Phase Documentation

Release 0.1

ChangeToMyName

Jun 28, 2021

Contents

CHAPTER 1

Getting started

1.1 Introduction

Phase is a document management system specifically designed for the needs of engineering and construction projects to manage the documentation of oil & gas, water treatment, nuclear, solar and wind facilities.

Phase offers the following characteristics:

- Management of document and data lists containing thousands of items
- Management of multiple metadata related to engineering, review, schedule, etc.
- Spreadsheet like filtering/search capabilities
- Document and data versioning
- Management of relationships between documents and data

Phase is intended to be used on projects where:

- Thousands of documents are generated
- Documents have to be produced, exchanged, reviewed, revised and used all along the project phases by multiple parties (owner/operator, contractors, vendors, partners, authorities, etc.)

1.2 Installation

Check the *deployment* doc to see how to properly install Phase on a local machine.

1.3 Contributing

To make Phase work on a local environment, you must have the following processes running:

- Phase (django runserver)

- Celery (run locally with *DJANGO_SETTINGS_MODULE=core.settings.local celery -A core.celery worker -l info*)
- RabbitMQ
- Postgres
- Elasticsearch
- Memcached

1.4 Available fabric commands

A fabric script is available to run custom commands. Check *fabfile.py* to have an up to date list.

CHAPTER 2

Basic usage

TODO : Define phase usage

- Category creation
- Category templates

2.1 Users Group

Users groups can be defined in django admin interface. Permissions must be assigned to control available actions. The permissions used in Phase are: documents.add_document, documents.can_control_document, documents.can_start_stop_review, and transmittals.add_outgoing_transmittal for transmittal generation.

CHAPTER 3

Customizing document models

Phase comes with predefined document models. However, it is designed so you can create your own.

All you need to do is to create a new application with a name ending by *"_documents"*:

```
mkdir myproject_app
cd myproject_app
django-admin.py startapp myproject_documents
```

You need to make sure that this application is accessible in the **PYTHONPATH**. If you use virtualenvwrapper, you can use **add2virtualenv**:

```
add2virtualenv myproject_app
```

Once this is done, add your application in the *core/settings/doc_apps.py* file and run **migrate**.

Sample *doc_apps.py*:

```
# -*- coding: utf-8 -*-
from __future__ import unicode_literals

DOC_APPS = (
    'epc2_documents',
    'sileo_documents',
)
```

This file is listed in .gitignore and must not be commited.

3.1 Document model definition

Every document model is made of two classes: a base metadata class and a revision class. The base class must inherit of *documents.models.Metadata* and the revision class must inherit of *documents.models.MetadataRevision*.

Check the *default_documents.models* package for an up to date working example.

3.2 Required fields and methods

On the metadata base class, you must define a **latest_revision** field as a foreign key to the corresponding metadata class.

Inside this class, you also must define a **PhaseConfig** class the same way you would define a **Meta** class. This is used to configure how your document model integrates itself into Phase.

To have the full list of methods that you must implement, take a look in *documents/models.py* and check all methods that throw a *NotImplementedError*.

3.3 Document unique identifier

Every document in Phase have a unique identifier, stored in the *document_key* field. However, every document type must define how this field is generated.

This must be done in the *generate_document_key* method. Here is a example :

```
def generate_document_key(self):
    return slugify(
        u"{contract_number}-{originator}-{unit}-{discipline}-"
        u"{document_type}-{sequential_number}"
        .format(
            contract_number=self.contract_number,
            originator=self.originator,
            unit=self.unit,
            discipline=self.discipline,
            document_type=self.document_type,
            sequential_number=self.sequential_number
        )).upper()
```

The fields that you will use to build unique identifiers should also be listed in a *unique_together* entry in the *Meta* subclass.

3.4 Document list columns

In *PhaseConfig*, the *column_fields* is used to define which fields will be displayed inside columns.

```
column_fields = (
    ('Document Number', 'document_key', 'document_key'),
    ('Title', 'title', 'title'),
    ('Rev.', 'current_revision', 'latest_revision.revision'),
    ('Rev. Date', 'current_revision_date', 'latest_revision.revision_date'),
    ('Status', 'status', 'latest_revision.status'),
)
```

Each entry is composed of three elements:

1. The name that will be displayed in the column header.
2. The class that will be given to the column.
3. The accessor to get the column value. You can use a field name or a property.

