

OpenSHORE

0.0

User Guide

2.01

www.openshore.org

This is the OpenSHORE User Guide

Project OpenSHORE

Copyright (C) 1999-2002 software design & management AG

OpenSHORE is Open Source Software; you can redistribute it and/or modify it under the sd&m Common Public License. You should have received a copy of this License along with OpenSHORE (see `sdm-cpl-vXX.html`).

OpenSHORE is distributed in the hope that it will be useful, but **WITHOUT ANY WARRANTY**; without even the implied warranty of **MERCHANTABILITY** or **FITNESS FOR A PARTICULAR PURPOSE**. See the sd&m Common Public License for more details.

If you find any bugs, unclear text passages or errors, feel free to report them to www.openshore.org (a suggestion how to correct it, will be appreciated).

History

<i>Version</i>	<i>State</i>	<i>Date</i>	<i>Author(s)</i>	<i>Description</i>
2.0	finished	15.11.2002	Tammo Schnieder	Translation from the “Anwenderhandbuch” Version 1.2.2 from Norma Korta, Klaus Mayr, Tammo Schnieder and Helge Schulz
2.01	Finished	25.11.2002	Tammo Schnieder	Added changes from Denis Kuniß

Content

1 Introduction	1
2 Overview	2
2.1 How does OpenSHORE work?	2
2.2 Model Abstraction And Model Layers	4
3 User Interface - OpenSHORE in the Browser	6
4 Example Application "Calculator"	7
5 First Steps	8
5.1 Open your Browser	8
5.2 Entering the documents	8
5.3 Navigation through documents	10
6 Working with OpenSHORE	12
6.1 Open / prepare your Browser	12
How to start OpenSHORE in your Browser	12
6.2 Start surfing your documents	12
How to search with document or object name	13
How to search from document type or object type	13
How to search from the meta-model	13
How to search with a full text search engine	14
Background of using wildcards and regular expressions	14
Dialogs for searching with document resp. Object name and full text search	16
6.3 Navigation in the documents	17
6.3.1 Navigation with info links	17
How to use the info link of a document	18
How to use the info link of an object	18
6.3.2 Navigation on links of relationships	19
How to navigate to the definition of a relationship	19
How to navigate to the source of a relationship	19
How to navigate to the sink of a relationship	19
How to navigate to the definition of a variable (object)	20
How to navigate to the definition of the meta model	20
6.4 Standard queries	20
6.4.1 Individual queries, standard queries and Prolog rules	21
6.4.2 How to execute a standard query	21
6.4.3 How to execute an individual query	22
6.5 Displaying undefined resources	23
How to find the source of an undefined resource	23
6.6 Display data base summary	24
6.7 Error messages in the Browser	24
6.7.1 SHORE - undefiniertes Objekt	24
Applied action	24
Message	24
6.7.2 SHORE - Fehler (Dateizugriff)	24
Applied action	24
Message	25
6.7.3 The requested URL could not be retrieved	25
Applied action	25
Message 1	25
Message 2	25

1 Introduction

What is OpenSHORE good for?

Information is very complex.

It is a difficult task to find a way to make information easier to learn or find a better way to transfer information.

OpenSHORE offers a way to present structured documents, so that you can faster get the information out of it.

What is OpenSHORE?

The semantic hypertext object repository (SHORE) combines features of a repository with those of a hypertext system with internet technologies. OpenSHORE can store every structured information and offers a comfortable navigation with a normal browser in an intranet, extranet or internet.

Additionally OpenSHORE offers possibilities for complex queries by using a logical language (Prolog).

Who should read this guide?

This guide is for everybody who works with OpenSHORE or is interested, how it works.

If you use OpenSHORE, you navigate (or surf) through the system and execute standard queries. Or if you are an advanced user, you write queries on your own. For writing queries, you need to have basic knowledge in the programming language Prolog.

Maybe you are also creating documents, sources or other structured text which you are bringing into OpenSHORE and surf them.

The installation and customization of OpenSHORE and the import of documents are not in the scope of this guide. These tasks are content of the OpenSHORE administrator guide.

How can I use this guide?

Everybody uses a handbook in his very own manner. Dependent on the knowledge or information needs, you can use this guide in several ways.

To help you to get used with this guide, here is, what we thought about the structure of this document.

In chapter 2 "Overview" you will find an explanation what the background and the concept of OpenSHORE is. We recommend this chapter for all users.

In this guide you will find an example that we use in some chapters. It is introduced in chapter 4 Example Application "Calculator".

If you want to get information about what is displayed in your browser, read chapter 3 "User Interface". Here we explain the areas in the browser and the menus. More detailed information of the dialogs are given in chapter 6 "Working with OpenSHORE".

If you have never worked with OpenSHORE before, we suggest to go to chapter 5 "First Steps" and to work through it with our example before you start to surf through your own documents.

Guides for solving typical questions are written in chapter 6 "Working with OpenSHORE". Also in this chapter, we recommend to get introduced to the single steps with our example.

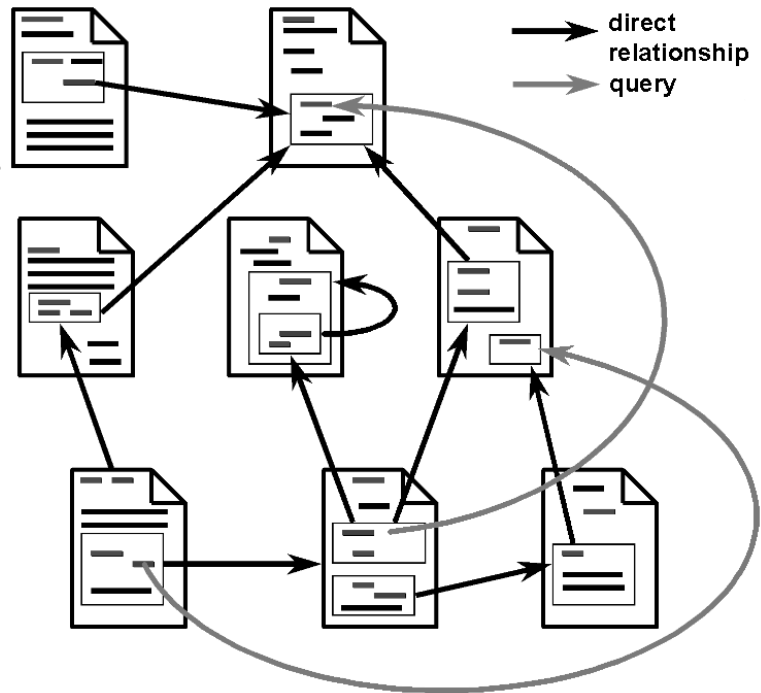
After you are introduced in OpenSHORE, we hope that you have an easy start for surfing your own documents.

2 Overview

2.1 How does OpenSHORE work?

What is stored in OpenSHORE?

