Annals of Library and Information Studies
Vol. 63, December 2016, pp. 241-260

The triumph of David: A case study in VuFind customization

Demian Katz

Library Technology Development Specialist, Falvey Memorial Library, Villanova University, United States of America,
Email: demian.katz@villanova.edu

Received: 19 September 2016; revised and accepted: 28 December 2016

When Villanova University undertook a major project to restore a large-scale painting with possible ties to the workshop of 17th-century painter Pietro da Cortona, an important component of the work was detailed data collection, including time-lapse and multispectral photography. When this data needed to be presented to the general public, a number of open source technologies were used, including VuFind, the discovery layer developed at Villanova's Falvey Memorial Library. This paper examines how VuFind was used as the glue to tie together the disparate components of the project, and by detailing the implementation strategies selected, demonstrates how VuFind's flexible and extensible architecture allows it to be adapted to a wide range of specialty projects.

Keywords: VuFind; Integrated search; Indexing

Introduction

From 2013 to 2015, a conservation project was undertaken at Villanova University to restore a huge painting of "The Triumph of David" that had hung in Falvey Memorial Library for decades and which was believed to have originated in Pietro da Cortona's 17th century workshop. This project was thoroughly documented, and one of its outputs was a wealth of digital imagery, including multispectral images and time-lapse photography showing progress on the restoration.

Upon the completion of the project, a public website was created at www.thetriumphofdavid.com to share information about the painting and its restoration. This site was composed using several open source technologies: WordPress[1] for blogging and simple content management; Omeka[2] with the NeatLine[3] plugin and a GeoServer[4] backend to allow dynamic interactions with the project's imagery; and VuFind[5] to provide a common search layer across the other technologies.

This paper will focus on the VuFind component of the project, showing some of the decision-making involved and demonstrating how VuFind's extensibility enabled a relatively lightweight solution to a complex problem.

Project Goals

The VuFind component of The Triumph of David needed to achieve three key goals:

1. Make the detailed information in the Omeka/Neatline exhibit searchable.
2. Make the blog posts in WordPress searchable.
3. Provide a simplified, more linear view of the Omeka/Neatline data as an alternative to the interactive timeline to improve accessibility.

The first two goals played to VuFind's strength as an integrated search tool. The third goal could likely have also been achieved in other layers of the system, but VuFind was chosen as the platform because the project team was familiar with its Zend Framework 2.x[6] architecture and because its Solr[7] index offered convenient access to all of the data needed to allow the site's "simple view" to be generated quickly and easily.

Step 1: Choosing an Index Schema

VuFind comes with support for a variety of different search systems. For projects like this one where a search engine is being created locally, it is a given that one of VuFind's Solr cores will be used;

however, the developer has several predefined Solr schemas (and supporting PHP code bases) to choose from. Two options were strong contenders for this project. The first was VuFind's standard "biblio" core, which was originally designed for indexing MARC (and MARC-like) records and which is complemented by PHP code that provides not just search capabilities but also rich functionality for interacting with individual records. The second was the lighter-weight "website" core, designed for indexing web pages and complemented by simpler PHP code providing links directly from search results to individual pages without an intervening record page.

For this project, the rich functionality and detailed schema of the "biblio" core were not really needed -- the primary goal was to simply help users locate pages within the site, not to provide additional information or capabilities related to those pages. Thus, the "website" core was chosen as the foundation for the project's VuFind implementation, since it provided existing code very close to the project's use case, and because its simple schema would make the indexing process straightforward, especially due to the presence of existing tools for web crawling.

Step 2: Indexing WordPress

Because VuFind includes a simple tool to update the "website" Solr core by harvesting a site's sitemap.xml file[8] and analyzing its contents using Apache Tika[9], including WordPress content in the index was one of the easiest parts of the project.

To begin the work, a VuFind instance was installed and configured as documented in the project's wiki[10]. During the installation process, a local Zend Framework module[11] called Cortona was created; this would be used to house all custom PHP code generated during the course of the project, effectively separating it from the VuFind core and making long-term maintenance easier by isolating all project code to a single location. Next, website indexing was set up, again following wiki documentation[12], to allow the WordPress content to be harvested and analyzed by Tika. At this stage, simply by following online instructions, a fully functional WordPress search was implemented.

In order to enrich the search experience, some custom code was written to take advantage of certain WordPress conventions in order to extract additional information into the index. VuFind's web crawling tool uses an XSLT stylesheet hooked to custom PHP code in order to map a sitemap.xml file into a set of documents in the Solr index. VuFind provides hooks to extend and customize this code. A Cortona\XSLT\Import\VuFindSitemap class (see Listing 1) was written. This code cleans up the content of title fields, adds a hard-coded "category" facet of "Blog" and uses regular expressions to extract "subject" facet values and update dates for sorting from the HTML. Instructing the VuFind web crawler to use this custom class is as simple as copying the standard import/sitemap.properties file into VuFind's local settings directory[13] and modifying the "custom_class[]" setting to point to the custom class instead of the default one.

