FirstSpirit™
Your Content Integration Platform

FirstSpirit™ Search
FirstSpirit Version 4.x

Version	**1.3**
Status	**RELEASED**
Date	**2011-06-24**
Department	FS-Core
Author/ Authors	R. Voß, S. Rusch
Copyright	2011 e-Spirit AG
File name	SEAR40EN_FirstSpirit_Modules_Search

e-Spirit AG

Barcelonaweg 14
44269 Dortmund | Germany

T +49 231 . 286 61-30
F +49 231 . 286 61-59

info@e-Spirit.com
www.e-Spirit.com

e-Spirit^AG

Contents

1 Introduction

The FirstSpirit™ Search documentation describes the licence-dependent FirstSpirit™ module for linking different search engines.

Different possible uses are conceivable:

- FirstSpirit™ Search links an external search engine or another data source (e.g. database) to the web applications managed by FirstSpirit™ (Preview, Generation).
- FirstSpirit™ Search provides an internal search engine.
- FirstSpirit™ Search can extend existing search engines to include additional functions. Examples of this are multilingualism, freely definable categories and dynamic filters.
- FirstSpirit™ Search in conjunction with FirstSpirit™ Personalisation can filter search results for certain users and groups. Only the search results for which the respective user (or group) has appropriate rights or permissions are output.

The contents of a website can grow relatively fast. It is necessary to use search engines so that the user can quickly obtain the required information.

One disadvantage of many available search engines is their poor extension capability. For example, indexing unknown document formats, unrestricted ability to define attributes (so-called meta information) or the linking of an external data source is frequently required. Many search engines can only fulfil these requirements incompletely or inadequately.

With the FirstSpirit™ Search module it is possible to optionally link internal or external search engines to FirstSpirit™. The FirstSpirit™ Search module can also be used to extend existing search engines to include additional functions.

2 Building block principle

The FirstSpirit Search module is subdivided into building blocks. These building blocks can be combined with each other depending on the required use.

FirstSpirit Search basically has three blocks:

- Engine (see Chapter 2.1, page 5)
- Server (see Chapter 2.2, page 8)
- Service (see Chapter 2.3, page 8)

The individual blocks and the corresponding terms are listed and described in the following sub-chapters.

2.1 Engine

FirstSpirit Search enables an internal or external search engine to be linked. A so-called search engine must be used to make a search functions available to a web application. The term (search) engine describes a facility which provides the standard functions for browsing through documents or data records. In addition, a search engine can be used to obtain information about the data source behind the results.

2.1.1 Implementation / adapter

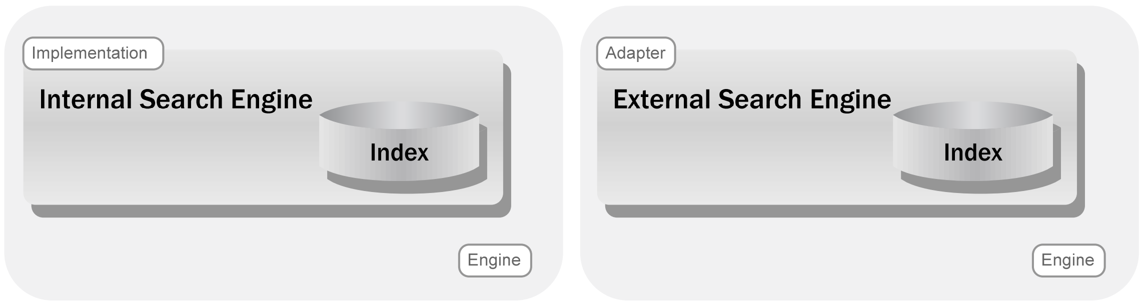


Figure 2-1: Engine implementation and adapter

Linking an internal search engine is called Implementation. With the implementation of an interface (here: the search engine), functions are independently made available without being dependent on other components.

At present, implementation of the Spider Engine (see Chapter 4.5.4.2 page 37) is available.

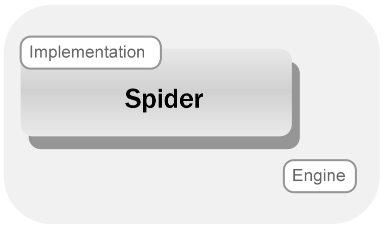


Figure 2-2: Engine implementations

In contrast to this the linking of an external search engine is called Adapter. An adapter enables incompatible interfaces to be accessed.

At present the engine adapters available are Lucene (see Chapter 4.5.4.3 page 46), and JDBC (see Chapter 4.5.4.4 page 48).

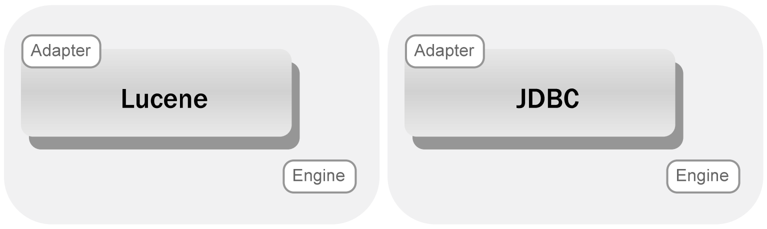


Figure 2-3: Engine adapters

2.1.2 Proxy

A further function fulfilled by the Engine block is the bundling (grouping) of search engines. In addition, search engines can be extended to add missing functions, e.g. categorisation and language assignment.

A Proxy delegates calls (invocations) to individual engines. In addition, it can affect the call (invocation) parameter and returned values. FirstSpirit Search uses a Proxy to bundle engines and to extend them to include other functions. One example for the extension of search engine functions is the definition of a filter to limit the quantity of results.

When combining engines to form a group it is unimportant whether the individual bundled engines are an implementation (see Chapter 2.1.1 page 5), an adapter (see Chapter 2.1.1 page 5) or another proxy.

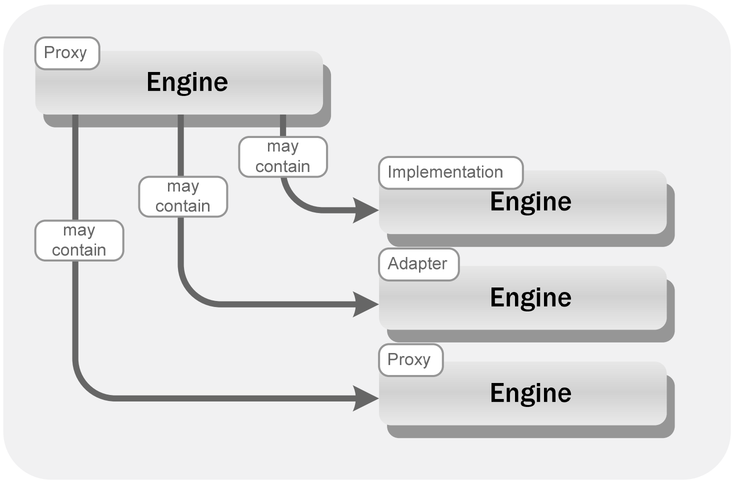


Figure 2-4: Engine proxy

At present the following engine proxies are available (see Figure 2-5):

▪ Simple filter	(Chapter 4.5.4.5, page 52)
▪ Multilingualism (filter)	(Chapter 4.5.4.6, page 54)
▪ Categorisation (filter)	(Chapter 4.5.4.7, page 54)
▪ Simple engine group	(Chapter 4.5.4.8, page 58)
▪ Multilingualism (engine group)	(Chapter 4.5.4.9, page 60)
▪ Categorisation (engine group)	(Chapter 4.5.4.10, page 62)
▪ Personalisation	(Chapter 4.5.4.11, page 64)
▪ Session management	(Chapter 4.5.4.12, page 67)

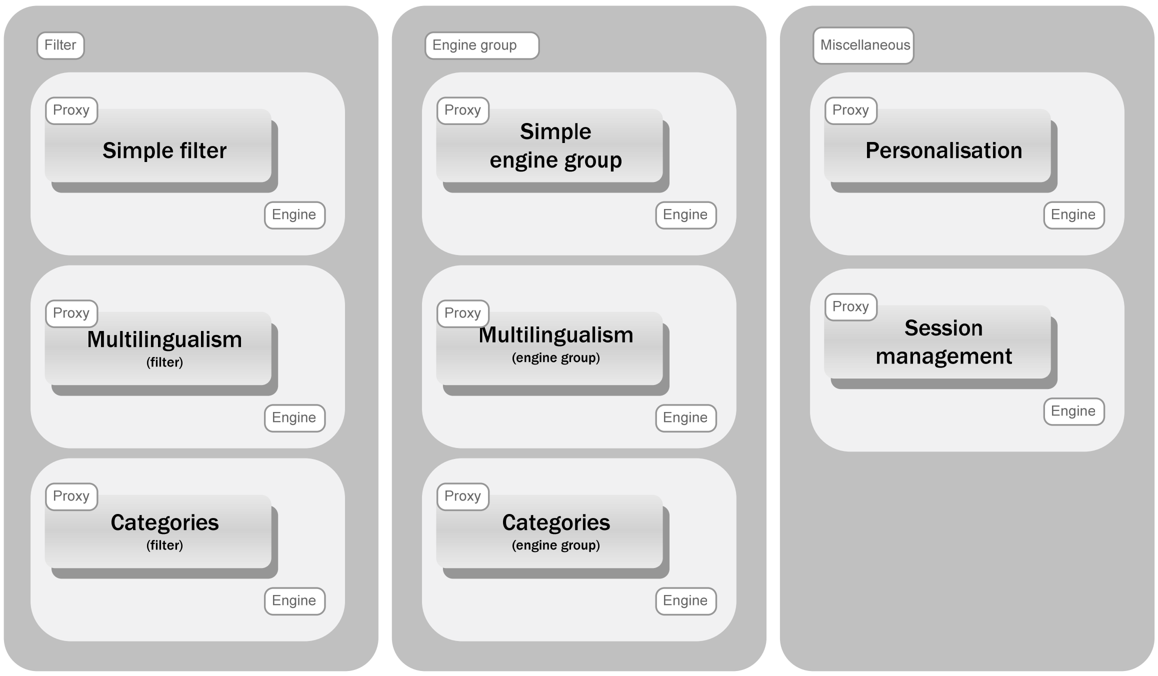


Figure 2-5: Engine proxies

2.2 Server

A further element of the FirstSpirit Search block is the (search) server. A (search) server is used to manage all defined engines (see Chapter 2.1, page 5). The server acts as the entry point to the search system and is given on configuring the "FS SEARCH" web component (see Chapter 3.3 page 12).

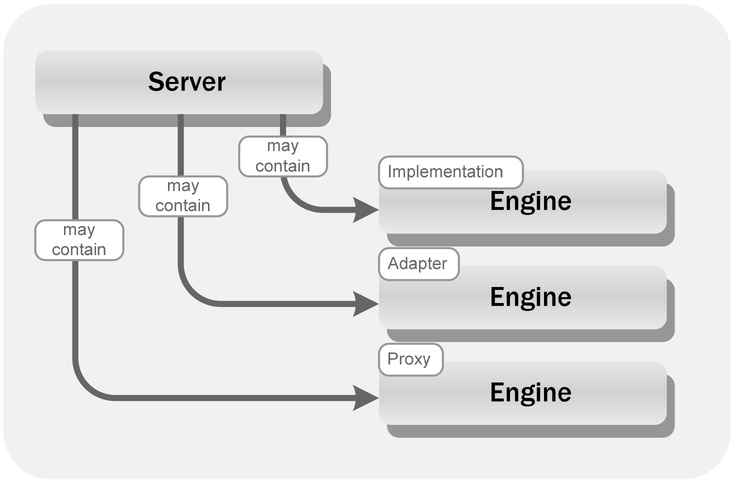


Figure 2-6: Server

2.3 Service

A service performs administrative functions. A service can be used to provide services which can be stopped or started as needed.

Possible services are:

- Control of a search server
  (see Chapter 2.2, page 8 and Chapter 4.5.1.1, page 20)
- Activating logging
  (see Chapter 4.5.1.2, page 22)
- Time-controlled rebuilding of a search index (re-indexing)
  (see Chapter 4.5.1.3, page 21)

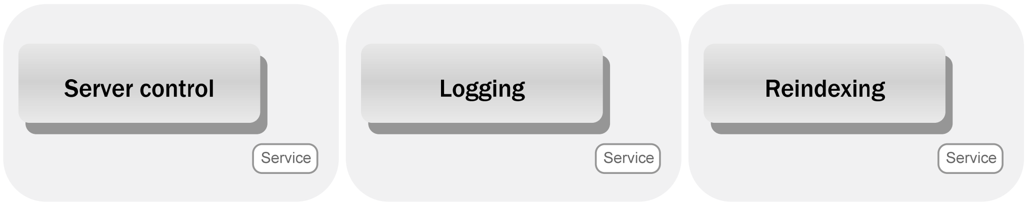


Figure 2-7: Services

Several services can be bundled via a proxy:

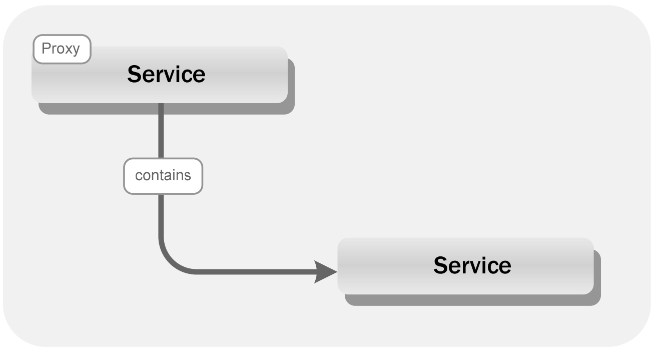


Figure 2-8: Service proxy

There are no other blocks above a service block. Therefore, only one service proxy or one service (several services must be bundled via a proxy) can be the highest element of the consideration.

3 Configuration

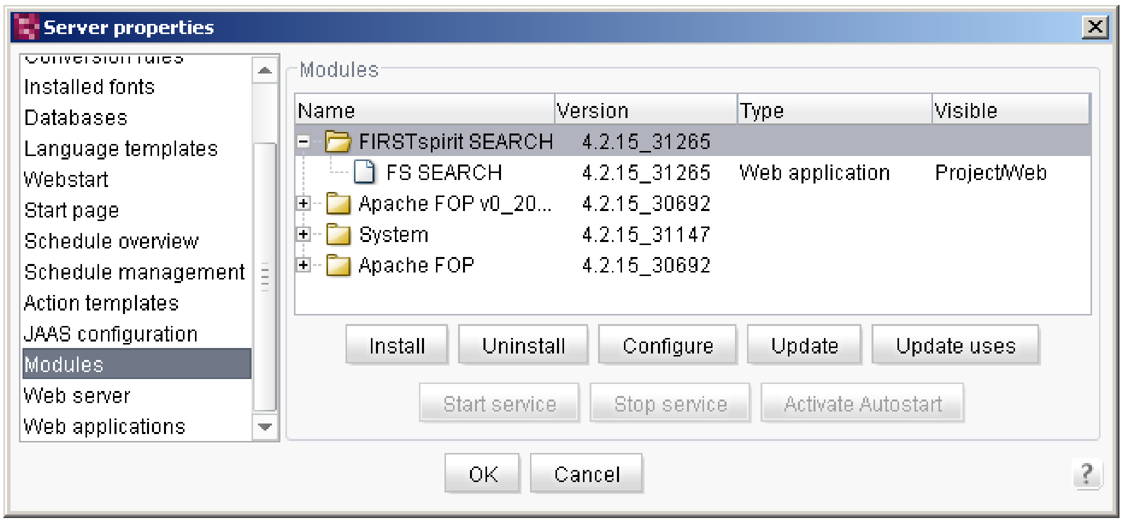
FirstSpirit Search is configured in three steps:

- Installation of the module (see Chapter 3.1, page 10)
- Installation of the web component (see Chapter 3.2, page 11)
- Configuration of the web component (see Chapter 3.3, page 12)

The three steps required are described in the following chapters.

3.1 Installing the module on the server

The FirstSpirit Search module must first be installed within the server and project configuration application. To do this, the "Modules" menu entry is selected in the Server Properties area. Click the "Install" button to open a file selection dialog box. The fsm file to be installed can be selected here. The successfully installed file is then displayed in the "Server Properties" dialog:



Name	Version	Type	Visible
FIRSTspirit SEARCH	4.2.15_31265		
FS SEARCH	4.2.15_31265	Web application	Project/Web
Apache FOP v0_20...	4.2.15_30692		
System	4.2.15_31147		
Apache FOP	4.2.15_30692		

Figure 3-1: Installing the module on the FirstSpirit server

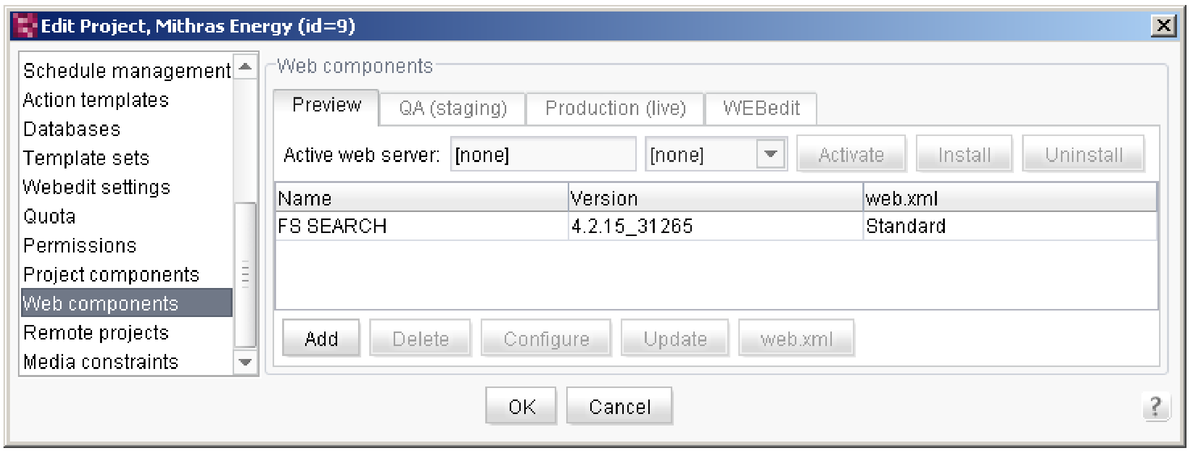
The "FS SEARCH" web application is part of the FirstSpirit Search module. The web application makes JSP tags and servlets available, which can be used and opened within the project.

