Open Legal Data Platform Documentation

Release 0.1

Malte Schwarzer

Dec 10, 2020

Contents:

OLDP is a web application, written in Python 3.5 and based on the Django web framework, It is used for processing legal text and providing a REST-API and Elasticsearch-based search engine. OLDP is being develop by the non-profit initiative Open Legal Data with the goal of building an Open Data platform for legal documents (mainly court decisions and laws). The platform makes legal information freely accessible for the general public and especially third-party apps.

CHAPTER 1

Getting started

The following we present a short guide on how to get started with OLDP. If you encounter any problems, do not hesitate to write an issue or contact us via email or Twitter.

1.1 Install dependencies

```
apt-get install -y $(cat apt-requirements.txt)
pip install -r requirements.txt
npm install
```

1.2 Run tests

Automated tests use Django's testing API. If you are not familiar with Django have a look at their extensive documentation first.

For testing we use settings slightly different to development and production. For instance, SQLite is used as database to speed up testing. To use the testing settings, set the configuration variable as following:

```
export DJANGO_CONFIGURATION=Test
```

Next, you can run either all or specific tests:

```
# all tests
./manage.py test

# tests from the laws app
./manage.py test oldp.apps.laws.tests

# tests only views
./manage.py test --tag=views
```

Some tests require external services (Elasticsearch or web access). To enable or disable them, set the configuration variables:

```
export DJANGO_TEST_WITH_ES=1
export DJANGO_TEST_WITH_WEB=0
```

1.3 Docker

To get the dependency services (database, search, cache) running we suggest to use Docker Compose. Compose is a tool for defining and running multi-container Docker applications.

- See *Docker*

1.4 Run server manually

Run webpack to create the website assets:

```
npm run-script build
```

Set the right environment:

```
export DJANGO_CONFIGURATION=Dev
```

Before running the server for the first time you need to set up the database schema and collect all static files to a single location.

```
./manage.py migrate
./manage.py collectstatic
```

Now you are ready to go:

```
./manage.py runserver
```

An admin account can be created using:

```
./manage.py createsuperuser
```

CHAPTER 2

Testing

See: https://realpython.com/blog/python/testing-in-django-part-1-best-practices-and-examples/

2.1 Fixtures

- Django fixtures
  - https://docs.djangoproject.com/en/dev/topics/testing/tools/#topics-testing-fixtures
  - https://docs.djangoproject.com/en/dev/howto/initial-data/

```
# Dump fixtures
./manage.py dumpdata --pks 1,2,3
./manage.py dumpdata courts --indent 4 --output oldp/apps/courts/fixtures/courts.json
./manage.py dumpdata laws --indent 4 --output oldp/apps/laws/fixtures/laws.json
./manage.py dumpdata cases --indent 4 --output oldp/apps/cases/fixtures/cases.json


# Load fixtures
./manage.py loaddata oldp/apps/laws/fixtures/laws/laws.json

./manage.py loaddata oldp/apps/courts/fixtures/locations/countries.json
./manage.py loaddata oldp/apps/courts/fixtures/locations/states.json
./manage.py loaddata oldp/apps/courts/fixtures/locations/cities.json
./manage.py loaddata oldp/apps/courts/fixtures/courts/courts.json

./manage.py loaddata oldp/apps/cases/fixtures/cases/cases.json
```

2.1.1 OLDP

- Courts: BGH+EUGH+AG...
- Laws: GG, BGB, with table...

- Cases:
  - bgh,

2.2 App tests

- db queries (get + update + create)
- processing tests

2.3 Browser Tests

- WebDriver / Selenium (firefox driver)
- test with local db + production db (ssh tunnel to production server)

2.4 Coverage Integration

```
export DJANGO_CONFIGURATION=Test
coverage run --source='.' manage.py test

# stdout report
coverage report --omit="env/*"

# html report
coverage html --omit="env/*"
```

CHAPTER 3

API

The OLDP API is based on Django's Rest framework extension (with Django-Rest-Swagger).

3.1 Authenticating with the API

Reading our data is possible without logging in. However, be aware of a stricter throttle policy for anonymous users (see Throttle rates). Write operators require authentication for which two methods exist:

- **Username and password**: Use HTTP BasicAuth to login with your login credentials.
- **API Key**: In your user settings you can request a specific API key that allows you to authenticating with the API.