Listing 1: Cortona\XSLT\Import\VuFindSitemap

```
<?php
/**
 * XSLT importer support methods for sitemaps.
 *
 * PHP version 5
 *
 * Copyright (c) Demian Katz 2016.
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2,
 * as published by the Free Software Foundation.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
```

```
* You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 *
 * @category VuFind2
 * @package  Import_Tools
 * @author   Demian Katz <demian.katz@villanova.edu>
 * @license  http://opensource.org/licenses/gpl-2.0.php GNU General Public License
 * @link       http://vufind.org/wiki/importing_records Wiki
 */
namespace Cortona\XSLT\Import;

/**
 * XSLT support class -- all methods of this class must be public and static;
 * they will be automatically made available to your XSL stylesheet for use
 * with the php:function() function.
 *
 * @category VuFind2
 * @package  Import_Tools
 * @author   Demian Katz <demian.katz@villanova.edu>
 * @license  http://opensource.org/licenses/gpl-2.0.php GNU General Public License
 * @link       http://vufind.org/wiki/importing_records Wiki
 */
class VuFindSitemap extends \VuFind\XSLT\Import\VuFindSitemap
{
    /**
     * Extract key metadata from HTML.
     *
     * @param string $html HTML content.
     *
     * @return array
     */
    protected static function getHtmlFields($html)
    {
        $fields = parent::getHtmlFields($html);

        // Try to find a date in the HTML (favor posted date over modified date):
        $patterns = [
            '|Posted Date:</strong>([^<]+)<|ms', '/Last Modified:([^<]+)</ms'
        ];
        foreach ($patterns as $pattern) {
            preg_match_all(
                 $pattern, $html, $matches
            );
            if (isset($matches[1][0]) && !empty($matches[1][0])) {
                $date = $matches[1][0];
                break;
            }
        }
        if (isset($date)) {
            $fields['sort_date'] = date('Y-m-d\TH:i:s\Z', strtotime(trim($date)));
        }

        // Extract subject tagging from the HTML:
        preg_match_all('|rel="category tag">([^<]+)</a>|ms', $html, $tags);
        if (isset($tags[1]) && !empty($tags[1])) {
            $fields['subject'] = $tags[1];
        }

        // Hard-code the category:
        $fields['category'] = 'Blog';
        return $fields;
    }

    /**
     * Support method for getDocument() -- retrieve associative array of field data.
     *
     * @param string $url URL of file to retrieve.
     *
     * @return array
     */
    protected static function getDocumentFieldArray($url)
    {
        $fields = parent::getDocumentFieldArray($url);
        // Trim trailing pipes:
        $fields['title'] = rtrim($fields['title'], '| ');
        return $fields;
    }
}
```

Step 3: Indexing Neatline

While the WordPress ingest was quite straightforward, indexing data from NeatLine provided a more significant challenge, as no existing tools were available to perform the task. After a detailed analysis of the MySQL database used by Omeka, it was determined that all of the data needed by VuFind could be found in two tables: omeka_neatline_exhibits and omeka_neatline_records. A PHP class, Cortona\Indexer (listing 2), was created to connect to MySQL using PDO[14], run queries to extract data, format the data into Solr <add> documents[15] for inclusion in the index, and post the data using VuFind's provided VuFind\Solr\Writer class. To avoid including server and password details in the code, a Cortona.ini file was introduced to store key elements. A simple controller, Cortona\Controller\CortonaController (listing 3), was created to inject dependencies into the indexer and to allow it to be invoked from the command line. Once registered in the Cortona module's module.config.php file, all data could be harvested from Omeka into VuFind with the command `php $VUFIND_HOME/public/index.php cortona import`.

While the details of the code shared here are very specific to this project's needs, they demonstrate a simple approach to mapping arbitrary database content into Solr documents using PHP. Also note the use of dynamic field suffixes like _str_mv (for multivalued strings) which allow arbitrary values to be stored in the index without the need to modify the VuFind-provided Solr schema configuration file. These custom values are used in the next step to build the "simple view."