The component is "visible" for the "Project/Web" areas. It is therefore a "web-local" component. This can be added to the different web areas (previews, staging, live, webedit) within the required projects following installation (see Chapter 3.2, page 11).

For further information on this dialog see "FirstSpirit Manual for Administrators".

3.2 Installing the web application in the project

The web application must now be installed in the required project. To this end, the "Web Components" menu entry is opened within the project properties. The web components for a project can be activated in this area.



Active web server: [none] [none]

Name	Version	web.xml
FS SEARCH	4.2.15_31265	Standard

Figure 3-2: Installing the web application within the web areas

Three different web areas exist for each project. The web components for each area can be individually activated and configured via the respective tab:

Figure 3-3: Web areas within a project

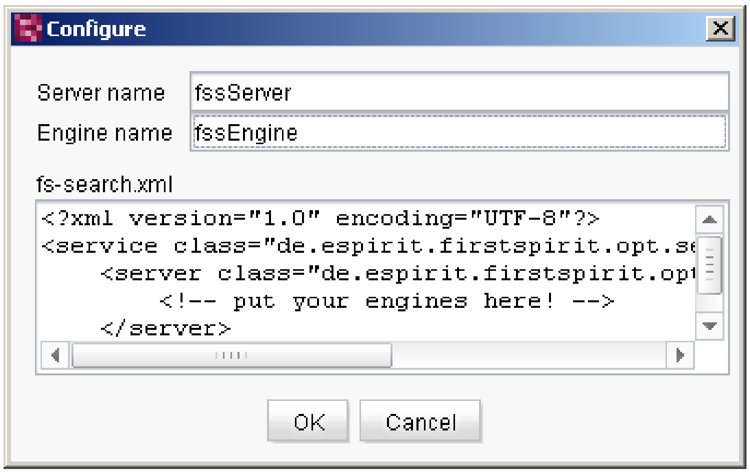
- Preview: Location for the project contents for which a preview has been requested.
- QA (staging): Location for the generated project content
- Production (live): Location for the deployed project content
- WEBedit: Location for the project content of WebEdit

Add: Click the button to open the "Add" dialog. The list shows all web components installed on the server (see Chapter 3.1, page 10).

After adding them to a web area it is possible to configure the components, either with a GUI generated by the component or with a generic GUI (see 3.3 page 12). After configuring the components then have to be activated. A component within a project can be activated or deactivated for specific areas only.

For further information on this dialog see "FirstSpirit Manual for Administrators".

3.3 Configuring the web application



Server name: fssServer
Engine name: fssEngine

fs-search.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<service class="de.espirit.firstspirit.opt.se
    <server class="de.espirit.firstspirit.opt
        <!-- put your engines here! -->
    </server>
```

Figure 3-4: Configuring the Integration web application

Different configuration parameters are available for configuring FirstSpirit Search.

Server name: The name of the search server to be used by the module as the entry point for the search is entered in this field. The name entered here must be the same as the name of the server within the configuration file "fs-search.xml" (see below).

Engine name: The name of the engine to be used by the module is entered here. The name entered here must be the same as the name of an engine within the configuration file "fs-search.xml" (see below).

fs-search.xml: The XML-based configuration of the module is carried out in this field. The configuration syntax follows the building block principle (see Chapter 2, page 5) and is explained in the following chapters (from Chapter 4.1 ff.).

4 Syntax

4.1 General information

FirstSpirit Search is configured with XML (see Chapter 3.3, page 12). Each block (see Chapter 2, page 5) is described with its own XML tag. The syntax also has general attributes and general XML tags. Both the general attributes and the general XML tags and the XML tags of the individual blocks are explained.

In addition, FirstSpirit Search has a tag library for controlling the web application. Both general, formal definitions are described (e.g. specification of a prefix for the JSP tag) and the use of available JSP tags is explained.

4.2 XML declaration

XML documents must be well-formed and valid so that they can be read out and interpreted by a parser. One requirement for this is the specification of an XML declaration.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
```

4.3 General XML attributes

General XML attributes can be used in all tags. General XML attributes can be divided into three categories depending on their intended use:

- Reuse:
  "id", "idref" and "fileref" attributes (see Chapter 4.3.1 page 14).
- Realisation:
  "type" and "class" attributes (see Chapter 4.3.2, page 15)
- Identifier:
  "name" attribute (see Chapter 4.3.3, page 18)

4.3.1 Reuse (id, idref and fileref)

These attributes can be used to fulfil two tasks:

- Reuse of an object (see Chapter 4.3.1.1, page 14)
- Reuse of XML files (see Chapter 4.3.1.2, page 15)

4.3.1.1 Reuse of an object (id and idref)

The "id" attribute can be used to define a unique identifier for an element. This unique identifier enables an element to be used again within the whole configuration.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<service ...>
  <engine id="ID" />
</service>
```

In the example the unique identifier "ID" is assigned to the "engine" tag.

The "idref" attribute is also necessary to re-use the unique identifier in the configuration. The expected value is the unique identifier with the assigned attribute "id". Use of the "idref" attribute results in reuse of the element of the referenced tag. Therefore, exactly the same element or object is used by both tags.

When assigning a unique identifier it is necessary to ensure that the first character is a letter. The following characters only should be used: A-Z, a-z, 0-9 and _. A valid identifier could therefore be called "server01". The identifier must be uniquely assigned within a configuration.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<service ...>
  <engine id="ID">
    ...
  </engine>
  ...
  <engine idref="ID" />
</root>
```

The "id" attribute is used to assign the unique identifier "ID" to the first tag "engine". In the second "engine" tag the element of the first "engine" tag is reused (referenced) with the "idref" attribute.

4.3.1.2 Reuse of XML files (fileref)

The "fileref" attribute can be given in an XML tag to insert the content of another XML file. In addition, either existing configurations can be reused or several configurations can be grouped together to form a configuration.

The value expected by the "fileref" attribute is a URL. This can be a path in the web application (e.g. "/WEB-INF/file.xml") or in the file system (e.g. "/home/user/file.xml").

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<service ...>
  <service fileref="URL" />
</service>
```

4.3.2 Realisation (type and class)

The individual blocks of FirstSpirit Search require both simple and complex objects for the processing.

An example of a simple object is the specification of a character string as a parameter for an engine (string). In contrast to this the engine itself is a complex object.

Due to the modular structure of FirstSpirit Search, the module for the individual tags can possibly not decide which implementation, which adapter or which proxy is to be used. Further, when a parameter is specified the interpretation may not be clearly defined. For example, the module cannot recognise whether the specification should be treated as a character string (string) or as a number (integer). For this reason, when using tags and attributes it is also necessary to define how these are to be converted or interpreted.

To this end, the attributes "type" and "class" can be given for each tag. These two attributes can be used to give the (abstract) type or specific realisation ("class", class name) of a type.

If the "type" attribute is not given the tag name is interpreted as a type. Therefore, specifying `<map />` or `<tag type="map" />` both result in "map" being used as the type. The default type for the "attribute" tag name is "string".

Examples:

```
<engine ... />
```

Here "engine" is used as the type.

```
<attribute type="list" ... />
```

"list" is used as the type in this specification.

As a type can be realised in different ways for many types it is necessary to specify which realisation is to be used. For example, in Java a map can be realised by "java.util.HashMap" nd "java.util.TreeMap".

The realisation is specified with the "class" attribute. The value expected by the attribute is the class name including the Java package (e.g. "java.util.HashMap").

If the "class" attribute is given, FirstSpirit Search checks whether the given realisation matches the type.

Default realisation *is stored for many types. For these types it is* ***not mandatory*** *to specify the "class" attribute. If* ***no default realisation*** *is stored for a type it is* ***mandatory*** *to specify the "class" attribute. It is advisable to use the "type" and "class" attributes only if it is absolutely necessary to specify the type and realisation.*

The useable types are given in the following tables. The tables show, for each type, whether default realisation is stored and whether an alternative realisation can be given.

The types are divided into:

- general types (see Chapter 4.3.2.1, page 17)
- special "First Spirit Search" types (see Chapter 4.3.2.2, page 17)

Examples of use of the two types are given in Chapter 4.3.2.3 (from page 18).

4.3.2.1 General types

The general types are used exclusively as a value for the "type" attribute. For this reason, general and special types (see Chapter 4.3.2.2, page 17) are combined with each other. All general types have in common that it is not mandatory to specify the "class" attribute.

Type	Description	"class" attribute	Default realisation	Alternative realisation
string	Character string	Optional	java.lang.String	None
int	Integer	Optional	java.lang.Integer	None
bool	Boolean value	Optional	java.lang.Boolean	None
map	Quantity of key value pairs	Optional	java.util.HashMap	java.util.TreeMap
list	List of objects	Optional	java.util.ArrayList	java.util.LinkedList
set	Quantity of objects	Optional	java.util.HashSet	java.util.TreeSet

4.3.2.2 Special types

Special types can only be used as tags and not as the value for the "type" attribute. For this reason, general (see Chapter 4.3.2.1, page 17) and special types are combined with each other.

It is mandatory to specify the "class" attribute for all special types with the exception of "attribute".

Type	Description	"class" attribute	Default realisation	Alternative realisation
engine	Search engine ("Engine" block)	Mandatory information	None	None
server	Search server ("Engine" block)	Mandatory information	None	None
filter	Search filter	Mandatory information	None	None

Type	Description	"class" attribute	Default realisation	Alternative realisation
attribute	General attributes (parameter)	Optional	java.lang.String	None

4.3.2.3 Examples

The following example generates an otherwise identical map in three different ways:

```
<attribute class="java.util.HashMap" />
<attribute type="map" />
<map />
```

The following entries are invalid because either the type is unknown or the given class is incompatible.

```
<unknowntag />
<attribute type="unknowntype" />
<map class="java.lang.String" />
<attribute type="map" class="java.util.ArrayList" />
```

4.3.3 Identifier (name)

The "name" attribute can be used to define an identifier for a tag. For example, an identifier for the engine can be given in the configuration of an engine with the attribute "name". In the special type "attribute" (see Chapter 4.3.2.2, page 17) the value of the attribute "name" corresponds to the parameter name for the surrounding special type.

In the first example the identifier "NAME" is given for the "engine" tag:

```
<?xml version="1.0" encoding="UTF-8"?>
<service>
  <engine name="NAME">
    ...
  </engine>
</service>
```

Unlike the first example, in the second example the parameter "PARAMETERNAME" is defined for the special type "engine":

```
...
<engine ... >
  <attribute name="PARAMETERNAME">...</attribute>
</engine>
...
```

If the "name" attribute is not given the tag name is interpreted as an identifier.

Therefore, the following two specifications are identical:

```
<engine name="engine" ... >
```

and

```
<engine ... >
```

4.4 General XML tags

General XML tags can be used in all tags. At present FirstSpirit Search has the general XML tag "attribute".

4.4.1 Parameter information (attribute)

The "attribute" tag can be used to give parameters for an element. The given parameters apply to the respective higher level element.

An identifier (the attribute "name") must be given for the tag (see Chapter 4.3.3, page 18). In this case the value expected by the attribute "name" is the parameter name:

```
<attribute name="PARAMETERNAME">
```

The parameter value must be given between the opening and closing tag:

```
<attribute name="PARAMETERNAME">PARAMETERVALUE</attribute>
```

4.5 Special XML tags

4.5.1 Services

A service provides a service. A service is used to realise administrative tasks.

At present the following services are available for FirstSpirit Search:

- Search server:
  Control and provision of a search server (see Chapter 4.5.1.1, page 20)
- Logging:
  Configuration of the log outputs (see Chapter 4.5.1.2, page 22)
- Re-indexing:
  Time-depending re-indexing of a search engine
  (see Chapter 4.5.1.3 page 21).

4.5.1.1 Search server

This service can be used to start all search engines contained (within the <service> tag) and the search server. Further, the service provides clients (e.g. a JSP page) with an interface with the search server via which, for example, queries can be made. The search server can be made available in different ways:

- in the current Java Virtual Machine (JVM)
- through Remote Method Invocation (RMI)
- through the Java Naming and Directory Interface (JNDI)

A <service> tag with the corresponding class name ("class" attribute) must be given to configure the search server. The service has several configuration parameters which can be given with an <attribute> tag (within the <service> tag).

The corresponding attributes and parameters are given in the following tables. The values are character strings. For this reason it is not necessary to give the "type" attribute (see Chapter 4.3.2.2, page 17).

<service> tag	
Attribute	**Value**
class[1]	de.espirit.firstspirit.opt.search.service.adapter.ServerService

[1] Mandatory parameter

<attribute> tag		
Parameter[1]	**Value**	**Default value**
bindTo	"bindTo" is used to define the location in which the search server is made available. Possible values are: ▪ `local`: local JVM ▪ `rmi`: RMI registration ▪ `jndi`: JNDI context Several comma-separated values can be given. If an incorrect value is given for "bindTo", "local" is used.	`local,rmi`
createRMI[2]	"createRMI" can be used to start an RMI server. The parameter expects a Boolean value. `true` is used to start an RMI server, `false` prevents an RMI server from starting. Instead the system tries to determine an existing RMI registration.	`true`
localName	Local identifier for the JVM.	`FIRSTseek.SearchServer`
rmiHost[2]	Computer name of the RMI server.	`localhost`
rmiName[2]	Name of the server within the RMI registration.	`FIRSTseek.SearchServer`
rmiPort[2]	RMI port	`1099`

[1] "name" attribute

[2] This Parameter is only evaluated if the value "rmi" was given for the "bindTo" parameter

Within the <service> tag the <server> tag must be given. The required engines (see Chapter 4.5.4, page 35 ff.) can be defined within the <server> tag. The mandatory "class" attribute must be given for the <server> tag.

The class name is:
`de.espirit.firstspirit.opt.search.server.SimpleServer`.

Alternatively the server can also be specified with an <attribute> tag (see Chapter 4.4.1, page 19) in conjunction with details of the special type "server" (see Chapter 4.3.2.2, page 17) (<attribute type="server"...>). However, it is advisable to specify the <server> tag to make the configuration as clear as possible.

<server> tag	
Attribute	**Value**
class[1]	de.espirit.firstspirit.opt.search.server.SimpleServer

[1] Mandatory parameter

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<service
class="de.espirit.firstspirit.opt.search.service.adapter.ServerSer
vice">
  <server
class="de.espirit.firstspirit.opt.search.server.SimpleServer">
    ...
  </server>
  <attribute name="createRMI">false</attribute>
  <attribute name="bindTo">local</attribute>
  <attribute name="localName">fssServer</attribute>
</service>
```

In the example the generation of an RMI server is prevented ("createRMI"). The identifier assigned is "fssServer" ("localName"). In addition, it is specified that the search server is to be made available in the local JVM ("bindTo").

4.5.1.2 Logging

This service configures the logging system used in FirstSpirit Search, Log4j[1]. It only makes sense to use the service if Log4j has not already been configured in the deployed environment.

[1] See http://logging.apache.org/log4j/

! *The logging service should be started as the first service so that early logging is possible.*

A <service> tag with the corresponding class name ("class" attribute) must be given to configure the logging. The configuration parameters which can be given with an <attribute> tag (within the <service> tag) are forwarded to Log4j unchanged.

The corresponding attributes and parameters are given in the following tables. The values are character strings. For this reason it is not necessary to give the "type" attribute (see Chapter 4.3.2.2, page 17).

