

SolrUpdate

```
<?

// VERSION SolrUpdate 0.200
// Written by Brian Lucas, use any which way you see fit.

// DESCRIPTION
//
// This class performs an update on an existing Solr repository
// It uses the handy ADODB database library, but you can substitute your own if you prefer.
// ADODB makes dealing with SQL much nicer than the php mysql_ calls and handy if you switch databases.
// http://adodb.sourceforge.net/

// SQL TABLE Notice:

// Inside your table that stores the primary key for your data or each record, create a tinyint(1) (or
enum) flag
// called "index_flag". Ex:
// ALTER TABLE `stories` ADD `index_flag` TINYINT( 1 ) NOT NULL ;
// This will allow the update process to continue where it left off in case there is an error.

// Example usages:

// From the command line
/*
#!/usr/local/bin/php
<?php

require_once("config.php"); // config file
require_once(PHYSICALDIR."/ops/lib/common.php");
$DB=createADODB();
$solr = new SolrUpdate;
$solr->updateIndex();
? >
*/

// From within a PHP function

/*
$solr = new SolrUpdate();

$array = array();
$array['story_id'] = $story_id;
$array['group_id'] = $group_id;
$array['lucene_date'] = $lucene_date;
$array['title'] = $title;
$solr->addIndex(array($array));
*/

// Change this to your server
define('SOLR_META_UPDATE', '127.0.0.1:8080');

class SolrUpdate {

// This function will update your solr index with information stored in the database.
// Use addIndex to add data from a function call.

function updateIndex () {
    global $DB;

    list($usec, $sec) = explode(" ", microtime());
    $start = ((float)$usec + (float)$sec);
```

```

$current_count = 0;
$batch_interval = 100;
$limit = 5000;

$loop_control=0;
$continue_index = true;
echo "Preparing to build index...";

while ($continue_index) {

    if ($loop_control++>50) break; //loop infinity control, change if this aborts for you

    $failed = false;
    $continue_index = false;

    // this is the query we are going to use to populate our index
    $dropSql = "drop table if exists temp_story_keywords";
    $ok = $DB->Execute($dropSql);
    if (!$ok) echo "ERROR".$DB->ErrorMsg()." sql:".$dropSql;

    // this is the query we are going to use to populate our index
    $createSql = "create table temp_story_keywords engine=memory select story_keywords.
story_id from story_keywords where story_keywords.index_flag=0 limit $limit";
    $ok = $DB->Execute($createSql);
    if (!$ok) echo "ERROR".$DB->ErrorMsg()." sql:".$createSql;

    // this is the query we are going to use to populate our index
    $selectSql = "select keywords story from (temp_keywords, keywords) where keywords.
story_id = temp_keywords.story_id";

    $rs = $DB->Execute($selectSql);
    if (!$rs) echo "ERROR".$DB->ErrorMsg()." sql:".$selectSql;

    echo "Building index...";

    while ($array = $rs->FetchRow()) {
        $continue_index=true;
        if(!$failed) {

            $result_array[] = $array;

            $current_count++;
            $modcount = $current_count % $batch_interval;

            if ($modcount == 0) {
                echo ".";

                $ok = $this->addIndex($result_array);

                if (!$ok) {
                    $failed = true;
                    echo "Error, restarting...";
                    sleep(1);
                    break;
                }
                $result_array = array();
            } // end if
        }
    } // end while

    if (!$failed) {
        // this is the query that updates the tables
        echo "Updating and committing ".$limit." records...";
        $updateSql = "update story_keywords, temp_story_keywords set story_keywords.
index_flag=1 where story_keywords.story_id = temp_story_keywords.story_id";

        $ok = $DB->Execute($updateSql);
        if (!$ok) echo $DB->ErrorMsg()." sql:".$updateSql;

        // commit the data
        if ($ok) {

```

```

        $this->sendCommit();
        echo "Success.";
    } else {
        echo "Failed.";
    }
}

} // end while

// this is the query we are going to use to populate our index
$finalDropSql = "drop table if exists temp_story_keywords";
$ok = $DB->Execute($finalDropSql);
if (!$ok) echo $DB->ErrorMsg()." sql:". $finalDropSql;

list($usec, $sec) = explode(" ", microtime());
$end = ((float)$usec + (float)$sec);
$retElapsed = ($end - $start)*1000;

$docsec = $current_count/(( $retElapsed)/1000);

echo "Done indexing. Took " . ($retElapsed) . " ms to index $current_count documents (" . $docsec . "
docs/sec)";

echo "Optimizing index...";
$this->sendUpdate('<optimize>');

} // end function

// Use this function to add data from a php call.

// Example usage:
/*
$solr = new SolrUpdate();

$array = array();
$array['story_id'] = $story_id;
$array['group_id'] = $group_id;
$array['lucene_date'] = $lucene_date;
$array['title'] = $title;
$solr->addIndex(array($array));
*/

function addIndex($resultarray) {

    $dom = new DomDocument();
    $root_element = $dom->createElement('add');
    $root = $dom->appendChild($root_element);

    // exclude these values from standard processing, handle them in special cases below
    $excludearray = array("dontincludeme");

    if($resultarray) foreach($resultarray as $num=>$array) {

        $doc_element = $dom->createElement('doc');
        $doc = $root->appendChild($doc_element);

        foreach ( $array as $elementname => $elementvalue ) {
            if (!(in_array($elementname, $excludearray))) { // only run if this isn't
excluded from processing
                if (($elementvalue != null)&&(!is_numeric($elementname))) { // ADODB
passes numeric entities alongside fieldnames
                    $element = $dom->createElement('field');
                    $element->setAttribute('name', $elementname);
                    $elementfinal = $dom->createTextNode($this->escapeChars
($elementvalue));

                    $element->appendChild($elementfinal);
                    $doc->appendChild($element);
                }
            }
        }

    } // end if /foreach

```

```

        $dom_string = $dom->saveXML();

        if ($dom_string) {
            $ok = $this->sendUpdate($dom_string);
        } else {
            echo "Error with xml document";
            print_r($resultarray);
        }
        return $ok;
    }

    function sendCommit() {

        // Choose what type of commit you want,
        // which is non-blocking or blocking and refreshing(flush) or not

        // non-blocking, non-refreshing index
        //          $this->sendUpdate('<commit waitFlush="false" waitSearcher="false"/>');

        // blocking
        $this->sendUpdate('<commit/>');
    }

    function escapeChars($string) {
        $string = str_replace("&", "&amp;", $string);
        $string = str_replace("<", "&lt;", $string);
        $string = str_replace(">", "&gt;", $string);
        $string = str_replace("'", "&apos;", $string);
        $string = str_replace('"', "&quot;", $string);
        return $string;
    }

    // the following update DOES work.  If you are having problems, make sure your SOLR install is working
    properly.
    function sendUpdate($post_string) {
        $url = "http://".SOLR_META_UPDATE."/solr/update";
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_URL,$url);
        curl_setopt($ch, CURLOPT_POST, 1);
        curl_setopt($ch, CURLOPT_POSTFIELDS,$post_string);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
        $data = curl_exec($ch);
        if (curl_errno($ch)) {
            print "curl_error:".curl_error($ch);
            return false;
        } else {
            curl_close($ch);
            if ( strstr ( $data, '<result status="0"></result>')) {
                return true;
            } else {
                return false;
            }
        }
    }
    } // end function send update
} // end class

?>

```