integrace
e-business solutions

Solr Site Search

Technical Implementation Guidelines

Introduction

Solr is an open source enterprise search platform. Solr is highly scalable and offers very powerful search features that go beyond the features of another popular search platform, Google in its various variants.

The main reason for this is the different approach of both platforms, the main difference being that Google spiders websites by following links, while Solr is feed-based. This has far-reaching consequences. An example of these consequences is the fact that removing an item from the Google search index can be a time-consuming task without immediate success, while updating the Solr index is done instantaneously. Another example is that searching structured collections of data, as opposed to Google, is very easily done with Solr.

However, one main drawback of the Solr platform is the knowledge and expertise that is needed to install, run and feed the Solr search engine. Google is offered as a service, through the cloud and also in a boxed version (Google Mini).

Solr Site Search, developed and offered by Integrace BV, removes this drawback. Solr Site Search is, like Google Site Search, a SaaS solution. Implementation is as easy as implementing Google Site Search. The solution is hosted by Integrace.

As said, Solr does not offer a spider that indexes full sites. To get data into Solr, one has to upload the data oneself. The Solr Site Search solution offers a straightforward interface to do this. This interface is described in this document.

Querying the search index and presenting the search results is done in roughly the same way as found in Google. Results are returned in XML-documents.

A future version of Solr Site Search, currently under development, will offer a spider that crawls URLs and also the capability to present formatted search results that are available through Javascript. This future version will make Solr a suitable solution for even the smallest of sites, that now use the basic free version of Google Site Search. The current version of Solr Site Search is primarily aimed at sites that are looking for a more powerful alternative for the (paid) business edition of Google Site Search.

This document describes the standard version of Solr Site Search. The interface implements parts of the facilities offered by Solr. However, in its current form Solr Site Search already covers the majority of search requirements of complex environments and of large institutions, while hiding all the complexities of this very powerful search environment. In future versions the interface will be expanded on the basis of input of our clients. For very client-specific requirements, there is always the possibility to develop a client-specific version of the interface or Solr itself.

Post data to Solr

Steps to get data into Solr:

1. Upload one or more XML documents to the server using a HTTP Post request.
2. When all the XML document are successfully on the server, call the 'process' action, all the documents will be processed.

Upload documents

Post one or more XML documents to the webserver using an HTTP Post request. There are two parameters, both are required:

- Filename (choose your unique filename (allowed characters: a-z, 0-9, dot and underscore)).
- Data (The XML document)

The XML document:

- Root node: Source
- One or more 'Object' elements
    - Attribute @id (required). The ID of this object, for example the ID of this object in your own database.
    - Attribute @type (required). The type of this object, you may use your own types, for example the types of your own database. The @type and @id together are the key of this object.
    - Attribute @source (required). The source of the data, this is for your own use, for example the name of your back office system,
    - Attribute @library (required). This value will be provided by Integrace, this is the name of the library where all your data will be stored. This library will always be the same.
    - Element Title (required). The title of the object, this data is searchable and will be returned in the search results. For example: the headline of the news item.
    - Element Description (required). The description of the object, this data is searchable and will be returned in the search results. For example: the first part of the news item.
    - Element Body. The body text of the object, this data is searchable but won't be returned in the search results. For example, the complete body text of the news item and all the relevant data that have to be searched.
    - Element Date (required). The data of this object, for example the date of publication. This date will be returned in the search results.

- Element Thumbnail. The thumbnail of this object, for example the image of the news item, or a screenshot of the page. This thumbail (the value, not the image selves) will be returned in the search results.
- Element Keywords. The keywords (one or more, or leave it blank when no keywords are available) of the object. Separate the keywords using a comma.
- Element Url. The URL where the object can be found on the internet (or extranet/intranet or something like that). This may be an absolute or a relative URL. The value will be returned in the search results.

If the XML document contains an object that already exists in Solr, the data will be updated. If not, a new object will be created in Solr. When the data is processed, it will be available in the search index instantly.

It's also possible to remove data from the Solr index, the attribute @action has to be added to the Object-element with the value 'delete'. All of the elements (also the required elements) don't have to be added. The attributes have to be added to identify the object in Solr.

It's possible to use all kind of objects in one XML document, for example to add a news item, to update a video item and to delete another news item. An example of the XML document is shown below:

```
<?xml version="1.0" encoding="utf-8"?>
<Source>
	<Object id="157452" type="news" source="SiteManager" library="yourlibrary">
		<Title>The title of this news object</Title>
		<Description>The description of this news object.</Description>
		<Body>
			<![CDATA[ The complete body of this news object ]]>
		</Body>
		<Date>2011-01-01T08:00:00</Date>
		<Thumbnail>NewsImage.jpg</Thumbnail>
		<Keywords>tag one, tag two, tag three, tag etc</Keywords>
		<Url>/an_absolute_or_relative_url.html</Url>
	</Object>
	<Object id="69806" type="video" source="SiteManager" library="yourlibrary">
		<Title>The title of this video object</Title>
		<Description>The description of this video object</Description>
		<Body><![CDATA[ The complete body of this video object ]]></Body>
		<Date>2011-01-01T21:00:00</Date>
		<Thumbnail>Video.gif</Thumbnail>
		<Keywords></Keywords>
		<Url>http://yourdomain.com/an_absolute_or_relative_url.html</Url>
	</Object>
	<Object id="157458" type="news" source="SiteManager" library="yourlibrary" action="delete" />
</Source>
```

Process documents

When all the XML documents are successfully uploaded to the server, another HTTP request has to be called to process al the uploaded XML documents. All the data and objects will be analyzed and the Solr index will be updated. When this is finished, the data will be available (or won't be available when an object has been removed) instantly.

