ETD Drop Documentation

Release 1.0

University of North Texas

November 12, 2014

Contents

ETD Drop is a simple web application for accepting online submissions of electronic theses and dissertations (ETDs), written in Django.

This manual is divided into the following parts:

CHAPTER 1

For System Administrators / I.T.

1.1 Technical Overview

ETD Drop is a simple web application for accepting online submissions of electronic theses and dissertations (ETDs), written in Django. Submissions are saved to a configurable location on disk in an easy to navigate structure, making them easy for your staff (or custom software) to review and move into the next stage of your ETD workflow.

A database is only required in order to facilitate user authentication, though with a bit of Django expertise it is possible to replace the default authentication system with a different one (e.g. using LDAP) potentially eliminating the need for a database altogether.

1.1.1 Data Storage Format

When an ETD submission is received from a user, the following steps take place:

1. The form data is validated according to which fields are marked with `'required': True` in your settings.py file.
2. A submission identifier is generated according to the following naming scheme: YYYYMMDD-HHMMSS-username (e.g. *20140401-182104-stephen* would be a submission made on April 1, 2014, at 18:21:04, by a user logged in as "stephen")
3. A directory in the ETD_STORAGE_DIRECTORY location is created with the following structure:

- (identifier)/
  - data/
    - etd.pdf (the main thesis/dissertation PDF file)
    - license.pdf (the license agreement PDF file, if provided)
    - form.json (JSON-encoded representation of what was submitted via the form)
    - form.xml (XML-encoded representation of what was submitted via the form)
    - supplemental/ (contents of the supplemental data ZIP file, if provided)
  - bagit.txt

  - bag-info.txt
  - manifest-md5.txt

You might recognize this structure as a BagIt bag. The submission package is stored in this format to allow for easier management and the ability to verify file checksums at a later point in time.

Form Data Representation

Along with the uploaded files, ETD Drop includes JSON and XML documents containing the values given by the user via the submission form (along with some basic metadata relating to the files that were uploaded). The documents are stored as `form.json` and `form.xml`, and are found within the `data` directory of a submission package. This document is small and should be easy to parse (even for a human). The presence of individual keys will depend on which form fields were enabled in `settings.py` and what was actually provided by the user. Here is a sample of the contents of a typical `form.json`:

```
{
  "document_file": {
    "content_type": "application/pdf",
    "original_filename": "thesis.pdf",
    "size": 2149036
  },
  "license_file": {
    "content_type": "application/pdf",
    "original_filename": "license.pdf",
    "size": 81439
  },
  "abstract": "Sample abstract.",
  "supplemental_file": {
    "content_type": "application/zip",
    "original_filename": "supplemental_data.zip",
    "size": 5181242
  },
  "title": "Sample Title",
  "author": "Sample Author"
}
```

And here is an equivalent example of a `form.xml` file:

```
<?xml version="1.0" ?>
<root>
  <author type="str">Sample Author</author>
  <abstract type="str">Sample abstract.</abstract>
  <title type="str">Sample Title</title>
  <document_file type="dict">
    <content_type type="str">application/pdf</content_type>
    <original_filename type="str">thesis.pdf</original_filename>
    <size type="int">2149036</size>
  </document_file>
  <license_file type="dict">
    <content_type type="str">application/pdf</content_type>
    <original_filename type="str">license.pdf</original_filename>
    <size type="int">81439</size>
  </license_file>
  <supplemental_file type="dict">
    <content_type type="str">application/zip</content_type>
    <original_filename type="str">supplemental_data.zip</original_filename>
    <size type="int">5181242</size>
```

```
  </supplemental_file>
</root>
```

1.1.2 Understanding the Project Code

For more in-depth information suited for developers, continue to the *Developer Overview*.

1.2 Installation

ETD Drop is distributed as a complete Django project, making it quite straightforward to run. It is essentially a WSGI application, needing only minor database-related setup and some Python package dependencies. Popular ways of running Django apps include gunicorn, nginx (with uwsgi), and Apache (with mod_wsgi).

1.2.1 System Requirements

- Python 2.7.4 (or any higher 2.7.x release)
- virtualenv (strongly recommended)
- A writable directory large enough to accommodate your needs

1.2.2 Local Testing

These instructions should help you to set up an instance of ETD Drop running locally on a personal machine or VM. This is useful for small scale testing or development, but not for serving the application to actual users.

