django-helmholtz-aai

Philipp S. Sommer, Housam Dibeh, Hatef Takyar

Mar 02, 2022

CONTENTS:

This small generic Django app helps you connect to the Helmholtz AAI and make use of it's virtual organizations.

CHAPTER

ONE

FEATURES

Features include

- ready-to-use views for authentification against the Helmholtz AAI
- a new `HelmholtzUser` class based upon djangos `User` model and derived from the Helmholtz AAI
- a new `HelmholtzVirtualOrganization` class based upon djangos `Group` model and derived from the Helmholtz AAI
- several signals to handle the login of Helmholtz AAI user for your specific application
- automated synchronization of VOs of on user authentification

Get started by following the *installation instructions* and have a look into the *Configuration options*.

1.1 Installation

To install the *django-helmholtz-aai* package for your Django project, you need to follow three steps:

1. *Install the package*
2. *Register an OAuth-client*
3. *Add the app to your Django project*

1.1.1 Installation from PyPi

The recommended way to install this package is via pip and PyPi via:

```
pip install django-helmholtz-aai
```

Or install it directly from the source code repository on Gitlab via:

```
pip install git+https://gitlab.hzdr.de/hcdc/django/django-helmholtz-aai.git
```

The latter should however only be done if you want to access the development versions.

1.1.2 Register your OAuth-Client at the Helmholtz AAI

To install this app in your Django application, you first need to register an OAuth client for the Helmholtz AAI. In short, this works the following way

1. head over to https://login.helmholtz.de
2. make sure that you are logged out at the Helmholtz AAI
3. click *No Acccount? Sign up* on the top-right on , and then by
4. click on *Oauth2/OIDC client Registration*
5. register your client. For more information on the necessary fields, see [client-registration] in the Helmholtz AAI docs.

Note: Make sure that you enter the correct return URL which should be something like `https://<link-to-your-django-website>/helmholtz-aai/auth/`.

The `/helmholtz-aai/` part is determined by the settings in your URL configuration *down below*. But you can also change this URL or add more once your client has been approved at https://login.helmholtz.de/oauthhome/

1.1.3 Install the Django App for your project

To use the *django-helmholtz-aai* package in your Django project, you need to add the app to your *INSTALLED_APPS*, configure your *urls.py*, run the migration, add a login button in your templates. Here are the step-by-step instructions:

1. Add the *django_helmholtz_aai* app to your *INSTALLED_APPS*
2. in your projects urlconf (see `ROOT_URLCONF`), add include *`django_helmholtz_aai.urls`* via:

```
from django.urls import include, path

urlpatterns += [
    path("helmholtz-aai/", include("django_helmholtz_aai.urls")),
 ]
```

   Note that the `helmholtz-aai/`-part has to match what you entered when you registered your client (see *above*).
3. Run `python manage.py migrate` to add the *`HelmholtzUser`* and *`HelmholtzVirtualOrganization`* models to your database
4. Add the link to the login view in one of your templates (e.g. in the *login.html* template from your `LOGIN_URL`), e.g. via

```
{% load helmholtz_aai %}

<a href="{% helmholtz_login_url %}">
  login via Helmholtz AAI
</a>
```

Note: To tell the user why he or should could not login, we are also using djangos `messaging` framework. See `django.contrib.messages`. To display these messages, you should add something in your django template, e.g. something like

```
{% if messages %}
   <ul class="messages">
     {% for message in messages %}
       <li{% if message.tags %} class="{{ message.tags }}"{% endif %}>
         {{ message }}
       </li>
     {% endfor %}
   </ul>
{% endif %}
```

5. Make sure to set the *HELMHOLTZ_CLIENT_ID* and *HELMHOLTZ_CLIENT_SECRET* settings in your *settings.py* with the username and password you specified during the *client registration*.

That's it! For further adaption to you Django project, please head over to the *Configuration options*. You can also have a look into the `testproject` in the source code repository for a possible implementation.

1.1.4 References

1.2 Configuration options

1.2.1 Configuration settings

Most important settings

HELMHOLTZ_ALLOWED_VOS	A string of lists specifying which VOs are allowed to log into the website.
HELMHOLTZ_CLIENT_ID	Client id for the Helmholtz AAI
HELMHOLTZ_CLIENT_SECRET	Client secret for the Helmholtz AAI

Two settings are necessary to use this package, this is the *HELMHOLTZ_CLIENT_ID* and the *HELMHOLTZ_CLIENT_SECRET* that you specified during the OAuth-Client registration (see *Register your OAuth-Client at the Helmholtz AAI*).

By default, the website allows all users to login and create an account via the Helmholtz AAI. This if often not desired and you can modify this with the *HELMHOLTZ_ALLOWED_VOS* setting, e.g. something like:

```
HELMHOLTZ_ALLOWED_VOS = [
    "urn:geant:helmholtz.de:group:hereon#login.helmholtz.de",
]
```

in your `settings.py`.

