FinSL-signbank Documentation

Release 1.0.0

Henri Nieminen

Jan 05, 2019

Contents

CHAPTER 1

Contents

1.1 FinSL-signbank documentation

Date Jan 05, 2019

Version 1.0.0

FinSL-signbank is a Django web framework based application for managing *sign language* lexicons.

1.1.1 Installation

Updated on June 15th 2018 by @henrinie

These instructions are written for linux operating systems. For Windows or MacOS some parts might be relevant, look up python docs and django docs for further instructions for those operating systems.

Set up the environment

Clone the git repository, create a python virtual environment, install dependencies with pip, and configure the relevant settings.

Git repository

Clone the repository to your machine from GitHub:

```
$ git clone https://github.com/Signbank/FinSL-signbank.git
```

Python Virtual Environment

Create a virtual environment in python3:

```
$ cd FinSL-signbank
$ python3 -m venv venv
$ source venv/bin/activate
```

If you need to deactivate the environment write:

```
$ deactivate
```

Install dependencies with pip

Install required python dependencies:

```
$ pip install -r /path/to/requirements.txt
Example: $ pip install -r FinSL-signbank/requirements.txt
```

Configure settings

Edit settings files in FinSL-signbank/signbank/settings/ and change the paths in:

```
# settings/production.py & settings/development.py
LOCALE_PATHS = (
    '/path/to/FinSL-signbank/locale',
)
STATIC_ROOT = '/path/to/FinSL-signbank/static' # For development only! Production dir␣
↪needs to be server by web server.
MEDIA_ROOT = '/path/to/FinSL-signbank/media' # For development only! Production dir␣
↪needs to be server by web server.
```

```
# settings/development.py
LOGGING = {
...
'filename': '/path/you/want/debug.log'
...
}
```

```
# settings/production.py

# IMPORTANT: The hostname that this signbank runs on, this prevents HTTP Host header␣
↪attacks
ALLOWED_HOSTS = ['yourhost.here.com']

STATIC_ROOT = '/path/to/static' # Served by the web server, e.g. /var/www/yourdomain/
↪static
MEDIA_ROOT = '/path/to/media' # Served by the web server, e.g. /var/www/yourdomain/
↪media

WSGI_FILE = '/path/to/FinSL-signbank/signbank/wsgi.py' # This will matter when you␣
↪want to use a web server
```

Rename settings/settings_secret.py.template

```
$ mv settings/settings_secret.py.template settings/settings_secret.py
```

Edit settings/settings_secret.py

```
# settings/settings_secret.py

# Make SECRET_KEY unique and do not share it with anyone
# You may use characters available in ASCII
SECRET_KEY = 'yoursecretkey!"#¤%&/()=?'
ADMINS = (
    ('Your Name', 'your.email@address.com'),
)
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': '/path/to/signbank.db',
    }
}
```

Tip: Generate a random secret key

Databases

We kindly recommend using PostgreSQL with FinSL-signbank, because django-framework is optimized to run on PostgreSQL. We have used MySQL in the past, but at least in our case we started to experience some problems with migrations.

PostgreSQL

When you are ready to switch to a database server, PostgreSQL is our recommendation, see django docs for more information about setting django up with PostgreSQL: https://docs.djangoproject.com/en/stable/ref/databases/#postgresql-notes.

In your postgresql.conf make sure you have the following:

```
client_encoding = 'UTF8'
default_transaction_isolation = 'read committed'
timezone: 'UTC' # Because USE_TZ = True in FinSL-signbank
```

Edit settings/secret_settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'mydatabase',
        'USER': 'mydatabaseuser',
        'PASSWORD': 'mypassword',
        'HOST': '127.0.0.1',
        'PORT': '5432',
    }
}
```

Then install psycopg2 with pip when your virtual environment is activated.