Listing 2: Cortona\Indexer

```
<?php
/**
 * Cortona Indexer
 *
 * PHP version 5
 *
 * Copyright (C) Villanova University 2016.
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2,
 * as published by the Free Software Foundation.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
*
 * You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 */
class Indexer
{
    /**
     * Index connection
     *
     * @var SolrWriter
     */
    protected $solr;

    /**
     * Cortona.ini contents
     *
     * @var Config
     */
    protected $config;

    /**
     * Database connection
     *
     * @var PDO
     */
    protected $db;

    /**
     * Success count
     *
     * @var int
     */
    protected $success = 0;

    /**
     * Failure count
     *
     * @var int
     */
    protected $failure = 0;

    /**
     * Total record count
     *
     * @var int
     */
    protected $total = 0;

    /**
     * Current work in progress record.
     *
     * @var string
     */
    protected $currentRecord;

    /**
     * Contents of "allfields" field for work in progress record.
     *
     * @var array
     */
    protected $allFields;
```

```
/**
 * Constructor
 *
 * @param SolrWriter $solr   Index connection
 * @param Config     $config Cortona.ini contents
 */
public function __construct(SolrWriter $solr, Config $config)
{
    $this->solr = $solr;
    $this->config = $config;
    $this->db = new PDO(
        $config->Database->dsn,
        $config->Database->username,
        $config->Database->password,
        []
    );
}

/**
 * Get count of successfully indexed records.
 *
 * @return int
 */
public function getSuccess()
{
    return $this->success;
}

/**
 * Get count of indexing failures.
 *
 * @return int
 */
public function getFailure()
{
    return $this->failure;
}

/**
 * Get count of total records encountered.
 *
 * @return int
 */
public function getTotal()
{
    return $this->total;
}

/**
 * Kick off the indexing process.
 *
 * @return void
 */
public function indexAll()
{
    // Load exhibit data:
    $exhibitStatement = $this->db->prepare("SELECT * FROM omeka_neatline_exhibits");
    $exhibitStatement->execute();
    $exhibits = [];
    while ($row = $exhibitStatement->fetch()) {
        $exhibits[$row['id']] = $row;
    }

    // Load records:
    $statement = $this->db->prepare("SELECT * FROM omeka_neatline_records");
    $statement->execute();
    while ($row = $statement->fetch(PDO::FETCH_ASSOC)) {
        $this->total++;
        $this->indexNeatlineRecord($row, $exhibits[$row['exhibit_id']]);
    }

    // If we encountered any records, we want to do a commit/optimize.
    if ($this->total) {
        // Issue a Commit Statement
        $this->solr->commit('SolrWeb');

        // Issue a Optimize Statement
        $this->solr->optimize('SolrWeb');
    }
}

/**
 * Index a single Neatline record.
 *
 * @param array $row     Neatline record details
 * @param array $exhibit Associated exhibit details
 *
 * @return void
 */
protected function indexNeatlineRecord($row, $exhibit)
{
    // Determine which Simile contains (or IS) the current record; we can't
    // add this to the index if we can't put it in proper context, since
    // we want to display records in VuFind as a single Simile record with
    // a series of associated items beneath it.
    try {
        $parentSimile = $this->getParentSimile($row);
    } catch (\Exception $ex) {
        $this->failure++;
        echo 'Skipping item: ' . $ex->getMessage() . "\n";
        return;
    }

    // Reset the internal "record in progress":
    $this->startNewRecord();

    // Fill in the values:
    $this->addFieldToRecord('id', $row['id']);
    $this->addFieldToRecord('category', $exhibit['title']);
    $this->addFieldToRecord('description', $row['body']);
    $this->addFieldToRecord('title', $row['title']);
    $titleSort = ltrim($row['title'], '- ');    // strip leading "-- "
    $this->addFieldToRecord('title_sort', $titleSort, false);
    $this->addFieldToRecord('url', $this->getNeatlineRecordUrl($row, $exhibit));
    $this->addFieldToRecord('sort_date', $row['start_date'] . 'T00:00:00Z');
    $tags = array_map('trim', explode(',', $row['tags']));
    $this->addFieldToRecord('subject', $tags);
    $widgets = empty($row['widgets']) ? [] : explode(',', $row['widgets']);
    $this->addFieldToRecord('widgets_str_mv', $widgets);
    $this->addFieldToRecord('simile_id_str', $parentSimile);
    $this->addFieldToRecord('weight_sint', $row['weight']);
    $urls = $this->getImageUrlDetails($row);
```

```
        $this->addFieldToRecord('url_orig_str_mv', $urls['orig']);
        $this->addFieldToRecord('url_full_str_mv', $urls['full']);
        $this->addFieldToRecord('url_wms_str_mv', $urls['wms']);

        // If this row IS a Simile, get the next and previous ones for
        // navigation purposes.
        if ($row['id'] == $parentSimile) {
            $nav = $this->getSimileNavigation($row);
            $this->addFieldToRecord('prev_simile_id_str', $nav['prev']);
            $this->addFieldToRecord('next_simile_id_str', $nav['next']);
        }

        // Close up and save the record:
        $this->endCurrentRecord();
        $this->saveCurrentRecord($row['id']);
    }

    /**
     * Return image URL details for the specified row.
     *
     * @param array $row Current row
     *
     * @return array
     */
    protected function getImageUrlDetails($row)
    {
        // Initialize the URL array. We have three types: original and full
        // images which will be represented as parallel arrays (two sizes of
        // the same things), plus WMS images pulled from the GeoServer.
        $urls = ['orig' => [], 'full' => [], 'wms' => null];

        // First, use the item API to retrieve information about associated
        // files; this will fill in the full and orig values:
        $itemDetails = $this->getItemDetails($row['item_id']);
        if (isset($itemDetails->files)) {
            $list = is_array($itemDetails->files)
                ? $itemDetails->files : [$itemDetails->files];
            foreach ($list as $file) {
                foreach ($this->getFileDetails($file->url) as $current) {
                    if (isset($current->file_urls->original)
                        && isset($current->file_urls->fullsize)
                    ) {
                        $urls['orig'][] = $current->file_urls->original;
                        $urls['full'][] = $current->file_urls->fullsize;
                    }
                }
            }
        }

        // Next, construct a GeoServer link if WMS settings are present. Note
        // that we use the "reflect" wrapper to cut down on the number of GET
        // parameters required to render the layer. See more details at this URL:
        // http://docs.geoserver.org/latest/en/user/services/wms/reference.html
        if (isset($row['wms_address']) && !empty($row['wms_address'])
            && isset($row['wms_layers']) && !empty($row['wms_layers'])
        ) {
            $base = isset($this->config->GeoServer->wms_base)
                && $this->config->GeoServer->wms_base
                ? $this->config->GeoServer->wms_base
                : $row['wms_address'];
            $urls['wms'] = $base . '/reflect?layers='
            . urlencode($row['wms_layers']);
    }

    // Send back our collected details:
    return $urls;
}

/**
 * Retrieve file details from the JSON API, and force them to an array for
 * consistent upstream processing.
 *
 * @param string $url URL to fetch
 *
 * @return array
 */
protected function getFileDetails($url)
{
    $fileDetails = json_decode(file_get_contents($url));
    return is_array($fileDetails) ? $fileDetails : [$fileDetails];
}

/**
 * Retrieve information (in object format) about an item from the Omeka API.
 *
 * @param int $id Item ID
 *
 * @return object
 */
protected function getItemDetails($id)
{
    if (empty($id)) {
        return null;
    }
    $uri = $this->config->Omeka->api . '/items/' . $id;
    return json_decode(file_get_contents($uri));
}

/**
 * Get the next and previous Similes based on date.
 *
 * @param array $row Details of the current Simile.
 *
 * @return array
 */
protected function getSimileNavigation($row)
{
    $nav = ['prev' => false, 'next' => false];

    if (!empty($row['start_date'])) {
        $q1 = "SELECT * FROM omeka_neatline_records "
            . "WHERE (end_date < :start) "
            . "AND widgets LIKE '%Simile%'"
            . "ORDER BY end_date DESC LIMIT 1";
        $s1 = $this->db->prepare($q1);
        $s1->execute([':start' => $row['start_date']]);
        while ($c = $s1->fetch()) {
            $nav['prev'] = $c['id'];
        }
    }

    if (!empty($row['end_date'])) {
            $q2 = "SELECT * FROM omeka_neatline_records "
                . "WHERE (start_date > :end) "
                . "AND widgets LIKE '%Simile%'"
                . "ORDER BY start_date ASC LIMIT 1";
            $s2 = $this->db->prepare($q2);
            $s2->execute([':end' => $row['end_date']]);
            while ($c = $s2->fetch()) {
                $nav['next'] = $c['id'];
            }
        }

        return $nav;
    }

    /**
     * Figure out which Simile record contains (or IS) the specified row.
     *
     * @param array $row Row details
     * @return int
     *
     * @throws \Exception
     */
    protected function getParentSimile($row)
    {
        // We want to find a Simile record whose date range fully contains the
        // dates of the provided $row.
        $query = "SELECT * FROM omeka_neatline_records "
            . "WHERE (start_date IS NULL OR start_date <= :start) "
            . "AND (end_date IS NULL OR end_date >= :end) "
            . "AND exhibit_id = :exhibit "
            . "AND widgets LIKE '%Simile%'";
        $statement = $this->db->prepare($query);
        $params = [
            ':start' => $row['start_date'],
            ':end' => $row['end_date'],
            ':exhibit' => $row['exhibit_id']
        ];

        // Count how many rows came back, and save the $id from the most
        // recently retrieved row. (We can probably do this more efficiently,
        // but this approach is very cross-DB-platform friendly).
        $statement->execute($params);
        $count = 0;
        while ($parentRow = $statement->fetch(PDO::FETCH_ASSOC)) {
            $id = $parentRow['id'];
            $count++;
        }

        // If we had too few or too many matches, something is wrong and we
        // should abort indexing for this record.
        if ($count !== 1) {
            throw new \Exception(
                'Unexpected count (' . $count . ') of associated Simile rows for '
                . $row['id']
            );
        }

        // If we got this far, it's safe to assume that $id is set, since the
        // exception above covers cases where $id is unset.
        return $id;
    }
```

```
/**
 * Assemble a link into the Neatline exhibit.
 *
 * @param array $row     Neatline record details
 * @param array $exhibit Associated exhibit details
 *
 * @return string
 */
protected function getNeatlineRecordUrl($row, $exhibit)
{
    return "{$this->config->Omeka->base}/{$exhibit['slug']}#records/{$row['id']}";
}

/**
 * Reset the state of the current in-progress record.
 *
 * @return void
 */
protected function startNewRecord()
{
    // Create Record
    $this->currentRecord = '<?xml version="1.0" encoding="utf-8"?>' . "\n";
    $this->currentRecord .= '<add>' . "\n";
    $this->currentRecord .= '  <doc>' . "\n";

    // Initialize "all fields" list:
    $this->allFields = array();
}

/**
 * Add data to the record in progress.
 *
 * @param string $name             Name of field
 * @param mixed  $values           One or more values to insert into field
 * @param bool   $saveToAllFields  Should value(s) also go into allfields field?
 *
 * @return void
 */
protected function addFieldToRecord($name, $values, $saveToAllFields = true)
{
    $values = (array)$values;
    foreach ($values as $value) {
        // Normalize whitespace, do not index empty values:
        $value = trim($value);
        if (strlen($value) == 0) {
            continue;
        }

        // Sanitize the value for XML legality:
        $regex = '/[^\x{0009}\x{000a}\x{000d}\x{0020}-\x{D7FF}\x{E000}-\x{FFFD}]+/u';
        $value = trim(preg_replace($regex, ' ', $value, -1));

        // Add the field:
        foreach ((array)$name as $field) {
            $this->currentRecord .= '    <field name="' . $field . '">' .
                htmlspecialchars($value) . '</field>' . "\n";
        }

        // If requested, save the value into the "all fields" list:
        if ($saveToAllFields) {
                $this->allFields[] = $value;
            }
        }
    }

    /**
     * Close up the current record in progress.
     *
     * @return void
     */
    protected function endCurrentRecord()
    {
        // Put in the "all fields" list as the final value:
        $this->addFieldToRecord('keywords', implode(' ', $this->allFields), false);

        // Close the record:
        $this->currentRecord .= '  </doc>' . "\n";
        $this->currentRecord .= '</add>' . "\n";
    }

    /**
     * Save the record-in-progress to the index.
     *
     * @param string $id ID of record being saved (for debugging only).
     *
     * @return void
     */
    protected function saveCurrentRecord($id)
    {
        // Clean up content
        try {
            $this->solr->save('SolrWeb', new RawXMLDocument($this->currentRecord));
            $this->success++;
        } catch (\Exception $e) {
            $this->failure++;
            echo "Failure: " . $e->getMessage() . " : $this->currentRecord\n";
            file_put_contents('import-log', '[' . date('r') .
                '] Could not import record ' . $id . ':' .
                $e->getMessage() . "\n", FILE_APPEND);
        }
    }
}
```

