VOWR
Digital Catalogue Documentation

Jacob House

Version 1
August 21, 2018

Compatibility

Operating System

The new Digital Catalogue application is written to run on Windows Server. Development was done using Windows Server 2012 R2. The same or later versions of the Windows Server platform should be used in production to ensure optimal results. Earlier versions are not tested.

The application is written in Python 3, a cross-platform programming language. Hence portability to other operating systems may work, but is not supported.

Web Application Server

The application was developed and tested using Microsoft's Internet Information Services (IIS) web server which is standard in Windows Server 2012 R2.

The application is written in Python 3, a cross-platform programming language. Hence portability to other web servers may work, but is not supported.

SQL Database

The web application should be used with a MySQL database. Other databases such as MariaDB or MongoDB may work with minimal changes to sources but these are not supported. Use with Microsoft SQL Server will require large source rewrites as the MySQL connector used is not compatible with SQL Server; a suitable substitute must be found.

Installation

Operating System and Web Application Server

Follow the standard installation procedure for the operating system.

Installation of IIS should include the following features in the Add Roles and Features wizard.

- → Web Server (IIS)
  - → Web Server
    - → Common HTTP Features
      - → Default Document
      - → Directory Browsing

- → HTTP Errors
- → Static Content

→ Health and Diagnostics

- → HTTP Logging

→ Performance

- → Static Content Compression
- → Dynamic Content Compression

→ Security

- → Request Filtering
- → Basic Authentication

→ Application Development

- → CGI

→ Management Tools

- → IIS Management Console

These roles and features may also be installed using PowerShell with the `Install-WindowsFeature` cmdlet.

```
PS> Install-WindowsFeature -Name Web-Default-Doc, Web-Dir-Browsing, `
     Web-Http-Errors, Web-Static-Content, Web-Http-Logging, `
     Web-Stat-Compression, Web-Dyn-Compression, Web-Filtering, `
     Web-Basic-Auth, Web-CGI, Web-Mgmt-Console
```

Depending on your installation you may need to specify a source.

Ensure that the operating system is configured to use Microsoft Update and to install updates automatically. Optionally schedule weekly restarts for Monday mornings (03:00 should be fine).

IIS Web Platform Installer

To easily install IIS modules required for our installation, we require the Microsoft Web Platform Installer.

Open IIS Manager (`InetMgr.exe` or `iis.msc`). From the IIS Manager Start Page, click Web Platform Installer, located under Online Resources. This will bring you to the download page for the Web Platform Installer.

After installing Web Platform Installer, restart IIS Manager. Expand the server name tab on the left and in the main pane under Management, choose Web Platform Installer.

Under Products, choose All. From the list, find URL Rewrite 2.1. Click Add and then Install. Accept the terms.

Python 3

At the time of writing, Python 3.6 is the latest stable version. This is what was used for testing.

Python may be downloaded from https://www.python.org.

When installing Python, be sure to choose the advanced installation options and then pick *Install For All Users* as well as *Add Python to PATH*.

Note the installation directory. Normally this is either

```
C:\Python3X
```

or

```
C:\Program Files\Python3X
```

Python Modules

Browse the file system to the Python installation directory. Note the executable name for Python. It may be `python.exe`, `python3.exe`, or, for version 3.6, `python3.6.exe`.

Open an administrative PowerShell instance. First we must update Pip, Python's package manager.

```
PS> python -m pip install --upgrade pip
```

Now we can begin installing the required modules.

```
PS> python -m pip install --ignore-installed flask
PS> python -m pip install --ignore-installed flask_login
PS> python -m pip install --ignore-installed flask_wtf
PS> python -m pip install --ignore-installed pymysql
PS> python -m pip install --ignore-installed wfastcgi
```

Python FastCGI

We installed wfastcgi in the last section. Now we must configure the handler.

Browse the file system to the Python installation directory. There should be a folder called `Scripts` that contains `wfastcgi.py`. Copy this Python file to the web application root. In our case, this is `C:\inetpub\wwwroot`.

Open IIS Manager. Rename "Default Web Site" to "VOWR Digital Cataloguer". Click on the site in the left sidebar. From the main pane, double click Handler Mappings. From the Actions menu on the right, choose Add Module Mapping. Use the values in Table 1.

Request Path:	*
Module:	FastCgiModule
Executable (optional):	C:\Path\To\Python.exe|C:\inetpub\wwwroot\wfastcgi.py
Name:	FlaskHandler

Table 1

Click Request Restrictions and make certain that the "Invoke handler only if request is mapped to:" checkbox is unchecked.

Click OK twice and then click Yes.

Go to the root server in IIS Manager's lefthand display. Double click on FastCGI Settings in the main pane. There should be a single entry with our Python path and the WFastCGI script. Double click this entry. Under FastCGI Properties > General, click the box to the right of "Environment Variables" that says "(Collection)". A button with an ellipsis should appear. Click this to open the entity attributes panel.

We need to add two entries to this list. These are shown in Table 2.

Name:	PYTHONPATH
Value:	C:\inetpub\wwwroot
Name:	WSGI_HANDLER
Value:	vowr.app

Table 2

MySQL

At the time of writing, the latest version of MySQL is version 8.0.12.0. The Windows installer may be downloaded from https://dev.mysql.com/get/Downloads/MySQLInstaller/mysql-installer-community-8.0.12.0.msi.

Follow the on-screen instructions to install MySQL. When choosing a setup type, select "Server Only". The other parts of the installation are useful for development however in the spirit of minimizing attack surface, we will not install them on the server.

When you have installed MySQL and are configuring it, on the Type and Networking page, for Config Type, choose "Server Computer". This will allow MySQL to use a suitable amount of memory for our application (i.e., good performance while sharing resources with IIS).

NOTE: MySQL requires that the .NET Framework 4.5.2 be installed. If this is not already installed, it may be downloaded from https://www.microsoft.com/en-us/download/details.aspx?id=42642.

Clean Up

Our installation is now complete.

Copy the source code to C:\inetpub\wwwroot.

Finally, we must ready the server for production. To do this, we remove the Desktop Experience server roles. In Server 2012 R2, we use the following PowerShell cmdlets.

```
PS> Uninstall-WindowsFeature -Name Server-Gui-Shell, `
     Server-Gui-Mgmt-Infra
PS> Restart-Computer
```

Overview

Implementation

Logic

vowr.py

```
#!/usr/bin/python3
#####################################################################
# File: vowr.py
# Written by: Jacob House
# Created: August 12, 2018
# Last Modified: "
# Modifications by: Jacob House
#
# Description: Python Flask application vowr.app for new digital
# catalogue to replace the one running on DOS
#####################################################################


# System imports
import sys
import os
import csv

# Local imports
import music_manager
import auth_manager
import db_manager
import datetime

# Flask magic
import flask
import flask_login
app = flask.Flask(__name__)
```

```
29 app.secret_key = '\xf0n\x94x\xdfK\x98\xbdN3\xb4\xd0\x1a\x1f\xd1\xd1\xf3P\xd1\xa6I~\x93@'
30 login_manager = flask_login.LoginManager()
31 login_manager.init_app(app)
32 login_manager.login_view = '/gatekeeper/sign-in' # This is dependent on the routes below
33
34 # General-use dictionary to pass things to the HTML processor
35 params = dict()
36 # Would be used to search for a page but I have them all in the navbar so this is
37 # redundant and awkward... and currently not linked to anything functional.
38 # Its location also clashes with the sign in/out buttton
39 params['searchBar'] = False
40 # Items to display in the navigation bar. I don't like having to use this dict but it ←↩
     is what it is
41 params['navbar'] = {
42     'Home':'/default',
43     'Request Song':'/request'}
44
45 # I want the index page to have a consistent URI so redirecting to that page
46 @app.route('/')
47 def rootPatge():
48     return(flask.redirect('/default', 301))
49
50 # The default splash/index page
51 @app.route('/default')
52 def defaultPage():
53     return(flask.render_template('default.htm', params=params))
54
55 # Music search page
56 @app.route('/search', methods=['POST', 'GET'])
57 def searchPage():
58     return(flask.render_template('search.htm', params=params))
59
60 # Accepts only POSTS (from JS) to help autocomplete search requests
61 # The logic in db_manager will return the first 10 matches for all
62 # full or partial matches to any and all words in the search bar...
63 # The results are NOT ordered by relevance. This is something I would
64 # like to implement but since I am using MySQL boolean full text
65 # searching, this may be difficult without a full rewrite.
66 @app.route('/search/auto', methods=['POST'])
67 def autocomplete():
68     try:
69         db = db_manager.DBQuery(flask.session.get('username'))
70     except Exception:
71         flask.abort(500)
72     searchParams, results = {'var':None, 'val':None}, []
73     for param in searchParams:
74         # First check GET... should be nothing since GET is disallowed in the route
75         searchParams[param] = flask.request.args.get(param)
76         if searchParams[param] == None:
77             # If nothing in get, try POST
78             searchParams[param] = flask.request.form.get(param)
79     results = db.autocomplete(searchParams['var'], searchParams['val'])
80     db.close()
81     return(flask.jsonify(results))
82
83 # This is used to add new entries to the database. The /append page is fairly flat...
84 # Unlike /modify, there is no logic going on here since the user must supply all values.
85 # The logic for /append is in the /append/commit backend page.
86 @app.route('/append')
```

```
87 # @flask_login.login_required
88 def insertPage():
89     return(flask.render_template('append.htm', params=params))
90
91
92 # Here is where the logic for /append happens. If the user tries to append an entry ↩
      that exists
93 # (or is similar), we will ask if they want to modify X and if so, send to the /modify ↩
      page
94 # with the song ID in the request so that /modify can pull up the song from the DB and ↩
      have the
95 # properties ready for edit.
96 # This function will send back (JSON?) for a delta of the changes.
97 # Ex. <green>[<date>-<time>] <user> added <song> by <artist> from <albnum>.</green>
98 # Ex. <red>[<date>-<time>] <user> FAILED to add <song> by <artist> from <albnum>. ↩
      Please try again.</red>
99 @app.route('/append/commit', methods=['POST'])
100 def commitAppend():
101     ret = list()
102     if flask.request.method == 'POST' and flask.request.form.get('song_id') != None:
103         # We need a delta between the new and old so that we can return what the ↩
              changes are
104         # Get the old
105         try:
106             db = db_manager.DBQuery(flask.session.get('username'))
107         except Exception:
108             flask.abort(500)
109         timestamp = '[' + datetime.datetime.now().strftime('%Y-%m-%d_%H:%M:%S') + ']'
110         ## BODY HERE
111         db.close()
112     return(flask.jsonify(ret))
113
114 # This page needs to get a POST of the song entry that the user is trying to edit.
115 # If no song_id in POST, just print the instructions.
116 # Otherwise, pull up the song from the DB by its ID and give the user access to edit it.
117 # We really should record when and by whom the last edit was made. Option to undo??
118 @app.route('/modify', methods=['POST', 'GET'])
119 # @flask_login.login_required
120 def modifyPage():
121     if flask.request.method == 'POST':
122         if flask.request.form.get('song_id') != None:
123             # We have a song. Retrieve it from the DB and return the info
124             try:
125                 db = db_manager.DBQuery(flask.session.get('username'))
126             except Exception:
127                 flask.abort(500)
128             # getSongById will throw a 404 if the entity is not found
129             params['entity'] = ↩
                    music_manager.Song(db.getSongById(flask.request.form.get('song_id')))
130             db.close()
131     return(flask.render_template('modify.htm', params=params))
132
133 @app.route('/modify/commit', methods=['POST'])
134 def commitModify():
135     ret = list()
136     if flask.request.method == 'POST' and flask.request.form.get('song_id') != None:
137         # We want a delta between the new and old so that we can return what the ↩
              changes are
138         # Get the old
```

```
        try:
            db = db_manager.DBQuery(flask.session.get('username'))
            old = db.getSongById(flask.request.form.get('song_id')) # Dict
        except Exception:
            db.close()
            flask.abort(500)
        timestamp = '[' + datetime.datetime.now().strftime('%Y-%m-%d_%H:%M:%S') + ']'
        changes = dict()
        if old['name'] != flask.request.form.get('song'):
            changes['name'] = flask.request.form.get('song')
        if old['album_code'] != flask.request.form.get('album'):
            artist_id = db.getArtistIdByName(flask.request.form.get('artist'), create=True)
            changes['album'] = db.getAlbumIdByCode(flask.request.form.get('album'), artist_id, create=True)
        if old['artist_name'] != flask.request.form.get('artist'):
            changes['artist_id'] = artist_id
        if old['genre'] != flask.request.form.get('genre'):
            changes['genre'] = flask.request.form.get('genre')
        if old['canadian'] == (flask.request.form.get('genre') == 'y'):
            changes['canadian'] = flask.request.form.get('genre') == 'y'
        errs = db.processSongChanges(changes) # returns dict of failed items. empty dict is good
        succs = sorted(list(set(changes.keys()) - set(errs.keys())))
        if succs:
            msg = '<div class="success-msg">' + timestamp + ' '
            for success in succs:
                msg += success + ' changed ("' + str(old[success]) + '" to "' + str(changes[success]) + '"), '
            msg = msg[:-2] + '</div>'
            ret.append(msg)
        if errs:
            msg = '<div class="err-msg">' + timestamp + ' '
            for err in errs:
                msg += err + ' failed ("' + old[err] + '" to "' + err[err] + '"), '
            msg = msg[:-2] + '</div>'
            ret.append(msg)
        db.close()
    return(flask.jsonify(ret))


@app.route('/playlists', methods=['GET', 'POST'])
def playlistPage():
    try:
        db = db_manager.DBQuery(flask.session.get('username'))
    except Exception:
        flask.abort(500)
    params['playlist'] = None
    params['existingPlaylists'] = db.getPlaylists()
    db.close()
    return(flask.render_template('playlists.htm', params=params))

@app.route('/playlists/edit', methods=['GET', 'POST'])
def playlistEdit():
    if flask.request.method == 'POST' and flask.request.form.get('playlist'):
        try:
            db = db_manager.DBQuery(flask.session.get('username'))
        except Exception:
```

```
194             flask.abort(500)
195         params['playlistId'] = flask.request.form.get('playlist')
196         params['playlistName'] = db.getPlaylistNameById(params['playlist'])
197 
198         db.close()
199         return(flask.render_template('playlists_edit.htm', params=params))
200     else:
201         return(flask.redirect(flask.url_for('playlistPage')))
202 
203 @app.route('/admin')
204 def adminRootPage():
205     return(flask.redirect('/admin/home', 301))
206 
207 @app.route('/admin/<page>')
208 #@flask_login.fresh_login_required
209 def adminPage(page):
210     params['page'] = page
211     if params['admin']:
212         params['subnavbar'] = {
213              'Home':'/admin/home',
214             'Users':'/admin/users',
215             'Database':'/admin/dbadmin',
216             'Setup':'/admin/setup'}
217     if params['page'] == 'home':
218         return(flask.render_template('admin_home.htm', params=params))
219     elif params['admin'] and params['page'] == 'users':
220         return(flask.render_template('admin_users.htm', params=params))
221     elif params['admin'] and params['page'] == 'dbadmin':
222         return(flask.render_template('admin_dbadmin.htm', params=params))
223     elif params['admin'] and params['page'] == 'setup':
224         try:
225             db = db_manager.DBQuery()
226             db.createTables()
227             params['setupDone'] = True
228         except:
229             params['setupDone'] = False
230         return(flask.render_template('admin_setup.htm', params=params))
231     else:
232         flask.abort(401)
233 
234 @app.route('/acknowledgements')
235 def acknowledgementsPage():
236     params['title'] = 'About The VOWR Digital Catalogue'
237     params['documentTitle'] = 'About'
238     return(flask.render_template('acknowledgements.htm', params=params))
239 
240 @app.route('/gatekeeper/<action>', methods=['POST', 'GET'])
241 def gatekeeper(action):
242     params['action'] = action
243     if params['action'] == 'sign-in':
244         if flask.request.method == 'POST':
245             username = flask.request.form.get('username', None)
246             if username != None:
247                 # try: u = UserClass(id=auth(uname, pwd)) then pass u to login_user?
248                 flask.flash("Logged in!")
249                 flask.session['username'] = username
250                 flask_login.login_user(username) # needs a user OBJECT
251         ret = flask.render_template('gatekeeper_sign-in.htm', params=params)
252     elif params['action'] == 'sign-out':
```

```
        flask_login.logout_user()
        ret = flask.render_template('gatekeeper_sign-out.htm', params=params)
    elif params['action'] == 'forgot':
        ret = flask.render_template('gatekeeper_forgot.htm', params=params)
    else:
        ret = flask.redirect('/default')
    return(ret)

@app.route('/export')
def exportQuery():
    flask.abort(404)

@app.route('/request')
def requestPage():
    return(flask.render_template('vowr_template.htm', params=params))









# @login_manager.user_loader
# def load_user(user_id):
# return flask_login.User.get(user_id)

@app.before_request
def preflight():
    params['route'] = flask.request.path
    if True:#flask_login.current_user.is_authenticated:
        params['navbar']['Search'] = '/search'
        params['navbar']['Playlists'] = '/playlists'
        params['navbar']['Append'] = '/append'
        params['navbar']['Modify'] = '/modify'
        params['navbar']['Admin'] = '/admin'
    params['admin'] = True # DEV

@app.after_request
def add_header(r):
    """
    Add headers to both force latest IE rendering engine or Chrome Frame,
    and also to cache the rendered page for 10 minutes.
    """
    r.headers["Cache-Control"] = "no-cache, no-store, must-revalidate"
    r.headers["Pragma"] = "no-cache"
    r.headers["Expires"] = "0"
    r.headers['Cache-Control'] = 'public, max-age=0'
    return(r)

if __name__ == "__main__":
    app.run(host='10.57.140.53', debug=True)
```