Other settings

Further settings can be used to specify how to connect to the helmholtz AAI and how to interpret the userinfo of the Helmholtz AAI.

HELMHOLTZ_AAI_CONF_URL	openid configuration url of the Helmholtz AAI
HELMHOLTZ_ALLOWED_VOS_REGEXP	Regular expressions for VOs that are allowed to login to the website.
HELMHOLTZ_CLIENT_KWS	Keyword argument for the oauth client to connect with the helmholtz AAI.
HELMHOLTZ_CREATE_USERS	Flag to enable/disable user account creation via the Helmholtz AAI.
HELMHOLTZ_EMAIL_DUPLICATES_ALLOWED	Allow duplicated emails for users in the website
HELMHOLTZ_MAP_ACCOUNTS	Flag whether existing user accounts should be mapped
HELMHOLTZ_UPDATE_USERNAME	Flag whether usernames should be updated from the Helmholtz AAI
HELMHOLTZ_USERNAME_FIELDS	Username fields in the userinfo

1.2.2 Customizing the login

If you are using the Helmholtz AAI, you likely want to combine it with the permission system of your Django project. You may want to set the *is_staff* attribute for users of a specific VO, or perform additional actions when a user logged in for the first time (e.g. send a welcome mail), enters or leaves a VO.

To perfectly adjust the django-helmholtz-aai framework to your projects need, you have two choices:

1. connect to the signals of the `signals` module, see *Configuration via Signals*
2. subclass the `HelmholtzAuthentificationView` view, see *Customization via the HelmholtzAuthentificationView*

The signals are the recommended way as they provide a more stable interface. As the *django-helmholtz-aai* is very new, we cannot guarantee that there won't be breaking changes in the `HelmholtzAuthentificationView`.

Configuration via Signals

The `signals` module defines various signal that are fired on different events:

`aai_user_created`	Signal that is fired when a user has been created via the Helmholtz AAI
`aai_user_logged_in`	Signal that is fired when a user logs in via the Helmholtz AAI
`aai_user_updated`	Signal that is fired when a user receives an update via the Helmholtz AAI
`aai_vo_created`	Signal that is fired if a new Virtual Organization has been created
`aai_vo_entered`	Signal that is fired if a Helmholtz AAI user enteres a VO
`aai_vo_left`	Signal that is fired if a Helmholtz AAI user left a VO

The purpose of these signals should be pretty much self-explanatory.

Examples

Suppose you want users of a specific VO to become superusers. Then you can do something like this using the *aai_vo_entered* and *aai_vo_left* signals:

```
from django.dispatch import receiver

from django_helmholtz_aai import models, signals

@receiver(signals.aai_vo_entered)
def on_vo_enter(
        sender,
        vo: models.HelmholtzVirtualOrganization,
        user: models.HelmholtzUser,
        **kwargs,
    ):
    vo_id = "urn:geant:helmholtz.de:group:hereon#login.helmholtz.de"
    if vo.eduperson_entitlement == vo_id:
        user.is_superuser = True
        user.save()


@receiver(signals.aai_vo_left)
def on_vo_leave(
        sender,
        vo: models.HelmholtzVirtualOrganization,
        user: models.HelmholtzUser,
        **kwargs,
    ):
    vo_id = "urn:geant:helmholtz.de:group:hereon#login.helmholtz.de"
    if vo.eduperson_entitlement == vo_id:
        user.is_superuser = False
        user.save()
```

Let's say you want to display a message in the frontend when a user logged in for the first time. Here you can use the *aai_user_created* signal:

```
from django.contrib import messages

from django_helmholtz_aai import models, signals

@receiver(signals.aai_user_created)
def created_user(
    sender,
    user: models.HelmholtzUser,
    request,
    **kwargs,
):
    messages.add_message(
        request, messages.success, f"Welcome on board {user}!"
    )
```

Customization via the `HelmholtzAuthentificationView`

Warning: Please bear in mind that this python package is still very new and we cannot guarantee that there won't be breaking changes in the *HelmholtzAuthentificationView* class.

Another way to customize the login is via the *HelmholtzAuthentificationView*. Your starting point should be the following two methods, one for checking the permissions and one for performing the request:

get(request)	Login the Helmholtz AAI user and update the data.
has_permission()	Check if the user has permission to login.

For a more fine-grained control of the authentification (such as user creation or update), you can make use of the following methods and reimplement to your needs.

create_user(userinfo)	Create a Django user for a Helmholtz AAI User.
login_user(user)	Login the Helmholtz AAI user to the Django Application.
synchronize_vos()	Synchronize the memberships in the virtual organizations.
update_user()	Update the user from the userinfo provided by the Helmholtz AAI.