```
$ pip install psycopg2
```

SQLite

Edit the following lines in settings/secret_settings.py:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': '/path/to/signbank.db',
    }
}
```

MySQL

If your database of choice is MySQL, create my.cnf for your MySQL credentials

```
[client]
database = yourdatabasename
user = yourusername
password = "yourpassword"
host = host.name.com # Could be localhost, if the database is hosted on the local␣
↪machine
port = 3306 # Or whichever is the correct one for your setting
default-character-set = utf8 # This is pretty much required with django
```

After done with my.cnf settings, make sure that the file is not accessible by anyone else than you

```
$ chmod 600 my.cnf
```

If you have problems with access by apache, place your my.cnf in a place where it can be accessed, or play with the user rights in the current location.

Edit settings/secret_settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'OPTIONS': {
            'read_default_file': '/path/to/my.cnf',
            'init_command': 'SET storage_engine=INNODB',
        },
    }
}
```

Then install MySQL-python with pip when your virtual environment is activated.

```
$ pip install MySQL-python
```

On RHEL and CentOS you might need additional packages, if the pip installing of MySQL-python is not working, you might try to install mariadb-devel. For debian based distributions the package name might be different.

```
$ sudo yum install mariadb-devel
```

It might be required that you install MySQL-python again with pip. Remove it and install it again without using the cache.

```
$ pip uninstall MySQL-python
$ pip install MySQL-python --no-cache
```

Other settings

Change these settings in settings/base.py according to your needs

```
# settings/base.py
TIME_ZONE = 'Europe/Helsinki'
LANGUAGE_CODE = 'fi' # examples: 'en-us', 'de', 'se'

# Enter the desired languages under this setting. These languages can be translated␣
↪in the app.
LANGUAGES = (
    ('fi', _('Finnish')),
    ('en', _('English')),
)
```

Django debug toolbar

Using django debug toolbar is optional, but recommended as it is very useful for evaluating of the actual SQL queries for example.

To install django debug toolbar (while your virtual environment is active):

```
$ pip install django-debug-toolbar
```

If you don't want to use django debug toolbar, remove or comment out the following lines in settings/development.py:

```
if DEBUG:
    # Setting up debug toolbar.
    MIDDLEWARE.append('debug_toolbar.middleware.DebugToolbarMiddleware')
    INSTALLED_APPS += ('debug_toolbar',)
```

and also remove or comment out the following lines in signbank/urls.py:

```
if settings.DEBUG:
    import debug_toolbar
    from django.conf.urls.static import static
    # Add debug_toolbar when DEBUG=True, also add static+media folders when in␣
↪development.
    # DEBUG should be False when in production!
    urlpatterns += [
        url(r'^__debug__/', include(debug_toolbar.urls)),
    ] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)\
        + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Database migration

Once we have handled all the settings, we can migrate the database.

Make sure you are in your environment

```
$ source /path/to/venv/bin/activate
```

First create migrations for django flatpages app to add translation fields with django-modeltranslation:

```
$ python FinSL-signbank/bin/develop.py makemigrations
```

Then migrate:

```
$ python FinSL-signbank/bin/develop.py migrate
```

Load fixture for flatpages:

```
$ python FinSL-signbank/bin/develop.py loaddata flatpages_initial_data
```

Note: In MySQL you might need to change the default collation, if the utf8_general_ci doesn't match your languages alphabetical order. You might need to do this to all the tables of the signbank app (not on the ones that begin with django_ or auth_). Take a look at: http://dev.mysql.com/doc/refman/5.7/en/charset-unicode-sets.html and https://docs.djangoproject.com/en/stable/ref/databases/#collation-settings

Run djangos test/development server to see if it works

```
# Run locally, only accessible from the machine you are running signbank with
$ python FinSL-signbank/bin/develop.py runserver localhost:8000

# Or run in your network/internet by entering your IPaddress or your hostname
$ python FinSL-signbank/bin/develop.py runserver 80.12.16.10:8000 # Change the port␣
↪if needed
```

Apache (httpd)

Apache + mod_wsgi

This process can differ between linux distributions. Take a look at django documentation.