AUTH_MANAGER.PY

```
307 import flask
308 import flask_login
309 import db_manager
310
311 class User(flask_login.UserMixin):
312     __slots__ = [
313         '_id',
314         '_username'
315     ]
316
317     def __init__(self, username, plaintextPassword):
318         # Check if user with same username exists
319         pass
320
321     def __eq__(self, other):
322         return(self._id == other._id)
323
324 def newUser(username, password, createdBy, isAdmin=False):
325     db = db_manager.DBQuery()
326     db.cursor.execute("SELECT id,username FROM users WHERE username LIKE %s", (username,))
327     results = db.cursor.fetchall()
328     if len(results) > 0:
329         # Users with that username already exist.
330         raise AssertionError('Username exists already: ' + str(results[0][1]) + ':' + ↩
              str(results[0][0]))
331
332     pass
333
334 def isAdmin():
335     return(True)
336     # if not flask_login.current_user.is_authenticated:
337     # return(False)
338     # else:
339     # if False:
340     # return(True)
341     # else:
342     # return(False)
343
344 def canEdit():
345     return(True)
346     # if not flask_login.current_user.is_authenticated:
347     # return(False)
348     # else:
349     # if False:
350     # return(True)
351     # else:
352     # return(False)
```

DB_MANAGER.PY

```
353 # System inports
354 import pymysql
355 pymysql.install_as_MySQLdb()
356 import re,os
357 from collections import OrderedDict
```

```
358 import flask
359
360 # Local imports
361 import music_manager
362 import xmlconf
363
364 ##############################################################################
365 # This class is used to deal with all querying and updating
366 # to the database for a particular user using a
367 # single connection and cursor to the database
368 ##############################################################################
369 class DBQuery:
370     """General class that allows you to Query the database and as data is fetched from ↩
        the database, an
371     internal data structure acts like a cache and as data is accessed it is stored in ↩
        this data structure """
372
373
374     ##########################################################################
375     # self.conn is MySQL connection object to the sys_config database
376     # self.cursor is the handler to the sys_config database
377     # self.username is the user who is responsible for the database connection
378     #
379     # close() should be called when an instance of this class is done with in
380     # order to cleanly kill of the connection with the sys_config database
381     ##########################################################################
382     def __init__(self, user=None):
383         self.conn = None
384         self.cursor = None
385         self._user = user
386         self.connect()
387
388     ##########################################################################
389     # This method connects to the database as a high user who has read
390     # and write privileges by using information in a config file on
391     # arlene to do the binding.
392     #
393     # After this method is called the instance variables self.conn and
394     # self.cursor of this class should be initialized and ready to be used.
395     ##########################################################################
396     def connect(self):
397         try:
398             curr_hosts,curr_user,curr_passwd,curr_db = ('','','','')
399             curr_hosts = xmlconf.getConfValue('../vowr.conf', 'mysql_hosts')
400             curr_user = xmlconf.getConfValue('../vowr.conf', 'mysql_user')[0]
401             curr_db = xmlconf.getConfValue('../vowr.conf', 'database')[0]
402             curr_passwd = xmlconf.getConfValue('../vowr.conf', 'mysql_pw')[0]
403         except:
404             raise AssertionError("Could not retrieve DB config from XML file.")
405
406         if (curr_user == None or curr_passwd == None or curr_db == None or curr_hosts ↩
            == None):
407             raise AssertionError("Config file did not have all required "
408                 + "connection info")
409
410         if len(curr_hosts) == 0:
411             raise AssertionError("Config file did not specify db hosts")
412
413         self.conn = None
```

```
414         self.cursor = None
415         # self.username = curr_user
416 
417         # Connect to the first available server in the list
418         # Note that on the computer that hosts the web tools that
419         # manage the master database the configuration file must
420         # contain only the master database host name along with the
421         # password and username of the database administrator so that
422         # the webtool programs can connect and make changes as necessary.
423         for host in curr_hosts:
424             try:
425                 self.conn = pymysql.connect(host=host, user=curr_user,
426                                             passwd=curr_passwd, db=curr_db)
427                 self.cursor = self.conn.cursor()
428                 break
429             except pymysql.Error as e:
430                 if self.conn != None:
431                         self.conn.close()
432                 self.conn = None
433                 continue
434         # End for
435 
436         if self.cursor == None:
437             if self.conn != None:
438                 self.conn.close()
439             raise pymysql.Error("MySQL Connection Error")
440     # End self.connect()
441 
442     ########################################################################
443     # Closes the connection to the sys_config database used by the given
444     # instance of this class.
445     ########################################################################
446     def close(self):
447         if self.cursor != None:
448             self.cursor.close()
449         if self.conn != None:
450             self.conn.close()
451 
452     def __del__(self):
453         self.close()
454 
455     ################################################################
456     # Returns a list of all possible entity types.
457     #
458     # Returns:
459     # ent_types a list of all possible entity types
460     ################################################################
461     # def get_ent_types(self):
462     # ent_types = []
463     # try:
464     # self.cursor.execute("DESCRIBE entity")
465     # results = self.cursor.fetchall()
466     # for row in results:
467     # if row[0] == 'type':
468     #  raw_ent_types = row[1]
469     #  raw_ent_types = raw_ent_types[5:-1]
470     #  raw_ent_types = re.sub("'","",raw_ent_types)
471     #  ent_types = raw_ent_types.split(',')
472     # except Exception as e:
```

```
473     # raise Exception("Error retrieving entity types: " + str(e))
474     # for index in range(0,len(ent_types)):
475     # if ent_types[index] == 'association':
476     # del ent_types[index]
477     # break
478     # return ent_types
479     #End get_ent_types()
480
481
482     ########################################################################
483     # Returns a list of all used variable values in the ent_var_vals
484     # table for the given variable of the given entity type.
485     #
486     # Parameters:
487     # ent_type the entity type of the given variable to search for all
488     #  used variable values for
489     # var the variable of the given entity type to search for all used
490     #  variable values for
491     #
492     # Returns:
493     # used_var_vals a list of all used variable values in the
494     #  ent_var_vals table for the given variable of the given
495     #  entity type
496     ########################################################################
497     def FindMatchingEntities(self,ent_type,pairs):
498         """
499            ###
500            # Find all entities that match the entity type and pairs,
501            # pairs is a python dictionary of key:value pairs that correspond
502            # to variables and their values
503            ###
504         """
505         entities = None
506         for var in pairs.keys():
507             try:
508                 self.cursor.execute("""
509                         SELECT DISTINCT entity.name
510                         FROM entity,ent_var_vals
511                         WHERE entity.type=%s AND entity.ent_id=ent_var_vals.ent_id
512                         AND ent_var_vals.var=%s AND ent_var_vals.val=%s
513                         """
514                         , (ent_type,var,pairs[var]))
515                 res = self.cursor.fetchall()
516             except pymysql.Error as e:
517                 raise pymysql.Error("MySQL Error retrieving all values of variable '"
518                                                         + str(var) + "' for entity ↩
                                                            type '"
519                                                         + str(ent_type) + "': " + ↩
                                                            e.args[1])
520             except Exception as e:
521                 raise Exception("MySQL Error retrieving all values of variable '" + ↩
                    str(e) + "'")
522
523             newEntities = {}
524             for row in res:
525                 if (entities == None) or entities.haskey(row[0]):
526                     newEntities[row[0]] = 1
527             # End for
528             entities = newEntities
```

```
529         # End for
530         return entities.keys()
531     # Ens FindMatchingEntities()
532 
533     # def get_all_used_var_vals(self, ent_type, var):
534     # try:
535     # self.cursor.execute("""
536     # SELECT DISTINCT ent_var_vals.val
537     # FROM entity,ent_var_vals
538     # WHERE entity.type=%s AND entity.ent_id=ent_var_vals.ent_id
539     # AND ent_var_vals.var=%s
540     # """
541     # , (ent_type,var))
542     # res = self.cursor.fetchall()
543     # except pymysql.Error as e:
544     # raise pymysql.Error("MySQL Error retrieving all values of variable '"
545     #     + str(var) + "' for entity type '"
546     #     + str(ent_type) + "': " + e.args[1])
547     # except Exception as e:
548     # raise Exception("MySQL Error retrieving all values of variable '"
549     #     + str(var) + "' for entity type '"
550     #     + str(ent_type) + "': " + e.args[1])
551     # used_var_vals = []
552     # for val_row in res:
553     # used_var_vals.append(val_row[0])
554     # used_var_vals.sort(key=str.lower)
555     # return used_var_vals
556     # # End get_all_used_var_vals()
557 
558 
559     ###################################################################
560     # Returns a list of the variable value for the given entity name
561     # of the given entity type for the given variable, or an empty
562     # list, i.e. [], if the variable doesn't exist for the entity
563     #
564     # Parameters:
565     # ent_type the entity type of the given entity name to get
566     #  the given variable values for
567     # ent_name the entity name of the given entity type to get
568     #  the given variable values for
569     # var the variable to get the variable values for the given
570     #  entity name of the given entity type for
571     #
572     # Returns:
573     # val_list a list of all the variable value for the given
574     #  variable for the given entity_type of the given entity
575     #  name, or an empty list, i.e. [], if the variable doesn't
576     #  exist for the entity
577     ###################################################################
578     def GetEntityVarVals(self, ent_name, ent_type, var):
579         try:
580             self.cursor.execute("""
581                 SELECT val FROM ent_var_vals,entity
582                 WHERE name=%s AND type=%s AND status='active'
583                 AND ent_var_vals.ent_id=entity.ent_id
584                 AND ent_var_vals.var=%s ORDER BY indx
585                 """
586                 , (ent_name,ent_type,var))
587             res = self.cursor.fetchall()
```

```
588             except pymysql.Error as e:
589                 raise pymysql.Error("MySQL Error retrieving " + str(var)
590                             + " value for host " + str(ent_name) + ": "
591                             + e.args[1])
592             except Exception as e:
593                 raise Exception("Error retrieving " + str(var)
594                             + " value for host " + str(ent_name) + ": "
595                             + str(e))
596             val_list = []
597             for val_row in res:
598                 val_list.append(val_row[0])
599             return val_list
600         # End GetEntityVarVals()
601
602
603         ######################################################################
604         # Returns a list of all the (variable,value) pairs as size 2
605         # tuples for the given entity name of the given entity type.
606         #
607         # Parameters:
608         # ent_type the entity type of the given entity name to get
609         #  all (variable,value) pairs for
610         # ent_name the entity name of the given entity type to get
611         #  all (variable,value) pairs for
612         #
613         # Returns:
614         # var_val_list a list of all the (variable,value) pairs as size 2
615         #  tuples for the given entity name of the given entity type,
616         #  this is an empty list if no such pairs exist
617         ######################################################################
618         # def get_ent_all_var_vals(self,ent_type,ent_name):
619         # try:
620         # self.cursor.execute("""
621         # SELECT var,val
622         # FROM ent_var_vals,entity
623         # WHERE ent_var_vals.ent_id=entity.ent_id
624         # AND entity.type=%s AND entity.name=%s
625         # ORDER BY var,val, indx
626         # """
627         # , (ent_type,ent_name))
628         # var_val_list = self.cursor.fetchall()
629         # except Exception as e:
630         # raise Exception("Failed to get all variable/values for entity '"
631         #    + ent_name + "' of type '" + ent_type + "':"
632         #    + str(e))
633         # return var_val_list
634         # # End get_ent_all_var_vals()
635
636         ############################################################################
637         # Set up an associative array of all variable values for a particular
638         # entity of a particular type.
639         #
640         # Parameters:
641         # ent_type the entity type of the entity to get all (variable,value)
642         #  pairs for
643         # ent_name the entity name of the entity to get all (variable,value)
644         #  pairs for
645         #
646         # Returns:
```

```
647     # table: an associative array of all variables as key and values as a list