Example

Let's say you want to approve users before you let them login to the website. One possibility is, to create a custom model with reference to a user and reimplement the *django_helmholtz_aai.views.HelmholtzAuthentificationView.login_user()*. Your custom app that reimplements this view then might look like

- models.py

```
from django.db import models
from django_helmholtz_aai.models import HelmholtzUser


class HelmholtzUserReview(models.Model):
    """A review of a helmholtz user"""

    class ReviewStatus(models.TextChoices):

        accepted = "accepted"
        rejected = "rejected"

    user = models.OneToOneField(HelmholtzUser, on_delete=models.CASCADE)

    review_status = models.CharField(
        choices=ReviewStatus.choices, blank=True, null=True
    )
```

- views.py

```python
from django.contrib import messages
from django_helmholtz_aai.views import HelmholtzAuthentificationView
from django_helmholtz_aai.models import HelmholtzUser
from .models import HelmholtzUserReview


class CustomHelmholtzAuthentificationView(HelmholtzAuthentificationView):
    def login_user(self, user: HelmholtzUser):
        review = HelmholtzUserReview.objects.get_or_create(user=user)[0]
        if (
            review.review_status
            == HelmholtzUserReview.ReviewStatus.accepted
        ):
            super().login_user(user)
        elif (
            review.review_status
            == HelmholtzUserReview.ReviewStatus.rejected
        ):
            messages.add_message(
                self.request,
                messages.error,
                f"Your account creation request has been rejected.",
            )
        else:
            messages.add_message(
                self.request,
                messages.success,
                f"Your account creation request is currently under review.",
            )
```

- urls.py

```python
from django.urls import include, path
from .views import CustomHelmholtzAuthentificationView

urlpatterns = [
    path(
        "helmholtz-aai/auth/",
        CustomHelmholtzAuthentificationView.as_view(),
    ),
    path("helmholtz-aai/", include("django_helmholtz_aai.urls")),
]
```

1.3 Common problems

In this document, we collect common problems and questions. If you cannot find your issue documented in here, you should create an issue at the source code repository and we'll try to find a solution and update this document with your problem.

1.3.1 Mapping to existing accounts

When you add this app to an existing django project, you might already have accounts in your database. If this is the case, you should have a look into the *HELMHOLTZ_MAP_ACCOUNTS* configuration variable.

1.3.2 Mapping of multiple accounts

One user can have multiple accounts in the Helmholtz AAI. You can, for instance create an account via GitHub and through your home institution. Both accounts can have the same email address. The Helmholtz AAI however treats them as separate accounts and both have different unique IDs and belong to different VOs. As we use the ID for mapping a user in the Helmholtz AAI to a user in Django, and we synchronize the VOs of the user in the Helmholtz AAI, we have to distinguish the two accounts as well.

As an example: One user can register two accounts in the Helmholtz AAI:

1. one via Google
2. one via GitHub but with the same Google-Mail

Then the user logs in to your project via the Helmholtz AAI and his Google account. If the user then logs in to your project via GitHub, this creates a new account, independent from the first one.

Usually you do not want to have this behaviour as both user-accounts will then have the same email-address. Therefore this is disabled by default. However, you can allow the creation of multiple user accounts using the *HELMHOLTZ_EMAIL_DUPLICATES_ALLOWED* configuration variable.

1.3.3 Too many VOs

Each time a user account is created, we create the VOs that the user participates in. These VOs remain, even if one deletes the user account. To remove these empty VOs, we therefore added the *remove_empty_vos* management command that you can use via `python manage.py remove_empty_vos`. Or you call it directly from python, e.g. via:

```
from django_helmholtz_aai import models
models.HelmholtzVirtualOrganization.objects.remove_empty_vos()
```

1.4 API Reference

1.4.1 App settings

This module defines the settings options for the `django_helmholtz_aai` app.

Data:

HELMHOLTZ_AAI_CONF_URL	openid configuration url of the Helmholtz AAI
HELMHOLTZ_ALLOWED_VOS	A string of lists specifying which VOs are allowed to log into the website.
HELMHOLTZ_ALLOWED_VOS_REGEXP	Regular expressions for VOs that are allowed to login to the website.
HELMHOLTZ_CLIENT_ID	Client id for the Helmholtz AAI
HELMHOLTZ_CLIENT_KWS	Keyword argument for the oauth client to connect with the helmholtz AAI.
HELMHOLTZ_CLIENT_SECRET	Client secret for the Helmholtz AAI
HELMHOLTZ_CREATE_USERS	Flag to enable/disable user account creation via the Helmholtz AAI.
HELMHOLTZ_EMAIL_DUPLICATES_ALLOWED	Allow duplicated emails for users in the website
HELMHOLTZ_MAP_ACCOUNTS	Flag whether existing user accounts should be mapped
HELMHOLTZ_UPDATE_USERNAME	Flag whether usernames should be updated from the Helmholtz AAI
HELMHOLTZ_USERNAME_FIELDS	Username fields in the userinfo

django_helmholtz_aai.app_settings.**HELMHOLTZ_AAI_CONF_URL = 'https://login.helmholtz.de/oauth2/.well-known/openid-configuration'**

openid configuration url of the Helmholtz AAI

Can also be overwritten using the *HELMHOLTZ_CLIENT_KWS* setting.

django_helmholtz_aai.app_settings.**HELMHOLTZ_ALLOWED_VOS: list[str] = []**

A string of lists specifying which VOs are allowed to log into the website.