<service> tag	
Attribute	**Value**
class[1]	de.espirit.firstspirit.opt.search.service.adapter.Log4jService

[1] Mandatory parameter

<attribute> tag
All parameters ("name" attribute) are forwarded to Log4j unchanged.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<service
class="de.espirit.firstspirit.opt.search.service.adapter.Log4jServ
ice">
  <attribute name="log4j.rootCategory">
    DEBUG, file
  </attribute>
  <attribute name="log4j.appender.file">
    org.apache.log4j.RollingFileAppender
  </attribute>
  <attribute name="log4j.appender.file.File">
    E:/FirstSpirit/log/fs-search.log
  </attribute>
  <attribute name="log4j.appender.file.MaxFileSize">
    5MB
  </attribute>
  <attribute name="log4j.appender.file.MaxBackupIndex">
    5
  </attribute>
  <attribute name="log4j.appender.file.layout">
    org.apache.log4j.PatternLayout
  </attribute>
  <attribute
name="log4j.appender.file.layout.ConversionPattern">%-5p %d (%c)
%m%n</attribute>
</service>
```

4.5.1.3 Re-indexing

The individual search engines have an interface to initiate rebuilding of their index. The re-indexing service can be used to start the rebuilding of the index at a specified time and at fixed time intervals.

A <service> tag with the corresponding class name ("class" attribute) must be given to configure the re-indexing. The service has several configuration parameters which can be given with an <attribute> tag (within the <service> tag).

The corresponding attributes and parameters are given in the following tables. The values are character strings. For this reason it is not necessary to give the "type" attribute (see Chapter 4.3.2.2, page 17).

<service> tag	
Attribute	**Value**
class[1]	de.espirit.firstspirit.opt.search.service.RebuildIndexTimerService

[1] Mandatory parameter

<attribute> tag		
Parameter[1]	**Value**	**Default value**
engineURL[2]	Specification of the complete search engine URL. The URL is made up of several parts: ▪ Server URL ▪ Engine identifier Example: *`SERVER_URL[ENGINE]`* An RMI URL, a JNDI URL or a local identifier can be given as the server URL. The local identifier corresponds to the value of the "localName" parameter of a search server (see Chapter 4.5.1.1, page 20). An RMI and JNDI URL must be given in the format *`//rmiHost:rmiPort/rmiName`*. The parameters "rmiHost", "rmiPort" and "rmiName" are given in Chapter 4.5.1.1, page 20. When the service is started, the system first tries to determine a local server. It then tries to establish a connection with a search server, with RMI first and then with JNDI.	None

<attribute> tag		
Parameter[1]	**Value**	**Default value**
startTime	Start time of the re-indexing. The specification is given in the format *`hh:mm`*. The hour must be given in 24 hour format first (*`00-23`*) and, separated by a colon, the minutes (*`00-59`*). Example: *`21:00`*	Start time of the service + 5 seconds
period[2]	Period between two re-indexings. The value expected is a number followed by a letter. This letter defines whether the specification is given in seconds (*`s`*), minutes (*`m`*), hours (*`h`*), days (*`d`*) or weeks (*`w`*). Example: *`1d`*	None
startNow	This parameter is used to specify whether re-indexing should be performed when the service is started – irrespective of the start time and indexing interval. If the value *`true`* is given, re-indexing is forced, however not if *`false`* is given.	*`false`*

[1] "name" attribute

[2] Mandatory parameter

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<service
class="de.espirit.firstspirit.opt.search.service.RebuildIndexTimer
Service">
  <attribute name="engineURL">fssServer[fssEngine]</attribute>
  <attribute name="startTime">6:00</attribute>
  <attribute name="period">24h</attribute>
  <attribute name="startNow">true</attribute>
</service>
```

Re-indexing is configured in the example. Re-indexing is always to take place when the web application is started ("startNow"). This re-indexing is carried out each day ("period") at 06:00 hrs ("startTime"). The re-indexing also applies to the "fssEngine" search engine of the local search server "fssServer" ("engineURL").

4.5.2 Service proxy

A service proxy can be used to group together several services. This is necessary as only one XML file can be configured when the web application is configured via the server and project configuration (see Chapter 3.3, page 12).

The service proxy is defined within a <service> tag. Within this <service> tags the <attribute> tag is used to give a list of services.

The parameter name for the list of services is: "services". "list" must be given as the type. Therefore, other <service> tags only may be given for the <attribute> tag.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<service class="...">
  <attribute name="services" type="list">
    <service class="...">
      ...
    </service>
    <service class="...">
      ...
    </service>
    ...
  </attribute>
</service>
```

The corresponding attributes and parameters are given in the following tables.

<service> tag	
Attribute	**Value**
class[1]	de.espirit.firstspirit.opt.search.service.proxy.MultiServiceProxy

[1] Mandatory parameter

<attribute> tag		
Parameter[1]	**Value**	**Default value**
services[2][3]	A list of services (<service> tags)	None

[1] "name" attribute

[2] Mandatory parameter

[3] "list" must be given as the "type" attribute

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<service
class="de.espirit.firstspirit.opt.search.service.proxy.MultiServic
eProxy">
  <attribute name="services" type="list">
    <service
class="de.espirit.firstspirit.opt.search.service.adapter.ServerSer
vice">
      <server
class="de.espirit.firstspirit.opt.search.server.SimpleServer">
        ...
      </server>
      <attribute name="createRMI">false</attribute>
      <attribute name="bindTo">local</attribute>
      <attribute name="localName">fssServer</attribute>
    </service>
    <service
class="de.espirit.firstspirit.opt.search.service.RebuildIndexTimer
Service">
      <attribute name="engineURL">fssServer[fssEngine]</attribute>
      <attribute name="startTime">6:00</attribute>
      <attribute name="period">24h</attribute>
      <attribute name="startNow">true</attribute>
    </service>
  </attribute>
</service>
```

In the example, two services are grouped together: a search server and a re-indexing.

4.5.3 Service runner

The so-called runner is the possibility of controlling a search service from various environments.

At present, FirstSpirit Search provides the following runners:

- Standalone: Starts a service in a separate JVM
  (see Chapter 4.5.3.1, page 29)
- Web: Integration of a service in a web application
  (see Chapter 4.5.3.2, page 32)
- JMX: Integrates a service in a Java Management Extension Container (JMX) see Chapter 4.5.3.3, page 34)

4.5.3.1 Standalone

The standalone service runner can be used to start services as an independent program in a separate JVM. The services are started in the given order and are stopped again in the reverse order when the program is ended.

It is useful to use the service runner if a separate RMI server is to be used in its own JVM.

The class name of the standalone service runner is:

de.espirit.firstspirit.opt.search.admin.standalone.ServiceRunner.

The class is located in the runtime jar file "fs-search-rt.jar" which is contained in the module file of FirstSpirit Search. The module file contains many of the required classes (e.g. Lucene-Core, Log4j) in the "lib" directory. Also, several services and engines require other classes which are contained in the "fs-server.jar".

The parameter expected by the service runner is one or several XML configuration files.

Syntax:

```
java –cp CLASSPATH
de.espirit.firstspirit.opt.search.admin.standalone.ServiceRunner
XMLFILE1 XMLFILE2 ...
```

A service proxy can be used to bundle several services. However, it is also possible to divide the individual services between several XML files.

Example of service bundling ("services.xml"):

```
<?xml version="1.0" encoding="UTF-8"?>
<service
class="de.espirit.firstspirit.opt.search.service.proxy.MultiServic
eProxy">
  <attribute name="services" type="list">
    <service
class="de.espirit.firstspirit.opt.search.service.adapter.Log4jServ
ice">
      <attribute name="log4j.rootCategory">
        DEBUG, file
      </attribute>
      <attribute name="log4j.appender.file">
        org.apache.log4j.RollingFileAppender
      </attribute>
      <attribute name="log4j.appender.file.File">
        E:/Standalone/fs-search.log
      </attribute>
      <attribute name="log4j.appender.file.MaxFileSize">
        5MB
      </attribute>
      <attribute name="log4j.appender.file.MaxBackupIndex">
        5
      </attribute>
      <attribute name="log4j.appender.file.layout">
        org.apache.log4j.PatternLayout
      </attribute>
      <attribute
name="log4j.appender.file.layout.ConversionPattern">%-5p %d (%c)
%m%n</attribute>
    </service>
    <service
class="de.espirit.firstspirit.opt.search.service.adapter.ServerSer
vice">
      <server
class="de.espirit.firstspirit.opt.search.server.SimpleServer">
        <engine name="fssEngine"
class="de.espirit.firstspirit.opt.search.engine.proxy.MonitorEngin
eProxy">
          <engine
class="de.espirit.firstspirit.opt.search.engine.spider.SpiderEngin
e">
            <attribute name="urls" type="list">

<attribute>E:/FIRSTsprit4/web/fs4staging_41038/41118/de/index.html
</attribute>
            </attribute>
            <attribute name="index">E:/Standalone/fs-
search.index</attribute>
          </engine>
        </engine>
      </server>
      <attribute name="createRMI">true</attribute>
      <attribute name="bindTo">rmi</attribute>
      <attribute name="rmiHost">localhost</attribute>
      <attribute name="rmiPort">7788</attribute>
      <attribute name="rmiName">fssServer</attribute>
    </service>
  </attribute>
</service>
```

In the example the logging is connected with a separate RMI server (search service). An exemplary call (invocation) in the command line looks like this:

```
java -cp log4j-1.2.14.jar;lucene-core-2.1.0.jar;commons-
httpclient.jar;fs-search-rt.jar
de.espirit.firstspirit.opt.search.admin.standalone.ServiceRunner
services.xml
```

This configuration can be divided into two configurations.

"logging.xml":

```
<?xml version="1.0" encoding="UTF-8"?>
<service
class="de.espirit.firstspirit.opt.search.service.adapter.Log4jServ
ice">
  <attribute name="log4j.rootCategory">
    DEBUG, file
  </attribute>
  <attribute name="log4j.appender.file">
    org.apache.log4j.RollingFileAppender
  </attribute>
  <attribute name="log4j.appender.file.File">
    E:/Standalone/fs-search.log
  </attribute>
  <attribute name="log4j.appender.file.MaxFileSize">
    5MB
  </attribute>
  <attribute name="log4j.appender.file.MaxBackupIndex">
    5
  </attribute>
  <attribute name="log4j.appender.file.layout">
    org.apache.log4j.PatternLayout
  </attribute>
  <attribute
name="log4j.appender.file.layout.ConversionPattern">%-5p %d (%c)
%m%n</attribute>
</service>
```

"rmi_server.xml":

```
<?xml version="1.0" encoding="UTF-8"?>
<service
class="de.espirit.firstspirit.opt.search.service.adapter.ServerSer
vice">
  <server
class="de.espirit.firstspirit.opt.search.server.SimpleServer">
    <engine name="fssEngine"
class="de.espirit.firstspirit.opt.search.engine.proxy.MonitorEngin
eProxy">
      <engine
class="de.espirit.firstspirit.opt.search.engine.spider.SpiderEngin
e">
        <attribute name="urls" type="list">

<attribute>E:/FIRSTsprit4/web/fs4staging_41038/41118/de/index.html
</attribute>
        </attribute>
        <attribute name="index">E:/Standalone/fs-
search.index</attribute>
      </engine>
    </engine>
  </server>
  <attribute name="createRMI">true</attribute>
  <attribute name="bindTo">rmi</attribute>
  <attribute name="rmiHost">localhost</attribute>
  <attribute name="rmiPort">7788</attribute>
  <attribute name="rmiName">fssServer</attribute>
</service>
```

The invocation in the console is then accordingly:

```
java -cp log4j-1.2.14.jar;lucene-core-2.1.0.jar;commons-
httpclient.jar;fs-search-rt.jar
de.espirit.firstspirit.opt.search.admin.standalone.ServiceRunner
logging.xml rmi_server.xml
```

4.5.3.2 Web

The web service runner is a servlet which enables the integration of a service in a new or existing web application. Apart from automatic starting and stopping of the service, the servlet also allows administration via the runtime parameters without having to restart the web application first.

As a default the service runner is configured in the "web.xml" file, which is created during configuration via the server and project configuration (see Chapter 3.2, page 11):

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4">
  ...
  <display-name>41133STAGING</display-name>
  <servlet>
    <servlet-name>fss-Init</servlet-name>
    <servlet-
class>de.espirit.firstspirit.opt.search.admin.web.ServiceServlet</
servlet-class>
    <init-param>
      <param-name>service</param-name>
      <param-value>/WEB-INF/fs-search.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  ...
</web-app>
```

For security reasons, as a default, no servlet mapping is given in the "web.xml" file. Such mapping must therefore be added manually.

After entering a servlet mapping the service can be started and stopped within the web application, if no other safeguards are taken.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4">
  ...
  <display-name>41133STAGING</display-name>
  <servlet>
    <servlet-name>fss-Init</servlet-name>
    <servlet-
class>de.espirit.firstspirit.opt.search.admin.web.ServiceServlet</
servlet-class>
    <init-param>
      <param-name>service</param-name>
      <param-value>/WEB-INF/fs-search.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  ...
  <servlet-mapping>
    <servlet-name>fss-Init</servlet-name>
    <url-pattern>*.service</url-pattern>
  </servlet-mapping>
  ...
</web-app>
```

After adding the servlet mapping the service can now be started and stopped.

Status information can be displayed with the following invocation:

```
PROTOCOL://HOST:PORT/APPLICATION/do.service
```

e.g. http://localhost/search/do.service

The invocation to start the service is:

```
PROTOCOL://HOST:PORT/APPLICATION/do.service?action=start
```

e.g. http://localhost/search/do.service?action=start

The invocation to stop the service is:

```
PROTOCOL://HOST:PORT/APPLICATION/do.service?action=stop
```

e.g. http://localhost/search/do.service?action=stop

The invocation to reload the service is:

```
PROTOCOL://HOST:PORT/APPLICATION/do.service?action=reload
```

e.g. http://localhost/search/do.service?action=reload

4.5.3.3 JMX

The service runner for JMX is a Managed Bean (MBean)[2] and can be integrated in any JMX agents (or JBoss).

The class name for the service runner is:
`de.espirit.firstspirit.opt.search.admin.jmx.ServiceMX`

<attribute> tag		
Parameter[1]	**Value**	**Default value**
Service[2][3]	Path to the configuration file	None

[1] "name" attribute
[2] Mandatory parameter
[3] The parameter must be written in upper case letters

[2] See http://java.sun.com/products/JavaManagement/index.html.

Example:

```
<mbean
    code="de.espirit.firstspirit.opt.search.admin.jmx.ServiceMX"
    name="fs:service=SearchMX">
  <attribute name="Service">META-INF/fs-search.xml</attribute>
</mbean>
```

4.5.4 Engines

▪ Overview	(see Chapter 4.5.4.1 page 35).
▪ Engine implementation "Spider Engine"	(see Chapter 4.5.4.2 page 37).
▪ Engine adapter "Lucene"	(see Chapter 4.5.4.3 page 46).
▪ Engine adapter "IDBC"	(see Chapter 4.5.4.4 page 48).
▪ Engine proxy "Simple filter"	(see Chapter 4.5.4.5 page 52).
▪ Engine proxy "Multilingualism (filter)"	(see Chapter 4.5.4.6 page 54).
▪ Engine proxy "Categorisation (filter)"	(see Chapter 4.5.4.7 page 56).
▪ Engine proxy "Simple engine group"	(see Chapter 4.5.4.8 page 58).
▪ Engine proxy "Multilingualism (group)"	(see Chapter 4.5.4.9 page 60).
▪ Engine proxy "Categorisation (group)"	(see Chapter 4.5.4.10 page 62).
▪ Engine proxy "Personalisation"	(see Chapter 4.5.4.11 page 64).
▪ Engine proxy "Session management"	(see Chapter 4.5.4.12 page 67).

4.5.4.1 Overview

As described in Chapter 2.1 (page 5 ff.), a search engine is a building block of FirstSpirit Search. The module supports a large number of engines (see Figure 2-2 on page 6, Figure 2-3 on page 6 and Figure 2-5 on page 7).

Refer to the following table for the individual search engines – sorted by type (implementation / adapter / proxy).

Name	Type	Description	Chapter
Spider engine	Engine implementation	Own search engine with integrated spider and indexer for different document formats.	Chapter 4.5.4.2, page 37
Lucene	Engine adapter	Integration of an existing Lucene index.	Chapter 4.5.4.2, page 37

Name	Type	Description	Chapter
JDBC	Engine adapter	Linking of a database.	Chapter 4.5.4.4, page 48
Simple filter	Engine proxy	Static filtering of certain results.	Chapter 4.5.4.5, page 52
Multilingualism (filter)	Engine proxy	Support for multilingualism by filtering certain results.	Chapter 4.5.4.6, page 54
Categorisation (filter)	Engine proxy	Support for categories by filtering certain results.	Chapter 4.5.4.7, page 56
Simple engine group	Engine proxy	Grouping together of several engines.	Chapter 4.5.4.8, page 58
Multilingualism (engine group)	Engine proxy	Simulation of multilingualism by several engines.	Chapter 4.5.4.9, page 60
Categorisation (engine group)	Engine proxy	Simulation of categories by several engines.	Chapter 4.5.4.10, page 62
Personalisation	Engine proxy	Personalised search results, depending on the group or user rights.	Chapter 4.5.4.11, page 64
Session management	Engine proxy	Automatic session timeouts	Chapter 4.5.4.12, page 67

The <engine> tag is used in the definitions and in the engine examples for improved clarity. Alternatively an engine can also be specified with an <attribute> tag (see Chapter 4.4.1, page 19) in conjunction with details of the special type "engine" (see Chapter 4.3.2.2, page 17) (<attribute type="engine"...>).

4.5.4.2 Engine implementation "Spider Engine"

The Spider engine is an independent implementation of the search engine function. It is based on a range of open libraries (Lucene[3], Apache POI[4], HTML Parser[5] and Jakarta Commons HttpClient[6]), which have been combined by their own spider for browsing websites.

The engine supports multilingualism and categories, if this information is available as meta tags in the HTML (see "categoryField", "categories", "localeField" and "locales" parameters). If the search query contains a restriction with respect to language or category the query transferred to Lucene is extended by the corresponding field search.

A document is indexed, if its URL allows and it is not forbidden (see "allowed" and "forbidden" parameters). The indexing process is finished when one of the following three conditions occurs:

- All sites/pages have been browsed through.
- The time limit has been exceeded (see "maxTime" parameter).
- The maximum number of documents has been exceeded (see "maxDocuments" parameter).