Listing 3: Cortona\Controller\CortonaController

```
<?php
/**
 * CLI Controller Module
 *
 * PHP version 5
 *
 * Copyright (C) Villanova University 2016.
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2,
 * as published by the Free Software Foundation.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 *
 * @category VuFind2
 * @package  Controller
 * @author   Demian Katz <demian.katz@villanova.edu>
 * @license  http://opensource.org/licenses/gpl-2.0.php GNU General Public License
 * @link       http://vufind.org/wiki/vufind2:building_a_controller Wiki
 */
namespace Cortona\Controller;
use Cortona\Indexer, Zend\Console\Console;


/**
 * This controller handles various command-line tools
 *
 * @category VuFind2
 * @package  Controller
 * @author   Demian Katz <demian.katz@villanova.edu>
 * @license  http://opensource.org/licenses/gpl-2.0.php GNU General Public License
 * @link       http://vufind.org/wiki/vufind2:building_a_controller Wiki
 */
class CortonaController extends \VuFindConsole\Controller\AbstractBase
{
    /**
     * Import records
     *
     * @return \Zend\Console\Response
     */
    public function importAction()
    {
        $importer = new Indexer(
            $this->getServiceLocator()->get('VuFind\Solr\Writer'),
            $this->getServiceLocator()->get('VuFind\Config')->get('Cortona')
        );
        $importer->indexAll();
        Console::writeLine("Import Completed\n");
        Console::writeLine(
            "Imported " . $importer->getSuccess() . " Records of " . $importer->getTotal() .
            " total with " . $importer->getFailure() . " failures"
        );

        return $this->getSuccessResponse();
    }
}
```