3.5 Search and filter form

In the document list, a document filter form is displayed to search and filter documents. Which field will be used is also defined in *PhaseConfig*.

```
# Here are the fields that fill appear in the filter form
filter_fields = ('leader',)

# Those fields will be searchable in the filter form
# You can use fields from the base document or the revision
searchable_fields = ('document_key', 'title')
```

3.6 Import fields

In *PhaseConfig*, the optionnal *import_fields* is used to define how to retrieve foreign keys when importing documents and how to generate import templates.

```
import_fields = OrderedDict(('document_key', {}),
    ('title', {}),
    ('originator', {
        'model': 'accounts.Entity',
        'lookup_field': 'trigram'}),
    ('discipline', {}),
    ('document_type', {}),
    ('vd_code', {}),
    ('received_date', {}),
    ('docclass', {}),
    ('client_document_number', {}),
    ('status_idc_planned_date', {}),
    ('status_ifr_planned_date', {}),
    ('status_afc_planned_date', {}),
    # Revision fields
    ('revision', {}),
    ('status', {}),
    ('purpose_of_issue', {}),)
```

Simple fields like *title* or *vd_code* are populated by inserted the imported value. For foreign key, like *originator*, we specifiy a dict containing the referenced model (here *'accounts.Entity'*) and the lookup field (*'trigram'*).

For revisions, the *created_on* field is always filled with the import date and should not belong to *import_fields*.

CHAPTER 4

Customizing reports

Phase outgoing transmittals reorts are generated with Reportlab package. Reports work out of the box but can be completely customized.

4.1 Company logo

A company logo can be added on the outgoing transmittals pdf on a per organisation basis by writing logo settings in a COMPANY_LOGOS dictionnary.

```
COMPANY_LOGOS = {
    'COMPANY_LOGO_ABC': {'path': abc_logo_path, 'wanted_height': 30, 'x': 13,'y': 40},
    'COMPANY_LOGO_XYZ': {'path': xyz_logo_path, 'wanted_height': 30, 'x': 13, 'y': 40}
↪,
}
```

where ABC and XYZ are the organisations trigrams. The logo appears on first page. This setting must define a path to the logo image file and optionally a wanted_height, logo_x and logo_y in mm. logo_x and logo_y define logo coordinates The logo aspect ratio is preserved.

4.2 Report templates

There is no templating mechanism per se, but a simple class defining pdf content and layout. The base class is transmittals.pdf.BaseTransmittalPdf. It can be overriden by subclassing it in a module, on a per organisation basis. Then, each custom pdf generator is referenced in PDF_CONFIGURATION settings which will provide the dotted path to it.

```
PDF_CONFIGURATION = {
    'TRANSMITTALS_PDF_GENERATOR_ABC': 'import.path.to.Class_1',
    'TRANSMITTALS_PDF_GENERATOR_XYZ': 'import.path.to.Class_2',
}
```

where ABC and XYZ are the organisations trigrams.

CHAPTER 5

Phase deployment

Phase is designed to be a lightweight alternative to traditional bloated and slow DMS. Hence a Phase instance can be run on a single virtual machine.

A single dedicated server can host several environments (pre-production, production).

Warning: Phase is *not* compatible with python 3.5.3 [because of this issue](https://bugs.python.org/issue29519). Either upgrade or downgrade.

5.1 Hosting Phase on a dedicated server

The recommanded settings is to install Phase in an LXC container on a debian stable (currently Stretch) host.

Also use a stretch container:

```
apt-get install lxc debootstrap bridge-utils
```

5.2 LXC configuration

The easiest way to configure the containers network is to give them public ips (using failover ips and a bridge). For other methods, [refer to the documentation](https://wiki.debian.org/LXC).

