLogger** | This class defines the logging mechanism. |
| **MSServletManager** | This class loads and unload a servlet. |
| **MSServletProfiler** | This class support time measurement for threads. |
| **MyErrorHandler** | This class helps to redefine the error handler. |
| **ReplaceString** | empty class needed for type-safety reasons |
| **TreeNode** | Simplified implementation of a Node from a Document Object Model (DOM) parse of an XML document. |
| **XMLUtil** | This class helps to read in the configuration file of the server and the configuration file of the targeted search engines. |

## Package metasearch.util.config

| Class Summary | |
|---|---|
| **Config** | The Config class stores all necessary configuration information for the OPAC and the search engines. |
| **Fields** | This class describes one field in the configuration file. |
| **Forms** | This class describes one search form in the in the configuration file. |
| **MappedField** | This class describes one mapped field in the configuration file. |
| **SearchEngine** | This class describes one field in the configuration file. |
| **SearchEngineMapping** | This class describes one mapped field in the configuration file. |

# 7 Annex II: Web Interface to the Broadcast OPAC

## 7.1 Overview

Server Pages) and JAVA Servlets. This section gives an overview about the interaction and data flow between these components.

There are a number web pages that integrate HTML-forms. A description of each page and its forms is given in the following sections.

Figure 12 shows which HTML and JAVA Server Pages submit data by forms to which servlets, and which servlets use JAVA Beans to pass information to JSP pages.

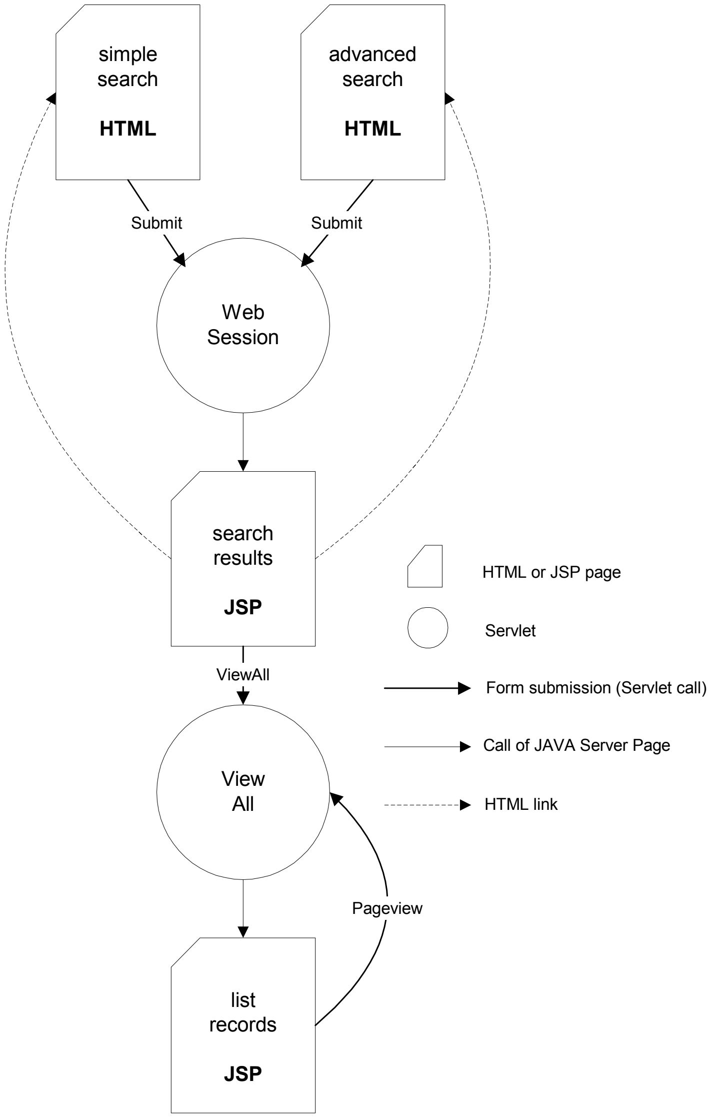


**Figure 12**: Servlets, HTML-pages and JSP-pages

On the pages search.html and searchadvanced.html (for which also an single-page implementation as JSP has been provided) the user may enter his query. An overview of the search results of the different search targets is displayed by searchresults.jsp. Then the user can select one search target and receives a page with brief presentations of the first ten of the found items shown by listrecords.jsp. This page can be repeatedly displayed to view more search results.

While the *.html pages are static web pages, the *.jsp pages are dynamically created every time they are shown to the user. The *.jsp pages merely define the format and layout of the pages, while the content of the pages is supplied by the various servlets.