Initial Setup

1. Clone this repository on your local filesystem, probably somewhere within your user's home directory.
2. `cd` into this repository's directory.
3. Create a new virtual environment here: `virtualenv.py venv`
4. Copy `etd_drop/settings.py.example` to `etd_drop/settings.py`.

5. Edit `etd_drop/settings.py` in a code or plain text editor and set up your any settings you need to override (see the Configuration section below).
6. Copy `env_example` to `.env`.
7. Edit `.env` with a text editor and configure the settings to your desired setup.

Each Time You Begin a New Work Session

1. `cd` into this repository's directory.
2. Activate your virtual environment: `source venv/bin/activate`
3. (Optional) Pull latest changes from git: `git pull`
4. Install/update dependencies: `pip install -Ur requirements.txt`
5. Ensure that the database is set up: `DJANGO_DEBUG=1 python manage.py syncdb`

Note: If this is your first time running the *syncdb* command, a new SQLite database will be generated, and you will be prompted to create a new administrative account. Follow the prompts and provide a username and password for this new account. (You can leave the "email" field blank.)

Running a Local Server

1. `cd` into this repository's directory.
2. Activate your virtual environment: `source venv/bin/activate`
3. Start the development server: `DJANGO_DEBUG=1 python manage.py runserver`
4. When you wish to stop the server, use CTRL+C in the terminal window.

1.2.3 Production Server Deployment

There are many approaches and choices to consider when deploying a Django project on a real server. The general strategy is something like the following:

1. Incoming requests are proxied by Nginx (which also directly serves static assets belonging to the project)
2. Nginx forwards requests on to a WSGI server (like uWSGI or Gunicorn)
3. The WSGI server handles the request and displays the page

If you would prefer to use Apache, you should use mod_wsgi and refer to this page for guidance in setting things up.

Otherwise, we recommend using Nginx with Gunicorn. The following steps outline the general process of setting up ETD Drop in this way on a Linux server.

Getting Started

This guide will assume the use of an up-to-date installation of Ubuntu Server 12.04, though the general principles should apply to any other reasonably- equipped Linux environment.

First, let's make sure everything we need is installed:

```
sudo apt-get install -y nginx git python-virtualenv
```

Setting up ETD Drop in a Virtualenv

We need a place where the ETD Drop code can live. Technically the location doesn't matter that much, but a popular convention is `/srv/django`. Let's create this location:

```
sudo mkdir -p /srv/django
```

Now we'll fetch the ETD Drop code repository:

```
cd /srv/django
sudo git clone https://github.com/metaarchive/etd-drop
```

ETD Drop is now fetched at `/srv/django/etd-drop`. Now, let's set up a virtualenv to contain our Python packages nicely:

```
sudo virtualenv /srv/venv/etd-drop
```

The following command "activates" the new virtualenv in your current shell so that we are in the correct environment for the next several steps:

```
sudo -i
source /srv/venv/etd-drop/bin/activate
```

Now go to the ETD Drop source code directory and fetch its dependencies:

```
cd /srv/django/etd-drop
pip install -r requirements.txt
```

Configure Nginx

To make things simple, we've provided a sample Nginx configuration file along with the ETD Drop source code. Install it as follows:

```
cp /srv/django/etd-drop/nginx/etd-drop.conf /etc/nginx/sites-available
ln -s ../sites-available/etd-drop.conf /etc/nginx/sites-enabled/
rm /etc/nginx/sites-enabled/default # Disables the default nginx config
```

Configuring Settings

Before going any further, you will need to edit `/srv/django/etd-drop/etd_drop/settings.py` and configure your project's settings (especially the DATABASES setting if you wish to use something other than SQLite3 to store user accounts). Refer to *Configuration* for details.

Initializing ETD Drop

Do the following in order to initialize ETD Drop:

```
source /srv/venv/etd-drop/bin/activate
cd /srv/django/etd-drop/
python manage.py collectstatic -c --noinput
python manage.py syncdb
python manage.py syncdb --noinput
```

At this point, you should create an initial "superuser" account (an administrative account which will be able to log in and manage other user accounts in ETD Drop). Run the following command and follow the prompts:

```
python manage.py createsuperuser
```

Choose these credentials wisely as this account will have full administrative privileges inside the application.