Step 4: Building the "Simple View"

The Neatline exhibit for The Triumph of David was constructed as a collection of SIMILE Timelines[16]. These interactive timelines feature layered images and pop-up boxes containing additional information. The "Simple View" in VuFind was designed to make the same information available in a more linear, less interactive format as an alternate presentation for users with mobile devices, disabilities, or other limitations or preferences restricting the usefulness of the full timeline view.

At index time, every point on every timeline is indexed as a distinct Solr record, allowing users to perform very precise searches. Each Solr record contains a field identifying its parent timeline as well as additional fields containing all of the text and image URLs necessary to display the content. The "Simple View" is implemented as a controller (Cortona\Controller\CortonaRecordController, listing 4) which performs a simple Solr query to retrieve all of the points on a specified timeline and passes the data to a template (cortonarecord/view.phtml, listing 5) which renders the data in a linear fashion, using HTML anchors to allow direct linking to any specific point on the timeline. Note that the custom template was created inside a custom VuFind theme[17] which, like the Cortona Zend Framework module, helps to clearly separate local customizations from the core VuFind distribution. Also note that some additional entries had to be added to the Cortona module's module.config.php to expose the new controller and related route.

Listing 4: Cortona\Controller\CortonaRecordController

```
<?php
/**
 * CLI Controller Module
 *
 * PHP version 5
 *
 * Copyright (C) Villanova University 2016.
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2,
 * as published by the Free Software Foundation.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 *
 * @category VuFind2
 * @package  Controller
 * @author   Demian Katz <demian.katz@villanova.edu>
 * @license  http://opensource.org/licenses/gpl-2.0.php GNU General Public License
 * @link       http://vufind.org/wiki/vufind2:building_a_controller Wiki
 */
namespace Cortona\Controller;
use VuFindSearch\ParamBag;


/**
 * This controller handles various command-line tools
 *
 * @category VuFind2
 * @package  Controller
 * @author   Demian Katz <demian.katz@villanova.edu>
 * @license  http://opensource.org/licenses/gpl-2.0.php GNU General Public License
 * @link       http://vufind.org/wiki/vufind2:building_a_controller Wiki
 */
class CortonaRecordController extends \VuFind\Controller\AbstractBase
{
    /**
     * Display a Cortona record in "static" mode.
     *
     * @return mixed
     */
    public function viewAction()
    {
        // The incoming ID will be the top-level "Simile" record....
        $id = $this->params()->fromRoute('id');

        // Connect to Solr and retrieve all of the records associated with the
        // specified Simile, sorting them by weight. There should never be more
        // than a dozen or so, so limiting to 1000 should ensure that we get
        // everything we need in a single request.
        $solr = $this->getServiceLocator()->get('VuFind\Search\BackendManager')
            ->get('SolrWeb')->getConnector();
        $params = new ParamBag(
            [
                'q' => 'simile_id_str:' . $id,
                'sort' => 'weight_sint asc',
                'rows' => '1000',
                'wt' => 'json',
            ]
        );
        $results = json_decode($solr->search($params));
        return $this->createViewModel(['docs' => $results->response->docs]);
    }
}
```