The generated Lucene index is created in its own directory underneath the base directory (see "index" parameter). The name of this directory represents the starting time of the indexing in coded form. If indexing fails, for troubleshooting reasons the directory created for it is not deleted. However, if it is successful, old directories are removed providing they are not still being used by running search sessions. The directory name of the currently valid index is located in the "index.cfg" file.

The indexed fields and documents in the Lucene index can be viewed using the Luke tool - Lucene Index toolbox[7] (see Chapter 8.1, page 96).

The indexing of parts of a document can be suppressed with the comment

```
<!-- noindex -->
```

[3] http://lucene.apache.org/

[4] http://poi.apache.org/

[5] http://htmlparser.sourceforge.net/

[6] http://hc.apache.org/httpclient-3.x/index.html

[7] http://www.getopt.org/luke/

From this comment the contents of the document are not indexed. Suppression of the indexing is exited with the comment

```
<!-- index -->
```

With a nesting of <!-- noindex --> comments the number of comments is determined. To end suppression of the indexing, in a nesting the same number of <!-- index --> comments must be given.

Example:

```
YES, this passage will be indexed
<!-- noindex -->
NO, this passage won't be indexed
<!-- noindex -->
NO, this passage won't be indexed
<!-- index -->
NO, this passage won't be indexed
<!-- index -->
YES, indexing will start from here again
```

<engine> tag	
Attribute	**Value**
class[1]	de.espirit.firstspirit.opt.search.engine.spider.SpiderEngine

[1] Mandatory parameter

<attribute> tag		
Parameter[1]	**Value**	**Default value**

<attribute> tag		
Parameter[1]	**Value**	**Default value**
analyzer	Specification of the Lucene analyzer to be used. A list of the possible analyzers is given in the API documentation provided by Lucene: http://lucene.apache.org/java/2_1_0/api/org/apache/lucene/analysis/Analyzer.html Attention: The analyzer is not configured by using the attribute-element with the class name as usual, but via the following expression: `<analyzer class="org.apache.lucene. analysis.SimpleAnalyzer"/>` (NOT: `<attribute name="analyzer" class="org.apache.lucene.analysis.SimpleAnalyzer"/>`)	StandardAnalyzer
index[2]	Specification of the base directory for the Lucene index	None
defaultField	This parameter can be used to define the default field name for the search. As a default the contents are stored in the "content" field by the indexing through Lucene.	content

<attribute> tag		
Parameter[1]	**Value**	**Default value**
categoryField[4]	The pages in a website can be equipped with meta information (<meta> tags). This information is stored in the index during indexing. The "categoryField" parameter is used to define which <meta> tag information is to be used for the categories. The value of "categoryField" therefore corresponds to the value of the "name" attribute of a <meta> tag, e.g. "keywords". The "categories" parameter can be used to specify the individual category values.	None
categories[4]	"categories" are a map with which an individual internal category name (e.g. "Cat_Common") can be displayed in a value in the <meta> tag specification (e.g. "common"). The advantage on assigning internal category names is that, in case of different engine configurations with the same category name – despite potentially different values, the <meta> tag can be accessed (see below).	None
credentials	Authentication information (see below)	None
localeField[4]	Corresponds to the "categoryField" parameter, however for specifying the language field. Here it is important that <meta> tags with the attribute "http-equiv" are not taken into account.	None
locales[4]	Corresponds to the "categories" parameter, however for language values and language abbreviation.	None
allowed	List of allowed URLs (see below)	All URLs are allowed
forbidden	List of not allowed URLs (see below)	No URLs are forbidden

<attribute> tag		
Parameter[1]	**Value**	**Default value**
maxContentLength	Size restriction for the content of a document to be saved. If the content of the document is larger than the restriction the first characters are stored up to where the size restriction is reached.	16384 (=16 kB)
maxDocuments	Maximum number of documents allowed for indexing (abort criterion)	Number is unlimited
maxFieldLength	Maximum number of words which can be saved in a field of a document. If a document contains more words it is cut-off during the indexing. If a larger value is selected, sufficient main memory must be available.	10000
maxTime	Maximum duration of the indexing process. The value expected is a number followed by a letter. This letter defines whether the information is specified in seconds (*s*), minutes (*m*), hours (*h*), days (*d*) or weeks (*w*). Example: *1d*	Duration is unlimited
maxThreads	Simultaneously running indexing threads. The more threads given here the higher the memory requirement during indexing.	1
threadPriority	Priority of the indexing thread (1=Minimum, 10=Maximum).	5
url or urls[3]	A URL or a list of URLs (see below), which are used as a starting point by the Spider engine.	None

[1] "name" attribute
[2] Mandatory parameter
[3] It Specification of one of the two parameters is mandatory

[4] The parameters "locales" and "localField" or "categories" and "categoryField" must each be given together

"allowed" / "forbidden" parameters:

An <attribute> tag is defined for the "allowed" and "forbidden" parameters. Within this tag, other <attribute> tags are used to specify a list of filters (parts of a URL). The following RegexWebLinkFilter, for which a regular expression (in Java syntax) must be given, must be used for these <attribute> tags.

Class:

`de.espirit.firstspirit.opt.search.engine.spider.link.RegexWebLinkFilter`

"list" must be given as the type for the "allowed" and "forbidden" parameters. The RegexWebLinkFilter must be used for the inner <attribute> tags.

Example:

```
<attribute name="allowed" type="list">
  <attribute
class="de.espirit.firstspirit.opt.search.engine.spider.link.RegexW
ebLinkFilter"> intranet.myServer.de/de/</attribute>
 <attribute
class="de.espirit.firstspirit.opt.search.engine.spider.link.RegexW
ebLinkFilter"> intranet.myServer.de/en/</attribute>
</attribute>
```

"urls" parameter:

Specification of the URL is similar to specification of the "allowed" and "forbidden" parameters, however no "class" attribute is given in the inner <attribute> tag.

Example:

```
<attribute name="urls" type="list">
<attribute>http://localhost:80/site1/index.html</attribute>
<attribute>http://localhost:80/site2/index.html</attribute>
</attribute>
```

"categories" / "locales" parameters:

A map with key/value pairs is given for these two parameters. The value "map" must be given as the "type" attribute. Within the parameter an <attribute> tag is given for each key / value pair. The key is given with the "name" attribute, the value within the opening and closing tag.

The key ("name" attribute) is the category name to be used internally or the language abbreviation to be used (e.g. "int"). The value is used to specify what the value for a category or language abbreviation stored in the search index is. "categoryField" or "localField“ is given to specify the field in which the value for a category or language abbreviation is stored in the search index.

Example:

```
<attribute name="locales" type="map">
  <attribute name="de">DE</attribute>
  <attribute name="en">GB</attribute>
</attribute>
<attribute name="localField">language</attribute>
```

The example contains two key/value pairs for the "locales" parameter. The first pair consists of the key "de" and the value "DE". The key of the second pair is "en" and the value is "GB". The parameter "localField" is used to specify that the values for the language abbreviation in the search field are to be found in the "language" field. The values for the language abbreviations in this field are "DE" (German) and "GB" (English). In FirstSpirit Search, "de" and "en" are used for these language abbreviations by specification with the "locales" parameters.

```
<attribute name="categories" type="map">
  <attribute name="int">intro</attribute>
  <attribute name="doc">documentation</attribute>
</attribute>
<attribute name="categoryField">categories</attribute>
```

"credentials" parameter:

If the website to be indexed requires authentication, the information required for this can be configured in the Spider engine. The "credentials" parameter is configured as a list of map entries, whereby each individual entry stands for an area and an authentication method. This enables different authentication with various servers in one index run.

The parameters of the map entries:

<attribute> tag		
Parameter[1]	**Value**	**Default value**
authType[2]	Type of authentication. Possible values are: "ntlm" or "basic".	None
authUsername[2]	Specification of the user name	None
authPassword[2]	Specification of the password for the given user	None

<attribute> tag		
Parameter[1]	**Value**	**Default value**
authHost[2][3]	Specification of the server against which the authentication is to take place	None
authDomain[2][3]	Authentication domain	None
scopeHost	The authentication applies to the given host.	ANY_HOST (i.e.: all hosts)
scopePort	The authentication applies to the given port	ANY_PORT (i.e.: all ports)

[1] "name" attribute
[2] Mandatory parameter
[3] Specification only for "authType" "ntlm"

Example:

```
<attribute name="credentials" type="list">
  <attribute type="map">
    <attribute name="authType">ntlm</attribute>
    <attribute name="authUsername">smith</attribute>
    <attribute name="authPassword">verysecret</attribute>
    <attribute name="authHost">myserver</attribute>
    <attribute name="authDomain">MYCOMPANY</attribute>
  </attribute>
</attribute>
```

Complete example:

```
<?xml version="1.0" encoding="UTF-8"?>
<service ...>
  ...
  <engine
class="de.espirit.firstspirit.opt.search.engine.spider.SpiderEngin
e">
    <attribute name="urls" type="list">
        <attribute>http://intranet.mydomain.com</attribute>
        <attribute>http://extranet.mydomain.com</attribute>
    </attribute>
    <attribute name="index">/var/www/search/index</attribute>
    <attribute name="maxThreads">2</attribute>
    <attribute name="threadPriority">1</attribute>
    <attribute name="allowed" type="list">
      <attribute class="
de.espirit.firstspirit.opt.search.engine.spider.link.RegexWebLinkF
ilter ">intranet</attribute>
      <attribute class="
de.espirit.firstspirit.opt.search.engine.spider.link.RegexWebLinkF
ilter ">extranet</attribute>
    </attribute>
    <attribute name="maxDocuments">1000</attribute>
    <attribute name="maxTime">60m</attribute>
    <attribute name="credentials" type="list">
      <attribute type="map">
        <attribute name="authType">ntlm</attribute>
        <attribute name="authUsername">smith</attribute>
        <attribute name="authPassword">verysecret</attribute>
        <attribute name="authHost">myserver</attribute>
        <attribute name="authDomain">MYCOMPANY</attribute>
      </attribute>
    </attribute>
  </engine>
</service>
```

4.5.4.3 Engine adapter "Lucene"

The Lucene engine adapter integrates an existing index created with the Lucene library.

The engine supports multilingualism and categories, if this information is available as fields in the documents (see "categoryField", "categories", "localField" and "locales" parameters). If the search query contains a restriction with respect to language or category the query transferred to Lucene is extended by the corresponding field search.

<engine> tag	
Attribute	**Value**
class[1]	de.espirit.firstspirit.opt.search.engine.adapter.LuceneEngineAdapter

[1] Mandatory parameter

<attribute> tag		
Parameter[1]	**Value**	**Default value**
analyzer	Details of the Lucene analyzer to be used. A list of the possible analyzers is given in the API documentation provided by Lucene: http://lucene.apache.org/java/2_1_0/api/org/apache/lucene/analysis/Analyzer.html Attention: The analyzer is not configured by using the attribute-element with the class name as usual, but via the following expression: `<analyzer class="org.apache.lucene.analysis.SimpleAnalyzer"/>` (NOT: `<attribute name="analyzer" class="org.apache.lucene.analysis.SimpleAnalyzer"/>`)	StandardAnalyzer
index[2]	Specification of the base directory for the Lucene index	None
defaultField	This parameter can be used to define the default field name for the search. As a default the contents are stored in the "content" field by the indexing through Lucene.	content
categoryField[3]	The pages in a website can be equipped with meta information (<meta> tags). This information is stored in the index during indexing. The "categoryField" parameter is used to define which <meta> tag information is to be used for the categories. The value of "categoryField" therefore corresponds to the value of the "name" attribute of a <meta> tag, e.g. "keywords". The "categories" parameter can be used to specify the individual category values.	None

<attribute> tag		
Parameter[1]	**Value**	**Default value**
categories[3]	"categories" are a map with which an individual internal category name (e.g. "Cat_Common") can be displayed in a value in the <meta> tag specification (e.g. "common"). The advantage on assigning internal category names is that, in case of different engine configurations with the same category name – despite potentially different values, the <meta> tag can be accessed (see below).	None
localField[3]	Corresponds to the "categoryField" parameter, however for specifying the language field. Here it is important that <meta> tags with the attribute "http-equiv" are not taken into account.	None
locales[3]	Corresponds to the "categories" parameter, however for language values and language abbreviation.	None

[1] "name" attribute

[2] Mandatory parameter

[3] The parameters "locales" and "localField" or "categories" and "categoryField" must each be given together

"categories" / "locales" parameters:

A map with key/value pairs is given for these two parameters. The value "map" must be given as the "type" attribute. Within the parameter an <attribute> tag is given for each key / value pair. The key is given with the "name" attribute, the value within the opening and closing tag.

The key ("name" attribute) is the category name to be used internally or the language abbreviation to be used (e.g. "int"). The value is used to specify what the value for a category or language abbreviation stored in the search index is. "categoryField" or "localField“ is given to specify the field in which the value for a category or language abbreviation is stored in the search index.

Example:

```
<attribute name="locales" type="map">
  <attribute name="de">DE</attribute>
  <attribute name="en">GB</attribute>
</attribute>
```

```
<attribute name="localField">language</attribute>
```

The example contains two key/value pairs for the "locales" parameter. The first pair consists of the key "de" and the value "DE". The key of the second pair is "en" and the value is "GB". The parameter "localField" is used to specify that the values for the language abbreviation in the search field are to be found in the "language" field. The values for the language abbreviations in this field are "DE" (German) and "GB" (English). In FirstSpirit Search, "de" and "en" are used for these language abbreviations by specification with the "locales" parameters.

```
<attribute name="categories" type="map">
  <attribute name="int">intro</attribute>
  <attribute name="doc">documentation</attribute>
</attribute>
<attribute name="categoryField">categories</attribute>
```

4.5.4.4 Engine adapter "JDBC"

The JDBC engine adapter links any databases to FirstSpirit Search, for which a JDBC driver is available. It converts the search query into a matching SQL command and extracts the required columns or column values from the database results. The results can then be accessed in the web application.

<engine> tag	
Attribute	**Value**
class[1]	de.espirit.firstspirit.opt.search.engine.adapter.JdbcEngineAdapter

[1] Mandatory parameter

<attribute> tag		
Parameter[1]	**Value**	**Default value**
driver[2]	Class name of the JDBC driver, e.g. `de.espirit.or.impl.mysql.MySQLLayer`	None
url[2]	JDBC URL	None
username[2]	User name for authentication via the JDBC driver in the database	None
password[2]	Password for authentication of the user via the JDBC driver in the database	None
columns[2]	Specification of the database columns (column name in the database), which is to be taken into account for the query	All columns ("*")
rawQuery[2]	This parameter can be used to limit the query (WHERE clause). The `$query$` wildcard can be used within the parameter. This wildcard is replaced with the search text	None

<attribute> tag		
Parameter[1]	**Value**	**Default value**
resultMapping[2]	The parameters are used to display the values of the individual database columns in variables which can be used in the web application. The value expected by the parameter is a comma-separated list of key/value pairs. The key (variable identifier) is separated from the value (database column) by an equals sign ("="). There is a peculiarity regarding the value or database column: The name of the database column must be enclosed in dollar symbols (e.g: $COLUMNNAME$). Example: `content=$DATA$,title=$TITLE$` For use in the web application it is important for the following variables to be displayed in the resultMapping in database columns if they are used in the application: ▪ "author" ▪ "categories" ▪ "content" ▪ "date" ▪ "locale" (language) ▪ "score" (number of hits) ▪ "size" ▪ "title" ▪ "url" (URL of the hit)	None
fixedCategory	Search results are returned with the given category.	None

[1] "name" attribute
[2] Mandatory parameter

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<service ...>
  ...
  <engine
class="de.espirit.firstspirit.opt.search.engine.adapter.JdbcEngine
Adapter">
    <attribute
name="driver">de.espirit.or.impl.mysql.MySQLLayer</attribute>
    <attribute
name="url">jdbc:mysql://localhost:3306/intranet</attribute>
```

```
    <attribute name="user">user</attribute>
    <attribute name="password">password</attribute>
    <attribute name="columns">*</attribute>
    <attribute name="tables">CALLS</attribute>
    <attribute name="rawQuery">(DESCRIPTION like "%$query$%") or
(SUBJECT like "%$query$%") or (SOLUTION like "%$query$%") or
(CLIENT like "%$query$%") or (CONTACT like "%$query$%") or
(CALLSTATUS like "%$query$%")</attribute>
    <attribute
name="resultMapping">title=$CLIENT$,content=$DESCRIPTION$,score=10
00,url=http://intranet/db/view.jsp?id=$ID$</attribute>
    <attribute name="fixedCategories">db</attribute>
  </engine>
</service>
```

4.5.4.5 Engine proxy "Simple filter"

This engine proxy adds a quantity of fixed specified filters to each search. This enables, for example, security relevant information to be filtered our or invalid data records to be hidden. The filters given here automatically use "OR" logic. Only the results to which the filter concern applies (positive filter) are returned.

Another engine must be defined or referenced within the <engine> tag of the proxy!

<engine> tag	
Attribute	**Value**
class[1]	de.espirit.firstspirit.opt.search.engine.proxy.FilterEngineProxy

[1] Mandatory parameter

<attribute> tag		
Parameter[1]	**Value**	**Default value**
filters	List of result filters. These filters are added to each search. The individual filters use "OR" logic (see below).	None

[1] "name" attribute

"filters" parameter:

The "filters" parameter defines an <attribute> tag. Within this tag the <filter> tag is used to specify a list of filters. For these <filter> tags, the RegexFilter (class: `de.espirit.firstspirit.opt.search.filter. RegexFilter`) must be used. Two other <attribute> tags are to be specified in the <filter> tag.

For the first attribute with the value "property" for the "name" attribute the property (e.g. the variable "url") to which the filter expression is to be applied is to be given.

The second attribute (with the "name" attribute value "pattern") contains the filter expression, which is to be given as a regular expression (in Java syntax).

"list" must be given as the type for the "filters" parameter. The class of the RegexFilter must be given for the <filter> tag. Within the <filter> tag, two <attribute> tags ("property" and "pattern") are to be given.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<service ...>
  ...
  <engine
class="de.espirit.firstspirit.opt.search.engine.proxy.FilterEngine
Proxy">
    <engine idref="_engine" />
    <attribute name="filters" type="list">
      <filter class="de.espirit.firstspirit.opt.search.filter.
RegexFilter">
        <attribute name="property">url</attribute>
        <attribute name="pattern">\.html?$</attribute>
      </filter>
      <filter class="de.espirit.firstspirit.opt.search.filter.
RegexFilter">
        <attribute name="property">url</attribute>
        <attribute name="pattern">\.php$</attribute>
      </filter>
      <filter class="de.espirit.firstspirit.opt.search.filter.
RegexFilter">
        <attribute name="property">url</attribute>
        <attribute name="pattern">\.jsp$</attribute>
      </filter>
    </attribute>
  </engine>
  ...
</service>
```

In the example, three filters are defined for the "url" variable. The first filter only allows elements whose URL ends with ".htm" or ".html". The second filter allows all elements whose URL ends with ".php". The third filter allows all elements whose URL ends with ".jsp".