Running the Server

Finally, the commands you will use to start up the servers:

```
sudo service nginx restart
cd /srv/django/etd-drop
sudo DOTENV=/srv/django/etd-drop/.env /srv/venv/etd-drop/bin/gunicorn -b unix:/tmp/gunicorn.sock etd_
```

1.3 Configuration

ETD Drop is configured using values defined in the `settings.py` file in the `etd_drop` directory of the project.

1.3.1 Environment Variables

For ease of deployment or local testing, some configuration values can be set using environment variables. Environment variables can be set up in web server configurations (like Apache and Nginx), set in your personal .bashrc or .profile files for easy personal use, or stated at the beginning of a terminal command (for example: `DJANGO_DEBUG=1 python manage.py runserver`).

ETD-Drop supports the 'dotenv' configuration convention, and will look for a .env file in the top level directory with values for runtime-configurable settings.

DJANGO_DEBUG

Accepted values: 1 (meaning debug mode is on) or 0 (meaning debug mode is off)

Overrides the default DEBUG setting.

DJANGO_SECRET_KEY

Accepted values: Any string

Overrides the SECRET_KEY setting, which should be set to a long, randomized string of characters used for security purposes (see the SECRET_KEY section later in this page).

ETD_STORAGE_DIRECTORY

Accepted values: Any valid pathname

Overrides the ETD_STORAGE_DIRECTORY setting, which is where the stored ETDs that are uploaded will be kept. Make sure that this directory exists and is writable by the ETD-Drop process.

ENABLE_CLAMD

Accepted values: 1 (on) or 0 (off) (default)

Set to 1 if you want ETD-Drop to use Clam Antivirus to scan uploaded ETDs. This requires that the Clam-AV Daemon be installed and running.

DJANGO_LOGGING_LEVEL

Accepted values: 'DEBUG', 'INFO', 'WARNING', 'ERROR' or 'CRITICAL'

Overrides the LOGGING_LEVEL setting, which controls the severity of events that will be reported by the logger.

DJANGO_LOGGING_PATH

Accepted values: Any valid pathname

Overrides the LOGGING_PATH setting, which controls where ETD-Drop will keep its logs. Make sure that the file is writable by the ETD-Drop process.

1.3.2 Core settings

These are standard Django settings you will want to pay special attention to:

ALLOWED_HOSTS

See: https://docs.djangoproject.com/en/1.6/ref/settings/#allowed-hosts

DATABASES

Specifies the connection information Django should use for all of its database operations (in the case of ETD Drop, this is for users/sessions).

By default, this is configured to use a SQLite3 file located in the project directory (which will be created automatically if it doesn't exist). SQLite3 should be sufficient for the needs of this application, but you may still prefer to change these settings to use an external database (e.g. MySQL) instead.

See: https://docs.djangoproject.com/en/1.6/ref/settings/#databases

DEBUG

Default: bool(int(get_env_setting('DJANGO_DEBUG', default=False)))

A boolean (True or False) value that decides if Django should run in "debug" mode. In debug mode, Django runs with fewer security restrictions and allows detailed error messages to be displayed in the browser. **It is very important not to use debug mode in production environments.**

The default value of DEBUG attempts to load the setting from an environment variable named DJANGO_DEBUG (which should be set to 1 if True or 0 if False). If this environment variable is not set, False will be used by default.

EMAIL_BACKEND

Default: 'django.core.mail.backends.smtp.EmailBackend'

The Django email backend to use for sending email. SMTP is the default, which is the most commonly-used server type for sending email.

For a full list of possible backends, see: https://docs.djangoproject.com/en/1.6/topics/email/#smtp-backend

See general information about this setting, see: https://docs.djangoproject.com/en/1.6/ref/settings/#email-backend

EMAIL_HOST

Default: 'localhost'

Hostname of SMTP server (or other selected backend type).

EMAIL_HOST_PASSWORD

Default: ''

Password for authenticating with SMTP server (or other selected backend type).

EMAIL_HOST_USER

Default: ''

Username for authenticating with SMTP server (or other selected backend type).