You can read about the settings in django documentation. These settings work with CentOS7 and apache httpd 2.4. The location of the configurations vary between linux distributions. It is important to note that you should definitely store FinSL-signbank and django files outside of the path your webserver serves to the web (f.ex. /var/www/), I suggest that you store the files inside your /home/ folder. This way you avoid the risk of your settings, code and files being accessible from the web. Your wsgi.py file should be located at FinSL-signbank/signbank/wsgi.py.

```
#/etc/httpd/conf/httpd.conf
# These lines set the WSGI directories for FinSL-signbank and django
WSGIScriptAlias / /path/to/FinSL-signbank/signbank/wsgi.py
WSGIDaemonProcess FinSL-signbank python-path=/path/to/FinSL-signbank:/path/to/FinSL-
↪signbank/venv/lib/python3.x/site-packages
WSGIProcessGroup FinSL-signbank

<Directory /path/to/FinSL-signbank/signbank>
    SetEnvIfNoCase Host your\.domain\.com VALID_HOST
    Require env VALID_HOST
    Options +FollowSymLinks -ExecCGI
    <Files wsgi.py>
        Require env VALID_HOST
    </Files>
</Directory>
```

(continues on next page)

(continued from previous page)

```
# Creates alias for /media as /static
# This will be the directory where static files are collected to, the web server␣
↪should serve them not django.
Alias /static /path/to/static # For example /var/www/yourdomain/static ,
# Sets robots.txt to be accessible at /robots.txt, you need to create the robots.txt␣
↪file to suit your needs
Alias /robots.txt /path/to/static/robots.txt
# Sets favicon.ico to be accessible at /favicon.ico, you need to create a favicon
Alias /favicon.ico /path/to/FinSL-signbank/favicon.ico

# Create alias for /media/ directory
Alias /media /path/to/media # For example /var/www/yourdomain/media
# Gives access to /media directory
<Directory /path/to/media>
    SetEnvIfNoCase Host your\.domain\.com VALID_HOST
    Require env VALID_HOST
</Directory>
```

Apache envvars

If you are running Signbank with apache (or probably any web server), make sure it is running on the right locale. For example in CentOS Apache seemed to run on LANG=C by default. To avoid problems with non-ascii characters, add these values to your web server evnvvars (in CentOS /etc/sysconfig/httpd):

```
LANG='en_US.UTF-8'
LC_ALL='en_US.UTF-8'
```

HTTPS

It is strongly recommended that you run your production server with HTTPS. For this take a look at the HTTPS specific settings in the settings files. Have a look at the django docs: https://docs.djangoproject.com/en/stable/topics/security/#ssl-https And also configure your domain properly for HTTPS. If you need free certificates check out LetsEncrypt at https://letsencrypt.org/.

1.1.2 Settings

Here you can find explanations for some of the settings, and how the settings files are distributed.

The different settings files

You can find all the settings files in the directory `signbank/settings/`.

There are several different files for different purposes:

- **base.py**: settings shared with all environments.
- **production.py**: production environment specific settings.
- **development.py**: development environment specific settings.
- **testing.py**: settings for tests.

- **settings_secret.py:**
  - Settings you don't want push to a public git repository.
  - `SECRET_KEY` and database passwords and such.

base.py

signbank.settings.base.**ACCOUNT_ACTIVATION_DAYS**
How many days a user has until activation time expires. Django-registration related setting.

signbank.settings.base.**AUTHENTICATION_BACKENDS**
A list of authentication backend classes (as strings) to use when attempting to authenticate a user.

signbank.settings.base.**INSTALLED_APPS**
A list of strings designating all applications that are enabled in this Django installation. Dotted Python path to: an application configuration class (preferred), or a package containing an application. The order of the apps matter!

signbank.settings.base.**LANGUAGES**
A list of all available languages. The list is a list of two-tuples in the format (language code, language name) - for example, ('ja', 'Japanese').

signbank.settings.base.**LANGUAGE_CODE**
A string representing the language code for this installation. This should be in standard language ID format. For example, U.S. English is "en-us".

signbank.settings.base.**LOGIN_REDIRECT_URL**
The URL where requests are redirected after login when the contrib.auth.login view gets no next parameter.

