Anacleto documentation

user interface

In this part help is offered for the users of this customized version on how to use this site and how to query.

Visual components

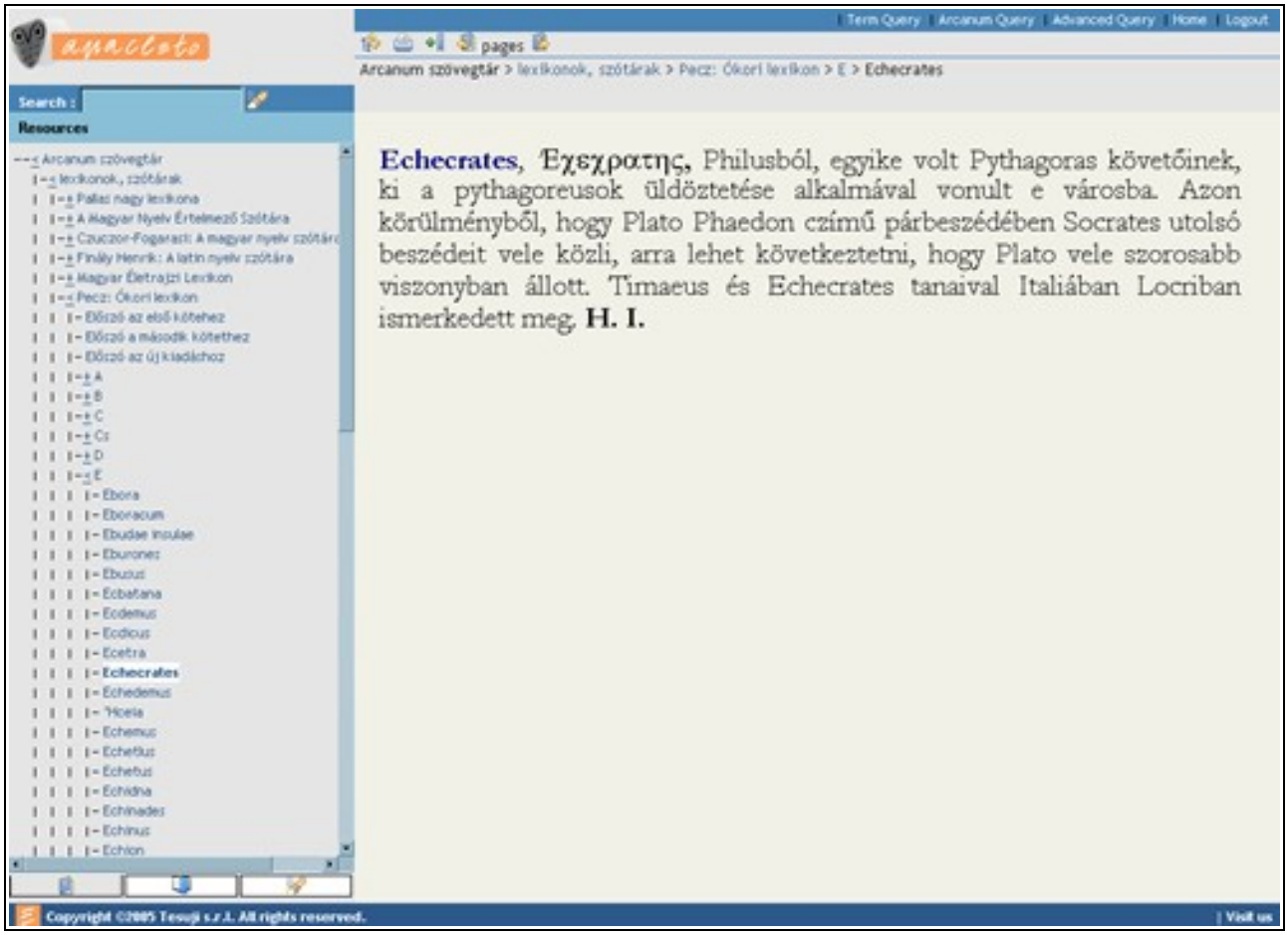
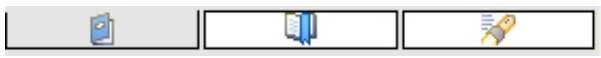


Table of Contents

In Anacleto every document has a hierarchical position, that is, it can have parent and child elements etc. (see hierarchy). The table of contents is the presentation of this structural disposition and the instrument to navigate through it.



The table of contents view can be activated from the left side icon situated on the lower toolbar of the left side bar. When opening the Anacleto only the highest level of the hierarchy is visible, namely the shelves. The table of contents can define the status of the document in three different ways:

- (document icon) – a document that has no child elements
- [+] (closed parent icon) – a document that has child elements but these are not visible since the document is 'unopened'
- [<] (open parent icon) – a document that has child elements and these are visible since the document is 'open'

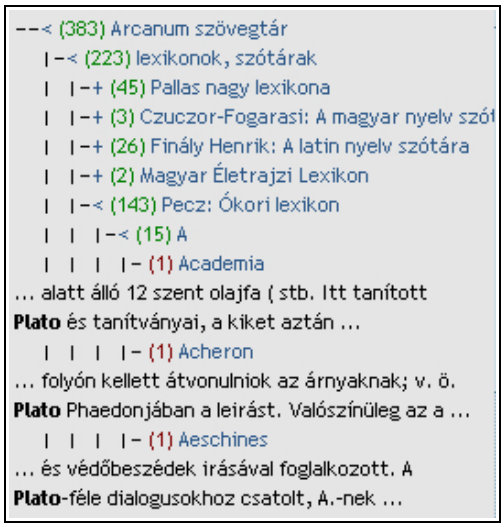
Documents can be opened and closed clicking on the [+] and [<] icons.

It may happen that a document has many child elements. To make all these visible, is time and resource consuming, therefore the number of the child elements to be active at one level has been limited. According to the default option this number is 50 but the system administrator can customize this number as suitable. The elements that are not visible can be activated by clicking on the [next] icon – again only the next 50 will be visible.

The document belonging to the element can be activated by clicking on its title and this will appear in the document-window.

The special feature of the table of contents is that it can be both ways (there and back) synchronized with the document. This option is set by the 'synchronizing the table of contents' icon situated in the navigation bar of the document. More details in the documentation on Navigation bar.