Configure the host network by editing */etc/network/interfaces*:

```
# Choose ONE of the following options:

# With a DHCP config
auto br0
iface br0 inet dhcp
    bridge_ports eth0
    bridge_fd 0
    bridge_maxwait 0
```

(continues on next page)

(continued from previous page)

```
# With a static config
# Check your hosting provider doc to get the exact parameters to use
auto br0
iface br0 inet static
    address xx.xx.xx.xx
    netmask xx.xx.xx.xx
    network xx.xx.xx.xx
    broadcast xx.xx.xx.xx
    gateway xx.xx.xx.xx
    bridge_ports eth0
    bridge_fd 0
    bridg_maxwait 0
```

Edit the file */etc/lxc/default.conf* with the following content:

```
lxc.network.type = veth
lxc.network.link = br0
lxc.network.flags = up
lxc.network.hwaddr = 00:16:3e:xx:xx:xx
```

Create the container:

```
lxc-create -n <name> -t debian -- -a amd64 -r stretch
```

Edit the container network configuration in */var/lib/lxc/<name>/config*:

```
lxc.network.type = veth
lxc.network.link = br0
lxc.network.flags = up
lxc.start.auto = 1

lxc.network.hwaddr = 00:16:3e:yy:yy:yy
lxc.network.ipv4 = yy:yy:yy:yy
lxc.network.ipv4.gateway = yy:yy:yy:yy
```

Note the *hwaddr* parameter: it's your vm mac address. You need to get this parameter from your hosting provider's interface to bind your vm with a failover ip.

The *ipv4* is the ip failover you want to use, and *ipv4.gateway* comes from you provider doc.

Restart the host's network (check twice or you risk losing access to the server):

```
service networking restart
```

Start the container to check that everything is ok:

```
lxc-start -n <name> -d
```

You can check that your vm is running:

```
lxc-ls --fancy
```

Use this command to access a shell in the vm:

```
lxc-attach -n <name>
```

5.3 Server installation

Some package won't be used and must be uninstalled:

```
apt-get purge apache2 apache2-doc apache2-mpm-prefork apache2-utils apache2.2-bin␣
↪apache2.2-common
```

Some package are needed and must be installed:

```
apt-get update
apt-get upgrade
apt-get install build-essential libpq-dev python3-dev wget curl zlib1g-dev
apt-get install vim postgresql postgresql-contrib nginx nginx-extras git supervisor␣
↪rabbitmq-server
```

5.4 NodeJS installation

Some tools used in Phase require a node.js installation. Get the latest version url on the Node.js site. Let's install it:

```
curl -sL https://deb.nodesource.com/setup_6.x | bash -
apt-get update
apt-get install nodejs
npm install -g npm@lts
```

5.5 Memcache installation

Phase uses Memcached as a cache tool. To install pylibmc, the python memcached backend, you need to install the libs first.

```
apt-get install memcached libmemcached-dev
```

5.6 Database creation

```
su - postgres
createuser -P phase

    Enter password for new role: phase
    Enter it again: phase

createdb --owner phase phase
```

5.7 Python configuration

Install pip and virtualenv (as root):

```
apt-get install python3-pip
pip3 install virtualenv virtualenvwrapper
```

Create user:

```
adduser phase --disabled-password
su - phase
```

Add those lines in the `~/.profile` file:

```
export VIRTUALENVWRAPPER_PYTHON=`which python3`
export WORKON_HOME=~/.virtualenvs
mkdir -p $WORKON_HOME
source `which virtualenvwrapper.sh`
workon phase
export DJANGO_SETTINGS_MODULE=core.settings.production
```

Then:

```
source ~/.profile
```

5.8 Elasticsearch configuration

Phase uses Elasticsearch to index documents and provides search features.

You need to install java for ES to work:

```
apt-get install openjdk-8-jre
```

You can install ES by downloading the apt package on the elastic site:

```
wget -qO - https://packages.elastic.co/GPG-KEY-elasticsearch | apt-key add -
echo "deb http://packages.elastic.co/elasticsearch/2.x/debian stable main" > /etc/apt/
↪sources.list.d/elastic-2.x.list
apt-get install apt-transport-https
apt-get update
apt-get install elasticsearch
```

The default Elasticsearch installation is enough, but remember that ES listens on 0.0.0.0 by default, which can be inconvenient.

To limit ES connections to localhost, one can update the config file `/etc/elasticsearch/elasticsearch.yml` as this:

```
...
network.host: 127.0.0.1
...
```

You also need to make sure that your virtual machine has enough memory available.