signbank.settings.base.**MIDDLEWARE**
A list of middleware classes to use. The order of middleware classes is critical!

signbank.settings.base.**REGISTRATION_OPEN**
A boolean indicating whether registration of new accounts is currently permitted.

signbank.settings.base.**STATICFILES_FINDERS**
The list of finder backends that know how to find static files in various locations.

signbank.settings.base.**TIME_ZONE**
A string representing the time zone for this installation.

signbank.settings.base.**USE_I18N**
A boolean that specifies whether Django's translation system should be enabled.

signbank.settings.base.**USE_L10N**
A boolean that specifies if localized formatting of data will be enabled by default or not.

signbank.settings.base.**USE_TZ**
A boolean that specifies if datetimes will be timezone-aware by default or not.

signbank.settings.base.**VIDEO_UPLOAD_LOCATION**
Location for upload of videos relative to MEDIA_ROOT, videos are stored here prior to copying over to the main storage location

development.py

signbank.settings.development.**DEBUG**
Debug should be True in development but not in production!

signbank.settings.development.**EMAIL_BACKEND**
To test emailing, use this to show emails in the console

signbank.settings.development.**LOCALE_PATHS**
A list of directories where Django looks for translation files.

production.py

signbank.settings.production.**ALLOWED_HOSTS**
IMPORTANT: The hostname that this signbank runs on, this prevents HTTP Host header attacks

signbank.settings.production.**CACHES**
Use Local-memory caching for specific views (if you have bigger needs, use something else).

signbank.settings.production.**DEBUG**
IMPORTANT: Debug should always be False in production

signbank.settings.production.**DO_LOGGING**
Turn off lots of logging.

signbank.settings.production.**EMAIL_BACKEND**
The backend to use for sending emails.

signbank.settings.production.**LOGGING**
A sample logging configuration. The only tangible logging performed by this configuration is to send an email to the site admins on every HTTP 500 error when DEBUG=False. See http://docs.djangoproject.com/en/stable/topics/logging for more details on how to customize your logging configuration.

signbank.settings.production.**MEDIA_ROOT**
Absolute filesystem path to the directory that will hold user-uploaded files.

signbank.settings.production.**STATIC_ROOT**
The absolute path to the directory where collectstatic will collect static files for deployment. Example: "/var/www/example.com/static/"

signbank.settings.production.**UPLOAD_ROOT**
Location and URL for uploaded files.

settings_secret.py

signbank.settings.settings_secret.**ADMINS**
A list of all the people who get code error notifications. When DEBUG=False and a view raises an exception, Django will email these people with the full exception information.

signbank.settings.settings_secret.**DATABASES**
A dictionary containing the settings for all databases to be used with Django.

signbank.settings.settings_secret.**DB_IS_PSQL**
Is the database engine used is postgresql?

signbank.settings.settings_secret.**PSQL_DB_QUOTA**
Maximum size of database in bytes, controlled outside of this application. Fill it in if you have a quota.

signbank.settings.settings_secret.**PSQL_DB_NAME**
The name of a database used.

signbank.settings.settings_secret.**DEFAULT_FROM_EMAIL**
Default email address to use for various automated correspondence from the site manager(s). Note: You can also use the following form 'Webmaster <webmaster@yourdomain.com>'

signbank.settings.settings_secret.**EMAIL_HOST**
: The host to use for sending email.

signbank.settings.settings_secret.**EMAIL_PORT**
: Port to use for the SMTP server defined in EMAIL_HOST.

signbank.settings.settings_secret.**SECRET_KEY**
: Make this unique, and don't share it with anybody. This is used to provide cryptographic signing.

signbank.settings.settings_secret.**SERVER_EMAIL**
: The email address that error messages come from, such as those sent to ADMINS and MANAGERS. Note: You can also use the following form 'Webmaster <webmaster@yourdomain.com>'

testing.py

The **testing.py** settings file currently only imports **development.py** settings. Edit this file to customize test settings when runnings tests with **bin/runtests.py**.

When is which settings file applied?