Table of Contents Filtered by Query



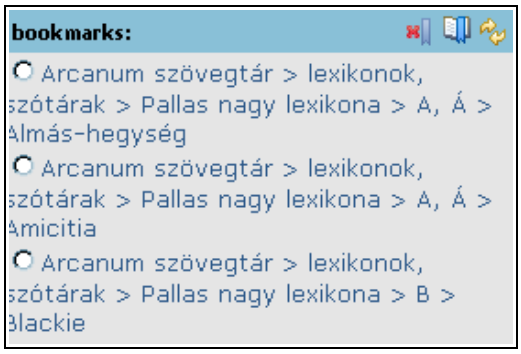
The query adapted table of contents is a view of the left side bar.

Its operation is entirely similar to that of the table of contents, but it activates only the contents branch in which there is a hit, and not only the branch but also the word environment of the hit.

This type of table of contents can only be activated from the hit list. It is active as long as it is not overwritten from another hit list. Unless brought into operation, a 'no result' sign will be displayed instead.

This view can be switched on by the right icon situated on the left side bar of the lower toolbar.

Bookmark



The function of the bookmark is to simplify and unify for saving the different text positions for later usage. Although browsers usually have a similar function, in this respect the frame mode raises constraints against most browsers.

Bookmarks are stored in cookies. Therefore, in order to be able to use them the browser has to allow cookies.

bookmark view

The bookmark view is a view of the left side toolbar. It has two parts: above in a thin bar there is the toolbar connected to the bookmark (delete, rename, go to), below the bookmark list.

new bookmark

A new bookmark can be added in the navigation bar of the document by clicking on the adequate icon. Then, the left side toolbar switches to bookmark-view, in which the entire access path of the document appears.

switching to bookmark view

Clicking on the bookmark-looking icon located in the middle of the navigation bar that is on the bottom of the left side toolbar, the toolbar will switch to bookmark view.

rename

A bookmark can always be renamed. Switch to the bookmark view, select the bookmark to be renamed and in the bookmark toolbar click on the middle – ‘rename selected bookmark’ – icon. An input field appears containing the current name. Rename/overwrite and click OK. Instead of the old name the new one appears.

delete bookmark

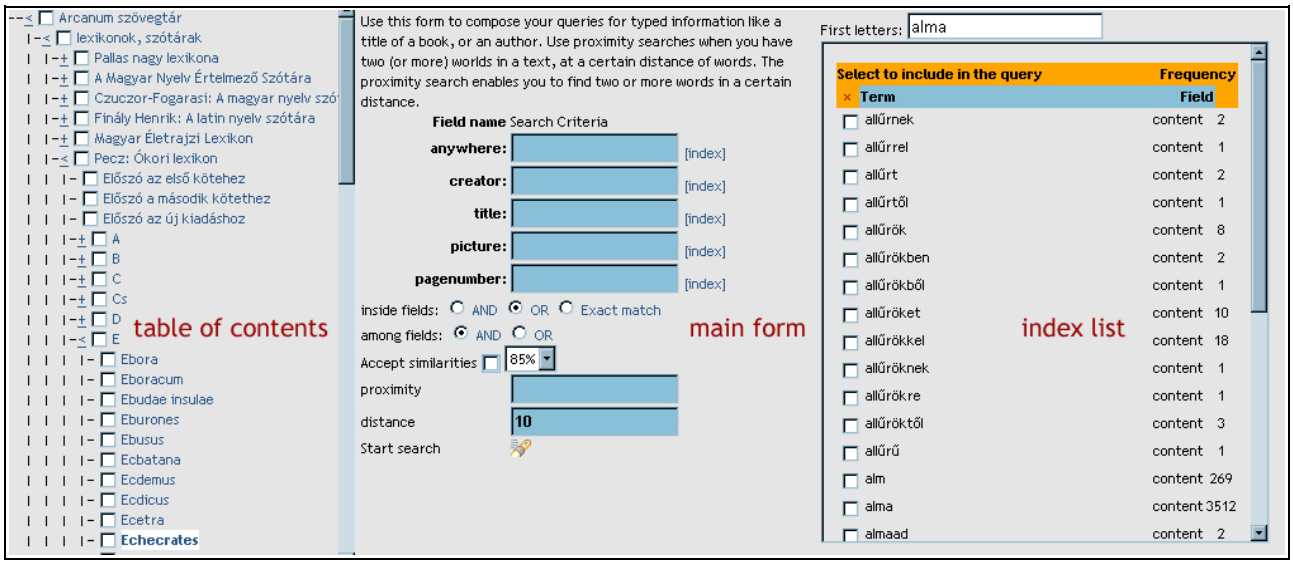
Switch to the bookmark view, select the bookmark to be deleted and in the bookmark toolbar click on the left side – ‘delete selected bookmark’ – icon. The selected bookmark will disappear.

synchronizing the document

The document linked to the bookmark can be activated by simply clicking on the chosen bookmark when in the bookmark view. In the document window the respective document appears.

Complex Query

The complex query form consists of one main form and two additional help forms. One of the additional help forms is the index list located on the right side of the window. The other one is the left side table of contents. In general about the query and about the query options see the query syntax documentation.



Main form

The main form is the ‘soul’ of the complex query form. The help forms offer only additional possibilities, but the essence is to be found in the main form.

anywhere

In the first input field (‘anywhere’) the query runs in the entire text, irrespective of the location of the text within the database. By clicking on the [x] symbol next to the input field, the right side index list is activated – and in this case the index list will contain every terms of all fields.

fields

The following four input fields offer support for special field query (or to name it differently, semantic fields). These fields are ‘Author’, ‘Title’, ‘Signature’ and ‘Page Number’. For example, if the query is to find authors named John, type 'john' in the ‘Author’ field. The query is not sensitive to lower/upper case differences, in this respect differences like JOHN or jOHn do not matter, the result will be the same. The [x] sign next to the input fields operates similarly as described before, namely it restricts the index list to the terms in the field.

linking the terms

When searching for more than one term, these terms should be separated by comma ‘,’. Every habitual query element can be used (truncated words, quotation marks). See query syntax.