EMAIL_PORT

Default: 25

EMAIL_USE_TLS

Default: `False`

See: https://docs.djangoproject.com/en/1.6/ref/settings/#email-use-tls

FILE_UPLOAD_TEMP_DIR

Default: `None`

The location where user-submitted files are temporarily kept before the submission package is built. If not defined (or set to `None`), the system's default temporary directory (e.g. `/tmp`) will be used.

To account for large uploads, you may wish to change this setting to a path on a volume where storage is plentiful.

FILE_UPLOAD_MAX_MEMORY_SIZE

Default: 2621440

Uploaded files smaller than this size (in bytes) will be temporarily stored in memory (RAM) instead of being stored as a file in `FILE_UPLOAD_TEMP_DIR`. This results in faster uploads, but will consume more system memory during uploads depending on how high this limit is set.

Note: 2621440 bytes = 2.5 MB

SECRET_KEY

Default: `SECRET_KEY = get_env_setting('DJANGO_SECRET_KEY', default=None)`

A string containing a unique, unpredictable set of characters known only to the server.

The default value attempts to do two things:

1. If an environment variable called DJANGO_SECRET_KEY is set, it will use that value for this setting.
2. Otherwise, the setting will be set to `None` and the application will not be able to start.

One way of generating a good random key is using the following command:

```
python -c 'import random; import string; print "".join([random.SystemRandom().choice(string.digits +
```

See: https://docs.djangoproject.com/en/1.6/ref/settings/#std:setting-SECRET_KEY

TIME_ZONE

Default: 'UTC'

See: https://docs.djangoproject.com/en/1.6/topics/i18n/

1.3.3 ETD Drop settings

These settings apply specifically to the functionality of ETD Drop, and will allow you to customize some of the functionality and presentation of the ETD Drop web application itself:

ETD_STORAGE_DIRECTORY

Default: `get_env_setting('ETD_STORAGE_DIRECTORY', default=mkdtemp(prefix="etd-drop"))`

A string representing the absolute path of the directory where ETD submissions should be stored. In practice, you will want to use a directory on a volume that is

- large enough to accommodate the submissions you anticipate receiving
- able to be accessed by the people in your organization whose staff will be responsible for receiving and processing the submission packages (via SFTP, SCP, Windows shared directory (SMB), etc.).

The default value attempts to do two things:

1. If an environment variable called ETD_STORAGE_DIRECTORY is set, it will use that value for this setting.
2. Otherwise, it will try to create a directory in your system's temporary directory (e.g. `/tmp`) prefixed with the name "etd-drop" and use that location instead. (This is useful for local testing, but obviously should not be used in production since anything stored there will not be permanently saved!)

If you would rather not use an environment variable to specify the directory, you can replace this line with something as simple as:

ETD_STORAGE_DIRECTORY = "/mnt/data"

(replacing */mnt/data* with the actual path you wish to use).

CONTACT_PHONE

A string containing a phone number that will be displayed on the homepage for users to call if they need help. If this setting is blank or undefined, the phone number will be hidden.

CONTACT_EMAIL

A string containing an email address that will be displayed on the homepage for users to email if they need help. If this setting is blank or undefined, the email address will be hidden.

DESCRIPTION_SERVICE_URL

Default: Not set

A string containing a URL to a running instance of the DAITSS Format Description service (https://github.com/daitss/describe).

If set, the description service will be used to generate PREMIS-formatted identification/validation/characterization data (powered by DROID, JHOVE) for each of the files in the submission package when submissions are created.

For more information about this process and how to run your own instance of the Format Description Service in your environment, see: https://github.com/MetaArchive/bag-describe

Example: `DESCRIPTION_SERVICE_URL = "http://localhost:3000"`

SUBMISSION_EMAIL_RECIPIENTS

Default: `[]`

A list of strings representing email addresses to notify when a new submission is received. If this list is empty, no email will be sent.

SUBMISSION_EMAIL_FROM_ADDRESS

Default: "noreply@domain.edu"

A string containing the email address that will appear in the "From" header for notification emails sent by ETD Drop.

HOMEPAGE_HEADING

A string containing the title you wish to be shown on the homepage. By default, this is set to `"Submit Your Thesis"`.