By default when creating a new django project, a **manage.py** file is created. It is used to run all the management commands, and it applies all the settings.

FinSL-signbank useses separate management files to make it easier to run management commands in different environments with different settings. You can find these files in the **bin/** folder:

Note: **base.py** holds shared settings, and is imported in every settings file. **settings_secret.py** is then imported in **base.py**.

- **develop.py**: to run the development environment with `development.py` settings.
- **production.py**: to run management commands with `production.py` settings.
- **runtests.py**: to run management commands with `testing.py` settings.

1.1.3 Translation

How to translate the user interface into a desired language, or how to edit current translations?

Translating the interface is handled with Django's internalization features, see django translation docs for more information.

Create a new language to translate to

Follow these instructions when you want to add a new language to translate to.

Add a new translation language

Before beginning translating the user interface into a new language, you need to add the new language into the `LANGUAGES` setting in **signbank/settings/base.py**.

```
# A list of all available languages. The list is a list of two-tuples in the
# format (language code, language name) - for example, ('ja', 'Japanese').
LANGUAGES = (
    ('fi', _('Finnish')),
    ('en', _('English')),
)
```

You can find the correct ISO 639-1 codes here: https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes

Make database migrations for django-modeltranslation

Django-modeltranslation is used to dynamically add translatable fields into models. See django-modeltranslation docs here: http://django-modeltranslation.readthedocs.io/en/latest/index.html

After you have added a new language to the `LANGUAGES` setting, run the following in the commandline in your **development** or **production** environment

In development environment:

```
$ python bin/development.py makemigrations
$ python bin/development.py migrate
```

In production environment:

```
$ python bin/production.py makemigrations
$ python bin/production.py migrate
```

Create or update translations

Create the translation file (.po)

You'll want to do this in your development environment.

```
# -i venv ignores the python virtual environment folder.
$ python bin/develop.py makemessages -i venv
```

For more information see: https://docs.djangoproject.com/en/stable/ref/django-admin/#makemessages

This command creates/updates the **django.po** files for all the `LANGUAGES`. These files will be created in **locale/<ISO 639-1 CODE>/LC_MESSAGES/django.po**

Write the translations

To write the translations, open the **django.po** file. For each `msgid "texthere"` there is a `msgstr ""` where you should place the translation of the text inside the quotes of msgstr.

For example the locale/fi/LC_MESSAGES/django.po for Finnish translations:

```
msgid ""
"You are not allowed to edit this comment, because you are not the author of "
"the comment."
msgstr "Et voi muokata tätä kommenttia, koska et ole kommentin kirjoittaja."
```

Once you have written the translations, make sure you put the new file on the server (overwrite the old one).

To activate the translations in the application, you have to run the following command which compiles the translation file :

In development:

```
$ python bin/develop.py compilemessages
```

In production:

```
$ python bin/production.py compilemessages
# Make the server reload FinSL-signbank to update the translations.
$ touch signbank/wsgi.py
```

For more information see: https://docs.djangoproject.com/en/stable/ref/django-admin/#compilemessages

Translate Flat pages

Open the edit page for the Flat page you want to edit. For each Flat page you should be able to translate the **title** and the **content** of the page. Each language should have their own field for their version of the page, e.g. **Title [fi]** and **Content [fi]**.

1.1.4 Changelog

1.0.0 - dd/mm/2018

- First release

1.1.5 Applications in FinSL-signbank

Applications that form FinSL-signbank.

Dictionary

A django app that handles data related to Lexicons and their Glosses.

Models

Django models for dictionary app.

Here are all the models that are actually used within the application. Some models are missing because they are not being used, but haven't yet been removed.