In the following two lines the possibility to establish a connection between the terms of the query - even if typed at the same time in more than one field - can be set.

The ‘connection within field’ is set so as to introduce in between the terms - typed in one input field and separated by a comma - a logical operator of the kind AND or OR, and to extend the search for the entire phrase.

The logical connection `Kossuth AND Deák` in the title field will search for documents with titles that contain both Kossuth and Deák.

The logical connection `Kossuth OR Deák` in the title field will search for documents with titles that contain at least one of the terms Kossuth and Deák (and consequently, also documents in which both terms appear).

When searching for an `exact match`, the terms of the phrase should not be connected, but the entire phrase should be put in quotation marks, e.g. `"Kossuth Lajos"`. Consequently, the possible hits will contain both (or even more) terms that appear in exactly the same order. In this situation, the similarity operator is ineffective. (Note: when typing only one term in the field, it should not be put in quotation marks. The terms `"Kossuth"` and `Kossuth` are equivalent.)

On the basis of a similar logic the option ‘connection between fields’ has been set.

similar terms

The check box ‘similar terms included’ can activate the so-called Fuzzy query. Basically, the search will also find word forms that are not exact matches with the query term but to some extent these are similar. The degree of similitude is expressed as exact character match percentage. For example there is an 80% match in between the characters of street and streel. A similarity percentage of 50 to 95% can be set through the scroll down menu of the check box. By default the similarity level is set to 85%.

The fuzzy search is not available if within the field the exact phrase option has been set.

proximity query

The ‘proximity query’ option refers to the situation when documents in which not only two (or more) terms are to be found but also these terms should be juxtaposed. The measure of distance is set by the number of words that separate the two terms. 0 means the terms are next to each other, 1 means that in between the terms there is maximum one other word, and so on. Words that can be found in the stop list are not considered ‘in between words’. The words are separated by space.

The distance between the terms of the query can be set in the ‘maximum distance between words’ field.

In detail about the proximity search in the proximity query documentation.

Index List

The index list contains all phrases that occur in a given field. In order to choose the phrases of a given field, the [x] symbol that appears next to each field on the main list should be clicked on. If no field is chosen, the phrases from the ‘content’ field will be selected by default. The ‘content’ field contains each phrase from the database.

On the top of the index list there is an input field named ‘start with’. When a group of

letters is added to this field the search will return phrases that begin with the given letter combination. The list is automatically updated therefore, in order to add or delete the data from the field it is not necessary to click on any icon or symbol.

The list can contain simultaneously 100 phrases.

Given the fact that the program is set to track first the initial letter combination, in order to return right side truncates (e.g. 'Kossuth*') there is no need to complete the search field with the masking characters (?, *). It is enough to type only 'Kossuth'. There is also the possibility to search for left side truncates, meaning that not only the beginning but also the ending of a phrase can be unknown. In this last case though, complementary characters have to be used. The phrase *asszony will return terms ending in 'asszony'. Given the fact that the '*' character can give 0, 1 or even more responses, in order to avoid the response `'asszony'` in itself at the beginning of the phrase a '?' should be added, which replaces 1 character. Thus, the phrase `'?*asszony'` in the search field will return a list of words that contain the term asszony with something else at the beginning of the term (e.g. `kisasszony`, `öregasszony`, etc.)

Table of contents

Clicking on the Complex search template the left side table of contents is transformed. At the beginning of every branch of the table of contents there will be a checkbox. The search will run only through the selected branches. When no checkbox is selected the search is not restricted, which means the search will run through the entire database.

This is the tool to use when a search should run only through a given sector (e.g. the history shelf) instead of the entire database. The depth of the search can be restricted as desired (e.g. if in the checkbox only the terms and phrases starting with 'Are' from the Pallas encyclopedia are selected, then only these will be searched for).

Simultaneously any number of items can be checked.

The users only have to tick the checkboxes. The main form 'knows' which are the selected items.

Document

The document is the skeleton, the innermost core around which Anacleto is built. This is why the document window has not been crowded with other useful functions, thus striving to preserve its original form.

Hit list

The hit list's function is to allow a rapid and thorough survey of the responses after query and – if necessary – to perform changes to improve the query.

A hit is a document for which the query is valid (i.e. not a paragraph, not a book or any other possible unit of the text).

Quick statistics

On the top of the hit list there is a quick statistics. The following information is displayed:

- total number of hits
- total time span required to complete the search
- the positions on the list of the currently displayed hits (e.g. 1-10)

Table of contents restricted to hit list

In a field this icon serves an unusual function. By clicking on it, on the left side toolbar a view is displayed. This is a view that reminds the table of contents, but it contains only the branches in which there are hits. In detail about this see the table of contents restricted by query.

Query

Here the full query is displayed. In case the hit list proves unsatisfactory, here the query can be modified according to the query syntax.

Hits by page

The number of hits to be displayed by page can be set as desired. By default the number of hits to be displayed is set to 10.

Search

Clicking on the respective icon, the search is run on the basis of the query typed in the input field. Instead of clicking with the mouse, it is also possible to press `ENTER`.

Previous/Next hit list

The pair of icons, which can be found in the same line with the query and also displayed at the bottom of the hit list, goes through the entire list forward and backward by one page. When at the end/beginning of the hit list, the respective icons will be inactive therefore it is not possible to click on them.

Navigation to the N-th hit number

The links here located are able to go farther on the hit list than the previous/next option earlier described. The list is able to display apart from the first, the last, the previous also the list starting with hits number +/- 1 time, twice, 3 times 10., 100., 1000., 10000., according to the number of results turned by the query.

The hit list

So far the options described are the garnish to the actual main hit list. For each entry the hit list displays the following information:

- the serial number of the hit displayed (starting from 1)
- the full access path of the hit (as link)
- the word environment of the hit by highlighting the searched phrase within the found document

The document can be activated by clicking on the full access path

Navigation Bar

The navigation bar enables easy access to the document functions and facilitates leafing through the database on the basis of the hit list and of the natural structure of the database.

Icons marked with '*' are displayed only when in a current search.