3.1.1 Throttle rates

Spending resources carefully is of high importance for a non-profit project like ours. Hence, we limit the usage of our API as following (taken from `oldp/settings.py`):

- **Anonymous users**: 100 requests/day
- **Registered users**: 5000 requests/hour

If you need to do more requests, check out our data dumps or contact us. Our data is meant to be share - throttling is use a matter of limited resources.

3.2 Clients

- Python https://github.com/openlegaldata/oldp-sdk-python/
- Javascript https://github.com/openlegaldata/oldp-sdk-js/
- Java https://github.com/openlegaldata/oldp-sdk-java/

- PHP https://github.com/openlegaldata/oldp-sdk-php/

3.3 Usage

3.3.1 Python

Python examples can be found in our OLDP-Notebooks on GitHub.

3.3.2 Curl

For terminal-loving developers, `curl` is also always an option. Examples can be found on ${SITE_URL}api/schema/.

```
# List cities
curl -X GET "${SITE_URL}api/cities/" -H  "accept: application/json" -H  "api_key: $
↪{API_KEY}"

# Get cases from court with id=3
curl -X GET "${SITE_URL}api/cases/?court_id=3" -H  "accept: application/json" -H
↪"api_key: ${API_KEY}"
```

3.3.3 Data dumps and bulk downloads

To export all data that is exposed over the API at once, you can use the `dump_api_data` command.

```
./manage.py dump_api_data ./path/to/output_dir --override
```

CHAPTER 4

API Swagger

We use Swagger to generate API clients for several programming languages.

4.1 Clients generator

Client libraries can be auto-generated with Swagger code-gen based on OpenAPI specs.

For example you can use the following command the generate a Python API client. Python 3.7+ support is only available in recent Swagger versions (2.4.1+):

```
export DJANGO_DIR="/your/path/to/django/app"

# cd to the directory where you want to have the client files
# cd /your/path/to/oldp-client

# Python SDK
swagger-codegen generate \
    -i ${SITE_URL}api/schema/?format=openapi \
    -l python \
    -o ${PWD} \
    -c ${DJANGO_DIR}/oldp/api/swagger_codegen.python.json \
    --git-user-id openlegaldata \
    --git-repo-id oldp-sdk-python \
    --release-note "Minor changes"

    [(-t <template directory> | --template-dir <template directory>)]
    [--git-repo-id <git repo id>]

# Run swagger-codegen with Docker
docker run --rm -v ${PWD}:/local -v ${DJANGO_DIR}:/django swaggerapi/swagger-codegen-
↪cli:2.4.1 generate \
    -i ${SITE_URL}api/schema/?format=openapi \
    -l python \
```

(continues on next page)

(continued from previous page)

```
    -o /local \
    -c /django/oldp/api/swagger_codegen.python.json \
    --git-user-id openlegaldata \
    --git-repo-id oldp-sdk-python \
    --release-note "Minor changes"

# Javascript
docker run --rm -v ${PWD}:/local -v ${DJANGO_DIR}:/django swaggerapi/swagger-codegen-
↪cli:2.4.1 generate \
    -i ${SITE_URL}api/schema/?format=openapi \
    -l javascript \
    -o /local \
    -c /django/oldp/api/swagger_codegen.javascript.json \
    --git-user-id openlegaldata \
    --git-repo-id oldp-sdk-javascript

# Java
docker run --rm -v ${PWD}:/local -v ${DJANGO_DIR}:/django swaggerapi/swagger-codegen-
↪cli:2.4.1 generate \
    -i ${SITE_URL}api/schema/?format=openapi \
    -l java \
    -o /local \
    -c /django/oldp/api/swagger_codegen.java.json \
    --git-user-id openlegaldata \
    --git-repo-id oldp-sdk-java
```

To display configuration help run `swagger-codegen config-help -l python`.

4.2 Other notes

The following notes are useful to configure the API on your own.