HOMEPAGE_TEXT

A string containing the text you wish to appear on the homepage underneath the page title. Any line breaks you use in this string will be converted to line breaks in the HTML, and a blank line between two lines of text will convert to a paragraph break.

By default, this is set to:

```
"""
ETD Drop allows our graduate students to easily submit a copy of their thesis or dissertation electr

After logging in you will be asked to upload your document as a PDF. If you have any supplemental fi

If required, please make sure you have a signed and scanned Copyright License in PDF form available

Lastly, the submission form will ask for your document's title and abstract. You can copy and paste

It's that easy.
"""
```

FOOTER_TEXT

A string containing the text you wish to appear in the footer. Any line breaks you use in this string will be converted to line breaks in the HTML, and a blank line between two lines of text will convert to a paragraph break.

By default, this is set to:

```
"""
Footer text
"""
```

LOGO_IMAGE_URL

A string containing a URL to a logo image you wish to appear in the footer.

SUBMISSION_AGREEMENT

A string containing the text you wish to appear above the "agreement" checkbox at the bottom of the submission form. Typically this represents the terms that the user will be agreeing to when submitting their ETD. Any line breaks you use in this string will be converted to line breaks in the HTML, and a blank line between two lines of text will convert to a paragraph break.

By default, this is set to:

```
"""
By clicking the box below I agree that this submission is complete. Any errors in this submission wi
"""
```

SUBMISSION_FORM_FIELDS

This setting allows you to hide or make mandatory the various submission form fields that make up a submission. For instance, if you want to completely hide the "Subject" field from the form, you would change the following lines:

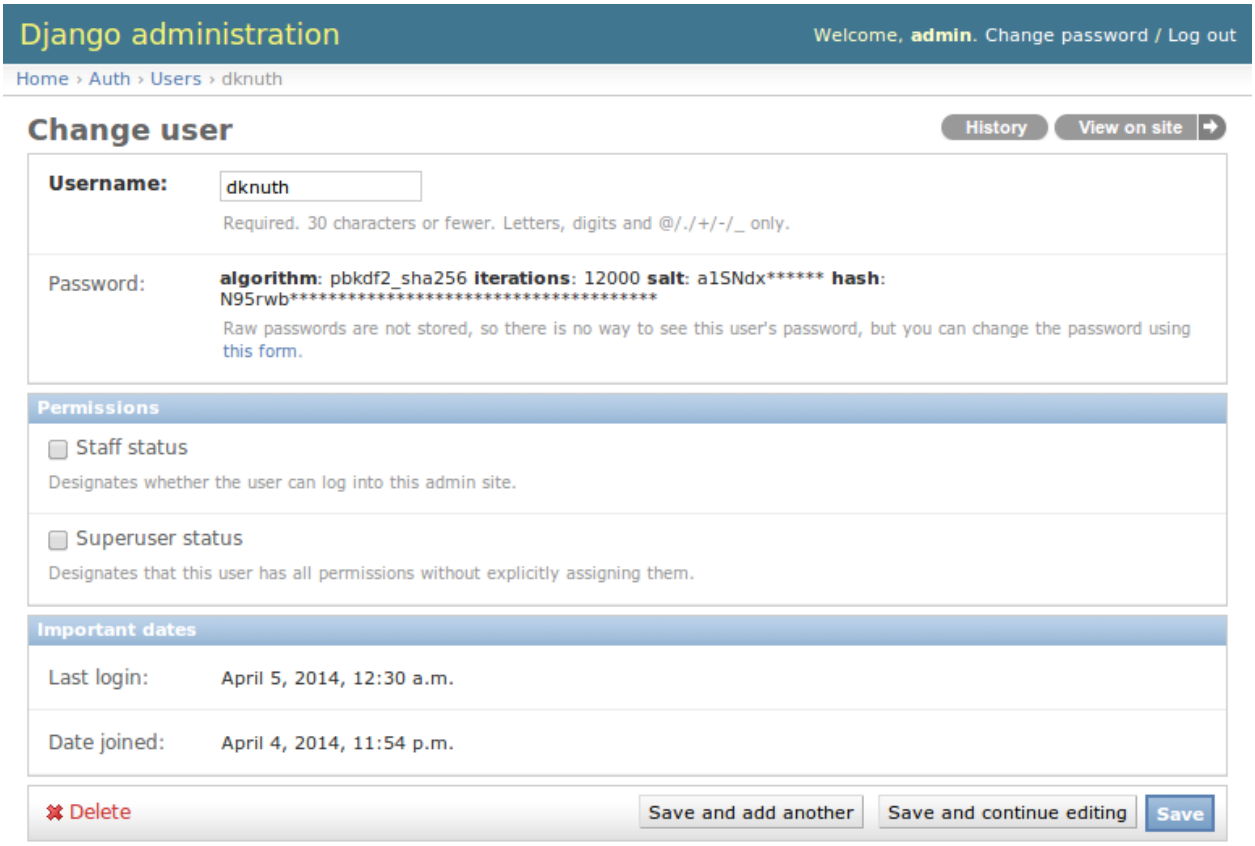
```
'subject': {
    'visible': True,
    'required': False,
},
```