By default, this is an empty list meaning that each and every user is allowed to login via the Helmholtz AAI. Each string in this list will be interpreted as a regular expression and added to *HELMHOLTZ_ALLOWED_VOS_REGEXP*

Examples

Assume you only want to allow people from the Hereon VO to login to the website. Then you can add the following to your `settings.py`:

```
HELMHOLTZ_ALLOWED_VOS = [
    "urn:geant:helmholtz.de:group:hereon#login.helmholtz.de",
]
```

or use a regex, e.g. something like:

```
HELMHOLTZ_ALLOWED_VOS = [
    r".*helmholtz.de:group:hereon#login.helmholtz.de",
]
```

django_helmholtz_aai.app_settings.**HELMHOLTZ_ALLOWED_VOS_REGEXP: list[re.Pattern] = []**

Regular expressions for VOs that are allowed to login to the website.

This attribute is created from the *HELMHOLTZ_ALLOWED_VOS* setting.

django_helmholtz_aai.app_settings.**HELMHOLTZ_CLIENT_ID: str = None**

Client id for the Helmholtz AAI

This is the username you use to login at https://login.helmholtz.de/oauthhome/, see [client-registration] for how to create a client

See also:

HELMHOLTZ_CLIENT_SECRET

django_helmholtz_aai.app_settings.**HELMHOLTZ_CLIENT_KWS = {'client_id': None, 'client_kwargs': {'scope': 'profile email eduperson_unique_id'}, 'client_secret': None, 'server_metadata_url': 'https://login.helmholtz.de/oauth2/.well-known/openid-configuration'}**

Keyword argument for the oauth client to connect with the helmholtz AAI.

Can also be overwritten using the *HELMHOLTZ_CLIENT_KWS* setting.

django_helmholtz_aai.app_settings.**HELMHOLTZ_CLIENT_SECRET: str = None**

Client secret for the Helmholtz AAI

This is the password you use to login at https://login.helmholtz.de/oauthhome/, see[client-registration]_ for how to create a client

See also:

HELMHOLTZ_CLIENT_ID

django_helmholtz_aai.app_settings.**HELMHOLTZ_CREATE_USERS: bool = True**

Flag to enable/disable user account creation via the Helmholtz AAI.

Use this flag if you want the Helmholtz AAI to create users when they login for the first time. This is enabled by default.

If you disable this setting, you should enable the *HELMHOLTZ_MAP_ACCOUNTS*, otherwise nobody will be allowed to login via the Helmholtz AAI.

django_helmholtz_aai.app_settings.**HELMHOLTZ_EMAIL_DUPLICATES_ALLOWED: bool = False**

Allow duplicated emails for users in the website

This setting controls if a user can register with multiple accounts from the Helmholtz AAI. An email is not unique in the AAI, but this might be desired in the Django application. This option prevents a user to create an account if the email has already been taken by some other user from the Helmholtz AAI

django_helmholtz_aai.app_settings.**HELMHOLTZ_MAP_ACCOUNTS: bool = False**

Flag whether existing user accounts should be mapped

Use this flag, if you want to map existing user accounts by their email address.

Examples

Suppose you just install django-helmholtz-aai to your already existing Django project and there exists already a user with the mail `user@example.com`. If this user now logs into your project, it would create a new *HelmholtzUser* which is probably not desired. To overcome this, you can set the *HELMHOLTZ_MAP_ACCOUNTS* configuration variable to `True` and the *HelmholtzUser* will be mapped to the already existing `User`

django_helmholtz_aai.app_settings.**HELMHOLTZ_UPDATE_USERNAME: bool = True**

Flag whether usernames should be updated from the Helmholtz AAI

Use this setting to control, whether the usernames are updated automatically on every login. If this is true, we will check the fields specified in the *HELMHOLTZ_USERNAME_FIELDS* setting variable on every login and update the username accordingly. If the user, for instance, changes his or her `preferred_username` on https://login.helmholtz.de/, we will update the username of the django user as well (if `preferred_username` is in the *HELMHOLTZ_USERNAME_FIELDS*).

django_helmholtz_aai.app_settings.**HELMHOLTZ_USERNAME_FIELDS: list[str] = ['preferred_username', 'eduperson_unique_id']**

Username fields in the userinfo

This setting determines how to get the username. By default, we use the `preferred_username` that the user can configure at https://login.helmholtz.de/oauthhome. If this is already taken, we use the unique `eduperson_unique_id` from the Helmholtz AAI. You can add more variables to this list but you should always include the `eduperson_unique_id` to make sure you do not end up with duplicated usernames.