648     # of values for the given entity name of the given entity type
649     ###########################################################################
650     def get_ent_vals_table(self, ent_type, ent_name):
651         try:
652             self.cursor.execute("""
653                 SELECT var, val
654                 FROM entity, ent_var_vals
655                 WHERE type='%s' and name='%s' and entity.ent_id=ent_var_vals.ent_id
656                 """
657                 % (ent_type, ent_name))
658             res = self.cursor.fetchall()
659         except Exception as e:
660             raise Exception("Error retrieving all entity variable values for '" +
661                                     ent_name + "' of type '" + ent_type +
662                                     "': " + str(e))
663         table = {}
664         for item in res:
665             val = item[1]
666             while val.rfind('"') != -1:
667                 val = str.replace(val,'"','&quot;')
668             if item[0] in table:
669                 table[item[0]].append(val)
670             else:
671                 table[item[0]] = [val]
672         return table
673     # End get_ent_vals_table()
674
675
676     def createTables(self):
677         tables = [
678             ['users',''' CREATE TABLE users(`id` INT AUTO_INCREMENT PRIMARY KEY,
679                                     `givenname` VARCHAR(255) NOT NULL,
680                                     `surname` VARCHAR(255) NOT NULL,
681                                     `username` VARCHAR(32) NOT NULL,
682                                     UNIQUE(`username`),
683                                     `password` CHAR(64) NOT NULL,
684                                     `edit` BIT(1) NOT NULL DEFAULT 0,
685                                     `admin` BIT(1) NOT NULL DEFAULT 0,
686                                     `lockout_count` INT DEFAULT 0
687                                     )'''],
688             ['artists',''' CREATE TABLE artists(`id` INT AUTO_INCREMENT PRIMARY KEY,
689                                         `name` VARCHAR(255) NOT NULL,
690                                         UNIQUE(`name`),
691                                         FULLTEXT(`name`)
692                                         )'''],
693             ['albums',''' CREATE TABLE albums(`id` INT AUTO_INCREMENT PRIMARY KEY,
694                                         `code` VARCHAR(10) NOT NULL,
695                                         UNIQUE(`code`),
696                                         `artist_id` INT,
697                                         FULLTEXT(`code`),
698                                         `name` VARCHAR(255) DEFAULT NULL,
699                                         FOREIGN KEY (`artist_id`) REFERENCES ↩
                                                `artists`(`id`)
700                                         )'''],
701             ['songs',''' CREATE TABLE songs(`id` INT AUTO_INCREMENT PRIMARY KEY,
702                                     `song_title` VARCHAR(255) NOT NULL,
703                                     `artist_id` INT DEFAULT NULL,
704                                     `album_id` INT DEFAULT NULL,
```

```
                                    `media` VARCHAR(10) DEFAULT NULL,
                                    `category` VARCHAR(75) DEFAULT NULL,
                                    `side` INT NOT NULL,
                                    `track` INT NOT NULL,
                                    `select` VARCHAR(10) DEFAULT NULL,
                                    `performance_type` VARCHAR(255) DEFAULT NULL,
                                    `canadian` BIT(1) NOT NULL DEFAULT 0,
                                    `display` BIT(1) NOT NULL DEFAULT 1,
                                    FULLTEXT(`song_title`),
                                    FOREIGN KEY (`album_id`) REFERENCES ↩
                                       `albums`(`id`),
                                    FOREIGN KEY (`artist_id`) REFERENCES ↩
                                       `artists`(`id`)
                                    )'''],
            ['playlists','''CREATE TABLE playlists(`id` INT AUTO_INCREMENT PRIMARY KEY,
                                          `user_id` INT NOT NULL,
                                          `name` VARCHAR(255) NOT NULL,
                                          FOREIGN KEY (`user_id`) REFERENCES ↩
                                             `users`(`id`),
                                          UNIQUE ↩
                                             `user_playlist_comb`(`user_id`, ↩
                                             `name`)
                                          )'''],
            ['playlist_songs','''CREATE TABLE playlist_songs(`id` INT AUTO_INCREMENT ↩
               PRIMARY KEY,
                                                    `playlist_id` INT NOT NULL,
                                                    `song_id` INT NOT NULL,
                                                    UNIQUE(`song_id`),
                                                    `position` INT NOT NULL,
                                                    UNIQUE ↩
                                                       `song_index`(`playlist_id`, ↩
                                                       `position`),
                                                    FOREIGN KEY ↩
                                                       (`playlist_id`) ↩
                                                       REFERENCES ↩
                                                       `playlists`(`id`),
                                                    FOREIGN KEY (`song_id`) ↩
                                                       REFERENCES ↩
                                                       `songs`(`id`)
                                                    )''']
        ]
        try:
            self.cursor.execute('''
            SHOW TABLES
            ''')
            existingTables = self.cursor.fetchall()
        except Exception as e:
            raise Exception("Failed to get all tables: " + str(e))
        existingTables = [row[0] for row in existingTables]
        for table in tables:
            # We will assume that if the table exists then it is in the right format
            if table[0] not in existingTables:
                try:
                    print('Creating table: ' + table[0])
                    self.cursor.execute(table[1])
                except Exception as e:
                    raise Exception("Failed to create table: " + table[0] + ", " + ↩
                       str(e))
```

```
750
751     def importCSVDB(self, file):
752         pass
753
754     def autocomplete(self, var, val):
755         LIMIT = 10
756         val = self.conn.escape_string(val)
757         val = ' '.join(['+' + x + '*' for x in re.findall(r'[a-zA-Z0-9]+', val)])
758         print(val)
759         if var == 'artist':
760             # query = "SELECT country FROM countries WHERE MATCH (country) AGAINST ←↩
                ('" + val + "' IN BOOLEAN MODE) LIMIT " + str(LIMIT) # Testing with ←↩
                country DB
761             self.cursor.execute("SELECT name FROM artists WHERE MATCH (name) AGAINST ←↩
                (%s IN BOOLEAN MODE) LIMIT %s", (val, LIMIT))
762         elif var == 'album':
763             self.cursor.execute("SELECT num FROM albums WHERE MATCH (num) AGAINST (%s ←↩
                IN BOOLEAN MODE) LIMIT %s", (val, LIMIT))
764         elif var == 'song':
765             self.cursor.execute("SELECT name FROM songs WHERE display = 1 AND MATCH ←↩
                (name) AGAINST (%s IN BOOLEAN MODE) LIMIT %s", (val, LIMIT))
766         elif var == 'category':
767             self.cursor.execute("SELECT category FROM songs WHERE display = 1 AND ←↩
                MATCH (category) AGAINST (%s IN BOOLEAN MODE) LIMIT %s", (val, LIMIT))
768         else:
769             return([])
770         results = self.cursor.fetchall()
771         results = [x[0] for x in results]
772         print(var, val, results)
773         return(results)
774
775     def getSongById(self, id):
776         ret = dict()
777         id = str(id).strip()
778         id = self.conn.escape_string(id)
779         id = re.sub(r'[^0-9]+', '', id)
780         self.cursor.execute("SELECT ←↩
                id,song_title,artist_id,album_id,media,category,side,track,select,performance_type,canadian,select ←
                FROM songs WHERE display = 1 AND id = %s", (id,))
781         results = self.cursor.fetchall()
782         if len(results) != 1:
783             flask.abort(404)
784         ret['id'] = results[0][0]
785         ret['song_title'] = results[0][1]
786         ret['artist_id'] = results[0][2]
787         ret['album_id'] = results[0][3]
788         ret['media'] = results[0][4]
789         ret['category'] = results[0][5]
790         ret['side'] = results[0][6]
791         ret['track'] = results[0][7]
792         ret['select'] = results[0][8]
793         ret['performance_type'] = results[0][9]
794         ret['canadian'] = results[0][10] == b'\x00'
795         songObj = music_manager.Song(ret)
796         return(songObj)
797
798     def getAlbumNumById(self, id):
799         id = str(id).strip()
800         id = self.conn.escape_string(id)
```

```
801         id = re.sub(r'[^0-9]+', '', id)
802         self.cursor.execute("SELECT num FROM albums WHERE id = %s", (id,))
803         results = self.cursor.fetchall()
804         if len(results) < 1:
805             return(None)
806         elif len(results) > 1:
807             flask.abort(500)
808         return(results[0][0])
809 
810     def getArtistNameById(self, id):
811         id = id.strip()
812         id = self.conn.escape_string(str(id))
813         id = re.sub(r'[^0-9]+', '', id)
814         self.cursor.execute("SELECT name FROM artists WHERE id = %s", (id,))
815         results = self.cursor.fetchall()
816         if len(results) < 1:
817             return(None)
818         elif len(results) > 1:
819             raise AssertionError('DB response indicates duplicate in unique key')
820         else:
821             return(results[0][0])
822 
823     def getPlaylistNameById(self, id):
824         id = id.strip()
825         id = self.conn.escape_string(str(id))
826         id = re.sub(r'[^0-9]+', '', id)
827         self.cursor.execute("SELECT name FROM playlists WHERE id = %s", (id,))
828         results = self.cursor.fetchall()
829         if len(results) < 1:
830             return(None)
831         elif len(results) > 1:
832             raise AssertionError('DB response indicates duplicate in unique key')
833         else:
834             return(results[0][0])
835 
836     def getArtistIdByName(self, name, create=False):
837         if name.strip() == str():
838             return(None)
839         name = name.strip()
840         name = self.conn.escape_string(name)
841         name = re.sub(r'[^a-zA-z0-9,& ]+', '', name)
842         self.cursor.execute("SELECT id FROM artists WHERE name LIKE %s", (name,))
843         results = self.cursor.fetchall()
844         if len(results) < 1:
845             if create:
846                 self.cursor.execute("INSERT INTO artists (name) values (%s)", (name,))
847                 return(self.getArtistIdByName(name, create))
848             else:
849                 flask.abort(404)
850         elif len(results) > 1:
851             raise AssertionError('DB response indicates duplicate in unique key')
852         else:
853             return(results[0][0])
854 
855     def getAlbumIdByNum(self, num, artist_id=None, create=False):
856         if num.strip() == str():
857             return(None)
858         num = num.strip()
859         num = self.conn.escape_string(num)
```

```
860         num = re.sub(r'[^a-zA-z0-9,& ]+', '', code)
861         self.cursor.execute("SELECT id FROM albums WHERE num LIKE %s", (num,))
862         results = self.cursor.fetchall()
863         if len(results) < 1:
864             if create and artist_id == None:
865                 self.cursor.execute("INSERT INTO albums (num) values (%s)", (num,))
866                 return(self.getAlbumIdByCode(code, create))
867             if create and artist_id != None:
868                 if not artist_id.isdigit() or self.getArtistNameById(artist_id) == None:
869                     raise AssertionError('Invalid artist_id: ' + str(artist_id))
870                 self.cursor.execute("INSERT INTO albums (num, artist_id) values (%s, ↩
                       %s)", (num, artist_id))
871                 return(self.getAlbumIdByCode(num, artist_id, create))
872             else:
873                 flask.abort(404)
874         elif len(results) > 1:
875             raise AssertionError('DB response indicates duplicate in unique key')
876         else:
877             return(results[0][0])
878 
879     def processSongChanges(self, changes):
880         # changes should be {'field':[old, new]}
881         pass
882 
883     def getPlaylists(self):
884         # Get playlists belonging to self.username
885         user_id = self._user.id
886         self.cursor.execute("SELECT id,name FROM playlists WHERE user_id = %s", ↩
               (user_id,))
887         results = self.cursor.fetchall()
888         ret = {row[0]:row[1] for row in results}
889         return(ret)
890 
891     def getPlaylistById(self, id):
892         # Returns Playlist object
893         name = getPlaylistNameById(id)
894         self.cursor.execute("SELECT song_id,position FROM playlist_songs WHERE ↩
               playlist_id = %s", (id,))
895         result = self.cursor.fetchall()
896         songs = {row[1]:getSongById(row[0]) for row in result}
897         ret = music_manager.Playlist(id, name, songs)
898         return(ret)
899 
900 #main() function for testing methods in the DBQuery class
901 def main():
902     # print 'Demo of all the methods of the DBQuery class'
903     dbquery = DBQuery()
904     # print '\nDemo of method: dbquery.get_ent_types()'
905     # print str(dbquery.get_ent_types())
906     # print '\nDemo of method: dbquery.FindMatchingEntities()'
907     # pairs = {'servers':'amazon'}
908     # print str(dbquery.FindMatchingEntities('client', pairs ))
909     # print '\nDemo of method: dbquery.get_all_used_var_vals()'
910     # print str(dbquery.get_all_used_var_vals('client', 'servers'))
911     # print '\nDemo of method: dbquery.GetEntityVarVals()'
912     # print str(dbquery.GetEntityVarVals('tiny', 'server', 'functions'))
913     # print '\nDemo of method: dbquery.get_ent_all_var_vals()'
914     # print str(dbquery.get_ent_all_var_vals('printer', 'icomprt1'))
915     # print '\nDemo of method: dbquery.get_ent_vals_table()'
```

```
    # print str(dbquery.get_ent_vals_table('printer', 'icomprt1'))
    dbquery.close()
    return 0

if __name__ == '__main__':
    main()
```