Listing 5: cortonarecord/view.phtml

```
<?
  // Set page title (first document in array is the top-level Simile).
  $this->headTitle(isset($docs[0]) ? $docs[0]->title : '');
  $this->layout()->searchClassId = 'SolrWeb';
  $this->layout()->timelineLink = isset($docs[0])
    ? $this->escapeHtmlAttr($docs[0]->url) : null;
?>
<div class="record row">
  <div class="<?=$this->layoutClass('mainbody')?>">
    <? /* we have an array of documents associated with the Simile... */ ?>
    <? $lastBodyHadContent = true; ?>
    <? $toc = ''; // build table of contents as we go.... ?>
    <? foreach ($docs as $i => $doc): ?>
      <div id="record-<?=$doc->id?>">
        <? /* the first record is the Simile itself, so it gets an h1 instead of h2. */ ?>
        <? $header = $i > 0 ? 'h2' : 'h1'; ?>
        <? $trimmedTitle = ltrim($doc->title, '- '); if ($doc->title != $trimmedTitle) { $header = 
'h3'; } ?>
        <? /* Skip displaying the header if the previous body was empty.... */ ?>
        <? if ($lastBodyHadContent): ?>
          <<?=$header?>><?=$this->escapeHtml($trimmedTitle)?></<?=$header?>>
          <? $toc .= '<li style="margin-left: ' . ((str_replace('h', '', $header) - 1) * 1.5) . 
'em;"><a href="#record-' . $doc->id . '">' . $this->escapeHtml($trimmedTitle) . '</a></li>'; ?>
        <? endif; ?>
        <? $lastBodyHadContent = false; // reset flag, recheck below... ?>
        <? /* we can assume that the "orig" and "full" image URL arrays have the same
              number of values, since this is guaranteed at index time. We just want
              to display the "full" web-friendly sizes with links to the originals. */ ?>
        <? if (isset($doc->url_orig_str_mv) && !empty($doc->url_orig_str_mv)): ?>
          <? $lastBodyHadContent = true; ?>
          <? foreach ($doc->url_orig_str_mv as $i => $orig): ?>
            <a href="<?=$this->escapeHtmlAttr($orig)?>">
              <img src="<?=$this->escapeHtmlAttr($doc->url_full_str_mv[$i])?>" />


            </a><br />
          <? endforeach; ?>
        <? endif; ?>
        <? /* next we want to display GeoServer links, if any; note that we might
              be able to append some GET parameters here to change background
              colors, scale to different sizes, etc. -- these are dynamic! */ ?>
        <? if (isset($doc->url_wms_str_mv) && !empty($doc->url_wms_str_mv)): ?>
          <? $lastBodyHadContent = true; ?>
          <? foreach ($doc->url_wms_str_mv as $wms): ?>
            <img src="<?=$this->escapeHtmlAttr($wms)?>" /><br />
          <? endforeach; ?>
        <? endif; ?>
        <? if (isset($doc->description) && !empty($doc->description)): ?>
          <? $lastBodyHadContent = true; ?>
          <p><?=$this->escapeHtml($doc->description)?></p>
        <? endif; ?>
      </div>
    <? endforeach; ?>
  </div>

  <div class="<?=$this->layoutClass('sidebar')?>">
    <? if (isset($docs[0]->prev_simile_id_str) || isset($docs[0]->next_simile_id_str)): ?>
      <div class="btn-group">
        <? if (isset($docs[0]->prev_simile_id_str)): ?>
          <a    href="<?=$this->url('cortona-record',    ['id'    =>    $docs[0]->prev_simile_id_str])?>"
class="btn btn-default">
            Previous Event
          </a>
        <? endif; ?>
        <? if (isset($docs[0]->next_simile_id_str)): ?>
          <a    href="<?=$this->url('cortona-record',    ['id'    =>    $docs[0]->next_simile_id_str])?>"
class="btn btn-default">
            Next Event
          </a>
        <? endif; ?>
      </div>
    <? endif; ?>
    <!--<a href="<?=$this->layout()->timelineLink?>">Timeline View</a>-->
    <h2>Contents</h2>
    <ul><?=$toc?></ul>
  </div>
</div>
```

Step 5: Customizing Record Displays

With the index fully populated and the "simple view" implemented, the next step is to ensure that search results display in the desired format and link both to the Omeka/Neatline timeline view and to the VuFind-hosted "simple view."