4.5.4.6 Engine proxy "Multilingualism (filter)"

This engine proxy simulates multilingualism with the help of filters. Here a language corresponds to precisely one filter. If the search is limited to a specific language, the corresponding filter is activated and used. When the results are returned they are analysed with the help of the filter and are assigned to a language.

Another engine must be defined or referenced within the <engine> tag of the proxy!

<engine> tag	
Attribute	**Value**
class[1]	de.espirit.firstspirit.opt.search.engine.proxy.FilterLocalizeEngineProxy

[1] Mandatory parameter

<attribute> tag		
Parameter[1]	**Value**	**Default value**
locales	A map is used to display language abbreviations (e.g. "de") in the filter objects of the search results (see below).	None

[1] "name" attribute

<u>"locales" parameter:</u>
The "locales" parameter defines an <attribute> tag. Within this tag the <filter> tag is used to specify a map of filters. The RegexFilter is to be used for these <filter> tags:
Class: `de.espirit.firstspirit.opt.search.filter.RegexFilter`
In addition, a language abbreviation (e.g. "de") must be given in the "name attribute" for each <filter> tag. This specifies which filter is used for which language.

Two other <attribute> tags are to be specified in the <filter> tag.

For the first attribute with the value "property" for the "name" attribute the property to which the filter expression is to be applied is to be given.

The second attribute (with the "name" attribute value "pattern") contains the filter expression, which is to be given as a regular expression (in Java syntax).

"map" must be given as the type for the "locales" parameter. The language abbreviation ('name") and the class of the RegexFilter must be given for the <filter> tag. Within the <filter> tag, two <attribute> tags ("property" and "pattern") are to be given.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<service ...>
  ...
  <engine
class="de.espirit.firstspirit.opt.search.engine.proxy.FilterLocali
zeEngineProxy">
    <engine idref="_engine" />
    <attribute name="locales" type="map">
      <filter name="de"
class="de.espirit.firstspirit.opt.search.filter.RegexFilter">
        <attribute name="property">url</attribute>
        <attribute name="pattern">/de/</attribute>
      </filter>
      <filter name="en"
class="de.espirit.firstspirit.opt.search.filter.RegexFilter">
        <attribute name="property">url</attribute>
        <attribute name="pattern">/en/</attribute>
      </filter>
    </attribute>
  </engine>
</service>
```

In the example, two filters are defined for the "url" variable. The first filter is defined for German (language abbreviation "de") and is used if the URL contains the character string "/de/". By contrast, the second filter is defined for English (language abbreviation "en") and is used if the URL contains the character string "/en/".

4.5.4.7 Engine proxy "Categorisation (filter)"

This engine proxy simulates categories with the help of filters. Here a category corresponds to precisely one filter. If the search is limited to specific categories, the corresponding filters are activated and used. When the results are returned they are analysed with the help of the filter and are categorised.

Another engine must be defined or referenced within the <engine> tag of the proxy!

<engine> tag	
Attribute	**Value**
class[1]	de.espirit.firstspirit.opt.search.engine.proxy.FilterCategorizeEngineProxy

[1] Mandatory parameter

<attribute> tag		
Parameter[1]	**Value**	**Default value**
categories	A map is used to display categories (e.g. "pdf") in the filter objects of the search results (see below).	None
inheritCategories	The parameter can be used to specify whether the categories of the search engine to be filtered are to be inherited (value "true") or ignored (value "false").	false

[1] "name" attribute

"categories" parameter:

The "categories" parameter defines an <attribute> tag. Within this tag the <filter> tag is used to specify a map of filters. The RegexFilter is to be used for these <filter> tags:
Class: `de.espirit.firstspirit.opt.search.filter.RegexFilter`.
In addition, a category (e.g. "pdf") must be given in the "name attribute" for each <filter> tag. This specifies which filter is to be used for which category.

Two other <attribute> tags are to be specified in the <filter> tag. For the first attribute with the value "property" for the "name" attribute the property to which the filter expression is to be applied is to be given. The second attribute (with the "name" attribute value "pattern") contains the filter expression, which is to be given as a regular expression (in Java syntax).

! *"map" must be given as the type for the "categories" parameter. The language abbreviation ('name") and the class of the RegexFilter must be given for the <filter> tag. Within the <filter> tag, two <attribute> tags ("property" and "pattern") are to be given.*

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<service ...>
  ...
  <engine
class="de.espirit.firstspirit.opt.search.engine.proxy.FilterCatego
rizeEngineProxy">
    <engine idref="_engine" />
    <attribute name="categories" type="map">
      <filter name="pdf"
class="de.espirit.firstspirit.opt.search.filter.RegexFilter">
        <attribute name="property">url</attribute>
        <attribute name="pattern">\.pdf$</attribute>
      </filter>
      <filter name="html"
class="de.espirit.firstspirit.opt.search.filter.RegexFilter">
        <attribute name="property">url</attribute>
        <attribute name="pattern">\.html?$</attribute>
      </filter>
      <filter name="office"
class="de.espirit.firstspirit.opt.search.filter.RegexFilter">
        <attribute name="property">url</attribute>
        <attribute name="pattern">(\.doc$)|(\.xsl$)</attribute>
      </filter>
    </attribute>
  </engine>
</service>
```

In the example, three filters are defined for the "url" variable. The first filter is used to assign the category "pdf" to all documents whose URL ends with ".pdf". The category "html" is assigned to all documents whose URL ends with ".htm" or ".html" (second filter). The category "office" is assigned to all documents whose URL ends with ".doc" or ".xsl" (third filter).

4.5.4.8 Engine proxy "Simple engine group"

This engine proxy, similar to a meta search machine, groups together several engines to form one. Such a combination is also called an engine group. The respective categories and languages are taken into account and a search is only forwarded to those engines which also support it. The results of the individual engines are grouped together and returned depending on the required sort criterion. The procedure is given in the following figure (see Figure 4-1).

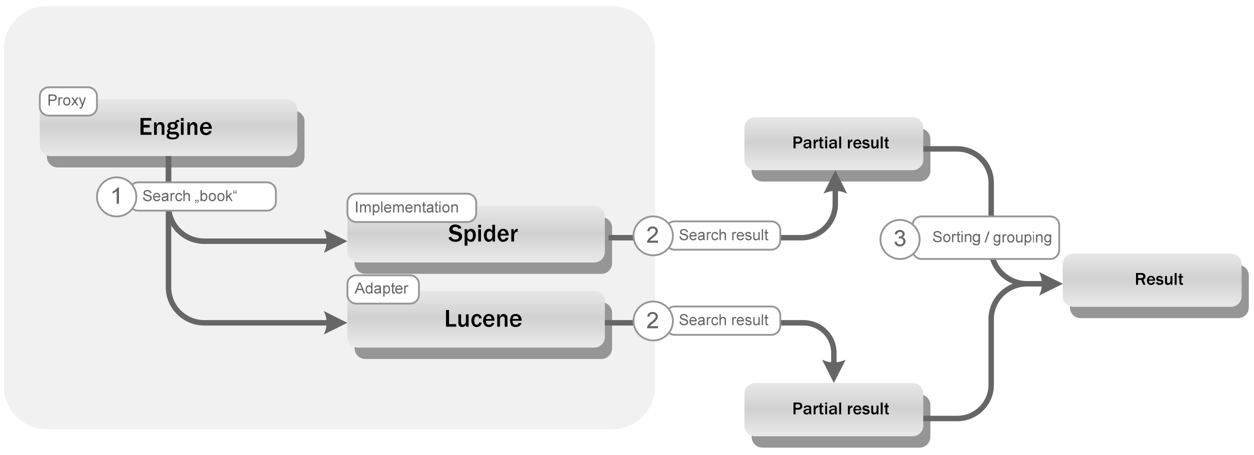


Figure 4-1: Search query / search results for multi-engine proxy

<engine> tag	
Attribute	**Value**
class[1]	de.espirit.firstspirit.opt.search.engine.proxy.MultiEngineProxy

[1] Mandatory parameter

<attribute> tag		
Parameter[1]	**Value**	**Default value**
engines[2]	A list of search engines (see below).	None

[1] "name" attribute

[2] Mandatory parameter

"engines" parameter:

The "engines" parameter defines an <attribute> tag. Within this tag the <engine> tag is used to specify a list of search engines.

! *"list" must be given as the type for the "engines" parameter. Further, one or several search engines must be specified within the <attribute> tag.*

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<service ...>
  ...
  <engine
class="de.espirit.firstspirit.opt.search.engine.proxy.MultiEngineP
roxy">
    <attribute name="engines" type="list">
      <engine idref="_engine_de" />
      <engine idref="_engine_en" />
    </attribute>
  </engine>
</service>
```

4.5.4.9 Engine proxy "Multilingualism (engine group)"

This engine proxy simulates multilingualism through the combination of several engines (engine group). A language is displayed on precisely one engine. If the search is limited to a specific language the query is forwarded to the corresponding engine only. On the other hand, depending on the engine (which delivers the result), the result is extended to include the relevant language information. The results of the individual engines are grouped together and returned depending on the required sort criterion.

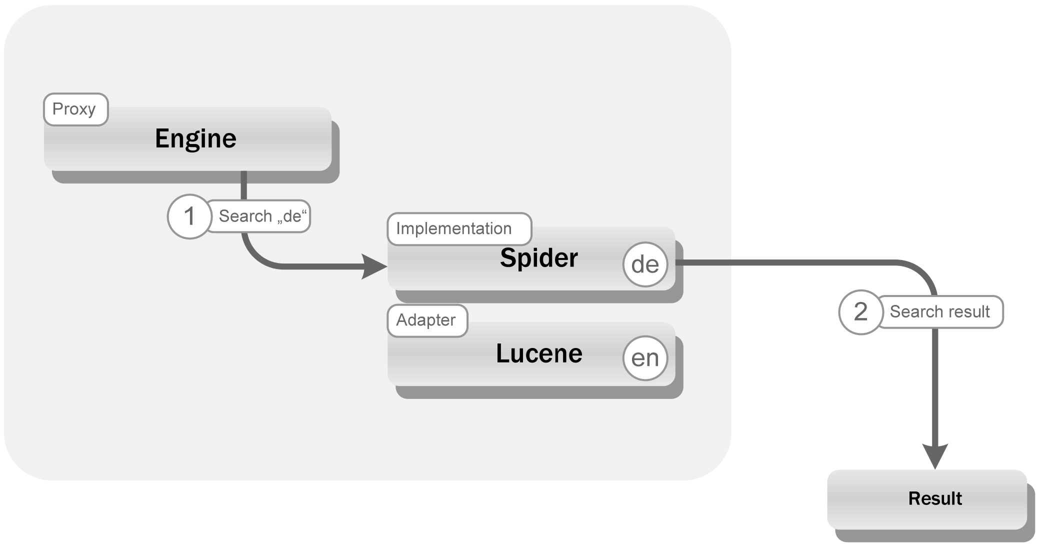


Figure 4-2: Search query / search results for MultiLocalizeEngineProxy

<engine> tag	
Attribute	**Value**
class[1]	de.espirit.firstspirit.opt.search.engine.proxy.MultiLocalizeEngineProxy

[1] Mandatory parameter

<attribute> tag		
Parameter[1]	**Value**	**Default value**
locales[2]	Display of language abbreviations (e.g. "de") in individual search engines (see below).	None

[1] "name" attribute
[2] Mandatory parameter

"locales" parameter:
A map with key/value pairs is given for this parameter. The value "map" must be given as the "type" attribute. Within the parameter an <engine> tag is given for each key / value pair. The key is given with the "name" attribute, the value within the opening and closing tag. The value is the engine object.

Example:

```
<attribute name="locales" type="map">
  <engine name="de" idref="_engine_de" />
  <engine name="en" idref="_engine_en" />
</attribute>
```

The example contains two key/value pairs for the "locales" parameter. The first pair consists of the key (language abbreviation) "de" and the engine object, which is referenced by the "idref" attribute ("_engine_de"). In the second pair the key (language abbreviation) is "en" and here the engine is referenced with the ID "_engine_en".

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<service ...>
  ...
  <engine
class="de.espirit.firstspirit.opt.search.engine.proxy.MultiLocaliz
eEngineProxy">
    <attribute name="locales" type="map">
      <engine name="de" idref="_engine_de" />
      <engine name="en" idref="_engine_en" />
    </attribute>
  </engine>
</service>
```

4.5.4.10 Engine proxy "Categorisation (engine group)"

This engine proxy simulates categories through the combination of several engines (engine group). A category is displayed on precisely one engine. If the search is limited to specific categories the query is forwarded to the corresponding engine only. On the other hand, depending on the engine (which delivers the result), the result is extended to include the relevant categories. A search query is made in each defined engine. The results of the individual engines are grouped together and returned depending on the required sort criterion.

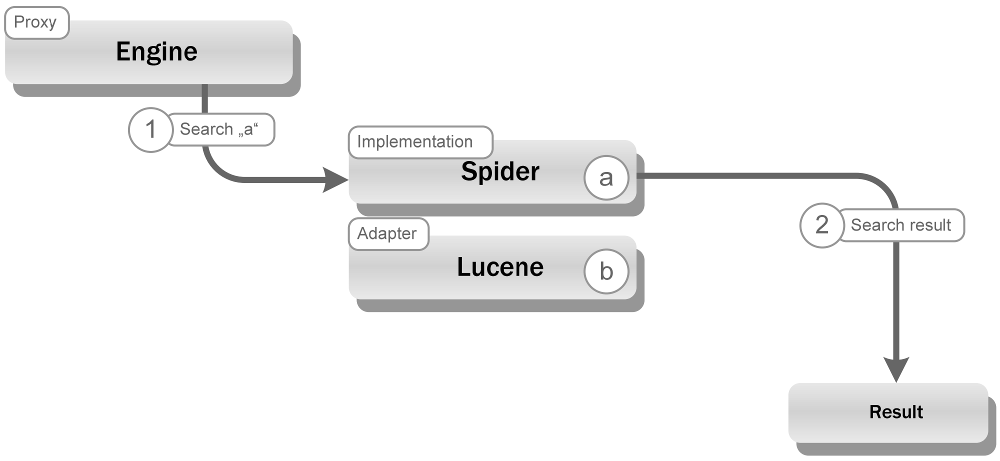


Figure 4-3: Search query / search results for MultiCategorizeEngineProxy

<engine> tag	
Attribute	**Value**
class[1]	de.espirit.firstspirit.opt.search.engine.proxy.MultiCategorizeEngineProxy

[1] Mandatory parameter

<attribute> tag		
Parameter[1]	**Value**	**Default value**
categories[2]	Display of categories (e.g. "pdf", "intranet") in individual search engines (see below).	None

[1] "name" attribute
[2] Mandatory parameter

<u>"categories" parameter:</u>
A map with key/value pairs is given for this parameter. The value "map" must be given as the "type" attribute. Within the parameter an <engine> tag is given for each key / value pair. The key is given with the "name" attribute, the value within the opening and closing tag. The value is the engine object.

Example:

```
<attribute name="categories" type="map">
  <engine name="intranet" idref="_engine_intranet" />
  <engine name="internet" idref="_engine_internet" />
</attribute>
```

The example contains two key/value pairs for the "categories" parameter. The first pair consists of the key (category) "intranet" and the engine object, which is referenced by the "idref" attribute ("_engine_intranet"). In the second pair the key (category) is "internet" and here the engine is referenced with the ID "_engine_internet".