Synchronizing the table of contents

This option serves to display the actual location of the document within the entire database. Clicking on the icon, the left side toolbar switches to table of contents view in which an updated table of contents is displayed. In the table of contents only the branches directly connected to the document will be open (the 'ancestor' and the 'sibling' documents). The current document is displayed highlighted in the table of contents and the window – according to the possibilities – will be aligned to this document, meaning that on the top of the visible part this document will be displayed. To put it differently, if the table of contents is long enough to enable scroll bar usage, it will scroll down or up until the document to be synchronized is displayed on the top of the page.

Printing the document

The respective icon is there to enable printing the document.

Important! The print icon on the browser is set - by default - to print the entire page (including the heading and the table of contents), while this one will print merely the actual margins of the current document.

New bookmark

The respective icon serves to add a new bookmark to Anacleto's already existing bookmarks. About the bookmarks and their usage see the documentation.

Leafing: previous page/next page

In Anacleto the documents have a hierarchical disposition – including documents originally part of (e.g. relational) databases in which hierarchy does not play a significant part. Consequently, every document has a parent and possible child branches, etc. It means that in the hierarchical structure every document is preceded and followed by other documents (except, of course, the first and the last document). With the help of the leafing icons and according to the system described, the previous and the next documents can be reached. The leafing order corresponds to the table of contents order.

Hits: previous/next*

If there is a current search, in order to be able to access the hits it is not necessary to return again and again to the hit list, it is enough to use the previous/next pair of icons. The 'previous' icon will go to the previous item, while the 'next' icon will go to the next item in the list.

Hits within the document: previous/next*

This pair of icons navigates between the hits within the document. The ‘previous’ icon is always active thinking that when leafing through the document (to the beginning/end of it), this icon secures possible to go back to the first hit within the document in just one move. The ‘next’ icon is active only when there is more than one hit within the document.

Hit list*

As expected, this icon goes back to the hit list. To be more precise, it goes back exactly to the part of the hit list where the current document is located.

Reference Bar

If the document is open, the full access path is always displayed in the frame above the document. When navigating away from here (e.g. to a main form or to the hit list) the bar is left empty.

Hierarchy

In Anacleto every document has a hierarchical disposition. The two upper items of the hierarchy are called ‘shelf’ and ‘book’. The shelves and the books are described in a document named 'books.xml'. Here is an excerpt from the file:

```
<?xml version="1.0" encoding="utf-8"?>
<xml-body>
  <shelf name="root" title="Arcanum szövegtár">
     <shelf name="f_lexikonok"
            title="lexikonok, szótárak"
            titlepage="titlepages/tp-lexikonok.html"
            encoding="utf-8">
        <book name="pallas"
              title="Pallas nagy lexikona"
              titlepage="titlepages/f_lexikonok/tp-pallas.html"
              contentStyleSheet="css/pallas.css"
              contentType="NXT3CONTENT"
              nxt3IndexingStyleSheet="html.xsl"
              nxt3Descriptor="pallas.xml"
              encoding="utf-8"
              location="e:/anacleto/f_lexikonok/pallas"/>
     </shelf>
  </shelf>
</xml-body>
```

This XML file is controlled by the `anacleto_books.dtd`:

```
<!ELEMENT xml-body (shelf)+>
<!ELEMENT shelf ( shelf | book )+>
<!ATTLIST shelf
  name CDATA #REQUIRED
  title CDATA #IMPLIED
  titlepage CDATA #IMPLIED
  encoding CDATA #IMPLIED
>
<!ELEMENT book EMPTY>
<!ATTLIST book
  name CDATA #REQUIRED
```

```
  title CDATA #IMPLIED
  titlepage CDATA #IMPLIED
  encoding CDATA #IMPLIED
  location CDATA #IMPLIED
  contentStyleSheet CDATA #IMPLIED
  contentType CDATA #IMPLIED
  nxt3IndexingStyleSheet CDATA #IMPLIED
  nxt3Descriptor CDATA #IMPLIED
  scheduled (true|false) "false"
  schedulingCronExpression CDATA #IMPLIED
  user CDATA #IMPLIED
  password CDATA #IMPLIED
>
```

shelf

The highest-ranking element in Anacleto. From the indexing point of view the shelf is not an independent unit, which means that the shelf structure can always be modified. The shelf plays part in organizing the table of contents and the hit list. Shelves can have independent address names.

book

The book is an independent content and indexing unit. A book can be potentially everything that stands unitary, for example a database (or occasionally, a part of the database which behaves like an independent unit). Generally, a book = a book/a collection of books/the collected volumes of a periodical/the complete works of an author/the documentation on a firm, a department or a product/the cards of a catalogue/etc. Ultimately, it can be anything that can be treated as a unit. A book can have address names. The books are described by a hierarchically constructed xml file.

document

A document is an independent entry in the xml file that describes a book. Generally, a document is understood to be also physically an independent file, although occasionally it is possible to find more than one document in a file.

title page

A specific HTML file, which contains basic data of the shelf or the book. It does not strictly belong to the database, meaning it is not indexed, it can always be modified and it primarily serves a navigation aim. The title pages are recorded in the books.xml file.

other, unnamed categories

Within the books generally there are natural or artificial lower hierarchical units (e.g. chapters or catalogue cards filed under the letter ‘A’). These are not given a specific name within the Anacleto. The xml files describing the hierarchy show if a certain item has child elements, and simply consider them as parent – irrespective of their character: chapters, paragraphs, volumes or cycles of poems.

Full access path

This name designates a string that contains the name of the given document and the names of its ancestor documents until the root of the database. The elements are separated by the ‘>’ ('greater than') symbol and its construction follows the hierarchical logic of the database.

To illustrate:

> *Arcanum szövegtár > történelem > Kossuth Lajos összes munkái > Kossuth Lajos összes munkái XV. > Kossuth Lajos 1848/49-ben. V. Kossuth Lajos kormányzóelnöki iratai 1849 április 15. – augusztus 15. > 301-400 > 394. Budapest, 1849 július 5. Kossuth válasza Simonffy József alezredes felterjesztésére: közli vele a magyar kormány álláspontját az erdélyi román felkelőkkel kötendő fegyverszünettel kapcsolatban*.

The full access path’s elements are occasionally linked, which means that – considering the example given above – by one click the document entitled 'Kossuth Lajos összes munkái' can be reached, and also the document entitled 'Kossuth Lajos 1848/49-ben. V. Kossuth Lajos kormányzóelnöki iratai 1849 április 15.'