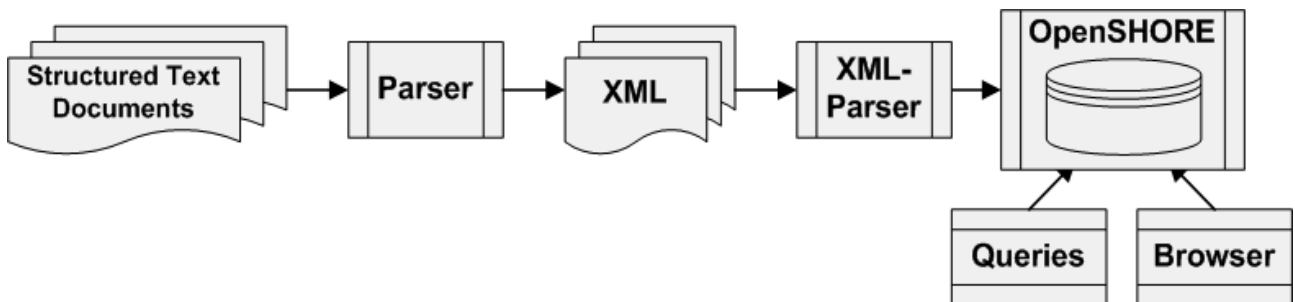
OpenSHORE is a repository for structured documents. It is open for all types of documents, for example all documents that are part of a software development project. OpenSHORE links the content of different kinds of documents of different file formats in one hypertext system. Parts of a document are marked as objects. Between these objects and also between documents there are relationships. The documents, their objects and relationships are stored in OpenSHORE and can be investigated by navigation, by searching and by executing queries.



Picture 1: Relationships between documents

How do my documents get into OpenSHORE?

All documents are internally stored in XML in OpenSHORE. Before a document is imported in OpenSHORE, it is converted from its original format by a parser (or pattern matcher) which analyzes the structure of the document for relevant objects and relationships. This document and object structure is additionally stored in OpenSHORE so that complex queries can be executed.



Picture 2: Schematic import mechanism of OpenSHORE

What are objects and relationships in OpenSHORE?

Imagine, that you are new in a long running software development project. You are interested in the structures and informational connections of the sources. And you are also interested in textual project documents like concepts or documentation. All these contents of all those files, are somehow connected with each other – maybe directly or indirectly. To realize these connections is not always easy.

Objects in OpenSHORE are sections in a text (like a use case description) or can also be single words (like a variable name). These objects can also be nested: e.g. in an object function, there is an

object „local variable“.

Relationships are the connections between objects and also documents (mixed relationships documents/objects are also allowed).

How are objects and relationships recognized?

If you want to automatically recognize objects and relationships, your texts must be structured. Then you can use a parser or a pattern matcher to recognize these structures.

An example for structured texts are computer programs. A software developer is interested in structures and connections. To figure these connections out is sometimes not very easy. To get these interconnections exactly, you need to have a parser for this computer language. This parser has to use symbol tables, do semantic analysis and compute implicitly contained informations. By this technique, name conflicts can be avoided and the correct definition of an object can be marked – even if you only know a part of its name (exactly: namespace).

For OpenSHORE there are some parsers and pattern matchers for programming languages available (see www.openshore.org).

How does OpenSHORE know about objects and relationships?

Objects and relationships are marked with specific XML tags. When OpenSHORE imports documents, marked objects and relationships are extracted and stored internally.

Which objects and relationships are known by OpenSHORE can be configured in a free definable meta-model.

The parser must insert markup accordingly to the defined meta-model.

How do I see objects, relationships and documents?

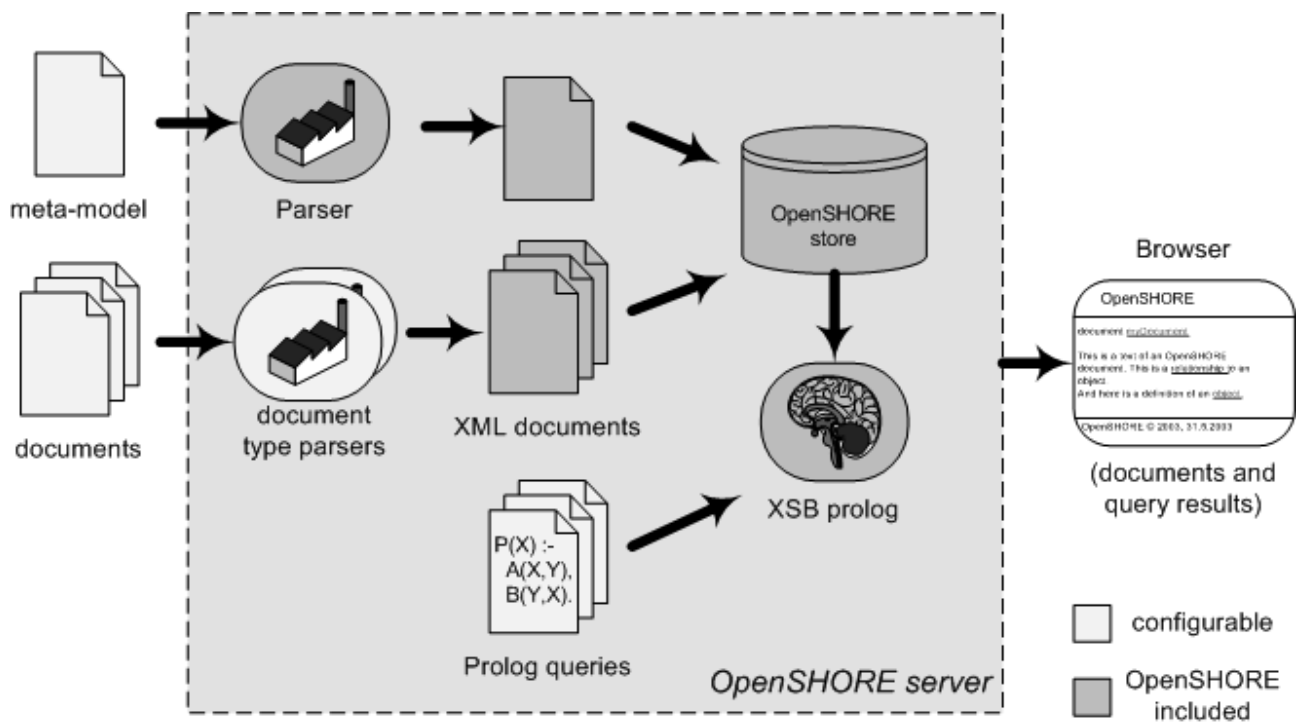
You can see all imported documents that are imported in OpenSHORE directly in your Browser.

The HTML that you see is dynamically generated from the internally stored XML. By this conversion, all entries of objects and relationships are displayed in specific colours (objects are red and relationships are blue). And they are linked with each other, so that a hypertext navigation is possible.

Each time, you click on a link in OpenSHORE, this will put a request to OpenSHORE and it returns the right document. All links are working in both directions: from the source to the sink and vice versa!

By using a logical programming language (XSB-Prolog), you can also define complex queries.

The answers of such a query is presented as HTML in your Browser and is linked to the appropriate documents and objects.



Picture 3: OpenSHORE schematic architecture

What kind of queries are possible?

Beside the Hypertext-Navigation, you can also define queries to get information out of the OpenSHORE database. You can define simple queries, or complex ones and also recursive queries! OpenSHORE has a complete Prolog engine included!

With this possibility, it is also possible to get tree-like structures out of a network of relationships. Typical queries of software developers that use OpenSHORE are:

- show me the calling hierarchies
- show me the inheritance hierarchies
- show me unused objects
- show me multiple object definitions
- show me the neighborhood of an object (and also limited by...)
- show me all Classes that realize a specific Use Case
- show me all requirements that are not yet reflected in the detailed specification

If you want to define a query, you must have a look at the meta-model. Then you can define which objects and which relationships you want to use in your query and how they are related to each other to get the right answer.

2.2 Model Abstraction And Model Layers

What is reflected in OpenSHORE?

OpenSHORE reflects a model of a system.