Also, make sure ES starts after boot:

```
update-rc.d elasticsearch defaults
```

Or, if your system uses systemd:

```
systemctl daemon-reload
systemctl enable elasticsearch.service
```

5.9 Phase installation

As root:

```
npm install -g cssmin uglify-js
```

As phase user:

```
cd
git clone https://github.com/Talengi/phase.git
cd phase/src
add2virtualenv .
pip install -r ../requirements/production.txt
export DJANGO_SETTINGS_MODULE=core.settings.production
python manage.py collectstatic
python manage.py migrate
```

You can load initial testing data if you need it:

```
python manage.py loaddata initial_accounts initial_values_lists initial_categories␣
↪initial_documents
```

5.10 Web server configuration

If you don't host any other site on the same server, you can replace nginx's default virtual host in */etc/nginx/sites-available/default*:

```
server {
        listen 80 default_server;
        return 444;
}
```

Create the Phase configuration file in `/etc/nginx/sites-available/phase`. Here is a working sample.

```
upstream phase {
    server localhost:8000;
}

server {
    server_name phase;
    access_log /var/log/nginx/phase.access.log;
    error_log /var/log/nginx/phase.error.log;

    client_max_body_size 1g;

    location /static/ {
        root   /home/thibault/code/phase/public/;
    }

    location /media/ {
        root   /home/thibault/code/phase/public/;
    }

    location /xprotected/ {
```

(continues on next page)

(continued from previous page)

```
        internal;
        alias /home/thibault/code/phase/protected/;
    }

    location /xprivate/ {
        internal;
        alias /home/thibault/code/phase/private/;
    }

    location / {
        proxy_pass http://phase;
        proxy_redirect off;
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}
```

Then create a link to enable it:

```
ln -s /etc/nginx/sites-available/phase /etc/nginx/sites-enabled/
```

Don't forget to restart nginx:

```
/etc/init.d/nginx restart
```

5.11 Running the application

Gunicorn is the recommanded WSGI HTTP server to run Phase. Supervisor will be used to monitor it.

Create the `/etc/supervisor/conf.d/phase.conf` config file. here is a working sample.

```
[program:phase]
environment=DJANGO_SETTINGS_MODULE='core.settings.production'
directory=/home/phase/phase/src
command=/home/phase/.virtualenvs/phase/bin/gunicorn -b localhost:8000 core.
↪wsgi:application
user=phase
autostart=true
autorestart=true
stdout_logfile=/var/log/supervisor/phase.log
redirect_stderr=true
```

Phase uses celery as a task queue. Here is the corresponding supervisor file.

```
[program:celery]
environment=DJANGO_SETTINGS_MODULE='core.settings.production'
directory=/home/phase/phase/src/
command=/home/phase/.virtualenvs/phase/bin/celery -A core.celery worker -l info
user=phase
numprocs=1
stdout_logfile=/var/log/celery_stdout.log
stderr_logfile=/var/log/celery_stderr.log
autostart=true
```

(continues on next page)

(continued from previous page)

```
autorestart=true
startsecs=10
```

Run this thing with:

```
supervisorctl reread
supervisorctl reload
```

5.12 Troubleshooting

5.12.1 RabbitMQ won't start after installation

If RabbitMQ fails to start after being installed, make sure the server hostname is set in */etc/hosts*. You can also check the exact hostname used by RabbitMQ by getting the failure detail in */var/log/rabbitmq/startup_log*.

5.12.2 No public key available

If you receive the "No public key available" upon the first *apt-get update*, run the following command:

```
apt-get install debian-keyring debian-archive-keyring
```

Then proceed normally.

5.12.3 Missing jpeg libs for Pillow

When you pip install requirements, Pillow might fail to install with an error related to jpeg management. To fix this, run this command as root:

```
apt-get install libjpeg-dev
```

CHAPTER 6

Development and test

6.1 Installation

Check the *deployment* doc to see how to properly install Phase on a local machine.

6.2 Configuration

You might need to override some local or test settings. You can create either a *local_private.py* or *test_private.py* and add you own settings. These files will be gitignored.

CHAPTER 7

Management tasks and cronjobs

7.1 Reindex all

Documents can be reindexed so that elastic search can stay in synch with actual document data. There is a dedicated task for it:

```
python manage.py reindex_all
```

Warning: This task will completely delete the index and recreate it from scratch.

7.2 Clear private media

Since Django 1.3, FileFields instances are not automaticaly deleted upon's the mode deletion anymore.

This is to preserve data integrity in case of transactions rollbacks.

The drawback is that cleaning file is our responsability.