4.2.1 Swagger settings

```
SWAGGER_SETTINGS = {
    'exclude_url_names': [],
    'exclude_namespaces': [],
    'api_version': '0.1',
    'api_path': '/',
    'relative_paths': False,
    'enabled_methods': [
        'get',
        'post',
        'put',
        'patch',
        'delete'
    ],
    'api_key': '',
    'is_authenticated': False,
    'is_superuser': False,
    'unauthenticated_user': 'django.contrib.auth.models.AnonymousUser',
    'permission_denied_handler': None,
```

(continues on next page)

(continued from previous page)

```
    'resource_access_handler': None,
    'base_path':'helloreverb.com/docs',
    'info': {
        'contact': 'apiteam@wordnik.com',
        'description': 'This is a sample server Petstore server. '
                       'You can find out more about Swagger at '
                       '<a href="http://swagger.wordnik.com">'
                       'http://swagger.wordnik.com</a> '
                       'or on irc.freenode.net, #swagger. '
                       'For this sample, you can use the api key '
                       '"special-key" to test '
                       'the authorization filters',
        'license': 'Apache 2.0',
        'licenseUrl': 'http://www.apache.org/licenses/LICENSE-2.0.html',
        'termsOfServiceUrl': 'http://helloreverb.com/terms/',
        'title': 'Swagger Sample App',
    },
    'doc_expansion': 'none',
}
```

CHAPTER 5

Django - Notes & Useful commands

Here you can find useful Django commands:

5.1 Django basics

```
python manage.py runserver --insecure
python manage.py shell
python manage.py makemigrations laws
python manage.py migrate
python manage.py createsuperuser
./manage.py changepassword admin  #to set the password for the Django admin user

python manage.py startapp appname
python manage.py test appname
```

5.2 Custom commands

```
python manage.py process_laws --empty --limit 100 --min-lines 1000
python manage.py process_cases --limit 10
python manage.py import_courts --input oldp/apps/courts/data/ecli.csv --empty
```

In dev & before deployment

```

python manage.py collectstatic --noinput
python manage.py collectstatic --noinput --ignore=*.scss

./manage.py compilescss

```

(continues on next page)

(continued from previous page)

```
python manage.py makemessages --locale=en --locale=de --ignore=env --ignore=workingdir
python manage.py compilemessages --l de --l en

# API token
./manage.py drf_create_token <username>
```

Migrations

```
./manage.py showmigrations

# Reset
./manage.py migrate appname zero --fake


# Login and drop tables
./manage.py dbshell

drop table cases_case;
drop table cases_relatedcase;

drop table references_casereference;
drop table references_casereferencemarker;
drop table references_lawreference;
drop table references_lawreferencemarker;
```

5.3 Search

```
./manage.py rebuild_index
./manage.py update_index
```

5.4 Tests

```
# All tests
./manage.py test

# Specific app tests
./manage.py test oldp.apps.cases --keepdb
```

5.5 Cache

```
# run in production shell
from django.core.cache import cache

# Empty specific cache
cache.delete('my_url')

# Empty all cache
cache.clear()
```

5.6 Flatpages

```
/imprint/
/privacy/
/api/
```

5.7 Sitemaps

Once the sitemaps application is added to your project, you may also ping Google using the ping_google management command:

```
python manage.py ping_google [/sitemap.xml]
```

5.8 Generate UML diagram from models

```
apt-get install python-pygraphviz
pip install django-extensions
# add 'django_extensions' to INSTALLED_APPS in settings.py
python manage.py graph_models trees -o test.png
```

CHAPTER 6

Development

This guide provides useful information for those who like to contribute to the project or just want into integrate it into their own.

6.1 Useful links

- Setting up a Django development environment by Mozilla
- PyCharm IDE with Django support

6.2 Write a custom processing step

A key part of OLDP is our data processing pipeline. All the data stored on the platform can be used as starting point for further processing, e.g. information extraction.

We illustrate how this can be done with the example of *topic extraction*. In our example we want to assign one or more topics to a law book. Hereby, a topic means a tag, category or list that helps users finding relevant information.

We start with a new Django app in `oldp/apps/topics` and add it to the `settings.py`. Next, we define simple models. `Topic` represents a topic with a descriptive `title`. `TopicContent` is an abstract class that we use as mixin to add the `topics` field to the `LawBook` model.

```
# oldp/apps/topics/models.py

class Topic(db.models):
    title = models.CharField(
        max_length=200,
        help_text='Verbose title of topic',
        unique=True,
    )

class TopicContent(db.models):
```

(continues on next page)

(continued from previous page)

```
    topics = models.ManyToManyField(
        Topic,
        help_text='Topics that are covered by this content',
        blank=True,
    )

    class Meta:
        abstract = True
```

Extent the target model (`LawBook` in our case) with TopicContent:

```
# oldp/apps/laws/models.py

class LawBook(TopicContent): # BEFORE: class LawBook(db.models):
    # ...

# multiple inherits are possible:
# class Law(TopicContent, SearchContent, SomeOtherContent, ...):
```

Make and apply migrations:

```
./manage.py makemigrations
./manage.py migrate
```

For the actual processing step we need to implement as class called `ProcessingStep` that inherits from `BaseProcessingStep` or the corresponding content class. Here, we inherit from `LawBookProcessingStep`.