Query Syntax

The query syntax can be applied within the program in all ‘simple’ input fields which serve to query. Possible examples are the search forms of the quick search or the hit list. Some of the elements used here can also be applied in search fields that are more complex (e.g. wildcard search), but the more complex search forms are primarily there in order to avoid the usage of sophisticated syntax.

terms

`Single term`: a word, e.g. ‘test’ or ‘hello’ `Phrase`: group of words typed in between inverted commas, e.g. “Project Gutenberg”

Multiple terms and phrases can be combined by logical operators.

Search phrases are insensitive to lower/upper case distinctions.

field

the syntax for the field search is the following:

```
<field name>:<term> e.g. title:Philosophy or title:philosophy
or
<field name>:"<phrase>" e.g. title:"Analitical philosophy"
```

The Lucene has a default field (in the Anacleto this is the ‘content’ that contains every text) and through this it is possible to run a search without naming the field:

```
<term>
e.g. philosophy
```

which equates

```
content:philosophy
```

Note: the expression title:analitical philosophy corresponds with the title:analitical AND philosophy expression, which means that only the term analitical is considered to be

part of the title field, and the philosophy will be searched in the default content field.

Term Modifiers

Joker characters: ? and *

Single term modifier: '?' (question mark) character zero or more character term modifier: '*' (star) character

examples:

`te?t` returns the following terms test, teet, teit, text, teüt (the Vogoul word for fire), etc.
`test*` returns the following terms test, testo, testa, testi etc.
`tes*t` returns the following terms (zero occurrence!), testület, testvérét, Tesszalonikát, Tesszáliát, tesztsorozat etc.

It becomes apparent that the term modifiers are valid not only for Latin characters but also to French, German, Italian, Hungarian ones (and many more).

Important! Term modifiers can also be used at the beginning of the words (left side truncates), for example: `*test` returns the terms Ürtest, hadtest, protest, potest, tettest, helyettest, önkénytest etc.

Fuzzy search: ~

In Lucene the fuzzy search is a query on the basis of a certain degree of similarity and it uses the so-called Levenshtein-distance (or editing distance) algorithm.

The fuzzy search operator is the '~' (tilde) symbol.

The operator can be used only after a single term

`test~` returns the terms: test (100% similarity), tett, telt, testi, teste, Pest etc.

The degree of similarity of the hit can be set as a decimal number in between 0 and 1. The default value is set to 0.5 (pay attention to the '.' , do not write a ',' instead!)
iparos~0.8 returns the terms iparosa, ivaros, Iharos, Ikaros etc.

Proximity search: ~<number>

The Lucene is able to find terms of a phrase in between which a set number of different terms appear.

The proximity search operator is the '~' (tilde) symbol placed after a phrase and a number that sets the maximum distance between the two terms.

The order of the words can be changed, phrases "magyarokra nézve"~10 and "nézve magyarokra"~10 are identical.

Terms that can be found in the stoplist are not considered in between words.

Examples:

`"magyarokra nézve"~0`: finds words next to each other `"magyarokra nézve"~1`: in between the two words there can be one different word `"magyarokra nézve"~10`: in between the two words there can be ten different words

Range: [x TO y] and {x TO y}

The Range query is searching for values that fall within a numerical or alphabetical defined range/interval. The borderline values can be set to be included or excluded from the hit list.

The inclusive query syntax: `<field_name>:[ <phrase> TO <phrase> ]`.

The exclusive query syntax: `<field_name>:{ <phrase> TO <phrase> }`.

The TO expression must be written in CAPITAL LETTERS.

Examples:

`content:[ alma TO almabimbó ]` returns alma, almabetegségek, Almaad (meaning: Almád), almaalakú ... etc. almabimbó terms `content:{ alma TO almabimbó }` returns almabetegségek, Almaad, almaalakú ... etc. terms, excluding the terms alma and almabimbó.

Note: given the fact that content is the default field, the above mentioned examples could also be written as [alma TO almabimbó] and { alma TO almabimbó }.

Boosting a Term: ^<number>

The terms of the query can be boosted with the '^' operator. The more weight a term is given, the more that term becomes relevant in terms of the query question. The number can be any positive number. The default value is 1, but it can take also values under 1 (e.g. 0.2). Assigning weight becomes really significant when the hit list is organized by relevance to the query. Weighting can also be applied to phrases. (When hit list is organized by table of contents, the boosting has no effect).

Examples:

`Kossuth^4 Lajos` in organizing the hit list the term Kossuth has a bigger relevance than the term Lajos `"Kossuth Lajos"^4 "Deák Ferenc"` in organizing the hit list the term Kossuth has a bigger relevance than the term Deák

Logical operators

The Lucene encourages the usage of the following logical operators: AND, '+', OR ('||'), NOT and '-'. Important: capital letter usage is mandatory!

OR

The OR operator connects two terms so that in the return hit list at least one of the terms of the query (if not both terms) has to be included. Instead of OR the || can also be used.

Examples:

```
Kossuth OR Lajos
"Kossuth Lajos" OR Deák
"Kossuth Lajos" OR title:Deák
```

AND

AND is a default logical operator, which means that if in the query no other operator is

specified than this operator will become valid. Both of the terms connected by the logical operator AND have to appear in the return hit list.

Examples:

```
Kossuth AND Lajos
"Kossuth Lajos" AND Deák
"Kossuth Lajos" AND title:Deák
```

+

The + (plus) symbol has to stand in front of a single term or phrase. This operator has the same function as the AND operator.

Examples:

```
Kossuth +Lajos
"Kossuth Lajos" +Deák
"Kossuth Lajos" +title:Deák
```

NOT

The term that stands after the NOT operator must not appear in the return document.

Examples:

```
Kossuth NOT Lajos
"Kossuth Lajos" NOT Deák
"Kossuth Lajos" NOT title:Deák
```

-

The term that stands after the - (minus) symbol must not appear in the document. Therefore, this operator's function equates that of the NOT operator.