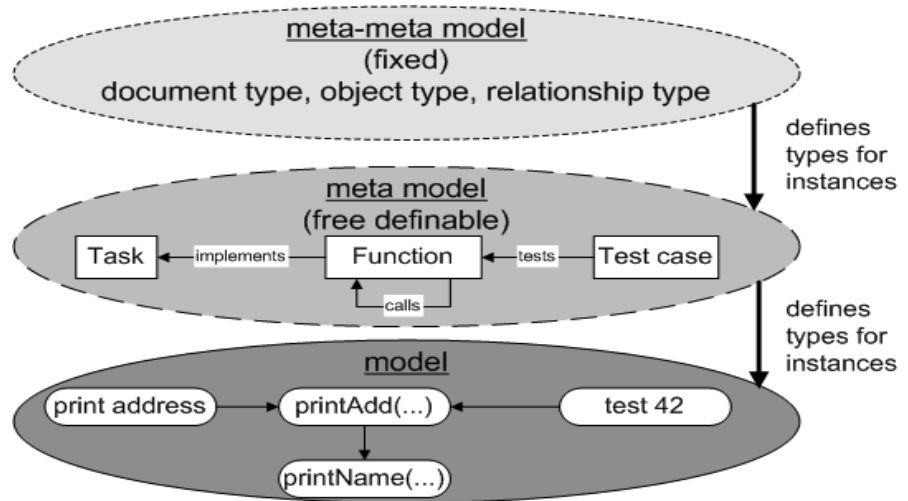
The structure of this model is defined by a meta-model. This meta-model can be free defined using some predefined types (they are fixed in the meta-meta-model). These fixed types are object type, document type and relationship type and are fixed programmed.

An example: in a software development project there are interface descriptions, concept papers, Java sources and COBOL sources. Each of these are own document types.

Object types could be: in a concept paper the use cases, in a use case the actors or the processes. Object types of an interface description could be the name of an interface, the destination system or used data types. In the programming sources there are following object types: names of programs and classes, methods, parameters, variables, data types and so on.

And there could be some relationship types like:

- "program implements use case": the name of the use case is part of the comment of the program.
- "implements interface": an identifying name in a program points to the interface description.
- "calls": a call of a subroutine or method.
- "calls": an embedded database call (e.g. SQL), which points to the database (or table) definition.
- "program includes copybook": a document uses other documents.



Picture 4: OpenSHORE model layers

3 User Interface - OpenSHORE in the Browser

In this chapter, you will be introduced to the elements of the user interface.

The starting page

After you have started OpenSHORE in your browser, you probably have to wait until all elements of the user interface are loaded and internally compiled (this depends on the browser and the used Java version). Meanwhile, you will see the picture of an open shore. Below the picture you will find links to the FAQ and to existing documents.



Picture 5: OpenSHORE is loading the user interface

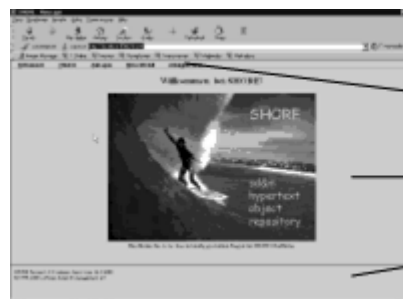
The user interface in the browser

The largest area of the browser window is the upper of two frames, the so-called *document area*. If you have just started OpenSHORE and have no document loaded, a picture of a surfer is displayed here.

This is the main area in the browser. While you are surfing through OpenSHORE, documents are displayed in the document area.

The lower frame is called the *info area*. It is used for displaying information about objects and documents.

You can change the height of this frame by moving the borderline between the document area and info area.



Picture 6: The areas of the user interface

OpenSHORE menu
document area
info area

In both areas, you will see hypertext markup. The meaning of the colours are explained inside of the chapter 6.3 under the section "Colours of links" on page 17.

Menu and entry dialogs

By choosing the menu entries „Dokumente“, „Objekte“ and „Metamodell“, you have a quick access to your documents inside your OpenSHORE instance. The menu entry „Anfragen“ (queries) opens a dialog to execute queries. Under „Administration“ you will find all functions to administer the OpenSHORE server from the browser.

Menu items which open a new dialog are marked with three dots after the name (e.g. „Suche Objekt...“).

All dialogs are Java dialogs, which partly can be resized by dragging the window border. You can jump between the input fields by using the TAB-key of your keyboard.

The dialogs also support the functions <Ctrl-x> (cut), <Ctrl-c> (copy) and <Ctrl-v> (paste). But only inside of OpenSHORE as the OpenSHORE user interface is not using the system clipboard. Input dialogs are explained in more detail in chapter 6 "Working with OpenSHORE" on page 12.

4 Example Application "Calculator"

This example shows you, how OpenSHORE can help you to get comfortable in foreign sources. You can navigate from the use cases to the implemented classes by using relationships. The example realizes a calculator. The user interface was implemented in Java, the arithmetic functions are implemented on a mainframe in COBOL. Between these languages is a middleware, and the specification are written down as a use case.

All files are already stored in XML that is OpenSHORE compliant. You will find these files in the folder "examples".

All examples and syntax descriptions in this guide are based on this calculator example.

In the Chapter 5 "First Steps" you will find some ways how to navigate in your documents. In an easy example you find how to navigate from your requirements to their implementation.

5 First Steps

Now we start. In this chapter we run through the example application so that you can see, how to use OpenSHORE.

You need the URL of the OpenSHORE system where you will find the example application "calculator".

We suggest, that you already have read the chapters "Example Application "Calculator"" on page 7 and "Overview" on page 2 but they are not mandatory for this chapter.

In the following chapter "Working with OpenSHORE" on page 12 all dialogs are explained more detailed.

5.1 Open your Browser

First, you should be sure that you fulfill these requirements. You need

- a network connection to an OpenSHORE server (or it runs on the same machine as localhost)
- a Java capable browser, where you are allowed to execute Java

1. Open your browser
2. Enter the URL of an OpenSHORE server where you find the example application „calculator“.

Syntax `http://<address of the server>:<port of the server>/openshore/`

Example `http://calculator.openshore.org:8080/openshore/`

Depending on the bandwidth and your system, it can take between a few seconds and a few minutes until the OpenSHORE user interface is loaded, because in most browser, after loading, the java applet it has to be compiled.

Note: If OpenSHORE is restricted for specific users, you will have to enter a user name and password before loading. You will get your user name and password from your OpenSHORE administrator.

3. Add the URL to the favorites of your browser.

If your connection was successful, you will see the starting page of OpenSHORE. You will find more information about the user interface in the chapter "User Interface".

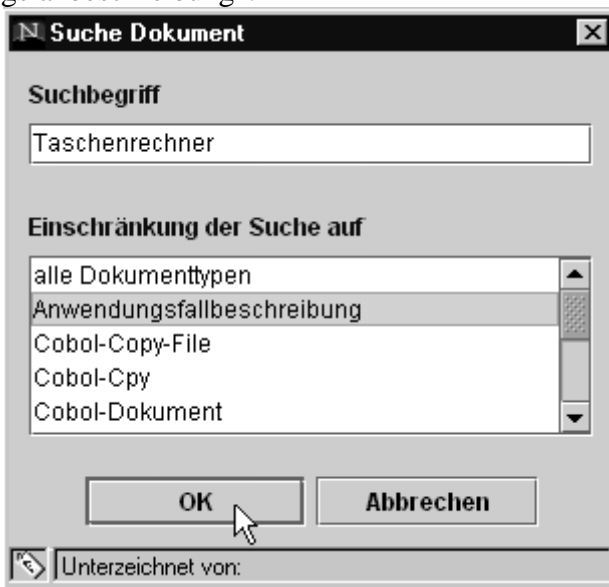
Tip: You stop working with OpenSHORE by navigating to a different web page or by closing your browser.

5.2 Entering the documents

After the OpenSHORE user interface is loaded by your browser, all imported documents are available for you. If you want to have more documents in OpenSHORE, talk to your administrator. He can import documents into OpenSHORE.