The cornerstone of each processing step is the `process` method which takes as input a `LawBook`, does the processing and then returns the processed `LawBook`.

In our example, we added an `__init__` that pre-loads all available topics. Then, the actual `process` only assigns five random topics to the law book:

```
# oldp/apps/topics/processing/processing_steps/assign_topics_to_law_book.py

class ProcessingStep(LawBookProcessingStep):
    description = 'Assign topics'

    def __init__(self):
        super().__init__()
        self.topics = Topic.objects.all()

    def process(self, law_book: LawBook) -> LawBook:
        # Remove existing topics
        law_book.topics.clean()

        # Select 5 random topics
        for t in random.sample(self.topics, 5):
            law_book.topics.add(t)

        law_book.save()

        return law_book
```

Add processing step to settings:

```
# oldp/settings.py
# ...

# Processing pipeline
PROCESSING_STEPS = {
    # ...
    'LawBook': [
        'oldp.apps.topics.processing.processing_steps.assign_topics_to_law_book',  #␣
↪Add this line
    ]
}

# ...
```

If the processing step is added to the settings and the corresponding admin class inherits from `ProcessingStepActionsAdmin`, the processing step gets automatically available as admin action.

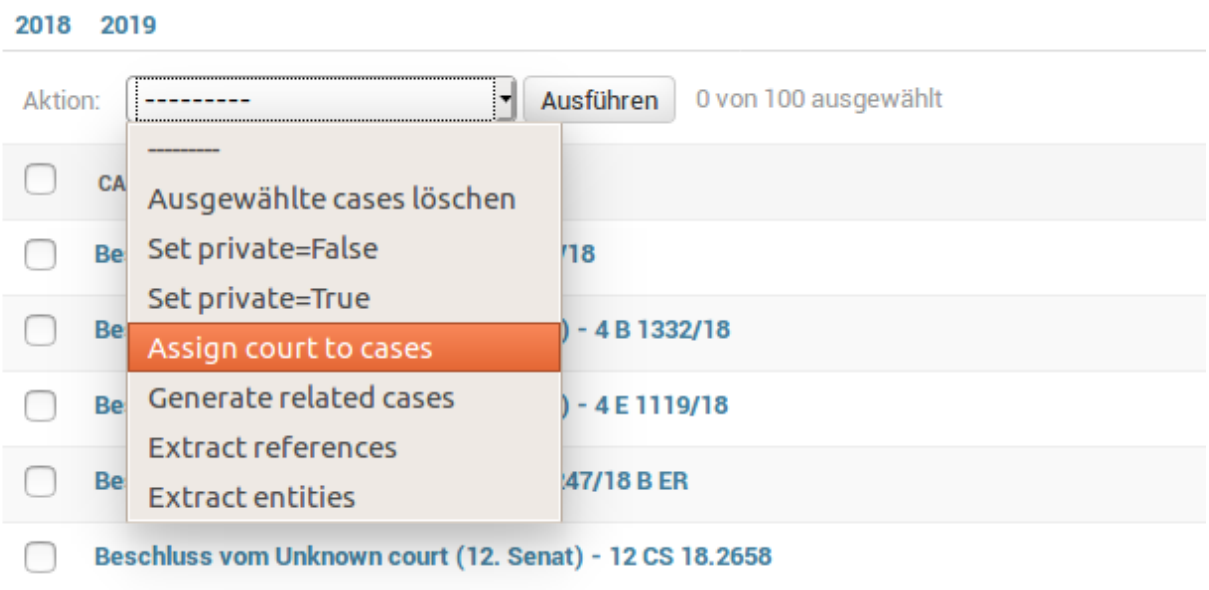
CHAPTER 7

Processing

OLDP comes with a data processing pipeline. How to write your own processing step is explained under *Development*. In the following we explain the execution of the processing pipeline.