Models for the Signbank dictionary/corpus.

class signbank.dictionary.models.**AllowedTags**(**args*, ***kwargs*)

Tags a model is allowed to use.

allowed_tags

The tags that are shown in tag lists.

content_type
The ContentType of the object whose AllowedTags we set.

class signbank.dictionary.models.**Dataset**(**args*, ***kwargs*)
Dataset/Lexicon of which Glosses are part of.

admins
The admins of this Dataset. Admins receive notifications when a user applies for permissins for the Dataset.

copyright
The copyright statement for the data in this Dataset, the license used for the videos etc.

description
A description of the Dataset: who maintains it, what is its purpose, etc.

is_public
Boolean defining whether to show this Dataset in the public interface.

name
A private name for the Dataset. Can include abbrevations not recognizable by the general users.

public_name
Public name for the Dataset, intended for users of the public interface.

signlanguage
The Sign Language of the Glosses in this Dataset.

translation_languages
The translation equivalent languages that should be available to the Glosses of this Dataset.

class signbank.dictionary.models.**Dialect**(**args*, ***kwargs*)
A dialect name - a regional dialect of a given Language

description
Description of the Dialect.

language
The Language of the Dialect.

name
Name of the Dialect.

class signbank.dictionary.models.**FieldChoice**(*id*, *field*, *english_name*, *machine_value*)

english_name
English (verbose) name of the FieldChoice.

field
The name of the FieldChoice.

machine_value
Machine value of the FieldChoice, its ID number.

class signbank.dictionary.models.**Gloss**(*id*, *published*, *exclude_from_ecv*, *dataset*, *idgloss*, *idgloss_en*, *notes*, *created_at*, *created_by*, *updated_at*, *updated_by*, *handedness*, *strong_handshape*, *weak_handshape*, *location*, *relation_between_articulators*, *absolute_orientation_palm*, *absolute_orientation_fingers*, *relative_orientation_movement*, *relative_orientation_location*, *orientation_change*, *handshape_change*, *repeated_movement*, *alternating_movement*, *movement_shape*, *movement_direction*, *movement_manner*, *contact_type*, *phonology_other*, *mouth_gesture*, *mouthing*, *phonetic_variation*, *iconic_image*, *named_entity*, *semantic_field*, *number_of_occurences*)

created_at
The DateTime when the Gloss was created.

created_by
The User who created the Gloss.

dataset
The Dataset (Lexicon) this Gloss is part of.

dialect
One or more regional dialects that this Gloss is used in.

exclude_from_ecv
Boolean: Exclude this gloss from all ELAN externally controlled vocabularies (ECV)?

field_labels()
Return the dictionary of field labels for use in a template

static get_choice_lists()
Return FieldChoices for selected fields in JSON, grouped by field, key=machine_value, value=english_name

get_translation_languages()
Returns translation languages that are set for the Dataset of the Gloss.

get_translations_for_translation_languages()
Returns a zipped list of translation languages and translations.

idgloss
Gloss in Finnish. This is the unique identifying name of the Gloss.

idgloss_en
Gloss in English. This is the English name of the Gloss.

notes
Notes about the Gloss.

published
Boolean: Is this Gloss published in the public interface?

updated_at
The DateTime when the Glosses information was last updated.

updated_by
The User who last updated the Glosses information.

class signbank.dictionary.models.**GlossRelation**(**args*, ***kwargs*)
Relation between two glosses

source
The source Gloss of the Relation.

tag()
The type of the Relation, a Tag.

target
The target Gloss of the Relation, the Gloss to which the source Gloss related to.

class signbank.dictionary.models.**GlossTranslations**(**args*, ***kwargs*)
Store a string representation of translation equivalents of certain Language for a Gloss.

get_keywords()
Returns keywords parsed from self.translations.

get_keywords_unique()
Returns only unique keywords from get_keywords()

gloss
The Gloss to translate

language
The written/spoken Language of the translations.

translations
The fields that contains the translations, a text field.

class signbank.dictionary.models.**GlossURL**(**args*, ***kwargs*)
URL's for gloss

gloss
The Gloss the URL belongs to.

url
The URL, a websites address.

class signbank.dictionary.models.**Keyword**(**args*, ***kwargs*)
A keyword that stores the text for translation(s)

text
The text of a Keyword.

class signbank.dictionary.models.**Language**(**args*, ***kwargs*)
A written language, used for translations in written languages.