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<service ...>
  ...
  <engine
class="de.espirit.firstspirit.opt.search.engine.proxy.MultiCategor
izeEngineProxy">
    <attribute name="categories" type="map">
      <engine name="intranet" idref="_engine_intranet" />
      <engine name="internet" idref="_engine_internet" />
    </attribute>
  </engine>
</service>
```

4.5.4.11 Engine proxy "Personalisation"

This engine proxy can be used to output search results depending on the rights of the groups or users. By connecting FirstSpirit Search and FirstSpirit Personalisation, search results can be withheld from certain user groups or certain search results can be made available to selected user groups.

The allowed groups must be deposited in the search index for the individual documents to enable personalisation of documents. To deposit group information for the documents, other meta information (<meta> tag) can be specified in the HTML files which is included in the search index by a Spider engine (see Chapter 4.5.4.2, page 37).

Example:

```
<meta name="groups" content="ceo,qeo" />
```

The <meta> tag is to be used to make the current document in the example visible to the "ceo" nd "qeo" groups only. During indexing by the Spider engine, relevant information was stored in the index for this document (under the "groups" property with "ceo,geo" content).

In a search query (client side or web application) the search is supplemented to include the groups (or roles) of the current user.

The group information of the currently logged in user and the individual documents enable the proxy to only return documents in the search result for which the user has sufficient permissions.

Another engine must be defined or referenced within the <engine> tag of the proxy!

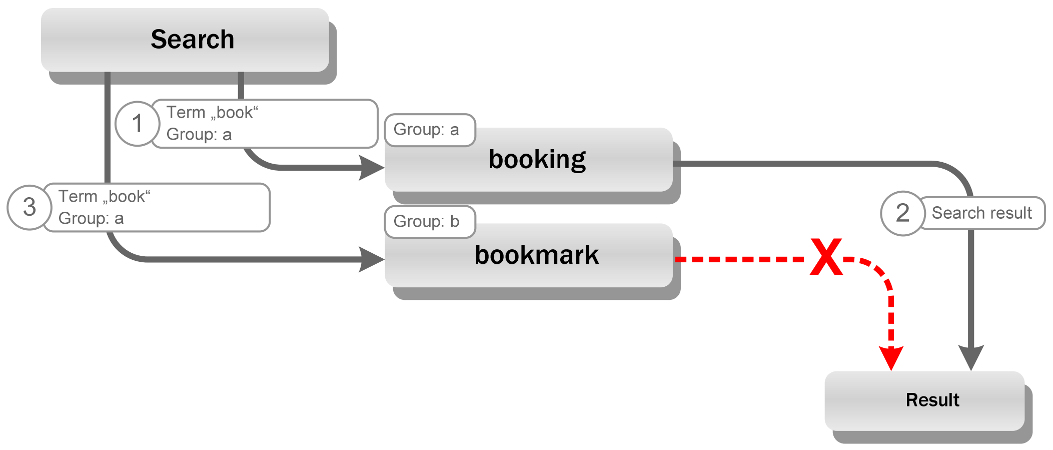


Figure 4-4: Search query / search results for SecurityEngineProxy

<engine> tag	
Attribute	**Value**
class[1]	de.espirit.firstspirit.opt.search.engine.proxy.SecurityEngineProxy

[1] Mandatory parameter

<attribute> tag		
Parameter[1]	**Value**	**Default value**
acceptUndefinedResults	This parameter can be used to specify whether search results for which no group or role information has been deposited are allowed. If "true" is specified search results are allowed without group information, if "false" they are not.	false
resultKey[2]	"resultKey" is used to specify the identifier or field name for the group information (property in the above example), which is deposited in the search index for a document.	None

propertyKey[2]	Specification of the attribute name, under which the group information of the currently logged in user is deposited in the HTTP session (web application). In FirstSpirit Personalisation the groups are stored under the session attribute "FIRSTpersonalisation.usergroups".	None
urlGroupMappings	Specification of a URL for a file with additional group information. This enables other files, for example PDF files, to be extended to include group information. The following format must be used in the given file: `URL=Groups` `URL` is the absolute URL starting from the web application. A comma separated list of groups must be given for `Groups`.	None

[1] "name" attribute

[2] Mandatory parameter

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<service ...>
  ...
  <engine
class="de.espirit.firstspirit.opt.search.engine.proxy.SecurityEngi
neProxy">
    <engine idref="_engine" />
    <attribute name="acceptUndefinedResults">false</attribute>
    <attribute name="resultKey">groups</attribute>
    <attribute
name="propertyKey">FIRSTpersonalisation.usergroups</attribute>
    <attribute name="urlGroupMappings">file:/META-
INF/secinfo.txt</attribute>
  </engine>
</service>
```

`/META-INF/secinfo.txt`:

```
/media/doc/howto-shutdown.pdf=admin
/media/office/payrolls.xls=office,ceo
```

4.5.4.12 Engine proxy "Session management"

This engine proxy monitors an engine and adds a timeout handling to the returned sessions. If these sessions are not accessed for a certain time they are automatically closed and the associated resources are released.

! *Another engine must be defined or referenced within the <engine> tag of the proxy!*

<engine> tag	
Attribute	**Value**
class[1]	de.espirit.firstspirit.opt.search.engine.proxy.MonitorEngineProxy

[1] Mandatory parameter

<attribute> tag		
Parameter[1]	**Value**	**Default value**
sessionTimeout	Specification of the session timeout in milliseconds.	1800000 (30 min)

[1] "name" attribute

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<service ...>
  ...
  <engine
class="de.espirit.firstspirit.opt.search.engine.proxy.MonitorEngin
eProxy">
    <engine idref="_engine" />
    <attribute name="sessionTimeout">30000</attribute>
  </engine>
</service>
```

5 JSP tags

5.1 General information

The following chapter explains the functions of the tag library for controlling the web application. Both general, formal definitions are described (e.g. specification of a prefix for the JSP tag) and the use of available JSP tags is explained.

5.2 Prefix

The relevant Taglib must be given in the JSP pages first for use of FirstSpirit Search. This documentation uses the prefix "fss" for FirstSpirit-Search tags.

Example of integration in JSP pages:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="fs-search" prefix="fss" %>
```

Here it must be noted that the URI for FirstSpirit is always "fs-search".

*If the prefix "fss" is changed to another value, this prefix must be used for the individual tags, i.e. "<**myPrefix**:isTrue>"" instead of "<**fss**:isTrue>".*

5.3 Variables defined by tags

Variables are defined in the tags `<fss:getSearchDetails>`, `<fss:iterateResults>`, `<fss:navFrame>`, `<fss:navTop>`, `<fss:navBottom>`, `<fss:url>`.

Overlapping with manually defined variables in the JSP page can occur.

To avoid overlapping the following variable identifiers should not be used (or circumspectly only):

- absNo
- ascending
- author
- categories
- content
- cursor
- cursorId
- date
- encodedUrl
- locales
- orderBy
- pageNo
- pageSize
- query
- relNo
- score
- se
- size
- title
- totalPages
- totalResults
- url
- urlPageNo
- urlPageSize

Please refer to the individual tag descriptions to find out which tag defines which variable (see Chapter 5.5 ff.).

5.4 Overview

FirstSpirit-Search JSP tags enable presentation of the search results in a web application as well as the display and determination of detailed information (total number of results, etc.) and display of suitable navigation.

Overview of the JSP tags:

Name	Description
`isTrue`, `isFalse`	Conditional output of contents.
`getSearchDetails`[1]	Query information on the search.
`iterateResults`[2]	Output the search results.
`isCategory`[1], `isLocale`[1]	Provides the decision for the question whether a search result belongs to a category or to a language.
`highlighter`	Highlighted display of the search words.
`navigation`	Display of the navigation bar.
`navTop`[1], `navBottom`[1]	Display the header or footer of the current navigation.
`navFrame`[1][2]	Display the window of the current navigation.
`url`[1]	Generate URL.

[1] Informal tags

[2] Output tags

Informal tags provide information about a state / condition or a circumstance, for example that a result belongs to a category. The output can be organised differently (conditional output) for the check for whether it belongs to a category (condition) depending on whether a result belongs to this category or not. The two JSP tags `<fss:isTrue>` and `<fss:isFalse>` are available for this. If the checked condition (result belongs to a certain category) is fulfilled, the content is displayed in `<fss:isTrue>`. If the condition is not fulfilled (result does not belong to a certain category) the content is displayed in `<fss:isFalse>`. Informal tags are executed precisely once only.

Output tags on the other hand are executed repeatedly, for example for each search result of the current results page.

5.5 Display search results

Search results are displayed in a web application page by page. The page size, i.e. the number of results per (virtual) page, is determined on starting the search by using the `pageSize` parameter. This value can however be changed at any time during the display.

5.5.1 Query information (<fss:getSearchDetails>)

This JSP tag provides several values of the current search as variables. If at least one result was found the content given within the `<fss:isTrue>` tag is output. If no result was found the content given within the `<fss:isFalse>` tag is output. If neither a `<fss:isTrue>`, nor a `<fss:isFalse>` is given the content is always output.

Within the tag `<fss:getSearchDetails>` the variables: "ascending", "cursor", "cursorId", "orderBy", "pageNo", "pageSize", "query", "se", "totalPages" and "totalResults" are available.

Variables:

Variable	Meaning	Return data type
ascending	Sorting of the results. The value is "true" if the results are sorted in ascending alphabetical order and "false" if they are sorted in descending order.	Boolean
cursor	Cursor object. Returns the object for the currently displayed (virtual) page.	WebCursor[8]
cursorId	Returns the ID of the cursor object	String
orderBy	Returns which field was used to sort the results ("author", "date", "score", "size" or "title").	String
pageNo	Number of the currently displayed (virtual) page (starting with "1").	Integer
pageSize	Number of elements to be displayed on the page.	Integer
query	Returns the search text.	String
se	Returns the name of the search engine used.	String
totalPages	Number of all (virtual) pages (calculation basis: Number of data records and "pageSize")	Integer
totalResults	Number of all elements (over all pages). If the number of elements not (yet) known, -1 will returned.	Integer

Example:

```
<fss:getSearchDetails>
  <fss:isTrue>
    <p><%= totalResults %> results found.</p>
  </fss:isTrue>
  <fss:isFalse>
    No results found!
  </fss:isFalse>
</fss:getSearchDetails>
```

[8] Complete class names: de.espirit.firstspirit.opt.search.web.WebCursor

5.5.2 Output of all elements of a page (<fss:iterateResults>)

The results of a search are spread over different (virtual) pages. The results of the current (virtual) page are output with the JSP tag `<fss:iterateResults>`. Apart from the known fields such as "Author", "Title" and "URL", the "properties" attribute can be used to make available any values from the results. In this way it is ensured that freely definable meta information from the documents can be accessed.

! *Use of the JSP tag `<fss:iterateResults>` is only possible if search results were found. The JSP tag `<fss:getSearchDetails>` can be used to check whether search results were found. For this reason the JSP tag `<fss:iterateResults>` should be used within `<fss:isTrue>` in an `<fss:getSearchDetails>` JSP tag.*

Attributes:

Attribute	Expected value	Mandatory parameter
dateFormat	Specification of the date format to be used, e.g. "dd.MM.yyyy". Please refer to the Sun-API documentation[9] for details of the syntax for the date format.	No

properties	Apart from the default variables, this attribute can be used to make additional results information available (e.g. rights).
	
	The attribute expects a comma-separated list of definitions. A definition is given in the format:
	
	`variableName`
	`variableName:variableType`
	`variableName:variableType=resultKey`
	`variableName=resultKey`
	
	"variableName" is used to specify what the variable is called within the JSP page. The optional information ":variableType" can be used to define
		No

[9] http://java.sun.com/j2se/1.5.0/docs/api/index.html?java/text/SimpleDateFormat.html

Attribute	Expected value	Mandatory parameter

	the type of variable (e.g. java.lang.String). If the type is not given, "java.lang.Object" is used.
	
	If the optional "=resultKey" is specified, the specified field is used for a result. If the field is not given the value of "variableName" is used as the field identifier.
	
	Example:
	`groups:java.lang.String`
	`groups`
	`g:java.lang.String:groups`	

Variables:

Variable	Meaning	Return data type
absNo	Number of the result in relation to the total number of results (over all pages) (starting with "1").	Integer
author	Returns the author.	String
categories	Returns the categories for a document.	String[]
content	Returns an extract from the document in which the search term was found.	String
date	Returns the date of the document. The date is formatted with the format used to specify the "dateFormat" attribute. If no date format was specified the default date format is used.	String
locale	Returns the language.	Locale
relNo	Number of the result within a (virtual) page.	Integer
score	Relevance or quality of a result (0 to 100).	Integer
size	Returns the size of the document. The unit of measurement depends on the search engine used and can vary from search engine to search engine. The unit of measurement used is usually bytes.	Long

title	Returns the title of the document.	String
url	Returns the URL of the document.	String

Example:

```
<fss:getSearchDetails>
  <fss:isTrue>

    <fss:iterateResults
properties="tel:java.lang.String=CUST_PHONE,fax:java.lang.String=C
UST_FAX">
      <%= relNo %> of <%= pageSize %>
      Telefon: <%= tel %>
      Telefax: <%= fax %>
    </fss:iterateResults>

  </fss:isTrue>
  <fss:isFalse>
    No results found!
  </fss:isFalse>
</fss:getSearchDetails>
```

In the example, two additional values from the results are made available as variables of the type "String". "CUST_PHONE" and "CUST_FAX" are given the identifiers "tel" and "fax" in the JSP page.

5.6 Checks

5.6.1 Category check (<fss:isCategory>)

The `<fss:isCategory>` tag can be used within the `<fss:getSearchDetails>` tag (see Chapter 5.5.1, page 70) and the `<fss:iterateResults>` tag (see Chapter 5.5.2, page 72).

When used within `<fss:getSearchDetails>`, `<fss:isCategory>` provides information about the search parameters used and within `<fss:iterateResults>` about the search result.

Within `<fss:isCategory>`, `<fss:isTrue>` and `<fss:isFalse>` are to be used. The content of `<fss:isTrue>` is output precisely when one of the given categories ("categories" attribute) corresponds to a category of the document (intersection). By contrast, the content of `<fss:isFalse>` is displayed precisely when no category matches a category of the document.

Attributes:

Attribute	Expected value	Mandatory parameter
categories	Comma separated list of categories for the comparison with the categories of the search result.	Yes

Example:

The first example demonstrates use of the JSP tag, to pre-assign the parameters of the current search to the search form:

```
<fss:getSearchDetails>
  <input type="text" name="query" value="<%= query %>" />
  | PDF <input type="checkbox" name="pdf"
    <fss:isCategory categories="pdf">
      <fss:isTrue>checked</fss:isTrue>
    </fss:isCategory>
  />
</fss:getSearchDetails>
```

In the second example the JSP tag is used to display a symbol in front of the results of a specific category:

```
<fss:iterateResults>
  <fss:isCategory cagegories="pdf">
    <fss:isTrue><img src="/icons/pdf.gif"></fss:isTrue>
  </fss:isCategory>
</fss:iterateResults>
```

5.6.2 Language check (<fss:isLocale>)

The `<fss:isCategory>` tag can be used within the `<fss:getSearchDetails>` tag (see Chapter 5.5.1, page 70) and the `<fss:iterateResults>` tag (see Chapter 5.5.2, page 72).

When used within `<fss:getSearchDetails>`, `<fss:isLocale>` provides information about the search parameters used and within `<fss:iterateResults>` about the search result.

Within `<fss:isLocale>`, `<fss:isTrue>` and `<fss:isFalse>` are to be used. The content of `<fss:isTrue>` is output precisely when one of the given language abbreviations ("locales" attribute) corresponds to a language abbreviation of the document (intersection). By contrast, the content of `<fss:isFalse>` is displayed precisely when no language abbreviation matches a language abbreviation of the document.

Attributes:

Attribute	Expected value	Mandatory parameter
locales	Comma separated list of language abbreviations for the comparison with the language abbreviations of the search result.	Yes

Example:

The first example demonstrates use of the JSP tag, to pre-assign the parameters of the current search to the search form:

```
<fss:getSearchDetails>
  <input type="text" name="query" value="<%= query %>" />
  | PDF <input type="checkbox" name="german"
    <fss:isLocale locales="de">
      <fss:isTrue>checked</fss:isTrue>
    </fss:isLocale>
  />
</fss:getSearchDetails>
```

In the second example the JSP tag is used to display a symbol in front of the results of a specific language:

```
<fss:iterateResults>
  <fss:isLocale locales="en">
    <fss:isTrue><img src="/icons/en.gif"></fss:isTrue>
  </fss:isLocale>
</fss:iterateResults>
```

5.7 Highlighting

Search terms are usually given for a search. If the search term is found within a document, the document and the place in which the term is found are returned as the result. As a default the place in which the term is found is not highlighted. The place in which the term is found can be highlighted with the `<fss:highlighter>` tag.

5.7.1 Highlight places in which search term found (<fss:highlighter>)

The `<fss:hightlighter>` tag is used to highlight the places within a text in which the search term is found. In addition, the output can be limited to a maximum number of characters. A list of words to be highlighted in the documents can also be specified instead of the search term.

<u>Attributes:</u>

Attribute	Expected value	Mandatory parameter
on	Specification of a (HTML) source text which is to be inserted in front of the place in which the search term is found. "<b>" is inserted as a default.	No
off	Specification of a (HTML) source text which is to be inserted behind the place in which the search term is found. "</b>" is inserted as a default.	No
maxSize	This attribute can be used to specify the maximum number of characters (including the place in which the search term is found) are to be output. An integer value is expected. It is useful to limit longer text passages (e.g. to 250 characters).	No
words	Comma separated list of words which are to be highlighted instead of the search text, e.g. `free,the`	No

Example: In the example the place in which the search term is found in the title is highlighted in bold and is highlighted with the background colour #DDDDDD (grey tone):

```
<fss:highlighter on="<b><span style='background-color:#DDDDDD'>"
off="</span></b>"><%= title %></fss:highlighter>
```

In the example the place in which the search term is found in also highlighted in the text section in bold and is highlighted with the background colour #DDDDDD (grey tone). In addition the number of characters output is limited to 250:

```
<fss:highlighter maxSize="250" on="<b><span style='background-
color:#DDDDDD'>" off="</span></b>"><%= content
%></fss:highlighter>
```

In the example the words "the" and "free" are highlighted in bold in the title and are highlighted with the background colour #DDDDDD (grey tone):

```
<fss:highlighter words="the,free" on="<b><span style='background-
color:#DDDDDD'>" off="</span></b>"><%= title %></fss:highlighter>.
```

5.8 Navigation

The results of a search are spread over virtual pages. FirstSpirit Search offers tags to generate a navigation bar for navigation between the individual, virtual pages.