VuFind uses a system of "record drivers"[18] that allow different records within the Solr index to be handled in different ways. Each record driver class is responsible for providing public methods to access Solr data elements, and each class is also associated with its own set of templates, allowing different sorts of records to be displayed in completely different ways if necessary.

For the purposes of this project, the standard SolrWeb record driver used by the code accompanying VuFind's "website" Solr core was almost completely suitable. However, the record driver needed to be extended to provide access to a couple of dynamic fields (see listing 6) and a custom search result template was created in the cortona theme to adjust the default display to include additional links to the timeline and simple views (see listing 7). In addition to these two files, some further Cortona module.config.php changes were needed, bringing it to its final state as shown in listing 8, as was the creation of a simple factory class (listing 9) to build the custom record driver.

Listing 6: Cortona\RecordDriver\SolrWeb

```
<?php
/**
 * Model for Solr web records (customized for Cortona project).
 *
 * PHP version 5
 *
 * Copyright (C) Villanova University 2016.
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2,
 * as published by the Free Software Foundation.
 *
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 *
 * @category VuFind2
 * @package  RecordDrivers
 * @author   Demian Katz <demian.katz@villanova.edu>
 * @license  http://opensource.org/licenses/gpl-2.0.php GNU General Public License
 * @link       http://vufind.org/wiki/vufind2:record_drivers Wiki
 */
namespace Cortona\RecordDriver;


/**
 * Model for Solr web records (customized for Cortona project).
 *
 * @category VuFind2
 * @package  RecordDrivers
 * @author   Demian Katz <demian.katz@villanova.edu>
 * @license  http://opensource.org/licenses/gpl-2.0.php GNU General Public License
 * @link       http://vufind.org/wiki/vufind2:record_drivers Wiki
 */
class SolrWeb extends \VuFind\RecordDriver\SolrWeb
{
    /**
     * Get the ID of the Simile record that contains the current record (or
     * false if this is inapplicable -- e.g. for blog posts). Note that if the
     * current record IS a Simile record, it will return its own ID here.
     *
     * @return int|bool
     */
    public function getContainingSimile()
    {
        return isset($this->fields['simile_id_str'])
            ? $this->fields['simile_id_str'] : false;
    }

    /**
     * Get the sort date.
     *
     * @return string|bool
     */
    public function getSortDate()
    {
        return isset($this->fields['sort_date'])
            ? $this->fields['sort_date'] : false;
    }
}
```

Listing 7: RecordDriver/SolrWeb/result-list.phtml

```
<?
  $url = $this->driver->getUrl();
?>
<div class="listentry col-xs-11">
  <div class="resultItemLine1">
    <a href="<?=$this->escapeHtmlAttr($url)?>" class="title">
      <?=ltrim($this->record($this->driver)->getTitleHtml(), '- ')?>
    </a>
  </div>
  <div class="resultItemLine2">
    <? $snippet = $this->driver->getHighlightedSnippet(); ?>
    <? $summary = $this->driver->getSummary(); ?>
    <? if (!empty($snippet)): ?>
      <?=$this->highlight($snippet['snippet'])?>
    <? elseif (!empty($summary)): ?>
      <?=$this->escapeHtml($summary[0])?>
    <? endif; ?>
  </div>


  <div class="resultItemLine3">
    <?=''//disabled 10/20/15 DDK - $this->escapeHtml($url)?>
    <? $id = $this->driver->getUniqueId(); $simileId = $this->driver->getContainingSimile(); if
(false !== $simileId): ?>
      <a href="<?=$this->escapeHtmlAttr($url)?>">
        <i class="fa fa-sliders"></i> timeline view
      </a>&nbsp;
      <a href="<?=$this->url('cortona-record', ['id' => $simileId]) . ($id === $simileId ? '' :
'#record-' . $id)?>">
        <i class="fa fa-sticky-note-o"></i> simple view
      </a>
    <? endif; ?>
    <? $sortDate = $this->driver->getSortDate(); list($sortDate) = explode('T', $sortDate); if
(!empty($sortDate)): ?>
      <? if (false !== $simileId): ?><br /><? endif; ?>
      <?=$this->transEsc('Date')?>: <?=$this->escapeHtml(trim($sortDate))?>
    <? endif; ?>
  </div>
</div>
```

Listing 8: Final Cortona module.config.php

```
<?php

return array (
  'controllers' => array (
    'invokables' => array (
      'cortona' => 'Cortona\Controller\CortonaController',
      'cortonarecord' => 'Cortona\Controller\CortonaRecordController',
    ),
  ),
  'router' => array (
    'routes' => array (
      'cortona-record' => array (
        'type' => 'Zend\Mvc\Router\Http\Segment',
        'options' => array (
          'route' => '/View/:id',
          'defaults' => array (
            'controller' => 'CortonaRecord',
            'action' => 'View',
          ),
        ),
      ),
    ),
  ),
  'vufind' => array (
    'plugin_managers' => array (
      'recorddriver' => array (
        'factories' => array (
          'solrweb' => 'Cortona\RecordDriver\Factory::getSolrWeb',
        ),
      ),
    ),
  ),
);
```