Search with Solr

You can search with Solr on two different ways:

1. Using the REST interface
2. Using the WCF service

The REST interface

An HTTP request (GET or POST) with parameters, for example: http://solr/?library=yourlibrary&q=test%20string&sort=publication%20desc&refinement=news&page=2 (this returns the second page of the results with news objects that matches 'test string', ordered by publication date descending).

The complete list of parameters:

Name	Parameter	Description
Library	library (required)	The name of the library
Search string	q (required)	The search query
Refinement	refinement (optional)	One or more refinements, separated by a comma. A refinement is the type of an object. For example, to search only news items, use "news". To search only news and video items, use "news, video". If you don't add a refinement, you're searching the complete index.
Sort	sort (optional)	The results are default sorted by 'score' (determined by the Solr algoritm) . You can sort the results by: • score asc (or score desc) • publication asc (or publication desc) • objecttype asc (or objecttype desc) • source asc (or source desc) You also can also use them in a combination
Numer of rows per page	rows (optional)	The number of rows per page in the search results. A number between 1 and 100. The default value is 20 rows per page.
Page	page (optional)	The page of the with search results. For example, there are 100 results and 20 results per page. On page=2 the results 21 to 40 will be returned.
Published	Published (optional)	If the value is 'true' (default), only the objects with a publication date in the past will be returned. To return all the results, set the value to 'false'.
Start and end date	start (optional) end (only required when start is used)	It's possible to add a start and end date, only the objects between those dates will be returned.

WCF Service

Instead of the REST service it is also possible to use a webservice with SOAP. A sample of the implementation using C#:

```
var query = new Solr.SearchQuery()
{
	Library = "yourlibrary",
	SearchString = "test string",
	ObjectRefinement = "news",
	Sort = "publication desc",
	MaxRows = 20,
	OnlyPublishedItems = true
};

// If necessary, you can use a start and end time
query.StartDate = DateTime.Now.AddMonths(-2);
query.EndDate = DateTime.Now.AddMonths(-1);

// 2 in this command means page 2 of the results
var document = Solr.Search(query, 2);
```

XML document with search results

The REST and the SOAP webservice both return the same output XML document:

```
<Solr version="2.2">
<Query>
	<Page>2</Page>
	<Library>yourlibrary</Library>
	<Searchstring>test string</Searchstring>
	<Refinement>news</Refinement>
	<Maxrows>3</Maxrows>
	<Sort>publication desc</Sort>
</Query>
<Result status="0" qtime="359" numfound="16621" start="3" end="6" page="2" lastpage="5541" maxscore="2.6089778">
	<Item id="157452" objecttype="news" score="1.5803182">
		<str name="source">SiteManager</str>
		<str name="title">Title of object 1</str>
		<str name="description">Description of object 1</str>
		<date name="publication">2011-01-01T06:42:52Z</date>
		<str name="url">/url/news/1</str>
		<arr name="keyword">
			<str>keyword 1</str>
			<str>keyword 2</str>
			<str>keyword 3</str>
			<str>keyword 4</str>
		</arr>
		<str name="thumbnail">thumbnail1.jpg</str>
	</Item>
	<Item id="157416" objecttype="news" score="1.3692553">
		<str name="source">SiteManager</str>
		<str name="title">Title of the object 2</str>
		<str name="description">Description of object 2</str>
		<date name="publication">2011-01-01T00:09:38Z</date>
		<str name="url">/url/news/2</str>
		<arr name="keyword">
			<str>keyword 1</str>
			<str>keyword 2</str>
```

```
                </arr>
                <str name="thumbnail">thumbnail2.jpg</str>
        </Item>
        <Item id="157446" objecttype="news" score="1.7141535">
                <str name="source">SiteManager</str>
                <str name="title">Title of object 3</str>
                <str name="description">Description of object 3</str>
                <date name="publication">2011-01-01T22:12:59Z</date>
                <str name="url">/url/news/3</str>
                <arr name="keyword">
                        <str>keyword 1</str>
                        <str>keyword 2</str>
                        <str>keyword 3</str>
                        <str>keyword 4</str>
                        <str>keyword 5</str>
                </arr>
                <str name="thumbnail">thumbnail3.jpg</str>
        </Item>
</Result>
</Solr>
```