A navigation bar consists of three areas:

- First virtual page or top navigation limit (fss:navTop)
- Sub-area or navigation section of virtual pages, starting from the current page (fss:navFrame)
- Last virtual page or bottom navigation limit (fss:navBottom)

Examples of navigation bars (the current page is highlighted with bold text in each case):

- Display of the first and last virtual page and the sub-area: [1] … [6] [7] **[8]** [9] [10] … [21]
- Display of the last, virtual page and the sub-area:
  (the first page is displayed in the sub-area):
  [1] **[2]** [3] [4] [5] … [21]
- Display of the first, virtual page and the sub-area
  (the last page is displayed in the sub-area):

[1] … [17] [18] [19] **[20]** [21]

- Display of the sub-area
  (the first and the last page are displayed in the sub-area):
  [1] [2] [3] [4] [5]

5.8.1 Navigation bar (<fss:navigation>)

The `<fss:navigation>` tag generates a navigation bar. Within the `<fss:navigation>` tag, the `<fss:navTop>`, `<fss:navFrame>` and `<fss:navBottom>` tags can be used.

The "frameSize" parameter can be used to define the maximum number of virtual pages in the sub-area (`<fss:navFrame>`).

Attributes:

Attribute	Expected value	Mandatory parameter
frameSize	Number of virtual pages (integer) to be displayed in <fss:navFrame>	Yes

Example:

```
...
<fss:navigation frameSize="3">

  <fss:navTop>
    ...
  </fss:navTop>

  <fss:navFrame>
    ...
  </fss:navFrame>

  <fss:navBottom>
    ...
  </fss:navBottom>

</fss:navigation>
...
```

5.8.2 First / last virtual page (<fss:navTop>, <fss:navBottom>)

The two JSP tags `<fss:navTop>` and `<fss:navBottom>` are available within the `<fss:navigation>` tag.

The `<fss:isTrue>` and `<fss:isFalse>` tags can be used to check whether the first (`<fss:navTop>`) or last (`<fss:navBottom>`) virtual page is not displayed within the sub-area (`<fss:navFrame>`). The content of `<fss:isTrue>` is displayed if the first or last virtual page is not displayed in the sub-area. In this way, for example, double output can be prevented.

```
...
<fss:navTop>
  <fss:isTrue>
    ...
  </fss:isTrue>
  <fss:isFalse>
    ...
  </fss:isFalse>
</fss:navTop>
...
```

The variables "urlPageNo" and "url" are available within the two tags. "urlPageNo" can be used to output the number and "url" the URL of the virtual target page. In addition, the number of virtual pages can be output with "urlPageSize".

Variables:

Variable	**Meaning**	**Return data type**
urlPageNo	Number of the virtual target page.	Integer
url	URL of the virtual target page (including parameter).	String
urlPageSize	Number of virtual pages.	Integer

Example:

```
...
<fss:navigation frameSize="3">

  <fss:navTop>
    <fss:isTrue>
      <a href="<%= url %>">[<%= urlPageNo %>]</a> ...
    </fss:isTrue>
  </fss:navTop>

  <fss:navFrame>
    <fss:isTrue>
      <a href="<%= url %>">[<%= urlPageNo %>]</a>
    </fss:isTrue>
    <fss:isFalse>
      <b>[<%= urlPageNo %>]</b>
    </fss:isFalse>
  </fss:navFrame>

  <fss:navBottom>
    <fss:isTrue>
      ... <a href="<%= url %>">[<%= urlPageNo %>]</a>
    </fss:isTrue>
  </fss:navBottom>

</fss:navigation>

...
```

5.8.3 Sub-area of all virtual pages (<fss:navFrame>)

The `<fss:navFrame>` tag, which is available within the `<fss:navigation>` tag, is used to display all virtual pages in the sub-area.

The number of pages to be displayed is specified by the attribute "frameSize".

The `<fss:isTrue>` tag can be used to check whether the page to be displayed in the sub-area does not correspond to the current page. The content of `<fss:isTrue>` is displayed if the page to be displayed by the sub-area is not the current page.

```
...
<fss:navFrame>
  <fss:isTrue>
    ...
  </fss:isTrue>
  <fss:isFalse>
    ...
  </fss:isFalse>
</fss:navFrame>
...
```

The two variables "navPageNo" and "navUrl" are available within the tag. "navPageNo" can be used to output the number and "navUrl" to output the URL of the virtual page to be displayed by the sub-area. In addition, the number of virtual pages can be output with "navPageSize".

Variables:

Variable	Meaning	Return data type
urlPageNo	Number of the virtual target page	Integer
url	URL of the virtual target page (including parameter)	String
urlPageSize	Number of virtual pages	Integer

Example:

```
...
<fss:navigation frameSize="3">

  <fss:navTop>
    <fss:isTrue>
      <a href="<%= url %>">[<%= urlPageNo %>]</a> ...
    </fss:isTrue>
  </fss:navTop>

  <fss:navFrame>
    <fss:isTrue>
      <a href="<%= url %>">[<%= urlPageNo %>]</a>
    </fss:isTrue>
    <fss:isFalse>
      <b>[<%= urlPageNo %>]</b>
    </fss:isFalse>
  </fss:navFrame>

  <fss:navBottom>
    <fss:isTrue>
      ... <a href="${navUrl}">[${navPageNo}]</a>
    </fss:isTrue>
  </fss:navBottom>

</fss:navigation>
...
```

5.8.4 Links to specific pages (<fss:url>)

`<fss:url>` can be used to generate a link to a specific virtual page. The selection can be made relative to the current page or absolutely to the first or last page. In addition, the link can be used to change the number of results to be displayed on a page ("pageSize"). The `<fss:isTrue>` tag can be used to check whether the page given is a valid page or not. If this is the case the content of `<fss:isTrue>` is displayed. If not, the content of `<fss:isFalse>` is displayed, provided the tag was given.

Parameters:

Attribute	Expected value	Mandatory parameter
jump	Relative jump to a virtual page. Possible values: ▪ `fp`: first page ▪ `pp`: previous page ▪ `np`: next page ▪ `lp`: last page	No
pageNo	Absolute value of the virtual target page. An integer value is expected	No
pageSize	Changes the value of the number of results displayed per virtual page.	No

Variables:

Variable	Meaning	Return data type
urlPageNo	Number of the virtual target page.	Integer
url	URL of the virtual target page (including parameter).	String
urlPageSize	Number of virtual pages.	Integer

Example:

The following example shows a simple navigation bar with four links to the first, previous, next and last virtual page:

```
<fss:url jump="fp"><a href="<%= url %>">[|<]</a></fss:url>
<fss:url jump="pp"><a href="<%= url %>">[<-]</a></fss:url>
<fss:url jump="np"><a href="<%= url %>">[->]</a></fss:url>
<fss:url jump="lp"><a href="<%= url %>">[>|]</a></fss:url>
```

The next example demonstrates the change in the number of results to be displayed per virtual page:

```
<fss:url>
  <form action="<%= url %>" method="GET">
    <select name="pageSize">
      <option value="10">10 entries/page</option>
      <option value="25">25 entries/page</option>
      <option value="50">50 entries/page</option>
    </select>
    ...
  </form>
</fss:url>
```

```
<fss:url jump="2000">
  <fss:isTrue>
    <a href="<%= url %>">2000</a>
  </fss:isTrue>
  <fss:isFalse>
    Link is not valid
  </fss:isFalse>
</fss:url>
```

6 Search queries (search servlet)

The search servlet can be used to perform a search query. The result of such a search query is saved in the user session and can be output with the help of the `<fss:iterateResults>` tag (see Chapter 5.5.2, page 72).

The search servlet is called (invoked) with:

```
"<%= request.getContextPath() %>/do.search"
```

The <form> tag is used in an HTML form:

```
<form method="post"
      action="<%= request.getContextPath() %>/do.search">
  ...
</form>
```

!

**post" is to be used for the HTTP transfer method as the length of the parameter names and values can be very long!*

The search servlet has the request parameter "pageSize" to spread results over different, virtual pages. The value expected by the parameter is the maximum number of results to be displayed on a virtual page. If the parameter is not given the default value 10 (results per virtual page) is used.

```
<input type="hidden" name="pageSize" value="1" />
```

The following tags are available for display of a navigation bar (for navigating through the individual virtual pages):

▪ <fss:navigation>	(see Chapter 5.8.1, page 79)
▪ <fss:navTop>	(see Chapter 5.8.2, page 80)
▪ <fss:navFrame>	(see Chapter 5.8.3, page 81)
▪ <fss:navBottom>	(see Chapter 5.8.2, page 80)
▪ <fss:url>	(see Chapter 5.8.4, page 83)

The mandatory request parameter "query" can be used to transfer one or several search terms. As a default, two search terms use "OR" logic. The form of the search term operation can be changed by the request parameter "op". The values "and" ("and" operation), "or" ("or" operation) and "not" (negation of the search term) are available for this request parameter.

Example:

```
<input type="hidden" name="query" value="city" />
<input type="hidden" name="op" value="and" />
<input type="hidden" name="query" value="tree" />
<input type="hidden" name="op" value="not" />
<input type="hidden" name="query" value="leaf" />
<input type="hidden" name="query" value="sun" />
```

Here the search is for "city" and "tree" and not "leaf" or "sun".

From FirstSpirit Version 4.2 Release 2 the request parameter “wildcardSearch” can be used to search for the beginnings of words only (right-hand side truncation) or for character strings (right and left-hand side truncation), whereby `*` is used as the truncation symbol. The following values are available for this parameter:

- “contains”: A wildcard search is performed, whereby the search text entered by the user is automatically supplemented with an asterisk operator (`*`), namely at the start **and** at the end (right and left-hand side truncation). I.e., a search for *list* finds *List*, *Listing* and *Contentlist*.
  Example: `wildcardSearch="contains"`
- “startsWith”: A search is performed for the start of a word, whereby the search text entered by the user is automatically supplemented with an asterisk operator (`*`), namely only at the end (right-hand side truncation). I.e., a search for *list* finds *List* and *Listing*, but not *Contentlist*.
  Example: `wildcardSearch="startsWith"`
- “none”: The search looks for precisely the same search term. I.e., a search for *list* finds only *List*, but not *Listing* or *Contentlist*. This is the default response, if the parameter is not set.
  Example: `wildcardSearch="none"`

*If the user uses inverted commas to search for a phrase (e.g. "List") or enters the wildcard * themselves (e.g. *List), automatic truncation is not performed.*

If using the parameter, you should take into account the fact that a search with `wildcardSearch="contains"` is more complicated and is therefore slower than a search with `wildcardSearch="startsWith"`.

The parameter can be used in the “web.xml” file as a configuration parameter and as a request parameter in an HTML form. If the parameter is defined both in the “web.xml” file and as a request parameter, the definition in the “web.xml” is used with priority.

Apart from specifying the search terms and their logical operation, the initial number of results to be sought can be limited using the "initialSize" request parameter. If this parameter is not specified, 10 results are initially requested.

The maximum number of search results can be limited using the "maxResults" request parameter. If more search results are determined than the specified maximum number, the first search results only are returned, until the maximum number is reached.

In addition, the "orderBy" request parameter can be used to specify which field is to be used for the sort. The "ascending" request parameter defines whether the sort is to be in ascending or descending order.

If, following a successful search, a different results page is to be displayed, the "resultsURL" request parameter can be used. If no results were found the "noResultsURL" request parameter can also be used to control a separate page.

The search can also be limited to a specific language abbreviation (via the "locale" request parameter) or certain categories (via the "categories" request parameter). If the "categories" request parameter is specified several times the specified categories use "or" logic. If the request parameter “categories” is set, but no value is given (`categories=`), **up to and including FirstSpirit Version 4.2** a search is performed for an empty category (`""`), **from Version 4.2R2** such a parameter is ignored in the search.

In the file **web.xml** the input fields "Server-Name" and "Engine-Name" in the server and project configuration are used to specify the default search server and the search engine to be used (see Chapter 3.3, page 12):

```
<servlet>
  <servlet-name>fss-Search</servlet-name>
  <servlet-
class>de.espirit.firstspirit.opt.search.web.SearchServlet</servlet
-class>
  <init-param>
    <param-name>serverURL</param-name>
    <param-value>fssServer</param-value>
  </init-param>
  <init-param>
    <param-name>searchEngine</param-name>
    <param-value>fssEngine</param-value>
  </init-param>
</servlet>
```

Here the search server ("serverURL") configured is "fssServer" and the search engine ("searchEngine") configured is "fssEngine".

The search engine can also be specified using the request parameter "se" for a search query. In this case the parameter "searchEngine" must be removed from the "web.xml" file.

For security reasons the values from the "web.xml" file cannot be overwritten by request parameters!

In addition, the number of possible search queries (per HTTP session), can be limited using the "singleton" request parameter.

Example:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="fs-search" prefix="fss" %>
...
<fss:getSearchDetails>
  <form method="post"
        action="<%= request.getContextPath() %>/do.search">
    Search text:
    <input type="text"
           name="query"
           value="<%= (query != null ? query : "word") %>"/>
    <input type="hidden" name="initialSize" value="10" />
    <input type="hidden" name="pageSize" value="5" />
    <input type="hidden" name="maxResults" value="500" />
    <input type="hidden" name="singleton" value="true" />
    <input type="hidden"
```

```
            name="resultsURL"
            value="$CMS_REF(pageref:"homepage", abs:1)$" />
     <input type="submit" value="Suchen" />
   </form>
</fss:getSearchDetails>
...
```

In the file “web.xml” the parameter “wildcardSearch” can be used **from FirstSpirit Version 4.2 Release 2**. If this parameter is defined both in the file “web.xml” and as request parameter, the definition in the file “web.xml” will be definition is used with priority.

A list of the available request parameters is given in the following table:

Configuration parameters ("web.xml"):

Parameter	**Expected value**
formEncoding	This parameter can be used to specify the encoding with which the form data is to be processed (e.g. "UTF-8").
searchEngine	Specification of the search engine. If the parameter "searchEngine" is removed from the file "web.xml" the request parameter "se" must be used.

sessionSearchProperties	Comma separated list of attributes which is automatically transferred to the search from the HTTP session. As a result, security-relevant information, e.g. the roles or groups of the current user, can be transferred too, although runtime parameters cannot influence the values.
	
	Example:
	`<init-param>`
	`  <param-name>sessionSearchProperties </param-name>`
	`  <param-value>FIRSTpersonalisation.usergroups </param-value>`
	`</init-param>`
	
	Here the session attribute: "FIRSTpersonalisation.usergroups", which is made available by FirstSpirit Personalisation, is transferred to the search.

serverURL	Specification of the URL of the search server, which is to be used for the search (see Chapter 4.5.1.3, page 24, parameter "engineURL"). This parameter is a mandatory parameter.
wildcardSearch (from FirstSpirit Version 4.2 Release 2)	This parameter can be used to search for the beginnings of words only (right-hand side truncation) or for character strings (right and left-hand side truncation). Use “contains” to perform a wildcard search, whereby the search text entered by the user is automatically supplemented with an asterisk operator (*), namely at the start and at the end (right and left-hand side truncation). Use “startsWith” to perform a search for the start of a word, whereby the search text entered by the user is automatically supplemented with an asterisk operator (*), namely only at the end (right-hand side truncation). Use “none” to search for precisely the entered search term. This is the default response, if the parameter is not set.

Request parameters:

Parameter	**Expected value**
ascending	Sort results in alphabetical ascending (value: "true") or descending (value: "false") order; default value: "false".
categories	Specification of one or several categories (= multiple specification of the request parameter), to which the search is to be limited. If several request parameters are specified the values use "OR" logic.
errorURL	Specification of a URL which is to be displayed if an error occurs during the search. If no value is given the value of the "noResultsURL" parameter is used.
field	If this parameter is specified the search relates to the specified field in the results (e.g. "title").
initialSize	Number of results requested at the beginning. The search engine waits either until a corresponding number of results are found or the search is quit.
locale	Specification of a language abbreviation (language), to which the search is to be limited.
maxResults	Maximum number of results to be found. If more results are found than specified the search is cancelled and displays the first results only until the maximum number is reached.
noResultsURL	This URL is called (invoked) if the search was unable to find any results. If no value is specified for the "noResultsURL" parameter the value of the "resultsURL" parameter is used.
op	This parameter is used to specify which logic is to be used for the search terms (see "query" parameter). The "op" parameter can be specified several times. The parameters are taken into account in the order specified for the logic. The available values are: "and" ("und" operation), "or" ("or" operation) and "not" (negation); default value: "or".
orderBy	Specification of the field to be used for the sort: "author", "date", "score", "size" or "title"; default value: "score".
pageSize	Number of elements to be displayed per page; default value: „10“.

Parameter	Expected value
query	Search word or text. If the parameter is specified several times the logic from the "op" parameter is used for the search terms.
regexPattern	This request parameter can be used to transfer one or several expressions (=multiple use of the request parameter) to the search in Java syntax. The specified expressions are attached to the search as "or" logic filters.
resultsURL	This URL is called (invoked) if the search was error free and the results were determined. If the parameter is not specified the page to be opened is opened again.
se	Synonym for the "searchEngine" parameter. The request parameter can only be used if the "searchEngine" parameter is removed from the "web.xml" file.
searchProperties	Comma separated list of request parameter names which is automatically transferred to the search.
singleton	Restriction to one simultaneous search query in an HTTP session (value: "true"). An unlimited number of search queries can be achieved with the value "false"; default value: "false".
wildcardSearch (from FirstSpirit Version 4.2 Release 2)	see documentation about “wildcardSearch” as configuration parameter (“web.xml”); if the parameter is used both in the file web.xml and as request parameter, the definition in the file web.xml will be used with priority.

In order to prevent Open-Redirect attacks, external redirects are prohibited in projects which use the FirstSpirit Personalisation module ***from FirstSpirit Version 4.2R4****. I.e. only relative and absolute URLs are possible, e.g.*
`start.jsp`
`../area/index.html`
`../area/search.jsp`)
Exception: Redirects to the same host or a host in the same domain are allowed. A forwarding (redirect) page must be created for redirecting to an external URL.

7 Examples

7.1 Categories and language simulation (URL)

In the example, three services are defined:

- a logging service
- a server service and
- a re-indexing service.

In the re-indexing service the index is updated daily at 06:00 hrs.

In the server service a Spider engine is defined which generates an index with two languages and two presentation channels on the basis of a project's generation directory.

The languages ("de" and "en") are simulated via a multilingualism filter and the categories ("web" and "print" are simulated via a categories filter.

