SolPHP

Solr PHP support

solr-php-client

A 3rd party PHP library for indexing and searching documents within an Apache Solr installation.

Zip / Tarballs can be found at SolrPhpClient

- Adding, Deleting (by id and query), committing, optimizing and of course searching against a Solr instance
- Written for PHP 5 in Zend Framework / PEAR coding style
- PHPDoc generated API documentation included
- See link above for example usage and further documentation

Apache Solr PHP Extension

The Apache Solr PECL extension is a light-weight, feature-rich library that allows developers using Apache Solr via PHP to communicate easily and efficiently with the Solr web service using an object-oriented API.

The documentation for the PECL extension contains instructions on how to install the extension and is available in the PHP Manual under Search Engine Extensions.

There are 2 parallel releases of the extension:

- PECL Apache Solr Extension 1.x which supports Apache Solr Server 3.x
- PECL Apache Solr Extension 2.x which supports Apache Solr Server 4.0+

The php extension can be downloaded from the Apache Solr PECL project home page. Windows binaries can also be found on the extension's page.

A quick list of some of the features of the API include :

- Built in support for adding, deleting, optimizing, searching, rollback.
- Ability to connect to Solr servers behind SSL-enabled containers.
- Users can optionally provide PEM-formatted private keys or certificates to connect in HTTPS mode.
- Users can optionally provide CA certificates to authenticate hostname and issuer of SSL certificate.
- Developers can now update the values of the servlets (such as search, update) after the SolrClient instance has been created.
- Built in, Serializable query string builder objects which effectively simplifies the manipulation of name-value pair request parameters across repeated requests.
- The query builder API has methods to add/set, remove or retrieve name-value pair values for the following features in Solr : SimpleFacetParameters, StatsComponent, MoreLikeThis, HighlightingParameters, TermsComponent etc.
- Ability to reuse of HTTP connections across repeated requests (within the same thread in ZTS mode or same process in non-ZTS mode).
- Advanced HTTP client that provides built-in support for connecting to Solr servers secured behind HTTP Authentication or HTTP proxy servers.
- Ability to obtain SolrInputDocument objects from SolrDocument in query response for possible resubmission or updates.
- Automatic parsing of Solr response into native php objects whose properties can be accessed as array keys or object properties without any additional configuration on the client-side. This is simplified interface to access server response data. Solr Objects can be treated as arrays or objects.
- Also the SolrDocument retrieved from the query response implements the following interfaces which gives the developer several options on how to manipulate the response : ArrayAccess, Iterator, Traversable, Serializable.

The extension currently uses version 2.2 of the xml response format internally.

The contents of the XML response is transformed into native PHP types and the result is returned as a Solr Object instance.

You may also install it by running the following command in the console :

$ pecl install solr

Solarium

Solarium is a Solr client library for PHP applications that not only facilitates Solr communication but also tries to accurately model Solr concepts.

Solr's PHP response format

Solr has a PHP response format that outputs an array (as PHP code) which can be eval'd.

Example usage:

```
$code = file_get_contents('http://localhost:8983/solr/select?q=iPod&wt=php');
eval("\$result = " . $code . ";");
print_r($result);
```

Solr's PHP Serialized response format

Solr has a PHP response format that outputs a serialized array.

Example usage:

```
$serializedResult = file_get_contents('http://localhost:8983/solr/select?q=iPod&wt=phps');
$result = unserialize($serializedResult);
print_r($result);
```

In order to use either PHP or Serialized PHP Response Writers, you may first need to uncomment these two lines in your solrconfig.xml:

```
<queryResponseWriter name="php" class="org.apache.solr.request.PHPResponseWriter"/>
<queryResponseWriter name="phps" class="org.apache.solr.request.PHPSerializedResponseWriter"/>
```

You can also use the new response writer plugin for PHP here

https://issues.apache.org/jira/browse/SOLR-1967

```
<code>
<queryResponseWriter name="phpnative" class="org.apache.solr.request.PHPNativeResponseWriter">
<!-- You can choose a different class for your objects. Just make sure the class is available in the client -->
<str name="objectClassName">SolrObject</str>
<!--
0 means OBJECT_PROPERTIES_STORAGE_MODE_INDEPENDENT
1 means OBJECT_PROPERTIES_STORAGE_MODE_COMBINED

In independed mode, each property is a separate property
In combined mode, all the properites are merged into a _properties array.
The combined mode allows you to create custom __getters and you could also implement ArrayAccess, Iterator and
Traversable
-->
<int name="objectPropertiesStorageMode">0</int>
</queryResponseWriter

<code>
```

Also check out how to use it on the client side here

http://www.php.net/manual/en/solrclient.setresponsewriter.php

http://www.php.net/manual/en/solrclient.construct.php

CategoryQueryResponseWriter

Historical

Original Client Code Contributed By Brian Lucas: *
_*
There are two classes for PHP: SolrUpdate and SolrQuery. *_
_*
⚠ :TODO: ⚠ *_
_*

- clean up some of the XML writing code – it's a tad "kludgy" right now. *_
  _*

- abstract out more of the logic into configurable variables *_
  _*
- add back in the logging and debugging classes that clean up the "echo" calls*_