Examples:

```
Kossuth -Lajos
"Kossuth Lajos" -Deák
"Kossuth Lajos" -title:Deák
```

Grouping

With the help of parentheses ('(' and ')') the query can be grouped into sub-queries. This operator is useful when complicated logical connections are to be formulated. The meaning of

```
( Kossuth OR Széchenyi ) AND Deák
```

expression is: the term Deák has to appear in the document and also at least one term of the following two.

Grouping fields

Grouping can also be valid in a field.

```
title:( ( Kossuth OR Széchenyi ) AND "Deák Ferenc" )
```

which means that: the phrase "Deák Ferenc" has to appear in the title of the document and also at least one term of the other two.

Domain level query

This function is characteristic only in Anacleto. It means it does not operate under Lucene. The path is a specific field which serves to record and query the hierarchy, the shelf, and it develops the full access path. query within a shelf:

```
path:/f_tortenelem/*
```

query within a database:

```
path:/f_tortenelem/marczali/*
```

Escaping special characters

The Lucene considers the following characters as special character: + - && || ! ( ) { } [ ] ^ " ~ * ? : \

When this characters are to be used, the \ symbol has to precede them. For example, if the query is:

```
(1+1):2
```

then this can be reformulated as

```
\(1\+1\)\:2
```

administrative interface

Default settings

Configuration directory
: A directory in which the configuration files can be found. Every configuration file has to be put in this directory.

Index directory.
: The directory contains the index files. This directory has to be empty. The files contained within cannot be deleted or altered by the users. The directory – unless the indexing is kept in the memory – should be assigned to the fastest partition/disk/system.

Log files directory.
: Here the log files will be created. See more at: logs.

Book files.
: The configuration files of shelves and books. This directory/file is responsible for the site content.

Language.
: The basic language of the texts. The language code defines the alphabetical order of the index lists.

Possible values:
be, bg, ca, cs, da, de (de_AT, de_CH, de_DE, de_LU), el, en (en_AU, en_CA, en_GB, en_IE, en_IN, en_NZ, en_ZA, en_US), es (es_AR, es_BO, es_CL, es_CO, es_CR, es_DO, es_EC, es_ES, es_GT, es_HN, es_MX, es_NI, es_PA, es_PE, es_PR, es_PY, es_SV, es_UY, es_VE), et, fi, fr (fr_BE, fr_CA, fr_CH, fr_FR, fr_LU), hr, hu, is, it (it_CH, it_IT), lt, lv, mk, nl (nl_BE, nl_NL), no (no_NO, no_NO_NY), pl, pt (pt_BR, pt_PT), ro, ru, sk, sl, sq, sv, tr, uk

Security level.
It sets the security level for the application. The first security level does not require a registered user name or password to access an application. The second level protects the administrative interface from unauthorized access, and the third level protects the entire site.
Possible values:
'No protection', 'Administrative interface protection', 'Entire site protection'

Indexing priority (0-9).
A higher value determines a faster indexing and a lower server performance, while a lower value determines a slower indexing speed, but in exchange the server can be used during indexing.

Default style sheet.
On the displayed page the CSS default style sheet is linked (books can overwrite this default value).

Default HTML index scheme.
The HTML files' default XSLT style sheet (books can overwrite this value).

Default TEI index scheme.
The TEI files' default XSLT style sheet (books can overwrite this value).

Default XML display scheme.
Default XSLT style sheet responsible for the XML-content display (books can overwrite this value).

Index maintenance

The index maintenance is one of the most important tasks of the administrative interface. On this page it is possible to index a document or to delete a document from the index, to control indexes. Furthermore, here it is possible to optimize the index directory.

The interface has two parts: in the upper part there is a status indicator table, while the lower part is the list of files to be indexed.

The site's databases, data units are bound by a common XML file, namely the books.xml (see more: hierarchy). The lower part displays the names and the titles from this 'books' database. The 'number of indexed documents' column shows how many documents from the database had been indexed. At the beginning – when not document is indexed yet – this number is naturally 0. To start indexing, the documents have to be selected and then it is possible to choose whether to start indexing right away (in this case, the

‘Index the selected resources now!’ icon should be clicked), or to start indexing at an arbitrary scheduled moment (in this case, the date and the time have to be set in the table positioned to the right of the previously described icon, then the icon should be clicked).

During indexing, in the upper part, the status indicator table shows the current status of indexing. To update the list click on the ‘refresh’ icon.

Currently Indexing	the currently indexed book
Books waiting for indexing	books waiting for indexing
One time jobs	
Scheduled jobs	Books waiting for scheduled indexing

Once the indexing is done it is worth optimizing, because during indexing the new items are put in new files, thus the search looses some of its speed. Optimizing merges these files.

In the lower table, the books appear as unique identifier links. When clicking on the link the book’s detailed data file is displayed.

General parameters	
Name	the book’s unique identifier (100falu)
Title	the title of the book (Száz magyar falu könyvesháza)
URL	the access path of the document (c:/<dir>/100falu)
Encoding	the encoding of the XML file descriptive of the book
Content type	content type
Authority	
User	
Password	
TEI support	
tei2IndexingStyleSheet	
NXT support	
NXT3 content descriptor	NXT files’ descriptive XML (100falu.xml)
NXT3 content descriptor	NXT files’ indexing scheme (html.xsl)
Scheduling details	
Scheduled	Is it scheduled (true/false)
Cron Expression	The time scheduled

Here all the documents belonging to the book are classified, with their unique ID, name and location of the file (compared to the root). Clicking on the unique ID the indexing details of the file, that is, from among the indexed items those that are stored appear. In this table the following data is recorded:

Field Name	the name of the field
Field Value	the value of the field, that is, the text in the field for example, if in the source document the following is

put:

```
<title>This is the title of the document</title>
```

and in index scheme – somehow reduced – it looks like:

```
<xsl:template match="title">
  <xsl:element name="in:index">
    <xsl:attribute name="field" >
      <xsl:value-of select="title" />
    </xsl:attribute>
    <xsl:apply-templates />
  </xsl:element>
</xsl:template>
```

then, when indexing the title element of the HTML content is put in the title field, and the value will be: This is the title of the document

Indexed Indexed, searchable index item

Stored The content of the field is stored in the index in its entirety

Tokenized The linguistic parser processes the field's text. This means that

- the field is processed as words and not as one phrase
- performs character conversion
- eliminates the stopwords
- when searching the field, it processes similarly – only backwards - the query question

TermVectorStored the term-frequency vector of the document is stored in the index

Attention! Since all these operations (indexing and optimization) are time- and resource-consuming, it is worth performing them at a time when circulation on the site is lowest, at night (especially, since in the case of a big database the period to optimize can exceed that of indexing).