Examples

You can use the email instead of the `preferred_username` via:

```
HELMHOLTZ_USERNAME_FIELDS = ["email", "eduperson_unique_id"]
```

1.4.2 Signals

This module defines the signals that are fired by the views in *django_helmholtz_aai.views* module.

django_helmholtz_aai.signals.aai_user_created = <django.dispatch.dispatcher.Signal object>

Signal that is fired when a user has been created via the Helmholtz AAI

This signal is called by the *HelmholtzAuthentificationView* when a new user has been created. Subscribers to this signal can accept the following parameters.

Parameters

- **sender** (*Type[django_helmholtz_aai.models.HelmholtzUser]*) – The type who sent the signal (implemented for reasons of convention)
- **user** (*django_helmholtz_aai.models.HelmholtzUser*) – The new user that has been created
- **request** (*Request*) – The request holding the session of the user.
- **userinfo** (*Dict[str, Any]*) – The userinfo as obtained from the Helmholtz AAI

See also:

django_helmholtz_aai.views.HelmholtzAuthentificationView.create_user

django_helmholtz_aai.signals.aai_user_logged_in = <django.dispatch.dispatcher.Signal object>

Signal that is fired when a user logs in via the Helmholtz AAI

This signal is called by the *HelmholtzAuthentificationView* when a user logged in via the Helmholtz AAI. Subscribers to this signal can accept the following parameters.

Parameters

- **sender** (*Type[django_helmholtz_aai.models.HelmholtzUser]*) – The type who sent the signal (implemented for reasons of convention)
- **user** (*django_helmholtz_aai.models.HelmholtzUser*) – The user who just logged in
- **request** (*Request*) – The request holding the session of the user.
- **userinfo** (*Dict[str, Any]*) – The userinfo as obtained from the Helmholtz AAI

See also:

django_helmholtz_aai.login, *django_helmholtz_aai.views.HelmholtzAuthentificationView.login_user*

django_helmholtz_aai.signals.**aai_user_updated = <django.dispatch.dispatcher.Signal object>**

Signal that is fired when a user receives an update via the Helmholtz AAI

This signal is called by the *HelmholtzAuthentificationView* when a user who does already have an account gets updated, e.g. because the email of the `preferred_username` changed in the Helmholtz AAI. Subscribers to this signal can accept the following parameters.

Parameters

- **sender** (*Type[*django_helmholtz_aai.models.HelmholtzUser*]*) – The type who sent the signal (implemented for reasons of convention)
- **user** (django_helmholtz_aai.models.HelmholtzUser) – The user that is supposed to be updated
- **request** (*Request*) – The request holding the session of the user.
- **userinfo** (*Dict[str, Any]*) – The userinfo as obtained from the Helmholtz AAI

See also:

django_helmholtz_aai.views.HelmholtzAuthentificationView.update_user

django_helmholtz_aai.signals.**aai_vo_created = <django.dispatch.dispatcher.Signal object>**

Signal that is fired if a new Virtual Organization has been created

This signal is called by the *HelmholtzAuthentificationView* when a new virtual organization has been created from the Helmholtz AAI because a of this VO registered on the website. Subscribers to this signal can accept the following parameters.

Parameters

- **sender** (*Type[*django_helmholtz_aai.models.HelmholtzUser*]*) – The type who sent the signal (implemented for reasons of convention)
- **user** (django_helmholtz_aai.models.HelmholtzUser) – The user that is about to become a member of the new VO
- **vo** (django_helmholtz_aai.models.HelmholtzVirtualOrganization) – The VO that has just been created
- **request** (*Request*) – The request holding the session of the user.
- **userinfo** (*Dict[str, Any]*) – The userinfo as obtained from the Helmholtz AAI

See also:

django_helmholtz_aai.views.HelmholtzAuthentificationView.synchronize_vos

django_helmholtz_aai.signals.**aai_vo_entered = <django.dispatch.dispatcher.Signal object>**

Signal that is fired if a Helmholtz AAI user enteres a VO

This signal is called by the *HelmholtzAuthentificationView* when a user enters a virtual organization as the user is a member in the Helmholtz AAI. Subscribers to this signal can accept the following parameters.

Parameters

- **sender** (*Type[*django_helmholtz_aai.models.HelmholtzUser*]*) – The type who sent the signal (implemented for reasons of convention)
- **user** (django_helmholtz_aai.models.HelmholtzUser) – The user that entered the VO.
- **vo** (django_helmholtz_aai.models.HelmholtzVirtualOrganization) – The VO that the user has just entered

- **request** (*Request*) – The request holding the session of the user.
- **userinfo** (*Dict[str, Any]*) – The userinfo as obtained from the Helmholtz AAI

See also:

django_helmholtz_aai.views.HelmholtzAuthentificationView.synchronize_vos

django_helmholtz_aai.signals.**aai_vo_left = <django.dispatch.dispatcher.Signal object>**
Signal that is fired if a Helmholtz AAI user left a VO

This signal is called by the *HelmholtzAuthentificationView* when a user leaves a virtual organization as the user is not anymore a member in the Helmholtz AAI. Subscribers to this signal can accept the following parameters.