This tasks cleans the private storage directory by removing all files that are not present in db anymore.

```
python manage.py clearmedia
```

7.3 Exports cleanup

Exported files are kept on disk for a certain duration. There is a dedicated task to clean old exported file.

```
python manage.py exports_cleanup
```

Warning: This task is unnecessary, since old exports are now cleaned on a new export creation.

7.4 Crontab

Setup a crontab to run scheduled tasks regularly. You must use your phase user to run the tasks. Here is a sample crontab file:

```
PYTHON="/home/phase/.virtualenvs/phase/bin/python"
DJANGO_PATH="/home/phase/phase/src/"
LOGS_PATH="/home/phase/django_logs/"
DJANGO_SETTINGS_MODULE="core.settings.production"

# m h  dom mon dow   command
# 42 0 * * * cd $DJANGO_PATH && $PYTHON manage.py reindex_all --noinput &>"$LOGS_PATH/
↪reindex.log"
42 1 * * * cd $DJANGO_PATH && $PYTHON manage.py clearmedia  &>"$LOGS_PATH/clearmedia.
↪log"
42 2 * * * cd $DJANGO_PATH && $PYTHON manage.py exports cleanup  &>"$LOGS_PATH/export_
↪cleanup.log"
```

Warning: Make sure you create the path pointed by the *$LOGS_PATH* variable.

CHAPTER 8

Transmittals upload

The transmittals upload feature allows a contractor to upload a bunch of documents into a Phase instance directly from a ftp upload.

8.1 Directory definition

The directory must be named XXX

dir content

8.2 Server configuration

Here are the instructions to install and configure the ftp server to activate this feature.

Note that Phase doesn't care how the files are transmitted to the server (ftp, ssh, nfs, etc.) so this section is for information only.

8.2.1 Ftp server installation and configuration

We will use the proftpd server to handle ftp communication, and configure the server to only accept ftps (ftp over ssl) connexions.

First, install the *proftpd* ftp server:

```
aptitude install proftpd
```

Choose the "standalone" start method.

Create the ssl certificates for the TLS connection.

```
openssl req -x509 -newkey rsa:2048 \
     -keyout /etc/ssl/private/proftpd.key -out /etc/ssl/certs/proftpd.crt \
     -nodes -days 365
chmod 0600 /etc/ssl/private/proftpd.key
chmod 0640 /etc/ssl/private/proftpd.key
```

Configure the server, using those examples files as starting points.

/etc/proftpd/proftpd.conf:

```
# Includes DSO modules
Include /etc/proftpd/modules.conf

# Set off to disable IPv6 support which is annoying on IPv4 only boxes.
UseIPv6                                 off

RootLogin                           off

# If set on you can experience a longer connection delay in many cases.
IdentLookups                            off

ServerName                           "Phase"
ServerType                           standalone
DeferWelcome                            off

MultilineRFC2228                    on
DefaultServer                            on
ShowSymlinks                            on

TimeoutNoTransfer                   600
TimeoutStalled                           600
TimeoutIdle                         1200

DisplayLogin                        welcome.msg
DisplayChdir                           .message true
ListOptions                            "-l"

DenyFilter                           \*.*/

# Use this to jail all users in their homes
DefaultRoot                             ~

# Users require a valid shell listed in /etc/shells to login.
# Use this directive to release that constrain.
RequireValidShell                   off

# Port 21 is the standard FTP port.
Port                                    21

# To prevent DoS attacks, set the maximum number of child processes
# to 30.  If you need to allow more than 30 concurrent connections
# at once, simply increase this value.  Note that this ONLY works
# in standalone mode, in inetd mode you should use an inetd server
# that allows you to limit maximum number of processes per service
# (such as xinetd)
MaxInstances                            30

# Set the user and group that the server normally runs at.
```

(continues on next page)