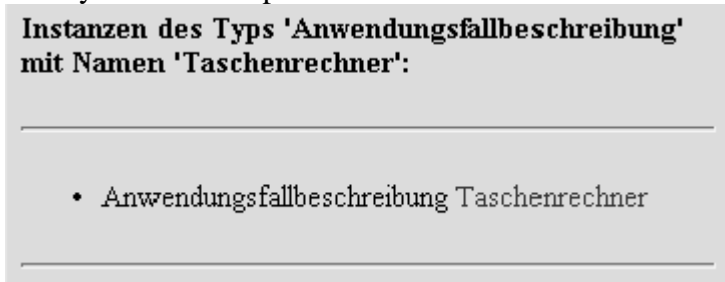
In our example, we are looking for the document "Taschenrechner" of the type "Anwendungsfallbeschreibung".

1. In the menu „Dokument“ click on „**Suche Dokument...**“
The dialog "Suche Dokument" is opened.
2. In the field "Suchbegriff" enter the word „Taschenrechner“.
3. In the field "Einschränkung der Suche auf" select the document type „Anwendungsfallbeschreibung“.

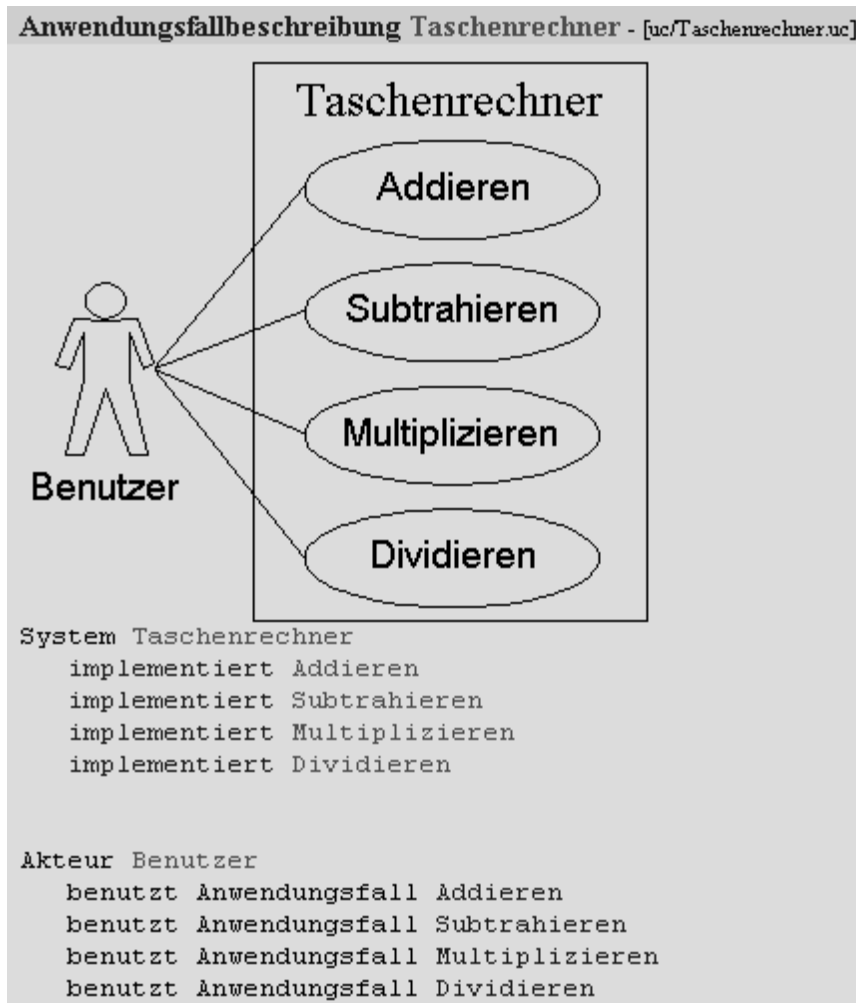


Picture 7: Dialog "Suche Dokument"

4. Click OK or press <Enter>.
The result of your search is presented in the document area as a linklist.



- Click on the filename „Taschenrechner“ which is displayed as a link. OpenSHORE opens the linked document and displays objects in red and relationships in blue.



Now you have opened the document "Taschenrechner" of the type Anwendungsfallbeschreibung. From here you can navigate into other documents.

5.3 Navigation through documents

In this example we want to navigate from the use case (Anwendungsfall) "Subtrahieren" to its implementing objects.

You have still opened the document Anwendungsfallbeschreibung "Taschenrechner".

- In the document area click on the red marked object "Subtrahieren". In the info-area OpenSHORE displays all information for this object.
- Click on the source document of the relationship „-- realisiert-->“. OpenSHORE opens the appropriate document in the document area, jumps to the selected object and marks the definition.
- In the info-area, click on the blue relationship „-- realisiert -->“. OpenSHORE marks in the document the definition of the relationship.

4. Click on the object "subtract".
In the info-area, OpenSHORE displays all information of this object.
5. In the info-area, click on the relationship „-- ruft_via_Middleware -->“.
In the document area, OpenSHORE opens the corresponding document, jumps to the selected relationship and marks the definition.
6. In the info-area, click on the target document of the relationship „-- ruft_via_Middleware -->“.
In the document area, OpenSHORE opens the corresponding document, jumps to the selected object and marks the definition.

Now you can see, that the use case calculator subtraction (Anwendungsfall Taschenrechner-Subtrahieren) is implemented by the Java object "JBC1Main.subtract()" supported by the Cobol Object "S1::SPC-SUBTRACT".

What you have done now...

Until now you have

- added the OpenSHORE URL to your favorites
- searched and opened a document
- looked at a use case (Anwendungsfall)
- searched its objects and navigated to them
- followed relationships beyond several programming languages.

After this short introduction you are now ready to surf in an OpenSHORE system by yourself. The calculator example will be present when you find explanations or syntax examples in the next chapters.

6 Working with OpenSHORE

You walked your first steps in OpenSHORE and are familiar with the example application "calculator". Now you can look into your own documents.

The order of the following actions is similar to a typical way of working with an OpenSHORE instance. If you haven't read the chapter „First Steps“ on page 8 yet, we recommend to read it first. You should also be familiar with the chapter „Overview“ on page 2 and „User Interface“ on page 6. This chapter offers you to look into basic use cases and to learn them easily.

6.1 Open / prepare your Browser

At first, make sure, that your browser fulfills the needed requirements as stated in chapter "Open your Browser" on page 8.

How to start OpenSHORE in your Browser

1. Start your browser.
2. Enter the URL of your OpenSHORE instance, `http://<address of your server>:<port of server>/openshore/`, and the user interface of OpenSHORE is loading.
Depending on bandwidth and the speed of your system, it may take some time.
3. Add the URL to your favorites (if you haven't already).
The favorite is probably named OpenSHORE.

When the connection is successful, the start page of OpenSHORE will be displayed. More detailed information regarding the user interface is given in chapter 3 "User Interface" on page 6.

6.2 Start surfing your documents

All imported files are stored in OpenSHORE as documents and objects with defined relationships (when we mean documents and objects, sometimes we write *ressource*). Every document and object has a name and a type as defined in the meta-model. You can search either by name or by type. Supposed, that you have a connection to an OpenSHORE server, the server is correctly installed by your administrator and your documents are imported to the system. Now you have several ways to open a specific ressource.

You can search a specific instance of a document or an object by

- the name of the ressource
- the type of the ressource or
- by navigating from the meta-model
- If there is a fulltext engine installed, you can also use that.