Listing 9: Cortona\RecordDriver\Factory

```
<?php
/**
 * Record Driver Factory Class
 *
 * PHP version 5
 *
 * Copyright (C) Villanova University 2016.
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2,
 * as published by the Free Software Foundation.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 *
 * @category VuFind2
 * @package  RecordDrivers
 * @author   Demian Katz <demian.katz@villanova.edu>
 * @license  http://opensource.org/licenses/gpl-2.0.php GNU General Public License
 * @link       http://vufind.org/wiki/vufind2:hierarchy_components Wiki
 */
namespace Cortona\RecordDriver;

/**
 * Record Driver Factory Class
 *
 * @category VuFind2
 * @package  RecordDrivers
 * @author   Demian Katz <demian.katz@villanova.edu>
 * @license  http://opensource.org/licenses/gpl-2.0.php GNU General Public License
 * @link       http://vufind.org/wiki/vufind2:hierarchy_components Wiki
 *
 * @codeCoverageIgnore
 */
class Factory
{
    /**
     * Factory for SolrWeb record driver.
     *
     * @param ServiceManager $sm Service manager.
     *
     * @return SolrWeb
     */
    public static function getSolrWeb(\Zend\ServiceManager\ServiceManager $sm)
    {
        $web = $sm->getServiceLocator()->get('VuFind\Config')->get('website');
        return new \Cortona\RecordDriver\SolrWeb(
            $sm->getServiceLocator()->get('VuFind\Config')->get('config'), $web, $web
        );
    }
}
```

Step 6: Finishing Touches

At this point, the goals of the project were met and the site was fully functional. The only remaining steps were cosmetic: adjusting some further templates and CSS files to introduce a consistent look and feel across the disparate components of the site. The interface unification goal was modest: providing a consistent set of navigation buttons and a search box at the top of all pages. VuFind's theme system makes these types of customizations quite simple, and no major obstacles were encountered in adjusting WordPress and Omeka to match. Obviously, maintaining a consistent style across multiple systems can be burdensome in the long term in environments where change is frequent; however, for this sort of one-off project where future redesigns are relatively unlikely, applying cosmetic changes to multiple systems was ultimately less difficult than attempting to shoehorn all of the desired functionality into a single unified system.

Conclusion

With a wide variety of indexing tools and options and an extensible architecture built on Zend Framework, VuFind is a strong candidate when a search system is needed to glue together disparate sites in a seamless fashion. The example of the Triumph of David project demonstrates some patterns that could be easily adapted to similar situations.

VuFind also boasts a strong and supportive development community. If anything in this paper remains unclear or if you encounter a use case that you are not sure how to approach, please feel free to reach out through the mailing lists or other communication channels listed on the VuFind website[19].

Acknowledgments

Thanks to David Lacy for leading the implementation of the Triumph of David project and to Chris Hallberg for his valuable input. This project would not have existed without Dave's leadership, and it would not have looked nearly as good without Chris' design talent.

References

1. For more information, see WordPress.org. Available at: https://wordpress.org/(accessed on 12 Aug 2016).
2. For more information, see the Omeka home page. Available at: https://omeka.org/(accessed on 12 Aug 2016).
3. For more information, see the Neatline home page. Available at: http://neatline.org/(accessed on 12 Aug 2016).
4. For more information, see the GeoServer home page. Available at: http://geoserver.org/(accessed on 12 Aug 2016).
5. For more information, see the VuFind home page. Available at: https://vufind.org/(accessed on 12 Aug 2016).
6. For more information, see the Zend Framework home page. Available at: https://framework.zend.com/ (accessed on 12 Aug 2016).
7. For more information, see the Apache Solr home page. Available at: http://lucene.apache.org/solr/ (accessed on 12 Aug 2016).
8. The XML sitemap standard is documented at sitemaps.org. Available at: http://www.sitemaps.org/(accessed on 12 Aug 2016).
9. For more information, see the Apache Tika home page. Available at: https://tika.apache.org/ (accessed on 12 Aug 2016).
10. VuFind Installation. Available at: https://vufind.org/wiki/-installation(accessed on 12 Aug 2016).
11. Zend\ModuleManager. Available at: https://framework.-zend.com/manual/2.4/en/modules/zend.module-manager.intro.html(accessed on 12 Aug 2016).
12. VuFind: Indexing a Website. Available at: https://vufind.-org/wiki/indexing:websites(accessed on 12 Aug 2016).
13. VuFind:Local Settings Directory. Available at: https://vu-find.org/wiki/configuration:local_settings_directory(accessed on 12 Aug 2016).
14. For more information, see PHP Data Objects. Available at: http://php.net/manual/en/book.pdo.php(accessed on 12 Aug 2016).
15. XML Messages for Updating a Solr Index. Available at: https://wiki.apache.org/solr/UpdateXmlMessages (accessed on 12 Aug 2016).
16. For more information, see Simile Widgets: Timeline. Available at: http://simile-widgets.org/timeline/ (accessed on 12 Aug 2016).
17. VuFind: User Interface Customization. Available at: https://vufind.org/wiki/development:architecture:user_interfa ce (accessed on 12 Aug 2016).
18. VuFind: Record Driver Plugins. Available at: https://vufind.org/wiki/development:plugins:record_drivers(a ccessed on 12 Aug 2016).
19. VuFind Support. Available at: http://vufind-org.github.io/-vufind/support.html(accessed on 12 Aug 2016).