(continued from previous page)

```
User                                proftpd
Group                               nogroup

# Umask 022 is a good standard umask to prevent new files and dirs
# (second parm) from being group and world writable.
Umask                               002  002

# Normally, we want files to be overwriteable.
AllowOverwrite                      off

# This is required to use both PAM-based authentication and local passwords
# AuthOrder                         mod_auth_pam.c* mod_auth_unix.c

TransferLog /var/log/proftpd/xferlog
SystemLog   /var/log/proftpd/proftpd.log

# In order to keep log file dates consistent after chroot, use timezone info
# from /etc/localtime.  If this is not set, and proftpd is configured to
# chroot (e.g. DefaultRoot or <Anonymous>), it will use the non-daylight
# savings timezone regardless of whether DST is in effect.
SetEnv TZ :/etc/localtime

DelayEngine on

# This is used for FTPS connections
Include /etc/proftpd/tls.conf

# List of authorized users
Include /etc/proftpd/users.conf

# Prevent files and directories rename / deletion
<Limit DELE>
DenyAll
</Limit>

<Limit RNFR>
DenyAll
</Limit>

<Limit RNTO>
DenyAll
</Limit>
```

/etc/proftpd/tls.conf:

```
TLSEngine                           on
TLSRequired                         on
TLSProtocol                         SSLv23
TLSVerifyClient                     off

TLSRSACertificateFile               /etc/ssl/certs/proftpd.crt
TLSRSACertificateKeyFile            /etc/ssl/private/proftpd.key

TLSLog                              /var/log/proftpd/tls.log
```

/etc/proftpd/users.conf:

```
<Limit LOGIN>
AllowUser test_ctr
DenyALL
</Limit>
```

8.2.2 User creation

Let's create a unix user "test_ctr" for the contractor, and configure the directory permissions.

```
adduser test_ctr --disabled-password --ingroup=phase --shell=/bin/false
chmod g+rwX /home/test_ctr
echo "umask 002" >> /home/test_ctr/.profile
```

Note that for safety reasons, the list authorized users are explicitely declared in the */etc/proftpd/users.conf* file.

CHAPTER 9

Audit trail

Phase features an audit trail, i.e activity stream logging users actions.

The audit trail is loosely based on Activity Stream specification http://activitystrea.ms/specs/json/1.0/

We log:

- The actor: the object that performed the activity (user or system)
- The verb of the action
- The action object : the object linked to the action itself
- The target: the object to which the activity was performed
- The action timestamp

Action object and target are optional Action object, Actor and target are also denormalized in a Charfield to keep the record even if related objects are deleted.

9.1 Actions logged

Currently, actions logged are defined in audit_trail.models.Activity:

```
VERB_CREATED = 'created'
VERB_EDITED = 'edited'
VERB_DELETED = 'deleted'
VERB_JOINED = 'joined'
VERB_STARTED_REVIEW = 'started_review'
VERB_CANCELLED_REVIEW = 'cancelled_review'
VERB_REVIEWED = 'reviewed'
VERB_CLOSED_REVIEWER_STEP = 'closed_reviewer_step'
VERB_CLOSED_LEADER_STEP = 'closed_leader_step'
VERB_CLOSED_APPROVER_STEP = 'closed_approver_step'
VERB_SENT_BACK_TO_LEADER_STEP = 'sent_back_to_leader_step'
```

A signal is defined in audit_trail.signals and sent in relevant part of the application.

9.2 For admins

The audit trail displaying all users activities is accessible in django admin interface for admin users.

9.3 For other users

User having *documents.can_control_document* permission can access the document audit trail by the action dropdown menu.

CHAPTER 10

Django administration

Superusers can access django admin interface.

10.1 Reports

Reports access and appearance in sidebar menu can be controlled by checking “display report section” in category template.

10.2 Contractors and outgoing transmittals

Third party users (contractor not belonging to main organisation) can receive a limited access to Phase in order to get outgoing tranmsmittals.

First, contractors users have to be created. The category user relationships must contains a link to the relevant Outgoing transmittal category. Then a contractor entity must be created (Contractor Type). Then, users have to be added to Entity users field.

CHAPTER 11

Colophon

This documentation is generated by sphinx, please edit *docs/index.rst* to add more content and use the *fab docs* command to compile it.

- Django: https://www.djangoproject.com/
- Bootstrap: http://twitter.github.io/bootstrap/
- Two Scoops of Django template: https://django.2scoops.org/
- Sphinx: http://sphinx-doc.org/
- Datepicker for Bootstrap: http://www.eyecon.ro/bootstrap-datepicker/
- File upload for Bootstrap: http://jasny.github.io/bootstrap/javascript.html#fileupload
- jQuery UI MultiSelect Widget: http://www.erichynds.com/blog/jquery-ui-multiselect-widget
- yuglify: https://github.com/yui/yuglify/