Parameters

- **sender** (*Type[django_helmholtz_aai.models.HelmholtzUser]*) – The type who sent the signal (implemented for reasons of convention)
- **user** (django_helmholtz_aai.models.HelmholtzUser) – The user that entered the VO.
- **vo** (django_helmholtz_aai.models.HelmholtzVirtualOrganization) – The VO that the user has just entered
- **request** (*Request*) – The request holding the session of the user.
- **userinfo** (*Dict[str, Any]*) – The userinfo as obtained from the Helmholtz AAI

See also:

django_helmholtz_aai.views.HelmholtzAuthentificationView.synchronize_vos

1.4.3 URL config

URL patterns of the django-helmholtz-aai to be included via:

```
from django.urls import include, path

urlpatters = [
    path("helmholtz-aai/", include("django_helmholtz_aai.urls")),
]
```

Data:

app_name	App name for the django-helmholtz-aai to be used in calls to django.urls.reverse()
urlpatterns	urlpattern for the Helmholtz AAI

django_helmholtz_aai.urls.**app_name = 'django_helmholtz_aai'**
App name for the django-helmholtz-aai to be used in calls to django.urls.reverse()

django_helmholtz_aai.urls.**urlpatterns = [<URLPattern 'login/' [name='login']>, <URLPattern 'auth/' [name='auth']>]**
urlpattern for the Helmholtz AAI

1.4.4 Models

Models to mimic users and virtual organizations of the Helmholtz AAI in Django.

Models:

HelmholtzUser(*args, **kwargs)	A User in the in the Helmholtz AAI.
HelmholtzVirtualOrganization(*args, **kwargs)	A VO in the Helmholtz AAI.

Classes:

HelmholtzUserManager(*args, **kwargs)	A manager for the helmholtz User.
HelmholtzVirtualOrganizationManager(*args, ...)	Database manager for the *HelmholtzVirtualOrganization* model.
HelmholtzVirtualOrganizationQuerySet([...])	A queryset with an extra command to remove empty VOs.

class django_helmholtz_aai.models.**HelmholtzUser**(**args*, ***kwargs*)

Bases: django.contrib.auth.models.User

A User in the in the Helmholtz AAI.

Parameters

- **id** (*AutoField*) – Id
- **password** (*CharField*) – Password
- **last_login** (*DateTimeField*) – Last login
- **is_superuser** (*BooleanField*) – Superuser status. Designates that this user has all permissions without explicitly assigning them.
- **username** (*CharField*) – Username. Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.
- **first_name** (*CharField*) – First name
- **last_name** (*CharField*) – Last name
- **email** (*EmailField*) – Email address
- **is_staff** (*BooleanField*) – Staff status. Designates whether the user can log into this admin site.
- **is_active** (*BooleanField*) – Active. Designates whether this user should be treated as active. Unselect this instead of deleting accounts.
- **date_joined** (*DateTimeField*) – Date joined
- **groups** (*ManyToManyField*) – Groups. The groups this user belongs to. A user will get all permissions granted to each of their groups.
- **user_permissions** (*ManyToManyField*) – User permissions. Specific permissions for this user.
- **user_ptr** (OneToOneField to User) – User ptr
- **eduperson_unique_id** (*CharField*) – Eduperson unique id

Miscellaneous:

DoesNotExist	
MultipleObjectsReturned	

Model Fields:

eduperson_unique_id	**Model field:** eduperson unique id
user_ptr	**Model field:** user ptr, accesses the User model.

Attributes:

objects	
user_ptr_id	**Model field:** user ptr

exception DoesNotExist
Bases: django.contrib.auth.models.User.DoesNotExist

exception MultipleObjectsReturned
Bases: django.contrib.auth.models.User.MultipleObjectsReturned

eduperson_unique_id
Model field: eduperson unique id

objects = <django_helmholtz_aai.models.HelmholtzUserManager object>

user_ptr
Model field: user ptr, accesses the User model.

user_ptr_id
Model field: user ptr

class django_helmholtz_aai.models.**HelmholtzUserManager**(**args*, ***kwargs*)
Bases: django.contrib.auth.models.UserManager

A manager for the helmholtz User.

Methods:

create_aai_user(userinfo)	Create a user from the Helmholtz AAI userinfo.

create_aai_user(*userinfo*)
Create a user from the Helmholtz AAI userinfo.

class django_helmholtz_aai.models.**HelmholtzVirtualOrganization**(**args*, ***kwargs*)
Bases: django.contrib.auth.models.Group

A VO in the Helmholtz AAI.