to this:

```
'subject': {
    'visible': False,
    'required': False,
},
```

1.4 User Management

Authenticated users with the "super user" role will see a link in the page header labelled "Admin". This link will take you to the Django administrative interface where you can manage user accounts.



When editing a user account, note the checkboxes granting the following permissions:

1.4.1 Staff status

Users with the "staff" role have access to the submissions page (see *Managing Submissions*) and can view the contents of submissions using the web interface.

1.4.2 Superuser status

Users with the "superuser" role have full access to all features of the web application, including the Django administrative interface (as described on this page).

1.4.3 No permissions

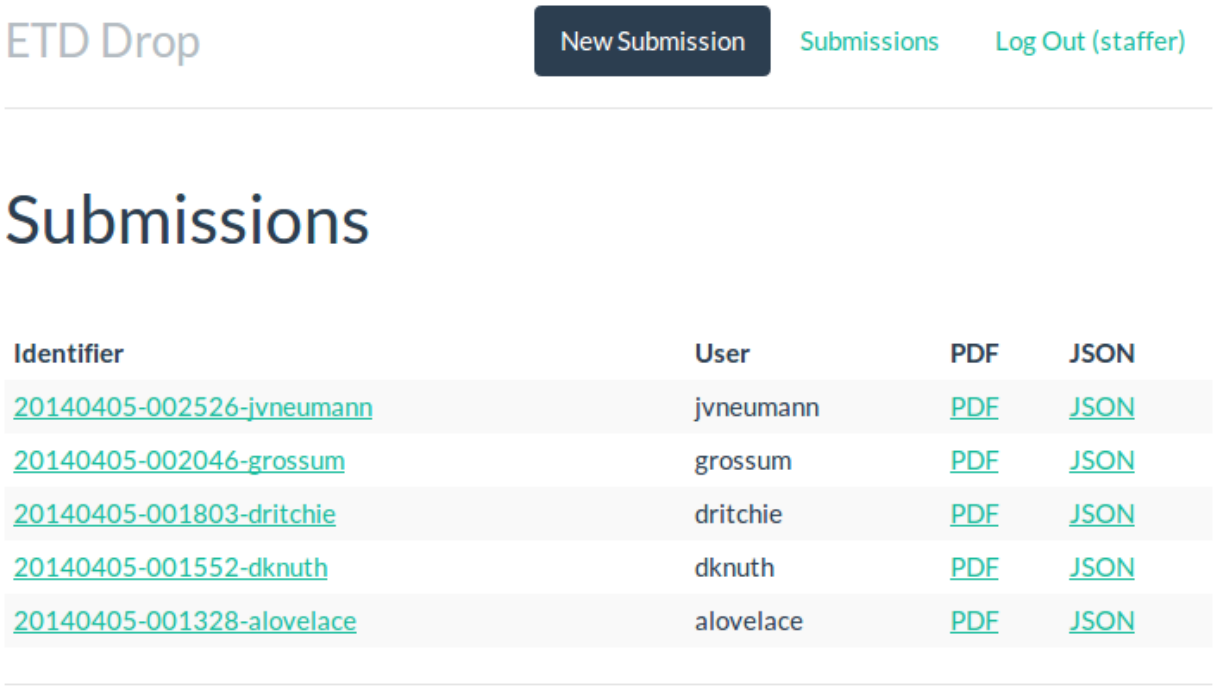
Users with neither of the "staff" or "superuser" roles described above are treated as normal users, with access only to the "submit" page and the home page.