7.1 Django Admin

The most convenient way to execute processing steps is to use the Django admin interface. For all models (e.g. cases or laws) that have available processing steps, you only need to select the target items from the list by clicking on the checkboxes and then select one processing step from the action drop-down menu:



Be aware that processing can take time, especially when running complex steps on a large number of items. Thus, the web server might time out and send you a 500 error message.

7.2 Available content processors

For large data-sets and complex selection of to be processed items, it is recommened to execute the processing over commandline. Each model comes with an own Django management command:

- Cases
  - Management command: `./manage.py process_cases`
- Laws
  - Management command: `./manage.py process_laws`
- Courts
  - Management command: `./manage.py process_courts`
- References
  - Management command: `./manage.py process_references`

7.2.1 Parameters

Use the `--help` argument to display a list of available parameters.

```
  --input INPUT [INPUT ...]
  --input-handler INPUT_HANDLER
                        Read input from file system
  --order-by ORDER_BY   Order items when reading from DB
  --filter FILTER       Filter items when reading from DB
  --limit LIMIT
  --start START
  --max-lines MAX_LINES
  --source SOURCE       When reading from FS process files differently
                        (serializer)
  --empty               Empty existing index
```

When using `db` as input handler, the parameter `--filter` and `--exclude` are URL-encoded and support Django model queries (See examples).

7.2.2 Examples

Find corresponding court to all cases that are currently assign to the default court (default court id = 1):

```
./manage.py process_cases --input-handler db --filter court__pk=1 assign_court
```

Limit the number of processed cases to 100 and order by last updated date, i.e., process oldest first.

```
# Assign court
./manage.py process_cases --input-handler db --filter court__pk=1 --order-by updated_
↪date --limit 100 assign_court

# Extract references
./manage.py process_cases --input-handler db --order-by updated_date --limit 100␣
↪extract_refs
```

Cases that are private and from courts located in state with id 5, exclude cases by type:

```
./manage.py process_cases --input-handler db --filter court__state_id=5&private=True -
↪-exclude type=Urteil all
```

Publish currently unpublished cases with a defined court:

```
./manage.py process_cases --input-handler db --order-by updated_date --filter court__
↪pk__gt=0  --limit 100 set_private_false
```

CHAPTER 8

Elasticsearch

As search backend we rely on Elasticsearch. In this document we collect useful commands or queries to work with ES.

8.1 Propagate database entries to search index

- Rebuild index: `./manage.py rebuild_index`
- Update existing index: `./manage.py update_index`

8.2 Queries

```
curl -XGET localhost:9200/oldp/law/_search?pretty&query=

curl -XGET localhost:9200/oldp/law_search?pretty -d '
{
    "query": {
        "match" : {
            "book_code" : "AbwV"
        }
    },
    "sort": [
        { "doknr": { "order": "asc" } },
        "_score"
    ],
    "_source" : ["doknr", "title"]
}'

curl -XGET localhost:9200/oldp/case/_search?pretty -d '
{
```

(continues on next page)