description
Description of the Language.

language_code_2char
The ISO 639-1 code of the Language (2 characters long).

language_code_3char
The ISO 639-3 code of the Language (3 characters long).

name
The name of a spoken/written Language.

class signbank.dictionary.models.**MorphologyDefinition**(**args*, ***kwargs*)
Tells something about morphology of a gloss

class signbank.dictionary.models.**Relation**(**args*, ***kwargs*)
A relation between two glosses

class signbank.dictionary.models.**RelationToForeignSign**(**args*, ***kwargs*)
Defines a relationship to another sign in another language (often a loan)

gloss
The source Gloss of the relation.

loan
Boolean: Is this a loan sign?

other_lang
The language of the related sign.

other_lang_gloss
The name of the Gloss in the related language.

class signbank.dictionary.models.**SignLanguage**(**args*, ***kwargs*)
A sign language.

language_code_3char
The ISO 639-3 code of the Sign Language (3 characters long).

name
The name of the Sign Language

class signbank.dictionary.models.**Translation**(**args*, ***kwargs*)
A translation equivalent of a sign in selected language.

gloss
The Gloss to translate.

keyword
The Keyword of the translation, the textual form.

language
The written/spoken Language of the translation.

order
The order number of the Translation within a Glosses Translations.

signbank.dictionary.models.**build_choice_list**(*field*)
This function builds a list of choices from FieldChoice.

signbank.dictionary.models.**model**
alias of *signbank.dictionary.models.GlossRelation*

Video

A django app that handles data all the videos.

Models

Django models for video app.

Models for the video application keep track of uploaded videos and converted versions

class signbank.video.models.**GlossVideo**(**args*, ***kwargs*)
A video that represents a particular idgloss

correct_duplicate_versions()
If glosses glossvideos have duplicate version numbers, reset version numbers.

create_filename()
Returns a correctly named filename

create_poster_filename(*ext*)
Returns a preferred filename of posterfile. Ext is the file extension without the dot.

dataset
The Dataset/Lexicon this GlossVideo is part of.

get_extension()
Returns videofiles extension.

get_glosses_videos()
Returns queryset of glosses GlossVideos.

get_videofile_modified_date()
Return a Datetime object from filesystems last modified time of path.

gloss
The Gloss this GlossVideo belongs to.

has_poster()
Returns true if the glossvideo has a poster file.

is_public
Boolean: Is this GlossVideo public? Do you want to show it in the public interface, for a published Gloss?

move_video_version(*direction*)
Move video back or forth in glosses videos.

next_version()
Return a next suitable version number.

posterfile
Poster image of the GlossVideo.

static rename_glosses_videos(*gloss*)
Renames the filenames of selected Glosses videos to match the Gloss name

rename_video()
Rename the video and move the video to correct path if the glossvideo object has a foreignkey to a gloss.

title
Descriptive title of the GlossVideo.

version
Version number of the GlossVideo within Glosses videos.

videofile
Video file of the GlossVideo.

class signbank.video.models.**GlossVideoStorage**(*location=u'/home/docs/checkouts/readthedocs.org/user_builds/fins signbank/checkouts/latest/media'*, *base_url=u'/media/'*)
Video storage, handles saving to directories based on filenames first two characters.

get_valid_name(*name*)
Generate a valid name, save videos to a 'base_directory', and under it use directories named for the first two characters in the filename to partition the videos

`url` (*name*)
Returns an absolute URL where the file's contents can be accessed directly by a Web browser.

1.1.6 Glossary

Allowed tags A model that controls which Tags are shown in Tag add lists for each types of objects.

CSV Comma Separated Values, a text file format used to import and export data from spreadsheets.

Dataset A model that holds the data for a *Lexicon*.

See *Models*.

Flat pages User editable content pages. These pages can be edited by admins with a WYSIWYG (what you see is what you get) editor.

Gloss Name of an entry of a Sign in a lexicon.

Gloss relation A relationship between two Glosses, stores a source Gloss and a target Gloss.

Gloss videos Videos to be attached to Glosses. Only the videos are able to show the form of a sign.