CHAPTER 2

For ETD Program Staff

2.1 Managing Submissions

When you are logged in as a user with the "staff" role, you will see a link in the page header labelled "Submissions". This link will take you to a page where you can see a listing of all current submissions awaiting review on the server.



From this table, you can click one of the links in any row to access more information about a submission. Clicking the identifier name in the left column will take you to a page with extensive details about that submission, including the user-submitted form data and a list of files and checksums.

ETD Drop

New Submission | Submissions | Log Out (staffer)

Submission Details

Download Links

Main Document | Form JSON | License Document

Basic Information and Metadata

Identifier 20140405-001552-dknuth

Title Finite Semifields and Projective Planes

Author Donald Ervin Knuth

Subject mathematics

Abstract This paper makes contributions to the structure theory of finite semifields, i.e., of finite nonassociative division algebras with unit. It is shown that a semifield may be conveniently represented as a 3-dimensional array of numbers, and that matrix multiplications applied to each of the three dimensions correspond to the concept of isotopy. The six permutations of three coordinates yield a new way to obtain projective planes from a given plane. Several new classes of semifields are constructed; in particular one class, called the binary semifields, provides an affirmative answer to the conjecture that there exist non-Desarguesian projective planes of all orders 2[...], if n is greater than 3. With the advent of binary semifields, the gap between necessary and sufficient conditions on the possible orders of semifields has disappeared.

Bag Info

```
Bag-Software-Agent: bagit.py <http://github.com/libraryofcongress/bagit-python>
Bagging-Date: 2014-04-05
Internal-Sender-Identifier: Finite Semifields and Projective Planes
Payload-Oxum: 5292829.5
```

Bag Manifest

```
7c2db699039188319bda3760f7d27fa2  data/etd.pdf
ebb4c3983bf66c1c1701c8a9a8ec9350  data/form.json
88003532cc1ebde09c3f1e3df0172313  data/license.pdf
6e34a2a0eaa61748ba3a33015a84e813  data/supplemental/How_fast.ogg
c9459a907b9345b289ba6c9e6517d4c2  data/supplemental/Josh
```

2.2 Suggested Workflow

ETD Drop is intentionally very simple, allowing for a multitude of options for how to implement your ETD program's workflow. For a basic use case, we might recommend the following ETD ingest workflow:

1. The student logs in to ETD Drop and submits their ETD and accompanying data.
2. An employee (or team of employees) takes the submission package from the server and moves it to a location where they can work on creating a final submission package suitable for the institutional repository. (This should typically include verifying that the submitted files adhere to institutional standards, creating cleaned metadata based on the information submitted by the student, virus-checking, etc.)
3. At this point, the submission package should be deleted from the location where ETD Drop created it.

CHAPTER 3

For End Users / Students

3.1 Submitting an ETD

1. Log in on the homepage.
2. Fill out the form to submit your thesis or dissertation. All fields are required unless marked as "(Optional)".
3. Agree to the terms at the bottom of the page.
4. Click the **Upload and Submit** button.

   Note: If there are any errors with your submission, correct them and try again. You will need to re-select any uploaded files you attached during the previous submission attempt.
5. Make a note of the submission ID number on the next page, after your submission has been received. This serves as a record that your submission was sent successfully.
6. Click the "Log out" button at the top of the page.

CHAPTER 4

For Developers

4.1 Technical Overview

ETD Drop is a simple web application for accepting online submissions of electronic theses and dissertations (ETDs), written in Django. Submissions are saved to a configurable location on disk in an easy to navigate structure, making them easy for your staff (or custom software) to review and move into the next stage of your ETD workflow.

A database is only required in order to facilitate user authentication, though with a bit of Django expertise it is possible to replace the default authentication system with a different one (e.g. using LDAP) potentially eliminating the need for a database altogether.