MUSIC_MANAGER.PY

```
# All information for these classes should be passed through the constructors.
# Objects of these classes are going to be passed back from DBManager so I don't
# want this module calling back to DBManager.

class Song:
    # We may (will) have a LOT of these objects made for searching or
    # doing the initial setup so __slots__ will make us MUCH more efficient
    __slots__ = [
        '_id',
        '_song_title',
        '_artist', # I want this to be an object, not text
        '_album', # I want this to be an object, not text
        '_media', # Legacy? No idea what this is
        '_category',
        '_side',
        '_track',
        '_select', # Legacy? No idea what this is
        '_performance_type',
        '_theme',
        '_canadian'
    ]

    def __init__(self, properties):
        if type(properties) == type(dict()):
            # If the key in question is missing, we get an exception
            # This is on purpose; all of the fields are MANDATORY
            self._id = properties['id']
            self._song_title = properties['song_title']
            self._artist = Artist(properties['artist_id'])
            self._album = Album(properties['album_id'])
            self._media = properties['media']
            self._category = properties['category']
            self._side = properties['side']
            self._track = properties['track']
            self._select = properties['select']
            self._performance_type = properties['performance_type']
            self._canadian = properties['canadian']
        elif type(properties) == type(self):
            # This will be our "deep copy"
            self._id = properties._id
            self._song_title = properties._song_title
            self._artist = properties._artist
            self._album = properties._album
            self._media = properties._media
            self._category = properties._category
            self._side = properties._side
            self._track = properties._track
```

```
970             self._select = properties._select
971             self._performance_type = properties._performance_type
972             self._canadian = properties._canadian
973         else:
974             raise TypeError('Song.__init__ takes dict or Song.')
975 
976     @property
977     def id(self):
978         return(self._id)
979 
980 
981 
982 
983     def __sub__(self, other):
984         # returns {field:[old, new]}
985         pass
986 
987 
988 class Artist:
989     __slots__ = [
990         '_id',
991         '_artist_name'
992     ]
993 
994     def __init__(self, artist_id, artist_name):
995         if type(artist_id) != type(int()) or type(artist_name) != type(str()):
996             raise TypeError('Artist.__init__ takes int of artist_id and str of ↩
                artist_id.')
997         self._id = artist_id
998         self._artist_name = artist_name
999 
1000    def __str__(self):
1001        return(self._artist_name)
1002 
1003    @property
1004    def artist_name(self):
1005        return(self._artist_name)
1006 
1007    @property
1008    def id(self):
1009        return(self._id)
1010 
1011 
1012 class Album:
1013    __slots__ = [
1014        '_id',
1015        '_album_number'
1016    ]
1017 
1018    def __init__(self, album_id, album_num):
1019        if type(album_id) != type(int()) or type(album_num) != type(str()):
1020            raise TypeError('Album.__init__ takes int of album_id and str of ↩
                album_name.')
1021        self._id = album_id
1022        self._album_number = album_num
1023 
1024    def __str__(self):
1025        return(self._album_number)
1026 
```

```
1027     @property
1028     def album_number(self):
1029         return(self._album_number)
1030 
1031     @property
1032     def id(self):
1033         return(self._id)
1034 
1035 class Playlist:
1036     __slots__ = [
1037         '_id',
1038         '_playlist_name',
1039         '_songs'
1040     ]
1041 
1042     def __init__(self, playlist_id, playlist_name, songs):
1043         if type(playlist_id) != type(int()) or type(playlist_name) != type(str()) or ↩
                type(songs) != type(dict()):
1044             raise TypeError('Album.__init__ takes int of playlist_id and str of ↩
                    playlist_name and dict of songs in format position:Song.')
1045         self._id = playlist_id
1046         self._playlist_name = playlist_name
1047         self._songs = songs
1048 
1049     @property
1050     def playlist_name(self):
1051         return(self._playlist_name)
1052 
1053     @property
1054     def id(self):
1055         return(self._id)
1056 
1057     @property
1058     def songs(self):
1059         return(dict(self._songs))
1060 
1061     @property
1062     def song_ids(self):
1063         return(sorted(self._songs.keys()))
1064 
1065     def __str__(self):
1066         return(self._playlist_name)
```

XMLCONF.PY

```
1067 import xml.parsers.expat, os
1068 
1069 configs = {}
1070 
1071 def getConfValue(confName, variable):
1072         global configs
1073         try:
1074                 info = os.stat(confName)
1075         except OSError as e:
1076                 raise AssertionError('Missing Config file: %s' % (e,))
1077         modTime = info.st_mtime
```

```
1078         config = None
1079         if confName in configs:
1080                 config = configs[confName]
1081                 if config and config.modtime >= modTime:
1082                         config = None
1083         if not config:
1084                 config = Config(confName, modTime)
1085                 configs[confName] = config
1086         if variable in config.varVals:
1087                 return config.varVals[variable]
1088 #End getConfValue()
1089
1090 class Config:
1091         def __init__(self, confName, modTime):
1092                 self.modtime = modTime
1093                 self.values = []
1094                 self.variable = ''
1095                 self.varVals = {}
1096                 self.inAP = self.inAN = self.inCD = self.inVR = self.inVL = False
1097                 try:
1098                         f = open(confName, 'r')
1099                         buf = f.read()
1100                 except IOError as e:
1101                         raise AssertionError('Missing Config file(%s): %s', ↩
                            (confName, e))
1102                 p = xml.parsers.expat.ParserCreate('ASCII')
1103                 p.StartElementHandler = self.start_element
1104                 p.EndElementHandler = self.end_element
1105                 p.CharacterDataHandler = self.char_data
1106                 p.Parse(buf, 1)
1107         # End __init()
1108
1109         # 3 handler functions
1110         def start_element(self, name, attrs):
1111                 if name == 'Entity_Profile':
1112                         self.inAP = True
1113                 elif name == 'Application' and self.inAP:
1114                         self.inAN = True
1115                 elif name == 'ConfigurationItem' and self.inAP:
1116                         self.inCD = True
1117                 elif name == 'variable' and self.inAP and self.inCD:
1118                         self.inVR = True
1119                 elif name == 'value' and self.inAP and self.inCD:
1120                         self.inVL = True
1121                 else:
1122                         raise AssertionError('Unexpected XML start element: ' + name)
1123
1124         def end_element(self, name):
1125                 if name == 'Entity_Profile':
1126                         self.inAP = False
1127                 elif name == 'Application' and self.inAP:
1128                         self.inAN = False
1129                 elif name == 'ConfigurationItem' and self.inAP:
1130                         if self.variable == '':
1131                                 raise AssertionError('Missing variable')
1132                         elif self.values == []:
1133                                 raise AssertionError('Missing values')
1134                         self.inCD = False
1135                         self.varVals[self.variable] = self.values
```

```
                        self.variable = ''
                        self.values = []
                elif name == 'variable' and self.inAP and self.inCD:
                        self.inVR = False
                elif name == 'value' and self.inAP and self.inCD:
                        self.inVL = False
                else:
                        raise AssertionError('Unexpected XML end element: ' + name)

        def char_data(self, data):
                if self.inAN:
                        file = data
                elif self.inVR:
                        self.variable = str(data)
                elif self.inVL:
                        self.values.append(str(data))
# End class Config

# print(getConfValue('vowr.conf', 'database'))
```

WFASTCGI.PY

```
# Python Tools for Visual Studio
# Copyright(c) Microsoft Corporation
# All rights reserved.
#
# Licensed under the Apache License, Version 2.0 (the License); you may not use
# this file except in compliance with the License. You may obtain a copy of the
# License at http://www.apache.org/licenses/LICENSE-2.0
#
# THIS CODE IS PROVIDED ON AN *AS IS* BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY
# IMPLIED WARRANTIES OR CONDITIONS OF TITLE, FITNESS FOR A PARTICULAR PURPOSE,
# MERCHANTABLITY OR NON-INFRINGEMENT.
#
# See the Apache Version 2.0 License for specific language governing
# permissions and limitations under the License.
from __future__ import absolute_import, print_function, with_statement

__author__ = "Microsoft Corporation <ptvshelp@microsoft.com>"
__version__ = "3.0.0"

import ctypes
import datetime
import os
import re
import struct
import sys
import traceback
from xml.dom import minidom

try:
    from cStringIO import StringIO
    BytesIO = StringIO
except ImportError:
    from io import StringIO, BytesIO
```

```
1189 try:
1190     from thread import start_new_thread
1191 except ImportError:
1192     from _thread import start_new_thread
1193 
1194 if sys.version_info[0] == 3:
1195     def to_str(value):
1196         return value.decode(sys.getfilesystemencoding())
1197 else:
1198     def to_str(value):
1199         return value.encode(sys.getfilesystemencoding())
1200 
1201 
1202 # http://www.fastcgi.com/devkit/doc/fcgi-spec.html#S3
1203 
1204 FCGI_VERSION_1 = 1
1205 FCGI_HEADER_LEN = 8
1206 
1207 FCGI_BEGIN_REQUEST = 1
1208 FCGI_ABORT_REQUEST = 2
1209 FCGI_END_REQUEST = 3
1210 FCGI_PARAMS = 4
1211 FCGI_STDIN = 5
1212 FCGI_STDOUT = 6
1213 FCGI_STDERR = 7
1214 FCGI_DATA = 8
1215 FCGI_GET_VALUES = 9
1216 FCGI_GET_VALUES_RESULT = 10
1217 FCGI_UNKNOWN_TYPE = 11
1218 FCGI_MAXTYPE = FCGI_UNKNOWN_TYPE
1219 
1220 FCGI_NULL_REQUEST_ID = 0
1221 
1222 FCGI_KEEP_CONN = 1
1223 
1224 FCGI_RESPONDER = 1
1225 FCGI_AUTHORIZER = 2
1226 FCGI_FILTER = 3
1227 
1228 FCGI_REQUEST_COMPLETE = 0
1229 FCGI_CANT_MPX_CONN = 1
1230 FCGI_OVERLOADED = 2
1231 FCGI_UNKNOWN_ROLE = 3
1232 
1233 FCGI_MAX_CONNS = "FCGI_MAX_CONNS"
1234 FCGI_MAX_REQS = "FCGI_MAX_REQS"
1235 FCGI_MPXS_CONNS = "FCGI_MPXS_CONNS"
1236 
1237 class FastCgiRecord(object):
1238     """Represents a FastCgiRecord. Encapulates the type, role, flags. Holds
1239     onto the params which we will receive and update later."""
1240     def __init__(self, type, req_id, role, flags):
1241         self.type = type
1242         self.req_id = req_id
1243         self.role = role
1244         self.flags = flags
1245         self.params = {}
1246 
1247     def __repr__(self):
```

```
1248         return '<FastCgiRecord(%d, %d, %d, %d)>' % (self.type,
1249                                                     self.req_id,
1250                                                     self.role,
1251                                                     self.flags)
1252
1253 #typedef struct {
1254 # unsigned char version;
1255 # unsigned char type;
1256 # unsigned char requestIdB1;
1257 # unsigned char requestIdB0;
1258 # unsigned char contentLengthB1;
1259 # unsigned char contentLengthB0;
1260 # unsigned char paddingLength;
1261 # unsigned char reserved;
1262 # unsigned char contentData[contentLength];
1263 # unsigned char paddingData[paddingLength];
1264 #} FCGI_Record;
1265
1266 class _ExitException(Exception):
1267     pass
1268
1269 if sys.version_info[0] >= 3:
1270     # indexing into byte strings gives us an int, so
1271     # ord is unnecessary on Python 3
1272     def ord(x):
1273         return x
1274     def chr(x):
1275         return bytes((x, ))
1276
1277     def wsgi_decode(x):
1278         return x.decode('iso-8859-1')
1279     def wsgi_encode(x):
1280         return x.encode('iso-8859-1')
1281
1282     def fs_encode(x):
1283         return x
1284
1285     def exception_with_traceback(exc_value, exc_tb):
1286         return exc_value.with_traceback(exc_tb)
1287
1288     zero_bytes = bytes
1289 else:
1290     # Replace the builtin open with one that supports an encoding parameter
1291     from codecs import open
1292
1293     def wsgi_decode(x):
1294         return x
1295     def wsgi_encode(x):
1296         return x
1297
1298     def fs_encode(x):
1299         return x if isinstance(x, str) else x.encode(sys.getfilesystemencoding())
1300
1301     def exception_with_traceback(exc_value, exc_tb):
1302         # x.with_traceback() is not supported on 2.x
1303         return exc_value
1304
1305     bytes = str
1306
```

```
1307     def zero_bytes(length):
1308         return '\x00' * length
1309 
1310 def read_fastcgi_record(stream):
1311     """reads the main fast cgi record"""
1312     data = stream.read(8) # read record
1313     if not data:
1314         # no more data, our other process must have died...
1315         raise _ExitException()
1316 
1317     fcgi_ver, reqtype, req_id, content_size, padding_len, _ = struct.unpack('>BBHHBB', ←↩
         data)
1318 
1319     content = stream.read(content_size) # read content
1320     stream.read(padding_len)
1321 
1322     if fcgi_ver != FCGI_VERSION_1:
1323         raise Exception('Unknown fastcgi version %s' % fcgi_ver)
1324 
1325     processor = REQUEST_PROCESSORS.get(reqtype)
1326     if processor is not None:
1327         return processor(stream, req_id, content)
1328 
1329     # unknown type requested, send response
1330     log('Unknown request type %s' % reqtype)
1331     send_response(stream, req_id, FCGI_UNKNOWN_TYPE, chr(reqtype) + zero_bytes(7))
1332     return None
1333 
1334 
1335 def read_fastcgi_begin_request(stream, req_id, content):
1336     """reads the begin request body and updates our _REQUESTS table to include
1337     the new request"""
1338     # typedef struct {
1339     # unsigned char roleB1;
1340     # unsigned char roleB0;
1341     # unsigned char flags;
1342     # unsigned char reserved[5];
1343     # } FCGI_BeginRequestBody;
1344 
1345     # TODO: Ignore request if it exists
1346     res = FastCgiRecord(
1347         FCGI_BEGIN_REQUEST,
1348         req_id,
1349         (ord(content[0]) << 8) | ord(content[1]), # role
1350         ord(content[2]), # flags
1351     )
1352     _REQUESTS[req_id] = res
1353 
1354 def read_encoded_int(content, offset):
1355     i = struct.unpack_from('>B', content, offset)[0]
1356 
1357     if i < 0x80:
1358         return offset + 1, i
1359 
1360     return offset + 4, struct.unpack_from('>I', content, offset)[0] & ~0x80000000
1361 
1362 
1363 def read_fastcgi_keyvalue_pairs(content, offset):
1364     """Reads a FastCGI key/value pair stream"""
```

```
1365
1366        offset, name_len = read_encoded_int(content, offset)
1367        offset, value_len = read_encoded_int(content, offset)
1368
1369        name = content[offset:(offset + name_len)]
1370        offset += name_len
1371
1372        value = content[offset:(offset + value_len)]
1373        offset += value_len
1374
1375        return offset, name, value
1376
1377
1378    def get_encoded_int(i):
1379        """Writes the length of a single name for a key or value in a key/value
1380        stream"""
1381        if i <= 0x7f:
1382            return struct.pack('>B', i)
1383        elif i < 0x80000000:
1384            return struct.pack('>I', i | 0x80000000)
1385        else:
1386            raise ValueError('cannot encode value %s (%x) because it is too large' % (i, i))
1387
1388
1389    def write_fastcgi_keyvalue_pairs(pairs):
1390        """Creates a FastCGI key/value stream and returns it as a byte string"""
1391        parts = []
1392        for raw_key, raw_value in pairs.items():
1393            key = wsgi_encode(raw_key)
1394            value = wsgi_encode(raw_value)
1395
1396            parts.append(get_encoded_int(len(key)))
1397            parts.append(get_encoded_int(len(value)))
1398            parts.append(key)
1399            parts.append(value)
1400
1401        return bytes().join(parts)
1402
1403    # Keys in this set will be stored in the record without modification but with a
1404    # 'wsgi.' prefix. The original key will have the decoded version.
1405    # (Following mod_wsgi from http://wsgi.readthedocs.org/en/latest/python3.html)
1406    RAW_VALUE_NAMES = {
1407        'SCRIPT_NAME' : 'wsgi.script_name',
1408        'PATH_INFO' : 'wsgi.path_info',
1409        'QUERY_STRING' : 'wsgi.query_string',
1410        'HTTP_X_ORIGINAL_URL' : 'wfastcgi.http_x_original_url',
1411    }
1412
1413    def read_fastcgi_params(stream, req_id, content):
1414        if not content:
1415            return None
1416
1417        offset = 0
1418        res = _REQUESTS[req_id].params
1419        while offset < len(content):
1420            offset, name, value = read_fastcgi_keyvalue_pairs(content, offset)
1421            name = wsgi_decode(name)
1422            raw_name = RAW_VALUE_NAMES.get(name)
1423            if raw_name:
```

```
            res[raw_name] = value
        res[name] = wsgi_decode(value)


def read_fastcgi_input(stream, req_id, content):
    """reads FastCGI std-in and stores it in wsgi.input passed in the
    wsgi environment array"""
    res = _REQUESTS[req_id].params
    if 'wsgi.input' not in res:
        res['wsgi.input'] = content
    else:
        res['wsgi.input'] += content

    if not content:
        # we've hit the end of the input stream, time to process input...
        return _REQUESTS[req_id]


def read_fastcgi_data(stream, req_id, content):
    """reads FastCGI data stream and publishes it as wsgi.data"""
    res = _REQUESTS[req_id].params
    if 'wsgi.data' not in res:
        res['wsgi.data'] = content
    else:
        res['wsgi.data'] += content


def read_fastcgi_abort_request(stream, req_id, content):
    """reads the wsgi abort request, which we ignore, we'll send the
    finish execution request anyway..."""
    pass


def read_fastcgi_get_values(stream, req_id, content):
    """reads the fastcgi request to get parameter values, and immediately
    responds"""
    offset = 0
    request = {}
    while offset < len(content):
        offset, name, value = read_fastcgi_keyvalue_pairs(content, offset)
        request[name] = value

    response = {}
    if FCGI_MAX_CONNS in request:
        response[FCGI_MAX_CONNS] = '1'

    if FCGI_MAX_REQS in request:
        response[FCGI_MAX_REQS] = '1'

    if FCGI_MPXS_CONNS in request:
        response[FCGI_MPXS_CONNS] = '0'

    send_response(
        stream,
        req_id,
        FCGI_GET_VALUES_RESULT,
        write_fastcgi_keyvalue_pairs(response)
    )

