

SolrQuery

```
<?

    // Change this to your server

    define('SOLR_META_QUERY', '127.0.0.1:8080');

// VERSION SolrQuery 0.1
// Written by Brian Lucas, use any which way you see fit.
//
// DESCRIPTION
//
// This class runs an external fulltext search on Solr
// Example Usage:

/*
    $query = new SolrQuery;

    $query->language_id = $this->language_id;
    $query->lowerdate = $day_3neg;
    $query->upperdate = $day_3pos;
    $query->limit = 10;
    $query->group_id = 1;

    $results = $query->runQuery($titlekeywords);
    print_r($results);
*/

class SolrQuery {

    // All of these init variables are yours to define, these are just examples
    var $language_id = 0;
    var $lowerdate = 0; // lower bound of date range
    var $upperdate = 0; // upper bound of date range
    var $limit = 100; // number of result to return
    var $min_score = 0.5; // only add these values if they are above this score
    var $httppost=false; // set to true to send via HTTP POST (form) instead of HTTP GET (url)
    var $debug=false; // turn on debugging mode (not fully complete)

    // Reset or initialize variables
    function init() {
        $this->language_id = 0;
        $this->lowerdate = 0;
        $this->upperdate = 0;
        $this->limit = 100;
        $this->min_score = 0.0; // only add these values if they are above this score
        $this->httppost=false;
    }

    // Example of a query -- different queries with their own logic should be made as separate functions
    function runQuery($titlekeywords) {

        $query = $this->setUpQuery();

        // remove spaces
        // change to mb_ereg_replace if using multibyte
        $keywords = preg_replace('@[\s]@', ' ', trim($titlekeywords));
        $string = "(title:($titlekeywords)) ";
        $query.=$string;

        $query = trim($query);

        $data = $this->fetchResults($query);
        $results = $this->handleResponse($data);
    }
}
```

```

        return $results;
    } // end function sortQuery

// Use this to set up default logic that will apply to all queries
function setUpQuery() {
    $query = "";

    // Set up your basic fields here that will be used in the query
    if ($this->language_id > 0) $query.="+language_id:$this->language_id ";
    if ($this->group_id > 0) $query.="+group_id:$this->group_id ";

    $lowerdate = "";
    $upperdate = "";

    if ($this->lowerdate > 0) $lowerdate = $this->lowerdate;
    if ($this->upperdate > 0) $upperdate = $this->upperdate;

    if (($lowerdate!="") || ($upperdate != "")) {
        $query.="+lucene_date:[\".$lowerdate.\" TO \".$upperdate.\" ] ";
    }
    return $query;
}

// Handle the response from the server and transform it into useful fields
function handleResponse($data) {
    if ($data) {

        $xml = simplexml_load_string($data );

        $results = array();
        foreach ($xml->result->doc as $story) {
            $xmlarray = array();

            try{
                foreach ($story as $item) {
                    $name = $item->attributes()->name;
                    $value = $item;

                    $xmlarray["$name"] = "$value";

                } // end foreach
            } catch (Exception $e) {
                echo 'Problem handling XML array.';
            }

            //
            if ($this->debug) echo "checking if ".$xmlarray['score']." > ".$this->min_score;
            if ($xmlarray['score'] > $this->min_score) $results[] = $xmlarray;

        } // end foreach

    } else {
        $results=false;
    } // end if / else data

    return $results;
}

// Send the query to the server
function fetchResults($query) {

    if ($this->debug) echo "parsing string $query";

    $url = "http://".SOLR_META_QUERY."/solr/select";

```

```

        $querystring = "stylesheet=&q=".trim(urlencode($query))."&version=2.1&start=0&rows=".$this->limit."&fl=*&score&q=standard";

        if (!$this->httppost) $selecturl = "?$querystring";
        $url .= $selecturl;

        $header[] = "Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5";
        $header[] = "Accept-Language: en-us,en;q=0.5";
        $header[] = "Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7";

        $ch = curl_init();
        curl_setopt($ch, CURLOPT_URL, $url); // set url to post to
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($ch, CURLOPT_TIMEOUT, 10);
        curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
        curl_setopt($ch, CURLOPT_ENCODING, "");
        curl_setopt($ch, CURLOPT_HTTPHEADER, $header);
        curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, 10);
        curl_setopt($ch, CURLOPT_DNS_USE_GLOBAL_CACHE, 0);

        if ($this->httppost) {
            curl_setopt($ch, CURLOPT_POST, 1);
            curl_setopt($ch, CURLOPT_POSTFIELDS, $querystring);
        }

        if ($this->debug) echo "\r\nRetrieving <A HREF='$url' target=_BLANK>$url</a>...\r\n";

        $data = curl_exec($ch);

        if (curl_errno($ch)) {
            logger "setting results to false, error";
            print curl_error($ch);
            $results=false;
        } else {
            curl_close($ch);
            if ( strstr ( $data, '<status>0</status>' ) ) {
                // Logger::decho("setting to true");
                $results = $data;
            }
        }
        if ($this->debug) echo $data;
        return $results;
    }

} // end class

?>

```