When data is passed from a HTML or a JAVA Server Page (JSP) to a servlet this is done by submission of HTML forms. When a servlet passes data to a JSP page, it creates (or re-uses) an instance of the SearchResultBean class, which can be accessed by the JSP page at compile time of the JSP page.

### search.jsp

The **search.jsp** page allows the user to enter a simple query. The user can pick one search attribute out of a list (e.g. full-text, creator, …) and enter a search term for that attribute (e.g. Clinton). It also allows him to select one or more search targets on which the specified search is performed.

The user’s input determines the following set of parameters, which is passed to the WebSession servlet via the post method.

| Parameter name | Values | Description |
|---|---|---|
| InstX<br>(X is a number) | Short name of institution (e.g. BBC, ORF, ...) | If a parameter is found that starts with "Inst", the value is used to search the SearchTargetList for this search target. |
| attrX | Attribute identifier | The meta data tag selected in the drop down lists, numbers from meta data document |
| termX | Text | The search term entered by the user |
| Language | String identifying to a language (e.g. ‘en’ for English, ‘fr’ for French) | Language selected in the user interface |

**Table 6:** Parameters of the form ‘PrestoSearch’ in search.html

### searchresults.jsp

After the WebSession servlet has executed performing the search, this JAVA server page is displayed. The servlet has created a Java Bean which is passed to the JSP page its data is displayed.

| Parameter name | Values | Description |
|---|---|---|

| Parameter name | Values | Description |
|---|---|---|
| Request ID | Number | An ID which refers uniquely to the search request the user initiated, generated JAVA-internally. |
| Institution Name | Long name of institution | A list of institutions is stored in the JAVA bean. For every single institution its name, its short name and number of records which were found at that search target is stored. |
| Institution ID | Short name of institution (e.g. BBC, ORF ...) | |
| NumSearchResults | Integer | |

**Table 7:** Information stored in the SearchResultBean passed to searchresults.jsp

The list of institutions is displayed on the JAVA server page and for every institution a button labelled "Show" is shown. When one of these buttons is clicked the ViewAll servlet is called and the parameters described in Table 8 are passed to it (via the post method).

| Parameter name | Values | Description |
|---|---|---|
| Request ID | Number | An ID which refers uniquely to the search request the user initiated. |
| Institution ID | Short institution name | Specifies the search target, from which the user wants to see the results |
| Page number | 1 | The ViewAll servlet displays data items in groups ('pages') of 10 (this is the default value but it is configurable) items. The value 1 here means that the first page (i.e. items 1 to 10) is to be displayed. |

**Table 8:** Parameters of the form 'ViewAll' in searchresults.jsp

### listrecords.jsp

The ViewAll servlet again creates a JAVA Bean (SearchResultBean), in which the data for presentation of the result records are stored.

| Parameter name | Values | Description |
|---|---|---|
| RequestID | Number | An ID which refers uniquely to the search request the user initiated. |
| InstitutionID | Short institution name | Specifies the search target, from which the user wanted to see the results |
| Page number | Integer | Number of the current result page |

| | | |
|---|---|---|
| NumResults | Integer | Number of results at the selected institution |
| AttributeValue | Text | Descriptive text for the item (brief present) |
| ItemNumber | Integer | Number of the record in the result set |

**Table 9:** Information stored in the SearchResultBean and passed to listrecords.jsp

This JAVA Server Page contains three HTML forms which are named Present and PageView and are submitted to differentiate the servlets WebSession and ViewAll respectively.

# 8 Annex III: References

[1] Annemieke de Jong: **Metadata in the audiovisual production environment**; ©2000 Netherlands Audiovisueel Archief

[2] European Broadcasting Union: **PMC Project P/META (Metadata exchange standards)**; http://www.ebu.ch/pmc_meta.html

[3] **MPEG-7 main page**; GMD - Forschungszentrum Informationstechnik GmbH; http://www.darmstadt.gmd.de/mobile/MPEG7/index.html

[4] **The New Zealand Digital Library Project**; http://www.nzdl.org/

[5] Philippe Salembier: **Status of MPEG-7: the Content Description Standard**; *International Broadcasting Conference Amsterdam, The Netherlands, September 8, 2000*; Universitat Politecnica de Catalunya, Barcelona, Spain

[6] The **Unicode Standard**, http://www.unicode.org

[7] **Character sets on the web**, http://www.w3.org/International/O-charset.html

[8] **Oracle**, http://www.oracle.com

[9] **Informix**, http://www.informix.com

[10] **Microsoft**, http://www.www.microsoft.com

[11] The **SOAP protocol**, http://www.w3.org/TR/SOAP/

[12] **Amicitia**, http://www.amicitia-project.net

[13] The **ISO Standard 8601**, http://www.iso.ch/markete/8601.pdf