(continued from previous page)

```
    "_source" : ["text", "title"]
}'
```

8.2.1 Check cluster health

```
curl -XGET https://localhost:9200/_cat/health?v
```

8.3 Load Index Mappings

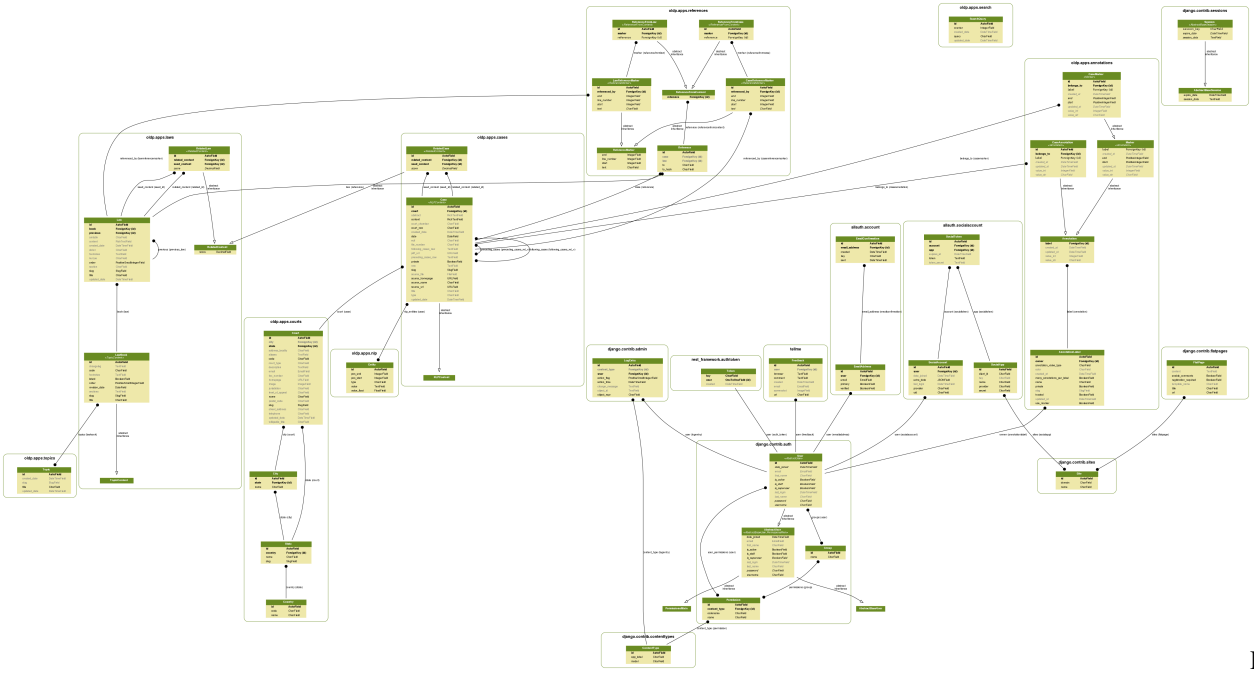
```
curl -XPUT localhost:9200/leegle -d @oldp/assets/es_index.json
```

CHAPTER 9

Database

As database backend you can use all Django-supported db adapters. However, the code is only tested with MySQL and SQLite.

9.1 Schema



DB Schema (Show full-size image)

9.2 Set encoding

Run the following commands to make MySQL support proper utf-8

```
# Check before
SHOW FULL COLUMNS FROM table_name;

ALTER TABLE logtest CONVERT TO CHARACTER SET utf8 COLLATE utf8_general_ci;

ALTER TABLE logtest DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;

ALTER TABLE logtest CHANGE title title VARCHAR(100) CHARACTER SET utf8 COLLATE utf8_
↪general_ci;

ALTER TABLE tablename MODIFY COLUMN col VARCHAR(255) CHARACTER SET utf8 COLLATE utf8_
↪general_ci NOT NULL;

ALTER TABLE courts_court MODIFY COLUMN description CHARACTER SET utf8 COLLATE utf8_
↪general_ci;
```

Change `cases_case.content` to utf-8:

```
# What is the current charset?
SELECT character_set_name FROM information_schema.`COLUMNS`
WHERE table_name = "cases_case"
  AND column_name = "content";

# Copy table as backup
CREATE TABLE cases_case__bak LIKE cases_case;
INSERT cases_case__bak SELECT * FROM cases_case;

# Change
ALTER TABLE cases_case MODIFY content LONGTEXT CHARACTER SET utf8;

# In case something went wrong, restore from backup
RENAME TABLE cases_case TO cases_case__changed;
RENAME TABLE cases_case__bak TO cases_case;
```

CHAPTER 10

Heroku

Deploy OLDP on Heroku:

```
heroku login
heroku create
git push heroku master
heroku config:set CONNECTION_MODE=offline
heroku config:set ES_URL=...

# to add demo data
# run from backend repo: sbin/heroku.sh

# Django
# for debugging
heroku config:set DEBUG_COLLECTSTATIC=1

# for production
heroku config:set DISABLE_COLLECTSTATIC=1

# mysql db
heroku config:set DATABASE_URL=mysql://...
```

CHAPTER 11

Production

11.1 Deployment

When pushing new changes into the production system the following routine should be performed:

- Check unit and integration tests
- Backup code and database
- Stop web service `sudo supervisorctl stop oldp`
- Pull changes from repo `git pull`
- Run
  - Activate production environment (only on oldp1) `source commands.sh`
  - Activate Python environment `source env/bin/activate`
  - `pip install -r requirements.txt`
  - `npm install`
  - `./manage.py render_html_pages`
  - `npm run-script build`
  - `./manage.py collectstatic --no-input`
  - `./manage.py compilemessages --l de --l en`
  - `./manage.py rebuild_index` or `./manage.py update_index`
- Run `./manage.py migrate`
- Start web service `sudo supervisorctl start oldp`