Index statistics

The index statistics page displays in a table the database's most frequently occurring phrases. The table contains three columns:

phrase
indexed word

field
the field in which the phrase was indexed

occurrence
shows the occurrence frequency of a given phrase.

Primarily this function serves best when setting the stoplist since it is easy to decide which from the most frequently occurring words is considered ‘expletive’, that is, a word with no meaning in itself and therefore inappropriate to be used as search word.

Stoplist

The stoplist contains words that should not be indexed. The site’s administrator can set the stoplist. It is important for the stoplist to be the same when indexing and searching. When analyzing the query question the program extracts the stopwords from the actual, behind the screen running query question. Which means that, if the stoplist is changed, in order to have a consistent search it is advisable to have it re-indexed.

A new stopword can be added in the input field named ‘New stopword’.

To delete the stopword select stopword and then click on the ‘Delete selected stopwords’ icon.

In the header of the stoplist table by clicking on the 'X' symbol the entire list is selected. In the same location, by clicking on the '0' symbol selected words can be unselected.

Accents

Word conversion is used only if the text contains words with accents, which cannot be typed by users (there are no such letters on the keyboard), or the right forms are unknown to the users (when the text is written in a foreign language). For example, on the Italian keyboard there is no letter ç. When the text contains this character, a conversion rule can be set to change this letter into c. The unicode characters (and others) can be found at:

http://www.unicode.org.
http://www.j-a-b.net.

Attention! If a new conversion rule is set, or an old one is deleted, it is advisable to re-index the entire document collection.

In order to set a new conversion rule in the two input fields, next to the ‘Input new conversion rule’ sign, type the original character and the its simplified version, and then click on the icon next to the input fields.

To delete characters, select the characters to be deleted, and then click on the ‘delete the selected characters’ icon.

Clicking on the ‘X’ located in header of the table the rule can be selected. In the same location, by clicking on the ‘0’ it can be unselected.

Authorization

User authority can be set either by authorization levels, or by user groups. Currently only the basic types of both possibilities are set.

Permission types:

- no protection, every function can be accessed

- using the administrative interface is permission dependant
- every function of the site is permission dependant

If needed other permission levels can be set. For example:

- query/search protection
- table of contents protection
- document protection to view

User groups:

- Administrator
- User

When entering, Anacleto is checking the user authority, and also that under the same user name nobody accessed Anacleto earlier from a different IP address or domain (thus in an institution with subscription – library, research center, editorial office – more users with one password can access the system at the same time, but only from within the institution).

To add new users there is a main form to be completed. The following data has to be recorded:

- User ID
- User name E-mail address
- Password
- Permission level (currently: Administrator/User)

The users' list has to be created by groups. Another setting option is possible: in a user's case, setting the 'Permission to filter ID' will set the program to check if somebody entered earlier from this domain under the same name (see above).

Logs

As already stated in the default settings, three kinds of logs are prepared about Anacleto's functioning:

- admin.log
- indexing.log
- userevents.log

These can be found in the set log directory. Logs, broken down by days, create files. It means that every day new log files are created. The archive files can be accessed in the

```
<log name>.<date>
```

form e.g.:

```
userevents.log.06-11-2005
```

The `admin.log` records events referring to administration, the `index.log` events referring to the indexing process, while the `userevents.log` contains primarily the data from the query and responses initiated by the users – currently in the anonymous mode, that is, the actual habits of the user are not revealed. The first two serves as control mechanism for the occasional errors, the third one offers a mirror image of database usage.

Every log file records three data:

1. the date and the time of the entry with a millisecond precision
2. which class of the program has declared the entry
3. the message – it can be a written explanation, a symptom description, the parameters, the time period required to fulfill a task etc.

The files can be accessed one by one.

customization

Basic principles

Java Server Pages

The structure of Anacleto follows the design model of the so-called Model-View-Controller (MVC) where the *Model* is the data collection, the *View* represents the user interface, and the *Controller* is the operational logic of the program. This three 'interfaces' are strictly separated from each other. Since Anacleto is an open source code, both the Controller and the Model interface can be overwritten; still it is the easiest and also the most typical to customize the user interface. In this chapter this aspect will be discussed at length. Here, view will be understood as interface.

The parts of the view that receive data directly from the server side will be created with the 'Java Server Pages' (JSP) technology. The JSP is like a regular HTML page, in which special elements are embedded, elements that while servicing the server can replace with concrete values from the program.

```
<bean:write name="showDocumentForm" property="name" />
```

This command inserts the `name` variable of the `showDocumentForm` program into the HTML page, which variable, in the current case, represents the unique ID of the document served.

Besides this 'data storing' function there are elements that accomplish relatively simpler functions (checking the logical conditions, array traversing etc.).

```
<logic:present name="searchResultForm" property="foundResult">
  ... // do something with the foundResult variable
</logic:present>
```

Here the hit list's (`searchResultForm`) `foundResult` variable is checked. If there is such a variable, it means there is a hit. In this case, in the ... block further commands are executed. If there is HTML in the block, it is displayed in case of positive answer.

It is important to note that this happens on the server side, so - compared to a client side JavaScript block - this commands do not run on the user computer, which will receive only the final result, the evaluated, the already data 'filled' HTML page.

Tag libraries

In JSP according to the function of the tags these are grouped in so-called tag libraries. Tags can only be used if these are previously imported on the page. There are a number of tag libraries. The basic variant of Anacleto uses the tag libraries belonging to the

Struts framework, which controls the entire MVC process.