Parameters

- **id** (*AutoField*) – Id
- **name** (*CharField*) – Name
- **permissions** (*ManyToManyField*) – Permissions
- **group_ptr** (OneToOneField to Group) – Group ptr

- **eduperson_entitlement** (*CharField*) – Eduperson entitlement

Miscellaneous:

DoesNotExist
MultipleObjectsReturned

Attributes:

display_name	
group_ptr_id	**Model field:** group ptr
objects	

Model Fields:

eduperson_entitlement	**Model field:** eduperson entitlement
group_ptr	**Model field:** group ptr, accesses the Group model.

exception DoesNotExist
Bases: django.contrib.auth.models.Group.DoesNotExist

exception MultipleObjectsReturned
Bases: django.contrib.auth.models.Group.MultipleObjectsReturned

property display_name: str

eduperson_entitlement
Model field: eduperson entitlement

group_ptr
Model field: group ptr, accesses the Group model.

group_ptr_id
Model field: group ptr

objects = <django_helmholtz_aai.models.HelmholtzVirtualOrganizationManager object>

class django_helmholtz_aai.models.**HelmholtzVirtualOrganizationManager**(**args*, ***kwargs*)
Bases: django.db.models.manager.GroupManagerFromHelmholtzVirtualOrganizationQuerySet

Database manager for the *HelmholtzVirtualOrganization* model.

class django_helmholtz_aai.models.**HelmholtzVirtualOrganizationQuerySet**(*model=None*, *query=None*, *using=None*, *hints=None*)
Bases: django.db.models.query.QuerySet

A queryset with an extra command to remove empty VOs.

Methods:

remove_empty_vos([exclude, without_confirmation])	Remove empty virtual organizations.

remove_empty_vos(*exclude: list[str] = []*, *without_confirmation: bool = True*) → list[*HelmholtzVirtualOrganization*]

Remove empty virtual organizations.

This method filters for virtual organizations in the queryset and removes them.

Parameters

- **exclude** (*list[str]*) – A list of strings that will be interpreted as regular expressions. If a *eduperson_entitlement* matches any of these strings, it will not be removed.
- **without_confirmation** (*bool*) – If True (default), remove the VO without asking for confirmation using python's built-in `input()` from the command-line.

Returns The list of virtual organizations that have been removed

Return type list[*HelmholtzVirtualOrganization*]

1.4.5 Views

Views of the django_helmholtz_aai app to be imported via the url config (see *django_helmholtz_aai.urls*). We define two views here: The *HelmholtzLoginView* that redirects to the Helmholtz AAI, and the *HelmholtzAuthentificationView* that handles the user login after successful login at the Helmholtz AAI.

Classes:

HelmholtzAuthentificationView(**kwargs)	Authentification view for the Helmholtz AAI.
HelmholtzLoginView(**kwargs)	A login view for the Helmholtz AAI that forwards to the OAuth login.

class django_helmholtz_aai.views.**HelmholtzAuthentificationView**(***kwargs*)

Bases: `django.contrib.auth.mixins.PermissionRequiredMixin`, `django.views.generic.base.View`

Authentification view for the Helmholtz AAI.

Classes:

PermissionDeniedReasons(value)	Reasons why permissions are denied to login.

Attributes:

aai_user	
is_new_user	True if the Helmholtz AAI user has never logged in before.
permission_denied_message_templates	Message templates that explain why a user is not allowed to login.
permission_denied_reason	The reason why the user cannot login.
userinfo	The userinfo as obtained from the Helmholtz AAI.

Methods:

create_user(userinfo)	Create a Django user for a Helmholtz AAI User.
get(request)	Login the Helmholtz AAI user and update the data.
get_permission_denied_message()	Get the permission denied message for a specific reason.
handle_no_permission()	Handle the response if the permission has been denied.
has_permission()	Check if the user has permission to login.
login_user(user)	Login the Helmholtz AAI user to the Django Application.
synchronize_vos()	Synchronize the memberships in the virtual organizations.
update_user()	Update the user from the userinfo provided by the Helmholtz AAI.

class PermissionDeniedReasons(*value*)

Bases: str, enum.Enum

Reasons why permissions are denied to login.

Attributes:

cannot_find_user	a user with the given email could not be found
email_changed_and_taken	the email changed and is already taken on the website
email_exists	the user is new and the email already exists
email_not_verified	the email has not yet been verified
new_user	the user is new and user creation is disabled by *HELMHOLTZ_CREATE_USERS*
vo_not_allowed	the virtual organization is not part of *HELMHOLTZ_ALLOWED_VOS_REGEXP*

cannot_find_user = 'cannot_find_user'

a user with the given email could not be found

email_changed_and_taken = 'email_changed_and_taken'

the email changed and is already taken on the website

email_exists = 'email_exists'

the user is new and the email already exists

email_not_verified = 'email_not_verified'

the email has not yet been verified

new_user = 'new_user'

the user is new and user creation is disabled by *HELMHOLTZ_CREATE_USERS*

vo_not_allowed = 'vo_not_allowed'

the virtual organization is not part of *HELMHOLTZ_ALLOWED_VOS_REGEXP*

aai_user: *models.HelmholtzUser*

create_user(*userinfo: Dict[str, Any]*) → *django_helmholtz_aai.models.HelmholtzUser*

Create a Django user for a Helmholtz AAI User.