4.1.1 Data Storage Format

When an ETD submission is received from a user, the following steps take place:

1. The form data is validated according to which fields are marked with `'required': True` in your settings.py file.
2. A submission identifier is generated according to the following naming scheme: YYYYMMDD-HHMMSS-username (e.g. *20140401-182104-stephen* would be a submission made on April 1, 2014, at 18:21:04, by a user logged in as "stephen")
3. A directory in the ETD_STORAGE_DIRECTORY location is created with the following structure:

- (identifier)/
  - data/
    - etd.pdf (the main thesis/dissertation PDF file)
    - license.pdf (the license agreement PDF file, if provided)
    - form.json (JSON-encoded representation of what was submitted via the form)
    - form.xml (XML-encoded representation of what was submitted via the form)
    - supplemental/ (contents of the supplemental data ZIP file, if provided)
  - bagit.txt

- bag-info.txt
- manifest-md5.txt

You might recognize this structure as a BagIt bag. The submission package is stored in this format to allow for easier management and the ability to verify file checksums at a later point in time.

Form Data Representation

Along with the uploaded files, ETD Drop includes JSON and XML documents containing the values given by the user via the submission form (along with some basic metadata relating to the files that were uploaded). The documents are stored as `form.json` and `form.xml`, and are found within the `data` directory of a submission package. This document is small and should be easy to parse (even for a human). The presence of individual keys will depend on which form fields were enabled in `settings.py` and what was actually provided by the user. Here is a sample of the contents of a typical `form.json`:

```
{
  "document_file": {
    "content_type": "application/pdf",
    "original_filename": "thesis.pdf",
    "size": 2149036
  },
  "license_file": {
    "content_type": "application/pdf",
    "original_filename": "license.pdf",
    "size": 81439
  },
  "abstract": "Sample abstract.",
  "supplemental_file": {
    "content_type": "application/zip",
    "original_filename": "supplemental_data.zip",
    "size": 5181242
  },
  "title": "Sample Title",
  "author": "Sample Author"
}
```

And here is an equivalent example of a `form.xml` file:

```
<?xml version="1.0" ?>
<root>
  <author type="str">Sample Author</author>
  <abstract type="str">Sample abstract.</abstract>
  <title type="str">Sample Title</title>
  <document_file type="dict">
    <content_type type="str">application/pdf</content_type>
    <original_filename type="str">thesis.pdf</original_filename>
    <size type="int">2149036</size>
  </document_file>
  <license_file type="dict">
    <content_type type="str">application/pdf</content_type>
    <original_filename type="str">license.pdf</original_filename>
    <size type="int">81439</size>
  </license_file>
  <supplemental_file type="dict">
    <content_type type="str">application/zip</content_type>
    <original_filename type="str">supplemental_data.zip</original_filename>
    <size type="int">5181242</size>
```

```
  </supplemental_file>
</root>
```

4.1.2 Understanding the Project Code

For more in-depth information suited for developers, continue to the *Developer Overview*.

4.2 Developer Overview

This section contains information that should come in handy if you are developing for (or on top of) ETD Drop. You should already be familiar with the information in *Technical Overview*.

Overall, ETD Drop is an ordinary single-application Django project. The primary difference between it and most Django apps is the total lack of database models; All application data is represented as files on the local filesystem, in order to make ETD Drop's scope as limited as possible and to facilitate easy atomic integration into larger workflows.

This project makes heavy use of Python docstrings. These are the best places to look for details about a particular module, class, method, or function.

4.2.1 Project Source Code Layout

The general structure of this repository is as follows:

```
etd-drop/                   # Top level git repository
    docs/                   ## Documentation (uses Sphinx Docs)
    etd_drop/               ## Django project files
        settings.py         ### Project settings
        urls.py             ### Project-level URL routing
        wsgi.py             ### Project WSGI application
    etd_drop_app            ## Main Django application code
        admin.py            ### Django Admin UI configuration
        forms.py            ### Form processing code
        static/             ### Static resources (CSS and images)
        templates/          ### HTML templates
        templatetags/       ### Custom template tag modules
            form_helpers.py ####  Template tags for forms
        tests.py            ### Unit tests
        urls.py             ### Application-level URL routing
        validators.py       ### Form validator functions
        vendor/             ### Modules included from elsewhere
            bag_describe.py ####  Description Service integration
        views.py            ### View generation code
    LICENSE                 ## Source code license
    manage.py               ## Project management script
    nginx/                  ## Sample configuration for nginx
    README.md               ## Project README
    requirements.txt        ## pip package dependencies
```