The Anacleto is mostly using: bean, html and logic.

bean
: it serves to control the implementation of data resources. As previously described, it implements into the page the given variant's valid value at that moment and at that place.

html
: it displays various HTML tags, occasionally updating them with the values sent by the program

logic
: tags that implement simple functions, e.g. branch or array traversing, or that implement logical conditions

It is extremely easy to import tags:

```
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"   prefix="bean"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"   prefix="html"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic"  prefix="logic"%>
```

The <%@ %> is a general JSP tag used to perform special commands. In this case, such a command is taglib. Its task is to import taglibriaries. It has two parameters:

uri
: the unique identifier of the taglibrary. Attention! This is not an URL, nor it is arbitrarily chosen. Every taglibrary has a recorded unique identifier.

prefix
: the name of a namespace reserved on the page for the taglibrary. If on the page a tag is used, then it has to be used together with the prefix. This is how it can be differentiated from other tags in different name spaces (e.g. in default HTML). The syntax is:

```
a name space, or the prefix
 |
<bean:message key="searchres.title" />
      |
      the tag name
```

Every tag (as in HTML, or generally in XML) has its own property list. These can be found at:

- bean: Struts Bean Tags
- html: Page Construction Tags
- logic: Struts Logic Tags

Besides the above described, other taglibraries' description can also be found, but in the basic implementation of Anacleto these play a smaller part.

Data resources

In Anacleto the JSP can receive two types of data from the server:

- Dynamic data from the so-called JavaBean
- And the static data from the so-called resource bundle

JavaBean

The JavaBeans are a special Java class, which contains the data sent by the user (by link or HTML main form), and where the active program, parameterized on these data, returns a response. For example, from the table of contents the title of a page is selected and through an URL the server receives the unique identifier of the page. In the URL the exported parameters arrive into a JavaBean (`showDocumentForm`) and from here it is read by the respective controller (`showDocumentAction`), which on the basis of the parameters decides, which program stream situated at a deeper level have to be activated in order to open from the database the respective document. The returned result (the text, title and other necessary data for accessing the document) is saved in the `showDocumentForm`.

Up to this point, only data traveled from one place to another, and the data can be collected in a formatted document only on a JSP page. To prepare formatting, one has to know, what data is at its disposal, respectively what kind of data is stored in the JavaBeans. This is all that the designers need to know about the functioning of the program.

To write out data the most common procedure is the following:

```
<bean:write name="showDocumentForm" property="name" />
```

This command simply writes out the name property of the showDocumentForm bean. Since the JSP tags are implemented on the server side, which happens earlier than the HTML code interpretation, the JSP tags can be also used embedded in the HTML tag:

```
<div id="<bean:write name="showDocumentForm" property="name" />">
```

Some important JavaBeans:

- searchResultForm – hit list
- showDocumentForm – the document

Resource bundle

Resource bundles are plain text files in which data are stored in a key=value form. According to its characteristic it is static, non-variable data are stored here, primarily the captions of the site (e.g. static texts, links, main form fields, alternative texts to pictures etc.). One of the Java characteristics of the resource bundle is that they may have different variants in more than one language and the current one will be the language of the respective linguistic environment. The files can be found in the

```
WEB-INF/classes/com/anacleto/struts
```

library. Their names are:

```
ApplicationResources.properties
ApplicationResources_it.properties
ApplicationResources_hu.properties
```

The default language is English. If the browser is set in a different language, Hungarian

or Italian, then the responses will be selected from the _hu, respectively, from the _it files, which means that on the interface there is NOTHING to be changed in oder to be able to read the user's own language variants.

In the ApplicationResources.properties file:

```
application.nexthit  = Next Hit
```

The ApplicationResources_hu.properties, respectively in the Hungarian language file:

```
application.nexthit  = K\u00F6vetkez\u00F5 tal\u00E1lat
```

or

```
application.nexthit  = Következő találat
```

The implementation of the data coming from here is possible as follows:

```
<bean:message key="application.nexthit" />
```

which on the response page according to the language of the browser is displayed as

```
Next Hit
```

or

```
Következő találat
```

Important: after editing ApplicationResources.properties file, the server has to be restarted if the new changes are to be activated.

Summary

Every data element supplied by Anacleto arrives separately on the JSP page. This gives the opportunity to the user to

- customize the service to an arbitrarily chosen extent (logos, pictures, colors, css, client side technologies can be used, changed as best suited without influencing in any way the functioning of Anacleto – which is not the case with NXT for example)
- create parallel interfaces next to each other (e.g. frames and simple interfaces without JavaScript for the simplest browsers, or wap interfaces for palmtops, mobile phones)
- easily create an XML communication interface
- send data not only to the browsers, but also to other programs, thus realizing the interoperability of the service
- easily create the i18n, that is, the internalization, since all there is to do is to translate only one language file to the newly added language, the rest is taken care of by Anacleto

further literature

There are a number of books, dissertations and articles on JSP.

- Hans Bergsten: JavaServer Pages. 2nd ed. O'Reilly, 2002. ISBN 0-596-00317-X., 3rd. ed. O'Reilly, 2002. ISBN 0-596-00563-6

- Larne Pekowsky: JavaServer Pages™, 2nd ed. Addison Wesley, 2003. ISBN 0-321-15079-1
- James Goodwill: Pure JSP -- Java Server Pages: A Code-Intensive Premium Reference. Sams, 2000. ISBN 0672319020.
- Goodwill–Hightower: Professional Jakarta Struts. Wrox, 2003. ISBN 0764544373
- Shachor–Chace–Rydin: JSP Tag Libraries. Manning, 2001. ISBN 1-930110-09-X
- David M. Geary: Core JSTL: Mastering the JSP™ Standard Tag Library. Prentice Hall, 2002. ISBN 0-13-100153-1
- Ted Neward: Server-Based Java Programming. Manning, 2000. ISBN 1884777716
- Fields-Kolb: Web Development with JavaServer Pages. Manning, 2000. ISBN 193011012X

Indexing scheme

The indexing scheme is an XSL file that serves to mark in the XML and HTML files the tags to be indexed. Anacleto is first of all filtering the XML and HTML documents through this indexing scheme. The XSLT processing appears here as a pipe: its output is not a file, but a variant stored in the memory, which will interpret the program executing the indexing.

The tags to be indexed are marked by the index name tag, which – in order not to collide with other tags - is embedded in a namespace.

```
<xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns:in="http://www.tesujionline.com.org/default-html-indexsheet" >
```

The index tag can have the following attributes:

indexed	to be indexed
stored	to be stored
tokenized	to be tokenized
termVectorStored	
field	field name