Searching by navigating through the meta-model is less direct, but the only way to search for relationships. The advantage is, that the types are displayed here in the context of other types as defined in the meta-model document.

You can search for any text or piece of text which is contained in an imported document, when a search engine is installed at your OpenSHORE server. You can use it, when you open in the menu the entry "Anfragen" and the menu entry "Volltextsuche" is activated.

How to search with document or object name

1. In the menu „**Dokument**“ or „**Objekt**“ click on „**Suche Dokument...**“ or „**Suche Objekt...**“ respectively.
The dialog „**Suche Dokument**“ or „**Suche Objekt**“ opens respectively.
2. In the field „**Suchbegriff**“ enter a regular expression for the name of the resource that you are looking for (e.g. '.*' for any characters, see also "Regular expressions" on page 14).
3. In the field „**Einschränkung der Suche auf**“ (filter the search) mark the document type resp. Object type you want to search. You can select multiple types by holding „Ctrl“ while you click.
4. Click „**OK**“ or press <Enter> to start searching.
The result is displayed in the document area as a list of links.
5. Click on the name of a resource – it is made as a link.
OpenSHORE opens the corresponding document.
If you searched for an object, your browser jumps to the selected object and marks the whole definition.

How to search from document type or object type

1. In the menu „**Dokument**“ or „**Objekt**“ respectively, click on the document or object type you are searching for.
In the document area all documents resp. objects of the selected type are displayed as a link list.
2. Click on the name of a resource, it is made as a link.
OpenSHORE opens the corresponding document. If you searched for an object, your browser jumps to the selected object and marks the whole definition.

How to search from the meta-model

1. In the menu „**Metamodell**“ click on a type you wish to search.
OpenSHORE opens the meta-model, jumps to the selected document, object resp. relationship type and marks the corresponding name.

The colours of the links in the meta-model are

- red for documents,
- green for objects and
- blue for relationships.

When documents are loaded, the colours have a different meaning (see inside of the chapter 6.3 under the section "Colours of links" on page 17).

Note: All colours can be changed by the administrator. In this guide we use the standard colours.

3. Click on the name of a type, it is made as a link.
In the info area you will see a link list with corresponding documents (or instances of this type).
4. Click on a name to open a document.
OpenSHORE opens the corresponding document. If you searched for an object, your browser jumps to the selected object and marks the whole definition.

How to search with a full text search engine

1. In the menu „Anfragen“, click on „Volltextsuche“ or press the shortcut <Strg-V>.
The dialog „Führe Volltextsuche durch“ opens.
2. Enter a search phrase.
Search phrases can be concatenated with the keywords AND and OR, and you can use wildcards. A "*" means any characters and "?" means exactly one character. You can not use regular expressions here, only "*" and "?".
3. In the list „Einschränkung der Suche auf“ (filter) you can choose document types which you want to be searched. Mark multiple types by holding "Ctrl" and clicking with the mouse.
4. Activate the checkbox "Groß- /Kleinschreibung" if you want to search case sensitive.
5. Select, how many hits you want to see on one result page.
6. Start searching by clicking „OK“.

Background of using wildcards and regular expressions

Usage of jokers

The parser determines the entire name of an object. The name is often a concatenation of the document name and the object name, because OpenSHORE needs unique names. They are the identifiers of resources. If you are searching an object, and you don't know if a document name is preceding the object name, then you should use ".*" as a "joker" to build the search string.

Tip: Not all parsers use the same logic and put the document name in front of the object name. So we recommend to use jokers.

Regular expressions

The string ".*" that we call joker is an example of using regular expressions. When you are searching for document or object names, you can use other regular expressions, too. Here is a very short introduction:

Regular Expressions are a replacement for a pattern of characters and strings.

Here are the most often used regular expressions:

The symbols *, + and ? are standing for repeating an alphanumeric character. They have the following meaning:

Symbol	Is the replacement for...
*	...no or any occurrence of the preceding character; the classic joker.
+	...one or more occurrences characters of the preceding characters..
?	...no or exactly one occurrence of the preceding character; also a classic joker.
.	...no or any single character.

Here are some examples, how you can use these symbols:

Expression	Description
12*	This expression matches strings that begin with a 1 and no or any occurrences of 2: "1", "12", "122", "1222" etc.
12+	The same like before, but there must be one or more occurrences of a 2: "12", "122", "1222" etc., but not "1".
12?	Here, there must be one or no occurrence of 2: only "1" or "12" are matching this expression.
1?2+	Here, the expression matches no or one occurrence of 1, followed by one or more occurrences of 2, for example: "12", "2", "122" or "2222".
12.*	Matches all strings, that begin with 12 like: "12", "123", "12th" or "12-hours-long".

You will find a table of the syntax of regular expressions in the appendix.

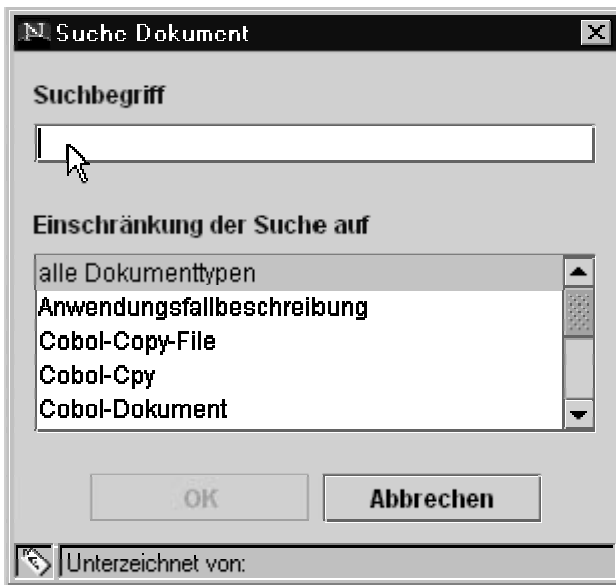
Full text searching

When you are searching with full text, you can not use regular expressions, but the special characters * and ? as used in the Internet. Here are some examples for search phrases:

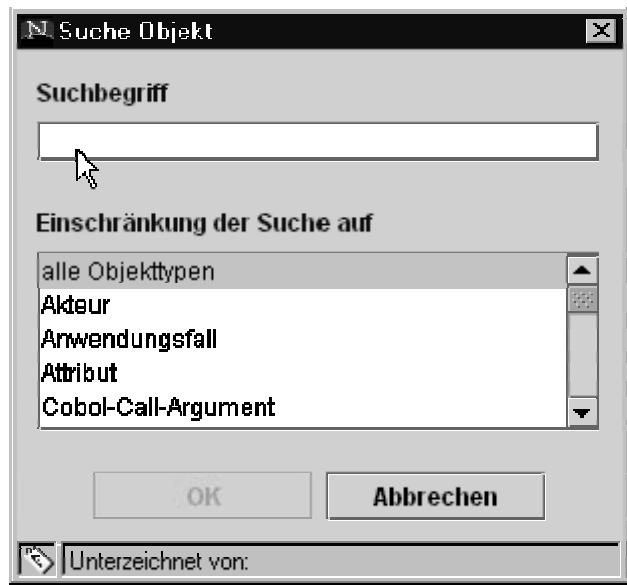
Search phrase	Documents are found that contain the following:
Währung	The word "Währung"
currenc*	Words that start with „currenc“
free climbing	The phrase „free climbing“
"free climbing"	Also the phrase „free climbing“
Eur??a	Words, that are beginning with „Eur“ followed by no or any of two characters and end with „a2 e.g. "Europa"; „Eura“ or "Eureka"
Euro and foreign currency	The word "Euro" and the phrase "foreign currency"
Jekyll and Hyde	The word "Jekyll" and the word "Hyde"
"Jekyll and Hyde"	The phrase "Jekyll and Hyde"

Dialogs for searching with document resp. Object name and full text search

"Suche Dokument" / "Suche Objekt"



Picture 8: Dialog "Suche Dokument"



Picture 9: Dialog "Suche Objekt"

In the field "Suchbegriff" (search phrase) you can enter the full name or use the joker ".*" or use other regular expressions. The full name of an object often is a concatenation of the document and object name.