11.2 Commands

Commands for running OLDP in production mode.

```
./manage.py process_cases --limit 20 --empty --input /var/www/apps/oldp/data/split001/
./manage.py set_law_book_order
./manage.py set_law_book_revision
```

11.2.1 Dump data

Create JSONL files from API data:

```
# Dump JSON files
./manage.py dump_api_data ./workingdir/2020-10-10-dump/

# Compress all dumps
gzip -r ./workingdir/2020-10-10-dump/*
```

11.3 Clean up database

```
DELETE FROM cases_case;
ALTER TABLE cases_case AUTO_INCREMENT=1;

DELETE FROM cases_relatedcase;
ALTER TABLE cases_relatedcase AUTO_INCREMENT=1;


DELETE FROM courts_court;
ALTER TABLE courts_court AUTO_INCREMENT=1;

DELETE FROM courts_city;
ALTER TABLE courts_city AUTO_INCREMENT=1;

DELETE FROM courts_state;
ALTER TABLE courts_state AUTO_INCREMENT=1;


DELETE FROM courts_country;
ALTER TABLE courts_country AUTO_INCREMENT=1;


DELETE FROM references_casereferencemarker;
ALTER TABLE references_casereferencemarker AUTO_INCREMENT=1;
```

11.4 Helper commands for migration

```
# Set missing previous law references to NULL
UPDATE laws_law
SET previous_id = NULL
```

(continues on next page)

(continued from previous page)

```
WHERE id in (
    SELECT * FROM (
        select l.id
        from laws_law l
        left join laws_law p
            on p.id = l.previous_id
        WHERE l.previous_id IS NOT NULL AND p.id IS NULL
        ) as t);
```

CHAPTER 12

Docker

OLDP has a containerized version based on Docker. If you just want to try out the platform locally, this is the recommended way to do it. The Docker image is available at Docker Hub.

12.1 Getting started

The OLDP web app depends services like search, db, cache. To run all service in orchestrated fashion use `docker-compose` as following:

```
# Build & start services
docker-compose up
```

To stop the services run `docker-compose down` or press `CRTL+C`.

In beginning the database will be empty, thus, we need to create all tables in the newly created database.

```
docker exec -it oldp_app_1 python manage.py migrate
```

You have probably noticed that you set the login credentials for the MySQL database in `docker-compose.yml`. By default, Django is using the same settings. But if you change those, you need to adjust the `DATABASE_URL` variable.

```
export DATABASE_URL="mysql://oldp:oldp@127.0.0.1/oldp"
```

Import some demo data (from fixtures - see more in testing docs)

```
docker exec -it oldp_app_1 python manage.py loaddata \
    locations/countries.json \
    locations/states.json \
    locations/cities.json \
    courts/courts.json \
    laws/laws.json \
    cases/cases.json
```

Compile localization files

```
docker exec -it oldp_app_1 python manage.py compilemessages --l de --l en
```

Create superuser (admin, pw: admin)

```
docker exec -it oldp_app_1 python manage.py shell -c \
    "from django.contrib.auth.models import User; User.objects.create_superuser('admin
↪', 'admin@example.com', 'admin')"
```

12.2 Common issues

12.2.1 Old image version

If you encounter any problems, please pull the latest image first.

```
docker pull openlegaldata/oldp:latest
```

12.2.2 Invalid file system permissions

Sometimes Elasticsearch has problems writing to its data directory. To solve this, set access rights:

```
# Quick & Dirty
chmod 777 docker/data/es

# Correct user group
chown docker:docker docker/data/es
```

12.3 Additional notes

```
# Build image from repo
docker build -t oldp .

# Tag image as latest
# - locally
docker tag oldp:latest

# - hub
docker tag oldp openlegaldata/oldp:latest

# Push to hub
docker push openlegaldata/oldp:latest

# Start a container
docker run oldp

# Override environment variables
docker run -e DATABASE_URL="sqlite:///db/db.sqlite" -it oldp python manage.py␣
↪runserver
```

CHAPTER 13

Indices and tables

- genindex
- modindex
- search