```

```
1483
1484 # Our request processors for different FastCGI protocol requests. Only those
1485 # requests that we receive are defined here.
1486 REQUEST_PROCESSORS = {
1487     FCGI_BEGIN_REQUEST : read_fastcgi_begin_request,
1488     FCGI_ABORT_REQUEST : read_fastcgi_abort_request,
1489     FCGI_PARAMS : read_fastcgi_params,
1490     FCGI_STDIN : read_fastcgi_input,
1491     FCGI_DATA : read_fastcgi_data,
1492     FCGI_GET_VALUES : read_fastcgi_get_values
1493 }
1494
1495 APPINSIGHT_CLIENT = None
1496
1497 def log(txt):
1498     """Logs messages to a log file if WSGI_LOG env var is defined."""
1499     if APPINSIGHT_CLIENT:
1500         try:
1501             APPINSIGHT_CLIENT.track_event(txt)
1502         except:
1503             pass
1504
1505     log_file = os.environ.get('WSGI_LOG')
1506     if log_file:
1507         with open(log_file, 'a+', encoding='utf-8') as f:
1508             txt = txt.replace('\r\n', '\n')
1509             f.write('%s: %s%s' % (datetime.datetime.now(), txt, '' if ↩
                txt.endswith('\n') else '\n'))
1510
1511 def maybe_log(txt):
1512     """Logs messages to a log file if WSGI_LOG env var is defined, and does not
1513     raise exceptions if logging fails."""
1514     try:
1515         log(txt)
1516     except:
1517         pass
1518
1519 def send_response(stream, req_id, resp_type, content, streaming=True):
1520     """sends a response w/ the given id, type, and content to the server.
1521     If the content is streaming then an empty record is sent at the end to
1522     terminate the stream"""
1523     if not isinstance(content, bytes):
1524         raise TypeError("content must be encoded before sending: %r" % content)
1525
1526     offset = 0
1527     while True:
1528         len_remaining = max(min(len(content) - offset, 0xFFFF), 0)
1529
1530         data = struct.pack(
1531             '>BBHHBB',
1532             FCGI_VERSION_1, # version
1533             resp_type, # type
1534             req_id, # requestIdB1:B0
1535             len_remaining, # contentLengthB1:B0
1536             0, # paddingLength
1537             0, # reserved
1538         ) + content[offset:(offset + len_remaining)]
1539
1540         offset += len_remaining
```

```
        os.write(stream.fileno(), data)
        if len_remaining == 0 or not streaming:
            break
    stream.flush()

def get_environment(dir):
    web_config = os.path.join(dir, 'Web.config')
    if not os.path.exists(web_config):
        return {}

    d = {}
    doc = minidom.parse(web_config)
    config = doc.getElementsByTagName('configuration')
    for configSection in config:
        appSettings = configSection.getElementsByTagName('appSettings')
        for appSettingsSection in appSettings:
            values = appSettingsSection.getElementsByTagName('add')
            for curAdd in values:
                key = curAdd.getAttribute('key')
                value = curAdd.getAttribute('value')
                if key and value is not None:
                    d[key.strip()] = value
    return d

ReadDirectoryChangesW = ctypes.windll.kernel32.ReadDirectoryChangesW
ReadDirectoryChangesW.restype = ctypes.c_uint32
ReadDirectoryChangesW.argtypes = [
    ctypes.c_void_p, # HANDLE hDirectory
    ctypes.c_void_p, # LPVOID lpBuffer
    ctypes.c_uint32, # DWORD nBufferLength
    ctypes.c_uint32, # BOOL bWatchSubtree
    ctypes.c_uint32, # DWORD dwNotifyFilter
    ctypes.POINTER(ctypes.c_uint32), # LPDWORD lpBytesReturned
    ctypes.c_void_p, # LPOVERLAPPED lpOverlapped
    ctypes.c_void_p # LPOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine
]
try:
    from _winapi import (CreateFile, CloseHandle, GetLastError, ExitProcess,
                         WaitForSingleObject, INFINITE, OPEN_EXISTING)
except ImportError:
    CreateFile = ctypes.windll.kernel32.CreateFileW
    CreateFile.restype = ctypes.c_void_p
    CreateFile.argtypes = [
        ctypes.c_wchar_p, # lpFilename
        ctypes.c_uint32, # dwDesiredAccess
        ctypes.c_uint32, # dwShareMode
        ctypes.c_void_p, # LPSECURITY_ATTRIBUTES,
        ctypes.c_uint32, # dwCreationDisposition,
        ctypes.c_uint32, # dwFlagsAndAttributes,
        ctypes.c_void_p # hTemplateFile
    ]

    CloseHandle = ctypes.windll.kernel32.CloseHandle
    CloseHandle.argtypes = [ctypes.c_void_p]

    GetLastError = ctypes.windll.kernel32.GetLastError
    GetLastError.restype = ctypes.c_uint32

```

```
    ExitProcess = ctypes.windll.kernel32.ExitProcess
    ExitProcess.restype = ctypes.c_void_p
    ExitProcess.argtypes = [ctypes.c_uint32]

    WaitForSingleObject = ctypes.windll.kernel32.WaitForSingleObject
    WaitForSingleObject.argtypes = [ctypes.c_void_p, ctypes.c_uint32]
    WaitForSingleObject.restype = ctypes.c_uint32

    OPEN_EXISTING = 3
    INFINITE = -1

FILE_LIST_DIRECTORY = 1
FILE_SHARE_READ = 0x00000001
FILE_SHARE_WRITE = 0x00000002
FILE_SHARE_DELETE = 0x00000004
FILE_FLAG_BACKUP_SEMANTICS = 0x02000000
MAX_PATH = 260
FILE_NOTIFY_CHANGE_LAST_WRITE = 0x10
ERROR_NOTIFY_ENUM_DIR = 1022
INVALID_HANDLE_VALUE = 0xFFFFFFFF

class FILE_NOTIFY_INFORMATION(ctypes.Structure):
    _fields_ = [('NextEntryOffset', ctypes.c_uint32),
                ('Action', ctypes.c_uint32),
                ('FileNameLength', ctypes.c_uint32),
                ('Filename', ctypes.c_wchar)]

_ON_EXIT_TASKS = None
def run_exit_tasks():
    global _ON_EXIT_TASKS
    maybe_log("Running on_exit tasks")
    while _ON_EXIT_TASKS:
        tasks, _ON_EXIT_TASKS = _ON_EXIT_TASKS, []
        for t in tasks:
            try:
                t()
            except Exception:
                maybe_log("Error in exit task: " + traceback.format_exc())

def on_exit(task):
    global _ON_EXIT_TASKS
    if _ON_EXIT_TASKS is None:
        _ON_EXIT_TASKS = tasks = []
        try:
            evt = int(os.getenv('_FCGI_SHUTDOWN_EVENT_'))
        except (TypeError, ValueError):
            maybe_log("Could not wait on event %s" % os.getenv('_FCGI_SHUTDOWN_EVENT_'))
        else:
            def _wait_for_exit():
                WaitForSingleObject(evt, INFINITE)
                run_exit_tasks()
                ExitProcess(0)

            start_new_thread(_wait_for_exit, ())
    _ON_EXIT_TASKS.append(task)

def start_file_watcher(path, restart_regex):
    if restart_regex is None:
        restart_regex = ".*((\\.py)|(\\.config))$"
```

```
    elif not restart_regex:
        # restart regex set to empty string, no restart behavior
        return

    def enum_changes(path):
        """Returns a generator that blocks until a change occurs, then yields
        the filename of the changed file.

        Yields an empty string and stops if the buffer overruns, indicating that
        too many files were changed."""

        buffer = ctypes.create_string_buffer(32 * 1024)
        bytes_ret = ctypes.c_uint32()

        try:
            the_dir = CreateFile(
                path,
                FILE_LIST_DIRECTORY,
                FILE_SHARE_READ | FILE_SHARE_WRITE | FILE_SHARE_DELETE,
                0,
                OPEN_EXISTING,
                FILE_FLAG_BACKUP_SEMANTICS,
                0,
            )
        except OSError:
            maybe_log("Unable to create watcher")
            return

        if not the_dir or the_dir == INVALID_HANDLE_VALUE:
            maybe_log("Unable to create watcher")
            return

        while True:
            ret_code = ReadDirectoryChangesW(
                the_dir,
                buffer,
                ctypes.sizeof(buffer),
                True,
                FILE_NOTIFY_CHANGE_LAST_WRITE,
                ctypes.byref(bytes_ret),
                None,
                None,
            )

            if ret_code:
                cur_pointer = ctypes.addressof(buffer)
                while True:
                    fni = ctypes.cast(cur_pointer, ↩
                       ctypes.POINTER(FILE_NOTIFY_INFORMATION))
                    # FileName is not null-terminated, so specifying length is mandatory.
                    filename = ctypes.wstring_at(cur_pointer + 12, ↩
                       fni.contents.FileNameLength // 2)
                    yield filename
                    if fni.contents.NextEntryOffset == 0:
                        break
                    cur_pointer = cur_pointer + fni.contents.NextEntryOffset
            elif GetLastError() == ERROR_NOTIFY_ENUM_DIR:
                CloseHandle(the_dir)
                yield ''
```

```
1716                return
1717            else:
1718                CloseHandle(the_dir)
1719                return
1720
1721    log('wfastcgi.py will restart when files in %s are changed: %s' % (path, ↩
           restart_regex))
1722    def watcher(path, restart):
1723        for filename in enum_changes(path):
1724            if not filename:
1725                log('wfastcgi.py exiting because the buffer was full')
1726                run_exit_tasks()
1727                ExitProcess(0)
1728            elif restart.match(filename):
1729                log('wfastcgi.py exiting because %s has changed, matching %s' % ↩
                       (filename, restart_regex))
1730                # we call ExitProcess directly to quickly shutdown the whole process
1731                # because sys.exit(0) won't have an effect on the main thread.
1732                run_exit_tasks()
1733                ExitProcess(0)
1734
1735    restart = re.compile(restart_regex)
1736    start_new_thread(watcher, (path, restart))
1737
1738def get_wsgi_handler(handler_name):
1739    if not handler_name:
1740        raise Exception('WSGI_HANDLER env var must be set')
1741
1742    if not isinstance(handler_name, str):
1743        handler_name = to_str(handler_name)
1744
1745    module_name, _, callable_name = handler_name.rpartition('.')
1746    should_call = callable_name.endswith('()')
1747    callable_name = callable_name[:-2] if should_call else callable_name
1748    name_list = [(callable_name, should_call)]
1749    handler = None
1750    last_tb = ''
1751
1752    while module_name:
1753        try:
1754            handler = __import__(module_name, fromlist=[name_list[0][0]])
1755            last_tb = ''
1756            for name, should_call in name_list:
1757                handler = getattr(handler, name)
1758                if should_call:
1759                    handler = handler()
1760            break
1761        except ImportError:
1762            module_name, _, callable_name = module_name.rpartition('.')
1763            should_call = callable_name.endswith('()')
1764            callable_name = callable_name[:-2] if should_call else callable_name
1765            name_list.insert(0, (callable_name, should_call))
1766            handler = None
1767            last_tb = ': ' + traceback.format_exc()
1768
1769    if handler is None:
1770        raise ValueError('"%s" could not be imported%s' % (handler_name, last_tb))
1771
1772    return handler
```

```
def read_wsgi_handler(physical_path):
    global APPINSIGHT_CLIENT
    env = get_environment(physical_path)
    os.environ.update(env)
    for path in (v for k, v in env.items() if k.lower() == 'pythonpath'):
        # Expand environment variables manually.
        expanded_path = re.sub(
            '%(\\w+?)%',
            lambda m: os.getenv(m.group(1), ''),
            path
        )
        sys.path.extend(fs_encode(p) for p in expanded_path.split(';') if p)

    handler = get_wsgi_handler(os.getenv("WSGI_HANDLER"))
    instr_key = os.getenv("APPINSIGHTS_INSTRUMENTATIONKEY")
    if instr_key:
        try:
            # Attempt the import after updating sys.path - sites must
            # include applicationinsights themselves.
            from applicationinsights.requests import WSGIApplication
        except ImportError:
            maybe_log("Failed to import applicationinsights: " + traceback.format_exc())
        else:
            handler = WSGIApplication(instr_key, handler)
            APPINSIGHT_CLIENT = handler.client
            # Ensure we will flush any remaining events when we exit
            on_exit(handler.client.flush)

    return env, handler

class handle_response(object):
    """A context manager for handling the response. This will ensure that
    exceptions in the handler are correctly reported, and the FastCGI request is
    properly terminated.
    """

    def __init__(self, stream, record, get_output, get_errors):
        self.stream = stream
        self.record = record
        self._get_output = get_output
        self._get_errors = get_errors
        self.error_message = ''
        self.fatal_errors = False
        self.physical_path = ''
        self.header_bytes = None
        self.sent_headers = False

    def __enter__(self):
        record = self.record
        record.params['wsgi.input'] = BytesIO(record.params['wsgi.input'])
        record.params['wsgi.version'] = (1, 0)
        record.params['wsgi.url_scheme'] = 'https' if record.params.get('HTTPS', ↩
            '').lower() == 'on' else 'http'
        record.params['wsgi.multiprocess'] = True
        record.params['wsgi.multithread'] = False
        record.params['wsgi.run_once'] = False