Even if you are searching for an object in an already opened document, you have to enter the complete name, as the search always uses the whole data base.

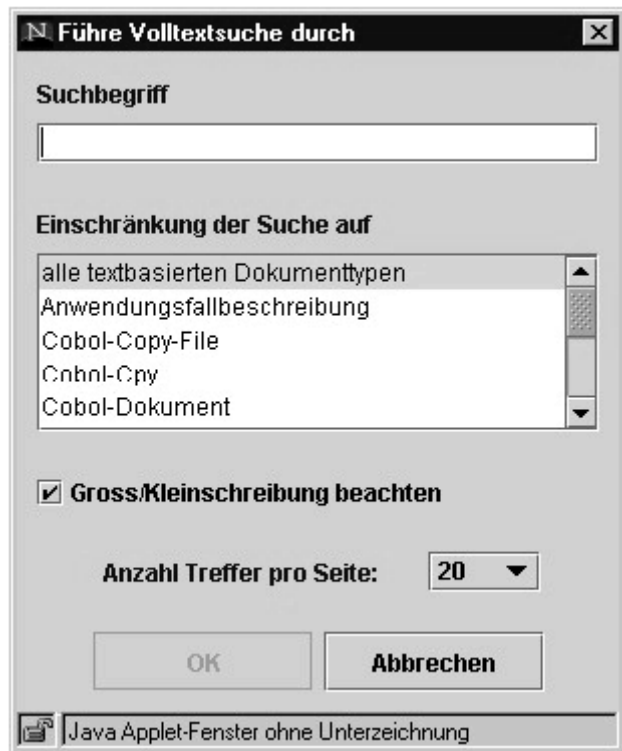
In the field "Einschränkung der Suche auf" (filter), you can select one or more document resp. Object types by holding <Ctrl> and clicking on the types with your mouse.

Full text search

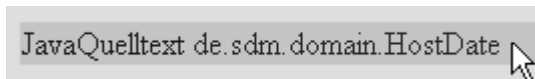
You can enter any search phrase as "Suchbegriff" and you can use "*" as a replacement for no or any occurrence of characters and "?" as a replacement for no or one occurrence of one character. You can concatenate search strings logically with AND and OR. Strings that are enclosed in double hyphens (" ") are searched as a phrase (not as single words).

6.3 Navigation in the documents

Inside the documents, you can navigate by clicking on links. In the status line of your browser you see the name of the loaded document. If you move the mouse pointer over a link, the path of the target is displayed. The type of the document and the document name are displayed in the headerline of each document (see "Picture 11: displayed document type and document name in the document area").



Picture 10: Dialog "Führe Volltextsuche durch"



Picture 11: displayed document type and document name in the document area

Colours of links

In the documents, links are distinguished between *links of relationships*, *info links* and *multi info links*. As defined in the default, links for relationships are displayed in blue colour, info links are displayed in red and multi info links are green. In the meta model these colours have a different meaning. Here are the colours of the standard configuration:

	Meaning in document	Meaning in meta model
blue	Link of relationship	relationship
red	Info link	document
green	Multi info link	object

Note: These are the standard colours that can differ from your installation, if the administrator uses different stylesheets.

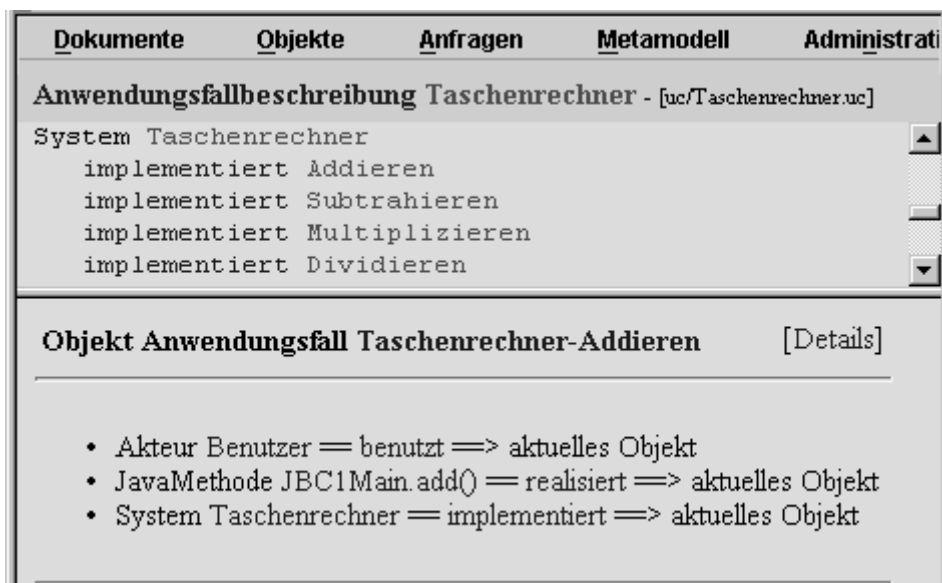
6.3.1 Navigation with info links

The red coloured info links are displayed in a document, where objects are defined. These are for example definitions of methods, use cases (Anforderungen), and also definitions of variables. All displayed objects are defined in the meta model.

If you click on an info link, this will be a request to the data base of OpenSHORE.

An info link is displayed on the top of each document. If you click on this info link, information of the document is displayed in the info area. These information gives you a quick overview of the document and you can use it as a kind of table of content of OpenSHORE objects and relationships of this document.

Inside of the document, all definitions of objects are displayed as a red info link.



Picture 12: After clicking on "Addieren", document structure in the info area

How to use the info link of a document

Click on the red coloured name of a document at the very top of the document area.

Then OpenSHORE displays the following information in the info area:

- document type and document name
- incoming and outgoing relationships of the document (document is source or sink of a relationship)
- all defined objects inside of the document with their incoming and outgoing relationships
- all matching standard queries. That is, all standard queries, where this document can be used as a parameter and which do not have other parameters. Standard queries are not automatically installed, they must be manually created (typically by your OpenSHORE administrator).

How to use the info link of an object

Click on the info link of an object to display structural information in the info area:

- type and name of the object
- type and name of the document where the object is defined
- incoming and outgoing relationships of the object
- matching standard queries

Note: In the info area, all relationships and objects are displayed in a specific order following these rules:

- Upper case letters prior to lower case letters
- german umlauts (like "ä", "ö" etc.) are following directly after their matching vocals

- "ß" follows after "s"

6.3.2 Navigation on links of relationships

A relationship link is displayed, where an object is used by a relationship inside of a document. For example a used method, a link to a use case (Anforderungen), and also the usage of variables. All displayable relationships are defined in the meta model.

If the relationship is "open ended" – this means, that the source or sink of the relationship is non-existent – OpenSHORE displays a message that you have clicked on an undefined object (undefiniertes Objekt).

Each navigation on a relationship link is internally a directed request on the data base.

How to navigate to the definition of a relationship

In the info area, click on the name of the relationship that is displayed as a blue link.

OpenSHORE opens the appropriate document and marks the definition.

Syntax == name of the relationship ==>

Example

```
• System Taschenrechner == implementiert ==> aktuelles Objekt
```

How to navigate to the source of a relationship

In the info area, click on the name of the source resource of the relationship that is displayed as a blue link.