```
<?xml version="1.0" encoding="UTF-8"?>
<service
class="de.espirit.firstspirit.opt.search.service.proxy.MultiServic
eProxy">
  <attribute name="services" type="list">

    <service
class="de.espirit.firstspirit.opt.search.service.adapter.Log4jServ
ice">
      <attribute name="log4j.rootCategory">DEBUG, file</attribute>
      <attribute
name="log4j.appender.file">org.apache.log4j.RollingFileAppender</a
ttribute>
      <attribute
name="log4j.appender.file.File">E:/FirstSpirit/log/fs-
search.log</attribute>
      <attribute
name="log4j.appender.file.MaxFileSize">5MB</attribute>
      <attribute
name="log4j.appender.file.MaxBackupIndex">5</attribute>
      <attribute
name="log4j.appender.file.layout">org.apache.log4j.PatternLayout</
attribute>
      <attribute
name="log4j.appender.file.layout.ConversionPattern">%-5p %d (%c)
%m%n</attribute>
    </service>

    <service
class="de.espirit.firstspirit.opt.search.service.adapter.ServerSer
```

```
vice">
      <server
class="de.espirit.firstspirit.opt.search.server.SimpleServer">
        <engine name="fssEngine"
class="de.espirit.firstspirit.opt.search.engine.proxy.MonitorEngin
eProxy">

          <engine
class="de.espirit.firstspirit.opt.search.engine.proxy.FilterCatego
rizeEngineProxy">

            <engine
class="de.espirit.firstspirit.opt.search.engine.proxy.FilterLocali
zeEngineProxy">

              <engine
class="de.espirit.firstspirit.opt.search.engine.spider.SpiderEngin
e">

                <attribute name="urls" type="list">

<attribute>http://localhost/fs4staging_12345/23456/de/dokumentatio
n/documentation.html</attribute>

<attribute>http://localhost/fs4staging_12345/23456/de_1/dokumentat
ion/documentation.html</attribute>

<attribute>http://localhost/fs4staging_12345/23456/en/dokumentatio
n/documentation.html</attribute>

<attribute>http://localhost/fs4staging_12345/23456/en_1/dokumentat
ion/documentation.html</attribute>
                </attribute>
                <attribute
name="index">E:/FirstSpirit/web/fs4staging_12345/WEB-
INF/lucene.index</attribute>
                <attribute name="maxThreads">1</attribute>
                <attribute name="threadPriority">1</attribute>
                <attribute
name="maxFieldLength">200000</attribute>
                <attribute name="maxContentLength">100</attribute>
                <attribute name="allowed" type="list">
                  <attribute
class="de.espirit.firstspirit.opt.search.engine.spider.link.RegexW
ebLinkFilter">/de/</attribute>
                  <attribute
class="de.espirit.firstspirit.opt.search.engine.spider.link.RegexW
ebLinkFilter">/de_1/</attribute>
                  <attribute
class="de.espirit.firstspirit.opt.search.engine.spider.link.RegexW
ebLinkFilter">/en/</attribute>
                  <attribute
class="de.espirit.firstspirit.opt.search.engine.spider.link.RegexW
ebLinkFilter">/en_1/</attribute>
                </attribute>
                <attribute name="maxTime">120m</attribute>

              </engine>
              <attribute name="locales" type="map">
                <filter name="de"
class="de.espirit.firstspirit.opt.search.filter.RegexFilter">
                  <attribute name="property">url</attribute>
                  <attribute
name="pattern">(/de/)|(/de_1/)</attribute>
                </filter>
                <filter name="en"
class="de.espirit.firstspirit.opt.search.filter.RegexFilter">
                  <attribute name="property">url</attribute>
                  <attribute
name="pattern">(/en/)|(/en_1/)</attribute>
                </filter>
              </attribute>

            </engine>

            <attribute name="categories" type="map">
              <filter name="web"
class="de.espirit.firstspirit.opt.search.filter.RegexFilter">
                <attribute name="property">url</attribute>
                <attribute
name="pattern">(/de/)|(/en/)</attribute>
              </filter>
              <filter name="print"
class="de.espirit.firstspirit.opt.search.filter.RegexFilter">
                <attribute name="property">url</attribute>
                <attribute
name="pattern">(/de_1/)|(/en_1/)</attribute>
              </filter>
            </attribute>

          </engine>

        </engine>
      </server>
      <attribute name="createRMI">false</attribute>
      <attribute name="bindTo">local</attribute>
      <attribute name="localName">fssServer</attribute>
    </service>

    <service
class="de.espirit.firstspirit.opt.search.service.RebuildIndexTimer
Service">
      <attribute name="engineURL">fssServer[fssEngine]</attribute>
      <attribute name="startTime">6:00</attribute>
      <attribute name="period">24h</attribute>
      <attribute name="startNow">false</attribute>
    </service>

  </attribute>
</service>
```

8 Appendix

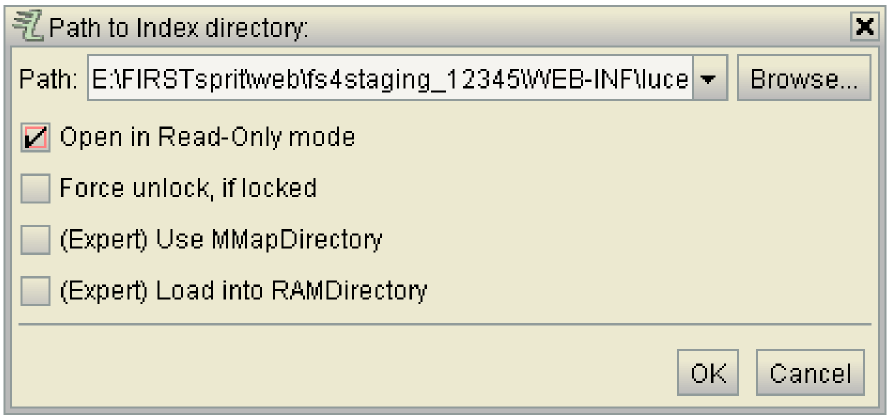
8.1 Debugging

The indexed fields and documents in the Lucene index can be viewed using the Luke tool - Lucene Index Toolbox[10]. The tool can be downloaded from the given address (see footnote) and either started via the command line with the command:

```
java -jar lukeall-0.8.1.jar
```

or via the JNLP link to the website.

After opening the index, e.g. E:\FIRSTsprit\web\fs4staging_12345\WEB-INF\lucene.index\fig4je34 (see Figure 8-1), the indexed fields and documents can be viewed and browsed through (see Figure 8-2 and Figure 8-3) (cf. Chapter 4.5.4.3, page 46).



Path to Index directory:

Path: E:\FIRSTsprit\web\fs4staging_12345\WEB-INF\luce

[✓] Open in Read-Only mode

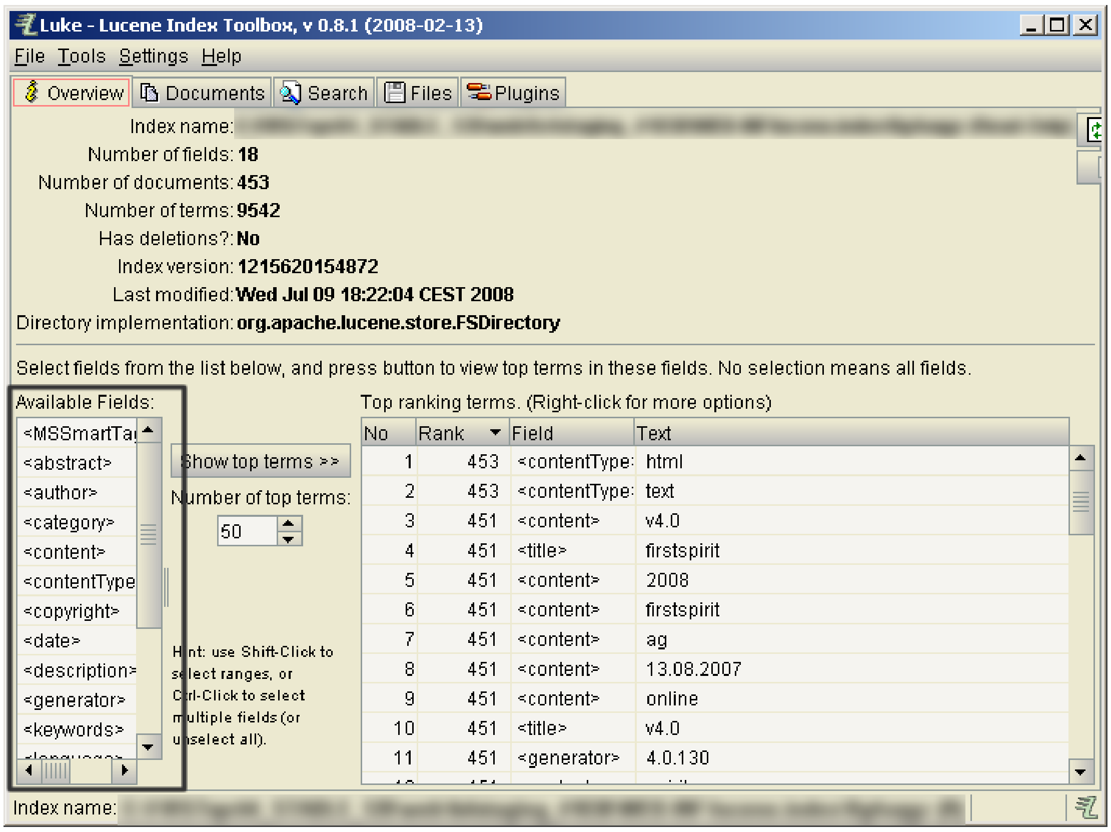
[ ] Force unlock, if locked

[ ] (Expert) Use MMapDirectory

[ ] (Expert) Load into RAMDirectory

Figure 8-1:Opening the Lucene index (Luke)

[10] http://www.getopt.org/luke/



Luke - Lucene Index Toolbox, v 0.8.1 (2008-02-13)

File Tools Settings Help

Overview | Documents | Search | Files | Plugins

Index name: [redacted]
Number of fields: **18**
Number of documents: **453**
Number of terms: **9542**
Has deletions?: **No**
Index version: **1215620154872**
Last modified: **Wed Jul 09 18:22:04 CEST 2008**
Directory implementation: **org.apache.lucene.store.FSDirectory**

Select fields from the list below, and press button to view top terms in these fields. No selection means all fields.

Available Fields: <MSSmartTa, <abstract>, <author>, <category>, <content>, <contentType, <copyright>, <date>, <description>, <generator>, <keywords>, <language>

Show top terms >>

Number of top terms: 50

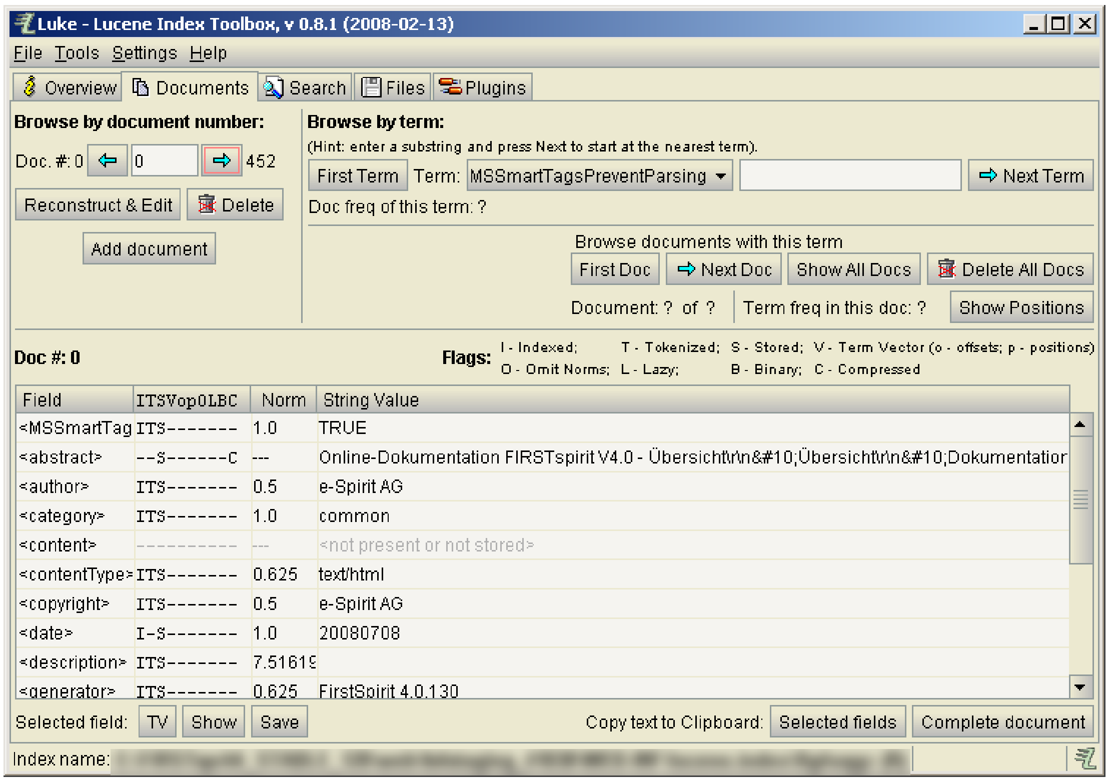
Hint: use Shift-Click to select ranges, or Ctrl-Click to select multiple fields (or unselect all).

Top ranking terms. (Right-click for more options)

No	Rank	Field	Text
1	453	<contentType>	html
2	453	<contentType>	text
3	451	<content>	v4.0
4	451	<title>	firstspirit
5	451	<content>	2008
6	451	<content>	firstspirit
7	451	<content>	ag
8	451	<content>	13.08.2007
9	451	<content>	online
10	451	<title>	v4.0
11	451	<generator>	4.0.130

Index name: [redacted]

Figure 8-2: View of the indexed fields (Luke)



Luke - Lucene Index Toolbox, v 0.8.1 (2008-02-13)

File Tools Settings Help

Overview | Documents | Search | Files | Plugins

Browse by document number:

Doc. #: 0 | 0 | 452

Reconstruct & Edit | Delete

Add document

Browse by term:

(Hint: enter a substring and press Next to start at the nearest term).

First Term | Term: MSSmartTagsPreventParsing | Next Term

Doc freq of this term: ?

Browse documents with this term

First Doc | Next Doc | Show All Docs | Delete All Docs

Document: ? of ? | Term freq in this doc: ? | Show Positions

Doc #: 0

Flags: I - Indexed; T - Tokenized; S - Stored; V - Term Vector (o - offsets; p - positions)
O - Omit Norms; L - Lazy; B - Binary; C - Compressed

Field	ITSVopOLBC	Norm	String Value
<MSSmartTag	ITS-------	1.0	TRUE
<abstract>	--S------C	---	Online-Dokumentation FIRSTspirit V4.0 - Übersicht\r\n&#10;Übersicht\r\n&#10;Dokumentatior
<author>	ITS-------	0.5	e-Spirit AG
<category>	ITS-------	1.0	common
<content>	----------	---	<not present or not stored>
<contentType>	ITS-------	0.625	text/html
<copyright>	ITS-------	0.5	e-Spirit AG
<date>	I-S-------	1.0	20080708
<description>	ITS-------	7.51619	
<generator>	ITS-------	0.625	FirstSpirit 4.0.130

Selected field: TV | Show | Save

Copy text to Clipboard: Selected fields | Complete document

Index name: [redacted]

Figure 8-3: View of the indexed documents (Luke)

9 Legal notices

The module "FirstSpirit™ Search" is a product of the e-Spirit AG, Dortmund, Germany.

When using this module only the licence agreed between the e-Spirit AG and the user is valid.

You can find information about third-party software which is potentially used fort he module but not produced by the e-Spirit AG, their own licences and – as the case may be – information about updates on the start page of each FirstSpirit server in the area "Legal notices".