```

```
        self.physical_path = record.params.get('APPL_PHYSICAL_PATH', os.path.dirname(__file__))

        if 'HTTP_X_ORIGINAL_URL' in record.params:
            # We've been re-written for shared FastCGI hosting, so send the
            # original URL as PATH_INFO.
            record.params['PATH_INFO'] = record.params['HTTP_X_ORIGINAL_URL']
            record.params['wsgi.path_info'] = record.params['wfastcgi.http_x_original_url']

        # PATH_INFO is not supposed to include the query parameters, so remove them
        record.params['PATH_INFO'] = record.params['PATH_INFO'].partition('?')[0]
        record.params['wsgi.path_info'] = record.params['wsgi.path_info'].partition(wsgi_encode('?'))[0]

        return self

    def __exit__(self, exc_type, exc_value, exc_tb):
        # Send any error message on FCGI_STDERR.
        if exc_type and exc_type is not _ExitException:
            error_msg = "%s:\n\n%s\n\nStdOut: %s\n\nStdErr: %s" % (
                self.error_message or 'Error occurred',
                ''.join(traceback.format_exception(exc_type, exc_value, exc_tb)),
                self._get_output(),
                self._get_errors(),
            )
            if not self.header_bytes or not self.sent_headers:
                self.header_bytes = wsgi_encode('Status: 500 Internal Server Error\r\n')
            self.send(FCGI_STDERR, wsgi_encode(error_msg))
            # Best effort at writing to the log. It's more important to
            # finish the response or the user will only see a generic 500
            # error.
            maybe_log(error_msg)

        # End the request. This has to run in both success and failure cases.
        self.send(FCGI_END_REQUEST, zero_bytes(8), streaming=False)

        # Remove the request from our global dict
        del _REQUESTS[self.record.req_id]

        # Suppress all exceptions unless requested
        return not self.fatal_errors

    @staticmethod
    def _decode_header(key, value):
        if not isinstance(key, str):
            key = wsgi_decode(key)
        if not isinstance(value, str):
            value = wsgi_decode(value)
        return key, value

    def start(self, status, headers, exc_info=None):
        """Starts sending the response. The response is ended when the context
        manager exits."""
        if exc_info:
            try:
                if self.sent_headers:
                    # We have to re-raise if we've already started sending data.
                    raise exception_with_traceback(exc_info[1], exc_info[2])
```

```
1886                finally:
1887                    exc_info = None
1888            elif self.header_bytes:
1889                raise Exception('start_response has already been called')
1890
1891            if not isinstance(status, str):
1892                status = wsgi_decode(status)
1893            header_text = 'Status: %s\r\n' % status
1894            if headers:
1895                header_text += ''.join('%s: %s\r\n' % handle_response._decode_header(*i) ↩
                       for i in headers)
1896            self.header_bytes = wsgi_encode(header_text + '\r\n')
1897
1898            return lambda content: self.send(FCGI_STDOUT, content)
1899
1900        def send(self, resp_type, content, streaming=True):
1901            '''Sends part of the response.'''
1902            if not self.sent_headers:
1903                if not self.header_bytes:
1904                    raise Exception("start_response has not yet been called")
1905
1906                self.sent_headers = True
1907                send_response(self.stream, self.record.req_id, FCGI_STDOUT, self.header_bytes)
1908                self.header_bytes = None
1909
1910            return send_response(self.stream, self.record.req_id, resp_type, content, ↩
                   streaming)
1911
1912    _REQUESTS = {}
1913
1914    def main():
1915        initialized = False
1916        log('wfastcgi.py %s started' % __version__)
1917        log('Python version: %s' % sys.version)
1918
1919        try:
1920            fcgi_stream = sys.stdin.detach() if sys.version_info[0] >= 3 else sys.stdin
1921            try:
1922                import msvcrt
1923                msvcrt.setmode(fcgi_stream.fileno(), os.O_BINARY)
1924            except ImportError:
1925                pass
1926
1927            while True:
1928                record = read_fastcgi_record(fcgi_stream)
1929                if not record:
1930                    continue
1931
1932                errors = sys.stderr = sys.__stderr__ = record.params['wsgi.errors'] = ↩
                       StringIO()
1933                output = sys.stdout = sys.__stdout__ = StringIO()
1934
1935                with handle_response(fcgi_stream, record, output.getvalue, errors.getvalue) ↩
                       as response:
1936                    if not initialized:
1937                        log('wfastcgi.py %s initializing' % __version__)
1938
1939                        os.chdir(response.physical_path)
1940                        sys.path[0] = '.'
```

```
1941
1942                    # Initialization errors should be treated as fatal.
1943                    response.fatal_errors = True
1944                    response.error_message = 'Error occurred while reading WSGI handler'
1945                    env, handler = read_wsgi_handler(response.physical_path)
1946
1947                    response.error_message = 'Error occurred starting file watcher'
1948                    start_file_watcher(response.physical_path, ←
                           env.get('WSGI_RESTART_FILE_REGEX'))
1949
1950                    # Enable debugging if possible. Default to local-only, but
1951                    # allow a web.config to override where we listen
1952                    ptvsd_secret = env.get('WSGI_PTVSD_SECRET')
1953                    if ptvsd_secret:
1954                        ptvsd_address = (env.get('WSGI_PTVSD_ADDRESS') or ←
                               'localhost:5678').split(':', 2)
1955                        try:
1956                            ptvsd_port = int(ptvsd_address[1])
1957                        except LookupError:
1958                            ptvsd_port = 5678
1959                        except ValueError:
1960                            log('"%s" is not a valid port number for debugging' % ←
                                   ptvsd_address[1])
1961                            ptvsd_port = 0
1962
1963                        if ptvsd_address[0] and ptvsd_port:
1964                            try:
1965                                import ptvsd
1966                            except ImportError:
1967                                log('unable to import ptvsd to enable debugging')
1968                            else:
1969                                addr = ptvsd_address[0], ptvsd_port
1970                                ptvsd.enable_attach(secret=ptvsd_secret, address=addr)
1971                                log('debugging enabled on %s:%s' % addr)
1972
1973                    response.error_message = ''
1974                    response.fatal_errors = False
1975
1976                    log('wfastcgi.py %s initialized' % __version__)
1977                    initialized = True
1978
1979                os.environ.update(env)
1980
1981                # SCRIPT_NAME + PATH_INFO is supposed to be the full path
1982                # (http://www.python.org/dev/peps/pep-0333/) but by default
1983                # (http://msdn.microsoft.com/en-us/library/ms525840(v=vs.90).aspx)
1984                # IIS is sending us the full URL in PATH_INFO, so we need to
1985                # clear the script name here
1986                if 'AllowPathInfoForScriptMappings' not in os.environ:
1987                    record.params['SCRIPT_NAME'] = ''
1988                    record.params['wsgi.script_name'] = wsgi_encode('')
1989
1990                # correct SCRIPT_NAME and PATH_INFO if we are told what our ←
                       SCRIPT_NAME should be
1991                if 'SCRIPT_NAME' in os.environ and ←
                       record.params['PATH_INFO'].lower().startswith(os.environ['SCRIPT_NAME'].lower()):
1992                    record.params['SCRIPT_NAME'] = os.environ['SCRIPT_NAME']
1993                    record.params['PATH_INFO'] = ←
                           record.params['PATH_INFO'][len(record.params['SCRIPT_NAME']):]
```

```
                        record.params['wsgi.script_name'] = wsgi_encode(record.params['SCRIPT_NAME'])
                        record.params['wsgi.path_info'] = wsgi_encode(record.params['PATH_INFO'])

                    # Send each part of the response to FCGI_STDOUT.
                    # Exceptions raised in the handler will be logged by the context
                    # manager and we will then wait for the next record.

                    result = handler(record.params, response.start)
                    try:
                        for part in result:
                            if part:
                                response.send(FCGI_STDOUT, part)
                    finally:
                        if hasattr(result, 'close'):
                            result.close()
        except _ExitException:
            pass
        except Exception:
            maybe_log('Unhandled exception in wfastcgi.py: ' + traceback.format_exc())
        except BaseException:
            maybe_log('Unhandled exception in wfastcgi.py: ' + traceback.format_exc())
            raise
        finally:
            run_exit_tasks()
            maybe_log('wfastcgi.py %s closed' % __version__)

def _run_appcmd(args):
    from subprocess import check_call, CalledProcessError

    if len(sys.argv) > 1 and os.path.isfile(sys.argv[1]):
        appcmd = sys.argv[1:]
    else:
        appcmd = [os.path.join(os.getenv('SystemRoot'), 'system32', 'inetsrv', 'appcmd.exe')]

    if not os.path.isfile(appcmd[0]):
        print('IIS configuration tool appcmd.exe was not found at', appcmd, file=sys.stderr)
        return -1

    args = appcmd + args
    try:
        return check_call(args)
    except CalledProcessError as ex:
        print(''''An error occurred running the command:

%r

Ensure your user has sufficient privileges and try again.''' % args, file=sys.stderr)
        return ex.returncode

def enable():
    executable = '"' + sys.executable + '"' if ' ' in sys.executable else sys.executable
    quoted_file = '"' + __file__ + '"' if ' ' in __file__ else __file__
    res = _run_appcmd([
        "set", "config", "/section:system.webServer/fastCGI",
```

```
2048            "/+[fullPath='" + executable + "', arguments='" + quoted_file + "', ←
                    signalBeforeTerminateSeconds='30']"
2049        ])
2050
2051        if res == 0:
2052            print('"%s|%s" can now be used as a FastCGI script processor' % (executable, ←
                    quoted_file))
2053        return res
2054
2055    def disable():
2056        executable = '"' + sys.executable + '"' if ' ' in sys.executable else sys.executable
2057        quoted_file = '"' + __file__ + '"' if ' ' in __file__ else __file__
2058        res = _run_appcmd([
2059            "set", "config", "/section:system.webServer/fastCGI",
2060            "/-[fullPath='" + executable + "', arguments='" + quoted_file + "', ←
                    signalBeforeTerminateSeconds='30']"
2061        ])
2062
2063        if res == 0:
2064            print('"%s|%s" is no longer registered for use with FastCGI' % (executable, ←
                    quoted_file))
2065        return res
2066
2067    if __name__ == '__main__':
2068        main()
```