OpenSHORE opens the appropriate document and marks the definition.

Syntax name of the source == name of the relationship ==>

Example

```
• JavaMethode JBC1Main.add() == realisiert ==> aktuelles Objekt
• System Taschenrechner == implementiert ==> aktuelles Objekt
```

How to navigate to the sink of a relationship

In the info area, click on the name of the target resource of the relationship that is displayed as a blue link.

OpenSHORE opens the appropriate document and marks the definition.

Syntax actual object == name of the relationship ==> name of the target

Example

```
• aktuelles Objekt == gehoert_zu ==> JavaKlasse JBC1Main
• aktuelles Objekt == ruft ==> JavaMethode JBC1Main.handRequest(java.lang.String)
```

How to navigate to the definition of a variable (object)

In the info area, click on the name of the variable that is displayed as a blue link. OpenSHORE opens the appropriate document and marks the definition.

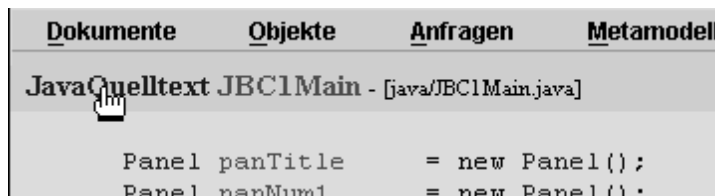
Syntax name of the variable

Example

```
m_blApplication        = false;  
SdmSysmm._applet = this;
```

How to navigate to the definition of the meta model

In the document area, click on the type of the document that is displayed as a blue link.



OpenSHORE opens the meta model and marks the definition.

```
Dokumenttyp JavaQuelltext  
ist ein Quelltext
```

6.4 Standard queries

A major strength of OpenSHORE is the ability to answer complex queries regarding the imported documents, objects and relationships.

OpenSHORE uses Prolog as a query language (Programming by Logic). You can learn Prolog quite easily and it is capable of answering complex queries also with recursion and path tracking (Pfadverfolgung).

If you would like to find single words or textual phrases, you can also use a full text search engine.

Queries in SHORE

The Prolog system in OpenSHORE has knowledge about all structures in imported documents that are marked with appropriate XML and are defined in the meta model.

More exactly, the Prolog system knows all stored documents, objects and relationships, their types and sources and sinks of all relationships. Moreover it knows, which objects and relationships are defined in which documents. This knowledge is presented as so called predicates.

If you write a query, you can write a Prolog rule. Prolog rules are Prolog programs (that is a set of Prolog clauses) which will serve you to offer additionally predicates to the basic predicates (document, object und relation).

How to define Prolog queries is described in an own handbook ("Anfragen in Prolog").

6.4.1 Individual queries, standard queries and Prolog rules

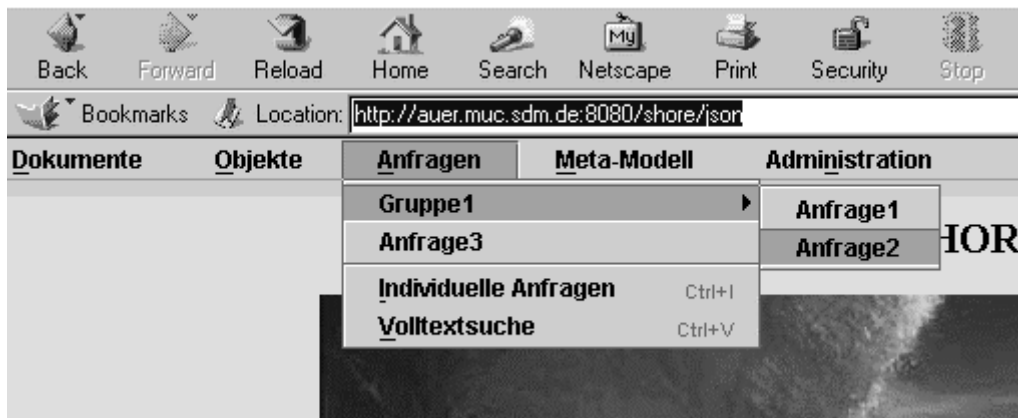
OpenSHORE knows to kinds of queries: individual queries and standard queries.

Individual queries

Individual queries are "per session" queries which are executed online by the system but are not stored after an OpenSHORE session. If you close your Browser, these queries vanish.

Standard queries

Standard queries are commonly used queries that are stored in OpenSHORE. You can recall and execute them as long as they are stored. With these queries, the parameters can be defined as variables so that the concrete values are inserted before the queries are executed.



Picture 13: Menu "Anfragen" with predefined standard queries

Individual queries and standard queries are under the menu entry "Anfragen". When there are no standard queries, you will only see the entries "Individuelle Anfragen" and "Volltextsuche".

Prolog rules and standard queries are created under the menu entry "Administration".

Standard queries can be grouped by defining a category in the dialog for maintaining standard queries (Standardanfragen). The names of these categories are displayed in the menu under "Anfragen" as own menu entries. Under these entries you will find the names of the applied standard queries. Standard queries without a category are directly displayed under "Anfragen".

6.4.2 How to execute a standard query

If you want to execute a standard query, there has to be at least one defined standard query in your system.

1. Under the menu „Anfragen“ click on a standard query.

2. A dialog opens, where you can enter values for the parameters of the query.

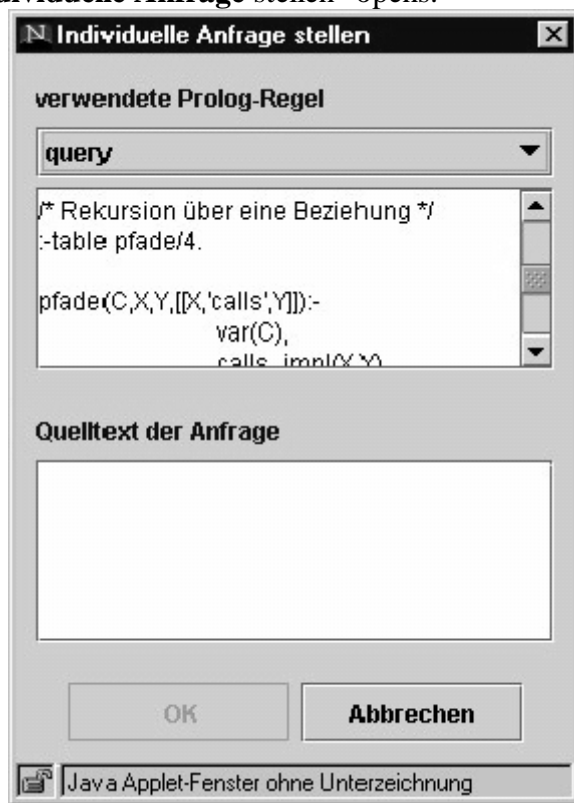
Note: If you leave a parameter empty, it is interpreted as an unbound variable. That means, that for this parameter, all possible values are inserted when the query executes. This may return a huge amount of results.

3. Execute the query by clicking „OK“.

The result of the query is displayed in the document area.