Lexicon A collection of Glosses. The name of the model in the application is *Dataset*, but it is also called a Lexicon in some places.

See *Models*.

Translation equivalent Translation of the Sign in the Gloss in some spoken language.

1.1.7 Features

- **Keep your sign language lexicon organized.**
  - Have as many lexicons as you like.
  - Control user permissions per lexicon.
  - Publish your lexicons to be viewed by the public.
- **Create Glosses and attach many kinds of data into them.**
  - Videos: Record videos with a webcam in the app, or simply upload from your computer.
  - Translation equivalents in any language(s) you want.
  - Relationships between Glosses.
  - Sign language, notes, URLs, comments, etc.
- Separate user interfaces for viewing and editing detailed gloss data, and a non-editable interface for the public.
- Export you lexicon to be used with annotation of videos with ELAN.
- **The user interface is translated into several languages.**
  - You can create translations for any language.
- View complete version history of Glosses and revert changes when needed.
- See a network graph of relationships between glosses per lexicon.

1.1.8 Requirements

- Python 3.4
- Django 1.11
- PostgreSQL (or MySQL, SQLite3)
- Apache+mod_wsgi (or nginx)

Python dependencies are listed in requirements.txt:

```
django==1.11.18
django-tagging==0.4.6
django-reversion==2.0.13
django-bootstrap3==11.0.0
django-summernote==0.8.8.8
django-modeltranslation==0.12.2
django-registration==2.4.1
django-contrib-comments==1.9.0
django-guardian==1.4.9
django-notifications-hq==1.4
```

JavaScript dependencies are listed in package.json:

```
{
  "name": "finsl-signbank",
  "version": "1.0.0",
  "description": "sign language lexicon database",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "collectjs": "cp -f ./node_modules/*/dist/*min.js ./signbank/static/js && cp -f ./
↪node_modules/recordrtc/RecordRTC.min.js ./signbank/static/js && cp -f ./node_
↪modules/at.js/dist/js/jquery.atwho.min.js ./signbank/static/js && cp -f ./node_
↪modules/cookieconsent/build/cookieconsent.min.js ./signbank/static/js && cp -f ./
↪node_modules/sigma/build/sigma.min.js ./signbank/static/js && cp -f ./node_modules/
↪sigma/build/plugins/sigma.layout.forceAtlas2.min.js ./signbank/static/js",
    "collectcss": "cp -f ./node_modules/at.js/dist/css/jquery.atwho.min.css ./
↪signbank/static/css && cp -rf ./node_modules/bootstrap/dist/* ./signbank/static/
↪bootstrap/ && cp -f ./node_modules/cookieconsent/build/cookieconsent.min.css ./
↪signbank/static/css"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/Signbank/FinSL-signbank.git"
  },
  "keywords": [
    "signlanguage"
  ],
  "author": "Henri Nieminen",
  "license": "BSD-3-Clause",
  "bugs": {
    "url": "https://github.com/Signbank/FinSL-signbank/issues"
  },
  "homepage": "https://github.com/Signbank/FinSL-signbank#readme",
  "dependencies": {},
  "devDependencies": {
    "TagManager": "git+https://github.com/max-favilli/tagmanager.git",
```

(continues on next page)

(continued from previous page)

```
    "at.js": "^1.5.4",
    "bootstrap": "^3.3.7",
    "jquery": "^3.3.1",
    "jquery.caret": "^0.3.1",
    "mark.js": "^8.11.1",
    "recordrtc": "^5.4.6",
    "sigma": "^1.2.1",
    "typeahead.js": "^0.11.1",
    "cookieconsent": "^3.0.6"
  }
}
```

1.1.9 Installation (in short)

1. Install with pip: `pip install finsl-signbank`.
2. Edit settings files.
3. Migrate: `python bin/develop.py migrate`

Note: See *Installation* and *Settings* for more detailed instructions.

CHAPTER 2

Indices, glossary and tables

- genindex
- modindex
- *Glossary*

Python Module Index

S

Index