HTML

templates/vowr_template.htm

```
2069    <!DOCTYPE html>
2070    <html>
2071        <head>
2072            <title>
2073    {% block title %}
2074    {% endblock %}
2075            </title>
2076            <meta http-equiv="Content-Language" content="en-ca">
2077            <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
2078            <link rel="stylesheet" href="/static/bootstrap/css/bootstrap.min.css">
2079            <link rel="stylesheet" href="/static/style/font-awesome.min.css">
2080            <link rel="stylesheet" href="/static/style/jqueryui.css">
2081            <link rel="stylesheet" href="/static/style/vowr.css">
2082            <link rel="stylesheet" href="/static/style/bg.css">
2083            <link rel="stylesheet" href="/static/style/menu.css">
2084    {% block style %}
2085    {% endblock %}
2086            <script type="text/javascript" src="/static/scripts/jquery-1.9.1.min.js"></script>
2087            <script type="text/javascript" src="/static/scripts/jquery-ui.min.js"></script>
2088            <script src="/static/bootstrap/js/bootstrap.min.js"></script>
2089    {% block scripts %}
2090    {% endblock %}
2091            <link rel="apple-touch-icon-precomposed" sizes="57x57" ←
                    href="/static/assets/apple-touch-icon-57x57.png" />
```

```
2092          <link rel="apple-touch-icon-precomposed" sizes="114x114" ↩
                 href="/static/assets/apple-touch-icon-114x114.png" />
2093          <link rel="apple-touch-icon-precomposed" sizes="72x72" ↩
                 href="/static/assets/apple-touch-icon-72x72.png" />
2094          <link rel="apple-touch-icon-precomposed" sizes="144x144" ↩
                 href="/static/assets/apple-touch-icon-144x144.png" />
2095          <link rel="apple-touch-icon-precomposed" sizes="60x60" ↩
                 href="/static/assets/apple-touch-icon-60x60.png" />
2096          <link rel="apple-touch-icon-precomposed" sizes="120x120" ↩
                 href="/static/assets/apple-touch-icon-120x120.png" />
2097          <link rel="apple-touch-icon-precomposed" sizes="76x76" ↩
                 href="/static/assets/apple-touch-icon-76x76.png" />
2098          <link rel="apple-touch-icon-precomposed" sizes="152x152" ↩
                 href="/static/assets/apple-touch-icon-152x152.png" />
2099          <link rel="icon" type="image/png" href="/static/assets/favicon-196x196.png" ↩
                 sizes="196x196" />
2100          <link rel="icon" type="image/png" href="/static/assets/favicon-96x96.png" ↩
                 sizes="96x96" />
2101          <link rel="icon" type="image/png" href="/static/assets/favicon-32x32.png" ↩
                 sizes="32x32" />
2102          <link rel="icon" type="image/png" href="/static/assets/favicon-16x16.png" ↩
                 sizes="16x16" />
2103          <link rel="icon" type="image/png" href="/static/assets/favicon-128.png" ↩
                 sizes="128x128" />
2104          <meta name="application-name" content=" "/>
2105          <meta name="msapplication-TileColor" content="#FFFFFF" />
2106          <meta name="msapplication-TileImage" ↩
                 content="/static/assets/mstile-144x144.png" />
2107          <meta name="msapplication-square70x70logo" ↩
                 content="/static/assets/mstile-70x70.png" />
2108          <meta name="msapplication-square150x150logo" ↩
                 content="/static/assets/mstile-150x150.png" />
2109          <meta name="msapplication-wide310x150logo" ↩
                 content="/static/assets/mstile-310x150.png" />
2110          <meta name="msapplication-square310x310logo" ↩
                 content="/static/assets/mstile-310x310.png" />
2111      </head>
2112      <body>
2113          <div class="navbar-fixed-top topnav">
2114 {% for navitem in params['navbar'] %}
2115 {% if params['route'].startswith(params['navbar'][navitem]) %}
2116              <a class="active" href="{{params['navbar'][navitem]}}">{{navitem}}</a>
2117 {% else %}
2118              <a href="{{params['navbar'][navitem]}}">{{navitem}}</a>
2119 {% endif %}
2120 {% endfor %}
2121 {% if current_user.is_authenticated %}
2122              <a class="right" href="/gatekeeper/sign-out">Hi, {{session['username']}}. ↩
                    Sign out.</a>
2123 {% else %}
2124              <a class="right" href="/gatekeeper/sign-in">Sign in</a>
2125 {% endif %}
2126 {% if params['searchBar'] == True %}
2127              <!--<div class="search-container">
2128                  <form action="/site-search">
2129                  <input type="text" placeholder="Search.." name="search">
2130                  <button type="submit"><i class="fa fa-search"></i></button>
2131                  </form>
2132              </div>-->
```

```
{% endif %}
        </div>
        <div class="content">
{% block body %}
{% endblock %}
        </div>
        <div class="acknowledgements navbar-fixed-bottom">
            <a class="acknowledgements" href="/acknowledgements">Site Design by Jacob House.</a>
        </div>
    </body>
</html>
```

templates/default.htm

```
{% extends "vowr_template.htm" %}

{% block title %}
VOWR Digital Library
{% endblock %}

{% block body %}

        <h1 class="sect-header">Welcome</h1>

        <p>This website allows DJs to view and edit the VOWR Digital Library, and to create music playlists from the catalogue. Unauthenticated users may also request songs to be played on a program.</p>

        {% if not current_user.is_authenticated %}
        <p>To get detailed song information, perform modifications or to create playlists, you must be <a href="/gatekeeper/sign-in">signed in</a>.</p>
        {% endif %}

        <h2 class="sect-header">Site Map</h2>

        <h3>Request Song</h3>
        <p>Lorepm ipsum dolor sit amet.</p>

{% if current_user.is_authenticated %}
        <h3>Search</h3>
        <p>Lorepm ipsum dolor sit amet.</p>

        <h3>Playlists</h3>
        <p>Lorepm ipsum dolor sit amet.</p>

        <h3>Append</h3>
        <p>Lorepm ipsum dolor sit amet.</p>

        <h3>Modify</h3>
        <p>Lorepm ipsum dolor sit amet.</p>

        <h3>Admin</h3>
        <p>Lorepm ipsum dolor sit amet.</p>

{% if params['admin'] %}
```

```
        <p>LOREM IPSUM DOLOR SIT AMET.</p>
{% endif %}
{% endif %}


{% endblock %}
```

templates/search_template.htm

```
{% extends "vowr_template.htm" %}

{% block style %}
        <link rel="stylesheet" href="/static/style/search.css">
{% endblock %}

{% block scripts %}
        <script type="text/javascript">
        /*
         * This script handles auto-completion for any text inputs with the "auto" class.
         */
        $(function() {
            $(".auto").autocomplete({
                minLength: 3,
                source: function (request, response) {
                    $.ajax({
                        url: "/search/auto",
                        data: { var: this.element.attr("id"), val: request.term },
                        dataType: "json",
                        success: response,
                        error: function () {
                            response([]);
                        }
                    });
            }})
        });
        </script>
        <script>
        /* Script to XMLHttpRequest the data from append up to the DB and update the
           status if the request goes through. Make sure in the Python to get the
           route and determine using that what to return.
         * Ex. For /modify/submit we might return "[<date/time>] <songs name> artist
            changed from <old> to <new>"
         * For /append/submit we might return "[<date/time>] Entity created - <song
            name> by <artist> in album <code>."
         */
        </script>
        <script>
        /*
         * This script is invoked by the /playlists page. If the playlist dropdown is
            set to create_new, we display the field for the new playlist name.
         */
        function checkNewPlaylist() {
            var e = document.getElementById("playlist");
            var newPlaylist = e.options[e.selectedIndex].value;
            if (newPlaylist === "create_new") {
```

```
                    document.getElementById("new_playlist_field").innerHTML = '<input type="text" class="auto input-block-level thin" id="playlist_name" placeholder="Playlist Name">';
                    document.getElementById("playlist_button").innerHTML = 'Add';
                } else {
                    document.getElementById("new_playlist_field").innerHTML = "";
                    document.getElementById("playlist_button").innerHTML = 'Edit';
                }
            }
            </script>
{% endblock %}

<!--<a href="firewall_rules_edit.php?id=17" class="fa fa-pencil" title="Edit"></a>
<a href="firewall_rules_edit.php?dup=17" class="fa fa-clone" title="Copy"></a>
<a href="?act=toggle&amp;if=wan&amp;id=17" class="fa fa-ban" title="Disable" usepost></a>
<a href="?act=del&amp;if=wan&amp;id=17" class="fa fa-trash" title="Delete this rule" usepost></a> -->
```

templates/search.htm

```
{% extends "search_template.htm" %}

{% block title %}
Search VOWR Digital Catalogue
{% endblock %}


{% block body %}

            <form class="form-search" method="POST">
                <input type="text" class="auto input-block-level" name="song" placeholder="Song">
                <input type="text" class="auto input-block-level" name="artist" placeholder="Artist">
                <input type="text" class="auto input-block-level" name="album" placeholder="Album">
                <input type="text" class="auto input-block-level" name="genre" placeholder="Genre">
                <button class="btn btn-large btn-primary thin" type="submit">Search</button>
            </form>

            Search results here.
{% endblock %}
```

templates/append.htm

```
{% extends "search_template.htm" %}

{% block title %}
Add to VOWR Digital Catalogue
{% endblock %}


```

```
{% block body %}
        <h2 class="sect-header">Instructions</h2>

        <p>To add a song to the catalogue, complete the fields above and click <em>Append</em>.</p>

        <p>To easily add songs to existing albums or by existing artists, the form will auto-complete with any existing entries. Moreover, to easily add multiple songs from the same album or, more generally, with any duplicate fields, the data entered will not disappear after submission. This is normal behaviour. You may confirm that the entry was recieved by consulting the <em>Status</em> section below. All changes will be listed here as they are committed to the catalogue.</p>

        <form class="form-modify" method="POST">
            <h2>New entity</h2>
            <label for="song">Song</label><input type="text" class="auto input-block-level" id="song" placeholder="Song">
            <label for="artist">Artist</label><input type="text" class="auto input-block-level" id="artist" placeholder="Artist">
            <label for="album">Album</label><input type="text" class="auto input-block-level" id="album" placeholder="Album">
            <label for="genre">Genre</label><input type="text" class="auto input-block-level" id="genre" placeholder="Genre">
            <label for="path">File Path</label><input type="text" class="input-block-level" id="path" placeholder="File Path">
            <label for="canadian">Canadian Content</label><select class="short" id="canadian">
                  <option selected disabled>Is Canadian</option>
                  <option value="n">No</option>
                  <option value="y">Yes</option>
            </select>
            <button class="btn btn-large btn-primary thin" type="submit">Append</button>
        </form>

        <h2 class="sect-header">Status</h2>

        <div id="status">
            <p>No changes made.</p>
        </div>
{% endblock %}
```

templates/modify.htm

```
{% extends "search_template.htm" %}

{% block title %}
Modify VOWR Digital Catalogue
{% endblock %}

{% block body %}
        <h2 class="sect-header">Instructions</h2>

        <p>To modify an entry in the catalogue, first use the <em>Search</em> page to find the entry you wish to modify. Then, click the edit button on the righthand side of the row in question.</p>
```

```
2307
2308        <p>This will bring you back to the <em>Modify</em> page. The attributes of the ↩
               entry you wish to modify will be populated in a form on this page. Make any ↩
               changes and click the <em>Modify</em> button to commit these to the ↩
               catalogue.</p>
2309
2310        <p>To easily change entity properties to match existing albums or artists, the ↩
               form will auto-complete with any existing entries. Moreover, to confirm the ↩
               changes made are indeed correct, the data entered will not disappear after ↩
               submission. This is normal behaviour. You may confirm that the entry was ↩
               recieved by consulting the <em>Status</em> section below. All changes will ↩
               be listed here as they are committed to the catalogue.</p>
2311
2312 {% if 'entity' in params %}
2313        <form class="form-modify">
2314            <h2>Modify entity</h2>
2315            <label for="song">Song</label><input type="text" class="auto ↩
                   input-block-level" id="song" value="{{ params['entity']['name'] }}" ↩
                   placeholder="Song">
2316            <label for="artist">Artist</label><input type="text" class="auto ↩
                   input-block-level" id="artist" value="{{ ↩
                   params['entity']['artist_name'] }}" placeholder="Artist">
2317            <label for="album">Album</label><input type="text" class="auto ↩
                   input-block-level" id="album" value="{{ params['entity']['album_code'] ↩
                   }}" placeholder="Album">
2318            <label for="genre">Genre</label><input type="text" class="auto ↩
                   input-block-level" id="genre" value="{{ params['entity']['genre'] }}" ↩
                   placeholder="Genre">
2319            <label for="canadian">Canadian Content</label><select class="short" ↩
                   id="canadian">
2320                    <option disabled>Is Canadian</option>
2321 {% if params['entity']['canadian'] %}
2322                    <option selected value="n">No</option>
2323                    <option value="y">Yes</option>
2324 {% else %}
2325                    <option value="n">No</option>
2326                    <option selected value="y">Yes</option>
2327 {% endif %}
2328            </select>
2329            <input type="hidden" id="song_id" value="{{ params['entity']['song_id'] }}">
2330            <button class="btn btn-large btn-primary thin" type="submit">Modify</button>
2331        </form>
2332
2333        <h2 class="sect-header">Status</h2>
2334
2335        <div id="status">
2336            No changes made.
2337        </div>
2338 {% endif %}
2339 {% endblock %}
```

templates/playlists.htm

```
2340 {% extends "search_template.htm" %}
2341
2342 {% block title %}
```

```
VOWR Playlist Editor
{% endblock %}


{% block body %}
        <form class="form-search" action="/playlists/edit" method="POST">
            <select required onchange="checkNewPlaylist()" class="short" name="playlist" id="playlist">
                <option selected disabled value="">Choose a playlist</option>
{% for playlistIndex in params['existingPlaylists'] %}
                <option value="{{ playlistIndex }}">{{ params['existingPlaylists'][playlistIndex] }}</option>
{% endfor %}
                <option value="create_new">Create new...</option>
            </select>
            <div class="inline" id="new_playlist_field"></div>
            <button id="playlist_button" class="btn btn-large btn-primary thin" type="submit">Edit</button>
        </form>
{% endblock %}
```