6.4.3 How to execute an individual query

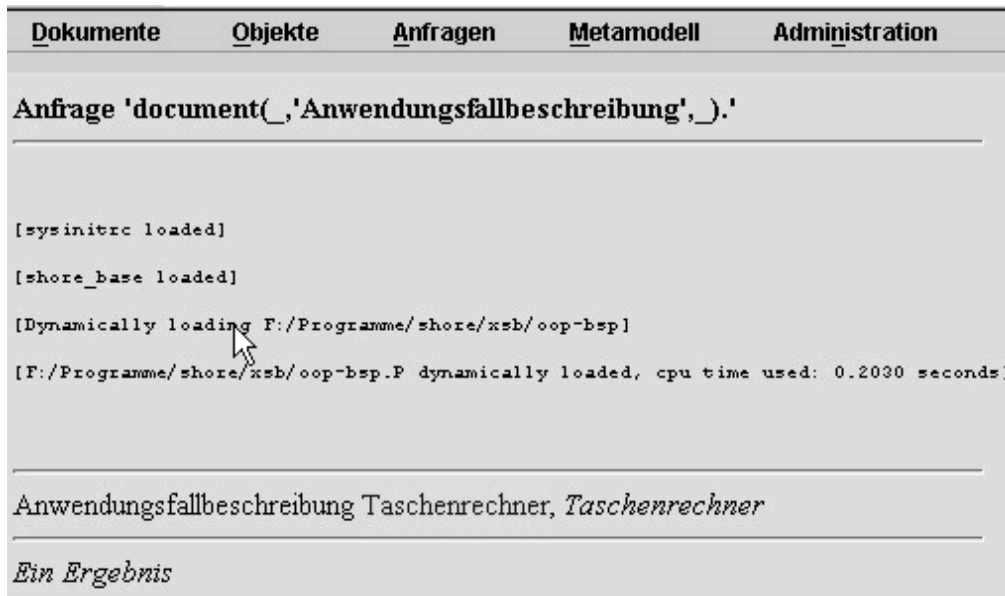
1. In the menu „Anfragen“, click on „Individuelle Anfragen“. The dialog „Individuelle Anfrage stellen“ opens.



Picture 14: Dialog "Individuelle Anfrage stellen"

2. Optionally, you can select a prolog rule from the drop down list „**verwendete Prolog-Regel**“ where predefined predicates that you would like to use are already defined. The source code of the selected prolog rule is displayed in the text field in the middle.
3. Enter the text of the query in the text field „**Quelltext der Anfrage**“. A query is a sequence of predicates that are separated by commas and a dot („.“) at its end.
4. Click „**OK**“.
Your query will be executed by the server.

The result of your query is divided in several areas which are separated by a horizontal line.



```
Anfrage 'document(_, 'Anwendungsfallbeschreibung', _)'  
  
[sysinitrc loaded]  
[shore_base loaded]  
[Dynamically loading F:/Programme/shore/xsb/oop-bsp]  
[F:/Programme/shore/xsb/oop-bsp.P dynamically loaded, cpu time used: 0.2030 seconds]  
  
Anwendungsfallbeschreibung Taschenrechner, Taschenrechner  
  
Ein Ergebnis
```

Picture 15: Result of a query.

1. The first line of the result shows the executed query text.
2. The next lines are the output of the Prolog system. It reports about loaded Prolog rules and eventually displays error messages.
3. In the next lines, you see the result of your query. It is either the string 'Ja' or 'Nein' or all combinations of possible values that makes the query true. Contained documents, objects and relationships are displayed as hyperlinks, so that you can directly jump to the appropriate entity.

6.5 Displaying undefined resources

In OpenSHORE you can store inconsistent data. You can import documents which refer to other objects and documents that are not included in OpenSHORE – maybe they are imported later or the name is mistyped. For that you easily find these objects and documents (or as we say "resources"), you can display a list of all these "dead ends".

How to find the source of an undefined resource

1. In the menu „Administration“ click on „Zeige undefinierte Ressourcen“.
In the document area, all undefined objects and documents are listed.
2. In the document area, click on an object or a document.
In the info area, you will see all relationships that are linking to this resource.
Note: If there are no incoming and outgoing relationships, talk to your administrator if the document can be deleted.
3. In the info area, click on a relationship.
OpenSHORE loads the document in the document area and marks the area where the relationship is defined.

Now you know, where the undefined resource was defined and you can decide what to do to get a consistent system.

6.6 Display data base summary

For every instance on an OpenSHORE server you can display a summary, how many items you have stored in the system. In the menu „**Administration**“, click on the entry „**Zeige Mengengerüst**“. You will get an appropriate list in the document area.

6.7 Error messages in the Browser

Note: Real messages can differ from these messages, depending on your browser or used servers in your network.

6.7.1 SHORE - undefiniertes Objekt

Applied action

You have clicked a link (also called "Hotspot").

Message

Das Objekt mit dem Namen <object name> vom Typ '<object type>' ist in keinem Dokument definiert.

Es existiert also nur als Quelle oder Ziel mindestens einer Beziehung. Die Beziehungen können hier abgefragt werden.

possible reason	How to resolve...
Document does not exist in OpenSHORE	Import the document, where the object is defined.
Document was renamed	1. change the link in the source document or 2. import the newest version of the source document, where the link is correctly included.

6.7.2 SHORE - Fehler (Dateizugriff)

Applied action

You have clicked a link (also called "Hotspot").

Message

Eine Datei für den Pfad <here is a file path and a file name> wurde nicht gefunden!

possible reason	How to resolve...
File not found	Inside of the document there is a link that directly points to a file that is not existant. Talk to your OpenSHORE administrator if this file should be manually copied to the server or if the link is wrong.

6.7.3 The requested URL could not be retrieved

Applied action

You have entered the URL of your OpenSHORE instance in your browser or you clicked on a link.

Message 1

While trying to retrieve the URL: <here is an URL>

The following error was encountered:

```
Connection Failed
```

The system returned:

```
(146) Connection refused
```

The remote host or network may be down. Please try the request again.

possible reason	How to resolve...
Entered wrong port (or forgot to enter a port number)	Be sure that you enter the correct port number in th URL.
OpenSHORE server is not available	The server which hosts OpenSHORE is not running or OpenSHORE was not started. Talk to your Administrator.

Message 2

While trying to retrieve the URL: <here is an URL>

The following error was encountered:

Unable to determine IP address from host name for <here is the hostname>

The dnsserver returned:

DNS Domain '<here is the hostname>' is invalid: Host not found (authoritative).

This means that:

The cache was not able to resolve the hostname presented in the URL.

Check if the address is correct.

possible reason	How to resolve...
Entered wrong servername	Be sure that you enter the URL exactly as your admin gave it to you. If the error remains, talk to your administrator.

Appendix

A Literature

1. K.F. Sagonas, T. Swift, D. S. Warren, J. Freire, P.Rao: The XSB Programmer's Manual Version 1.8.
2. Michael A. Covington (1994): Natural Language Processing for Prolog Programmers. Englewood Cliffs, New Jersey: Prentice Hall
3. Yoav Shoham (1994): Artificial Intelligence Techniques in Prolog. San Francisco: CAL
4. Leon Sterling und E. Shapiro (1986): The Art of Prolog. Advanced Programming Techniques. Cambridge, Mass.; London, England: The MIT Press
5. Ramin Yasdi (1995): Logik und Programmieren in Logik. München

B Links

Introduction to logical programming languages (and prolog) (in german).	http://ivs.cs.uni-magdeburg.de/~dumke/PSK/LP1.html
Prolog FAQ	http://www.cs.uu.nl/wais/html/na-dir/prolog/faq.html
An introduction to regular expressions.	http://etext.lib.virginia.edu/helpsheets/regex.html
A further introduction to regular expressions with theory behind (in german).	http://www.lrz-muenchen.de/services/schulung/unterlagen/regul/

C Syntax of regular expressions

Regular expressions	Matching text
.	No or any character
	alternative
[...]	One of the listed characters
[^...]	None of the listed characters
(...)	Grouping operator
*	No or more occurrences of the preceding character
+	One or more occurrences of the preceding character
?	No or one occurrence of the preceding character

W
wildcards

14

X
XML

2p.