templates/playlists_edit.htm

```
{% extends "search_template.htm" %}

{% block title %}
VOWR Playlist Editor
{% endblock %}


{% block body %}
        <h2 class="inline sect-header">Editing Playlist "{{ params['playlistName'] }}"</h2>
        <form class="inline" action="/playlists" method="POST"><button class="btn btn-large btn-primary vthin right" onClick="history.go(0)" type="submit">Back</button></form>


        <form class="form-search" action="/playlists/edit" method="POST">
            <input type="text" class="auto input-block-level" name="song" placeholder="Song">
            <input type="text" class="auto input-block-level" name="artist" placeholder="Artist">
            <input type="text" class="auto input-block-level" name="album" placeholder="Album">
            <input type="text" class="input-block-level" name="genre" placeholder="Genre">
            <input type="hidden" name="playlist" value="{{ params['playlistId'] }}">
            <button class="btn btn-large btn-primary thin" type="submit">Append</button>
        </form>
{% endblock %}
```

templates/admin_template.htm

```
{% extends "vowr_template.htm" %}

{% block style %}
<link rel="stylesheet" href="/static/style/admin.css">
{% endblock %}

{% block body %}
{% if params['admin'] %}
<div class="navbar">
    <div class="navbar-inner">
      <div class="container">
        <ul class="nav">
{% for subnavitem in params['subnavbar'] %}
{% if params['subnavbar'][subnavitem] == params['route'] %}
            <li class="active"><a ↩
               href="{{params['subnavbar'][subnavitem]}}">{{subnavitem}}</a></li>
{% else %}
            <li><a href="{{params['subnavbar'][subnavitem]}}">{{subnavitem}}</a></li>
{% endif %}
{% endfor %}
        </ul>
      </div>
    </div>
  </div><!-- /.navbar -->
{% endif %}
  {% block content %}
  {% endblock %}
{% endblock %}
```

templates/admin_home.htm

```
{% extends "admin_template.htm" %}

{% block title %}
Digital Catalogue Admin
{% endblock %}

{% block content %}
        <h1 class="sect-header">Edit Profile</h1>

        <p>User settings here.</p>

        <h1 class="sect-header">Edit Appearance</h1>

        <p>Appearance settings here.</p>
{% endblock %}
```

templates/admin_users.htm

```
{% extends "admin_template.htm" %}

{% block title %}
```

```
Digital Catalogue Users
{% endblock %}

{% block content %}
Users here.
{% endblock %}
```

templates/admin_dbadmin.htm

```
{% extends "admin_template.htm" %}

{% block title %}
Digital Catalogue DB Admin
{% endblock %}

{% block content %}
DB admin here.
{% endblock %}
```

templates/admin_setup.htm

```
{% extends "admin_template.htm" %}

{% block title %}
Digital Catalogue Setup
{% endblock %}

{% block content %}
{% if params['setupDone'] == True %}
        <!-- Main hero unit for a primary marketing message or call to action -->
        <div class="hero-unit">
            <h1>Way to go!</h1>
            <p>It looks like your database is all set up!</p>
        </div>
{% else %}
        <!-- Main hero unit for a primary marketing message or call to action -->
        <div class="hero-unit">
                <h1>Fix me!</h1>
                <p>It looks like your database is needs a little setup. This ↩
                   shouldn't take long.</p>
            </div>
{% endif %}
{{ params[] }}
{% endblock %}
```

templates/gatekeeper_sign-in.htm

```
{% extends "vowr_template.htm" %}

{% block style %}
```

```
<link rel="stylesheet" href="/static/style/gatekeeper.css">
{% endblock %}

{% block title %}
Sign In
{% endblock %}

{% block body %}

        <div css="text-align:center"><form class="form-signin" method="POST" action="/gatekeeper/sign-in">
          <h2 class="form-signin-heading">Please sign in</h2>
          {% if params['authFailed'] %}
          <p class="error">Invalid username or password!</p>
          {% endif %}
          <input type="text" class="input-block-level" id="username" placeholder="User name">
          <input type="password" class="input-block-level" id="password" placeholder="Password">
          <!-- <label class="checkbox">
              <input type="checkbox" value="remember-me"> Remember me
          </label> -->
          <button class="btn btn-large btn-primary" type="submit">Sign in</button>
        </form>
        <div style="text-align:center;"><a class="forgot" href="/gatekeeper/forgot">Forgot your password?</a></div>

{% endblock %}
```

templates/gatekeeper_sign-out.htm

```
{% extends "vowr_template.htm" %}

{% block style %}
<link rel="stylesheet" href="/static/style/gatekeeper.css">
{% endblock %}

{% block title %}
Sign Out
{% endblock %}

{% block body %}

{% endblock %}
```

templates/gatekeeper_forgot.htm

```
{% extends "vowr_template.htm" %}

{% block style %}
<link rel="stylesheet" href="/static/style/gatekeeper.css">
{% endblock %}

```

```
{% block title %}
Reset Password
{% endblock %}

{% block body %}

        <div css="text-align:center"><form class="form-signin" method="POST" action="/gatekeeper/forgot">
          <h2 class="form-signin-heading">Reset password</h2>
          <input type="text" class="input-block-level" id="username" placeholder="User name">
          <input type="text" class="input-block-level" id="secAnswer" placeholder="Answer">
          <button class="btn btn-large btn-primary" type="submit">Sign in</button>
        </form>

{% endblock %}
```

Style Sheets

static/style/vowr.css

```
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    margin: 0;
    padding: 0;
}

html{
  position:relative;
  min-height: 100%;
}

h1.sect-header, h2.sect-header {
    padding-top: 2pc;
}

.content {
    margin: 3pc 3pc 2pc 3pc;
    position: relative;
}


.acknowledgements {
    padding: 6pt;
    font-size: 80%;
    text-align: center;
}

.acknowledgements a {
    color: black;
    /* text-decoration: none; */
}

/* .acknowledgements {
```

```
  position: absolute;
  right: 0;
  bottom: 0;
  left: 0;
  vertical-align: bottom;
  clear: both;
  z-index: 10;
  height: 3em;
  font-size: 75%;
  text-align: center;
} */

.inline {
    display: inline;
    vertical-align: middle;
}

.right {
    float: right;
}
```

STATIC/STYLE/ADMIN.CSS

```
/* Customize the navbar links to be fill the entire space of the .navbar */
.navbar .navbar-inner {
  padding: 0;
}
.navbar .nav {
  margin: 0;
  display: table;
  width: 100%;
}
.navbar .nav li {
  display: table-cell;
  width: 1%;
  float: none;
}
.navbar .nav li a {
  font-weight: bold;
  text-align: center;
  border-left: 1px solid rgba(255,255,255,.75);
  border-right: 1px solid rgba(0,0,0,.1);
}
.navbar .nav li:first-child a {
  border-left: 0;
  border-radius: 3px 0 0 3px;
}
.navbar .nav li:last-child a {
  border-right: 0;
  border-radius: 0 3px 3px 0;
}
h1 {
  margin-top: 1pc 0pt;
}
```

static/style/bg.css

```
body {
    position: relative;
    background-attachment: fixed;
    background-position: center;
    background-repeat: no-repeat;
    background-size: cover;
    background-image: url("/static/assets/bg.jpg");
    height: 100%;
}



/* Turn off parallax scrolling for tablets and phones */
@media only screen and (max-device-width: 1024px) {
    .bgimg {
        background-attachment: scroll;
    }
}
```

static/style/gatekeeper.css

```
.form-signin {
    max-width: 33%;
    padding: 19px 29px 29px;
    margin: 10% auto 1pc;
    background-color: #fff;
    border: 1px solid #e5e5e5;
    -webkit-border-radius: 5px;
       -moz-border-radius: 5px;
            border-radius: 5px;
    -webkit-box-shadow: 0 1px 2px rgba(0,0,0,.05);
       -moz-box-shadow: 0 1px 2px rgba(0,0,0,.05);
            box-shadow: 0 1px 2px rgba(0,0,0,.05);
}

.form-signin .form-signin-heading,
.form-signin .checkbox {
    margin-bottom: 10px;
}

.form-signin input[type="text"],
.form-signin input[type="password"] {
    font-size: 16px;
    height: auto;
    margin-bottom: 15px;
    padding: 7px 9px;
  }

  .forgot {
      color: black;
      font-weight: 500;
  }

```

```
  .error {
      color: red;
      font-weight: 600;
  }
```

STATIC/STYLE/MENU.CSS

```
/* ul.nav {
    list-style-type: none;
    margin: 0;
    padding: 0;
    overflow: hidden;
    background-color: #333;
    position: fixed;
    top: 0;
    width: 100%;
}

li.nav {
    float: left;
}

li.nav a {
    display: block;
    color: white;
    text-align: center;
    padding: 0.5pc 1.5pc;
    text-decoration: none;
}

li.nav a:hover:not(.active) {
    background-color: #111;
}

.active {
    background-color: #018EEC;
} */
/*div.topnav {
    display: block;
    margin: 0pt;
    width: 100%;
    position: fixed;
    overflow: hidden;
    background-color: #333;
}

.topnav a {
    float: left;
    display: block;
    color: white;
    text-align: center;
    padding: 0.5pc 1.5pc;
    text-decoration: none;
}

.topnav a:hover {
```

```
    background-color: #111;
    color: white;
}

.topnav a.active {
    background-color: #018EEC;
    color: white;
}

.topnav a.active:hover {
    background-color: #0E91E8;
    color: white;
}
.topnav .search-container {
    float: right;
}

.topnav input[type=text] {
    padding: 0.25pc;
    margin-top: 0.25pc;
    font-size: 1pc;
    border: none;
}

.topnav .search-container button {
    float: right;
    padding: 0.25pc 0.75pc;
    margin-top: 0.25pc;
    margin-right: 1pc;
    background: #ddd;
    font-size: 1pc;
    border: none;
    cursor: pointer;
}

.topnav .search-container button:hover {
    background: #ccc;
}*/









* {box-sizing: border-box;}

.topnav {
  overflow: hidden;
  position: fixed;
  top: 0;
  left: 0;
```

```
2770    right: 0;
2771    width: 100%;
2772    background-color: #333;
2773  }
2774
2775  .topnav a {
2776    float: left;
2777    display: block;
2778    color: white;
2779    text-align: center;
2780    padding: 0.5pc 1.5pc;
2781    text-decoration: none;
2782    font-size: 1pc;
2783  }
2784
2785
2786  .topnav a:hover {
2787    background-color: #111;
2788    color: white;
2789  }
2790
2791  .topnav a.active {
2792      background-color: #018EEC;
2793      color: white;
2794  }
2795
2796  .topnav a.active:hover {
2797      background-color: #0077C5;
2798      color: white;
2799  }
2800
2801  .topnav .right {
2802      float: right;
2803   }
2804
2805  .topnav .search-container {
2806      float: right;
2807  }
2808
2809  .topnav input[type=text] {
2810    padding: 0.25pc;
2811    margin-top: 0.25pc;
2812    font-size: 1pc;
2813    border: none;
2814  }
2815
2816  .topnav .search-container button {
2817    float: right;
2818    padding: 0.25pc 0.5pc;
2819    margin-top: 0.25pc;
2820    margin-right: 1pc;
2821    background: #ddd;
2822    font-size: 1pc;
2823    border: none;
2824    cursor: pointer;
2825  }
2826
2827  .topnav .search-container button:hover {
2828    background: #ccc;
```

```
}

@media screen and (max-width: 600px) {
  .topnav .search-container {
    float: none;
  }
  .topnav a, .topnav input[type=text], .topnav .search-container button {
    float: none;
    display: block;
    text-align: left;
    width: 100%;
    margin: 0;
    padding: 14px;
  }
  .topnav input[type=text] {
    border: 1px solid #ccc;
  }
}
```

static/style/search.css

```
.form-search {
    /* max-width: 80%; */
    padding: 0.75pc;
    margin: 0.75pc auto;
    align-self: center;
    align-content: center;
    background-color: #fff;
    border: 1px solid #e5e5e5;
    -webkit-border-radius: 5px;
       -moz-border-radius: 5px;
            border-radius: 5px;
    -webkit-box-shadow: 0 1px 2px rgba(0,0,0,.05);
       -moz-box-shadow: 0 1px 2px rgba(0,0,0,.05);
            box-shadow: 0 1px 2px rgba(0,0,0,.05);
}

.form-search .form-search-heading,
.form-search .checkbox {
    margin-bottom: 10px;
}

.form-search input[type="text"],
.form-search input[type="password"] {
    width: 17%;
    font-size: 16px;
    height: auto;
    margin-bottom: 15px;
    padding: 0.5pc 0.5pc;
    margin: 1%;
}

.form-modify {
    max-width: 80%;
    padding: 0.75pc 1.5pc 1.5pc 1.5pc;
    margin: 2pc auto;
```

```
    align-self: center;
    align-content: center;
    background-color: #fff;
    border: 1px solid #e5e5e5;
    -webkit-border-radius: 5px;
       -moz-border-radius: 5px;
            border-radius: 5px;
    -webkit-box-shadow: 0 1px 2px rgba(0,0,0,.05);
       -moz-box-shadow: 0 1px 2px rgba(0,0,0,.05);
            box-shadow: 0 1px 2px rgba(0,0,0,.05);
}

.form-modify .form-modify-heading,
.form-modify .checkbox {
    margin-bottom: 10px;
}

.form-modify input[type="text"],
.form-modify input[type="password"] {
    width: 100%;
    font-size: 1pc;
    height: auto;
    display: block;
    margin-bottom: 1pc;
    padding: 0.5pc 0.5pc;
}

select {
    width: 100%;
    height: auto;
    display: block;
    padding: 0.5pc 0.5pc;
    margin-bottom: 1pc;
}

.short {
    width: 30%;
}

.form-modify label {
    font-size: 1.25pc;
}

.vphantom p {
    margin: 0pt;
}

.thin {
    margin: 0pt;
    padding: 0.5pc 1.25pc;
}

.vthin {
    padding: 0.33pc 1.25pc;
}
```

JavaScript