This method uses the *create_aai_user()* to create a new user.

Notes

Emits the *aai_user_created* signal

get(*request*)
Login the Helmholtz AAI user and update the data.

This method logs in the aai user (or creates one if it does not exist already). Afterwards we update the user info from the information on the Helmholtz AAI using the *update_user()* and *synchronize_vos()* methods.

get_permission_denied_message()
Get the permission denied message for a specific reason.

This method is called by the super-classes *handle_no_permission()* method.

handle_no_permission()
Handle the response if the permission has been denied.

This reimplemented method adds the `permission_denied_message` to the messages of the request using djangos messaging framework.

has_permission() → bool
Check if the user has permission to login.

This method checks, if the user belongs to the specified *HELMHOLTZ_ALLOWED_VOS* and verifies that the email does not exist (if this is desired, see *HELMHOLTZ_EMAIL_DUPLICATES_ALLOWED* setting).

is_new_user
True if the Helmholtz AAI user has never logged in before.

login_user(*user:* django_helmholtz_aai.models.HelmholtzUser)
Login the Helmholtz AAI user to the Django Application.

Login is done via the top-level *django_helmholtz_aai.login()* function.

Notes

Emits the *aai_user_logged_in* signal

permission_denied_message_templates: dict[*PermissionDeniedReasons*, str] = {PermissionDeniedReasons.cannot_find_user: 'A user with the email {email} is not available on this website and the account creation is disabled. Please sign up or contact the website administrators.', PermissionDeniedReasons.email_changed_and_taken: 'You email in the Helmholtz AAI changed to {email}. A user with this email already exists and on this website. Please contact the website administrators.', PermissionDeniedReasons.email_exists: 'A user with the email {email} already exists.', PermissionDeniedReasons.email_not_verified: 'Your email has not been verified.', PermissionDeniedReasons.new_user: 'Your email {email} does not yet have a user account on this website and the account creation is disabled. Please sign up or contact the website administrators.', PermissionDeniedReasons.vo_not_allowed: 'Your virtual organizations are not allowed to log into this website.'}
Message templates that explain why a user is not allowed to login.

via the Helmholtz AAI. Use in the *get_permission_denied_message()* method.

permission_denied_reason: *PermissionDeniedReasons*
The reason why the user cannot login.

This attribute is set via the *has_permission()* method

synchronize_vos()
Synchronize the memberships in the virtual organizations.

This method checks the `eduperson_entitlement` of the AAI userinfo and

1. creates the missing virtual organizations
2. removes the user from virtual organizations that he or she does not belong to anymore
3. adds the user to the virtual organizations that are new.

Notes

As we remove users from virtual organizations, this might end up in a lot of VOs without any users. One can remove these VOs via:

```
python manage.py remove_empty_vos
```

Notes

Emits the *aai_vo_created*, *aai_vo_entered* and *aai_vo_left* signals.

update_user()
Update the user from the userinfo provided by the Helmholtz AAI.

Notes

Emits the *aai_user_updated* signal

userinfo
The userinfo as obtained from the Helmholtz AAI.

The attributes of this dictionary are determined by the Django Helmholtz AAI[1]

References

class django_helmholtz_aai.views.**HelmholtzLoginView**(***kwargs*)
Bases: `django.contrib.auth.views.LoginView`

A login view for the Helmholtz AAI that forwards to the OAuth login.

Methods:

get(request)	Get the redirect URL to the Helmholtz AAI.
post(request)	Reimplemented post method to call *get()*.

get(*request*)
Get the redirect URL to the Helmholtz AAI.

post(*request*)
Reimplemented post method to call *get()*.

[1] https://hifis.net/doc/helmholtz-aai/attributes/

1.4.6 django_helmholtz_aai.management.commands package

Submodules

Remove empty virtual organizations

This command can be used to automatically remove empty virtual organizations.

```
usage: python manage.py remove_empty_vos [-h] [-e EXCLUDE] [-y] [-db DATABASE]
```

Named Arguments

-e, --exclude	Exclude VOs that match the following pattern. This argument can be specified multiple times. Default: []
-y, --yes	Remove the VOs without asking for confirmation. Default: False
-db, --database	The Django database identifier (see settings.py), default: "default" Default: "default"

Classes:

Command([stdout, stderr, no_color, force_color])	Django command to migrate the database.

class django_helmholtz_aai.management.commands.remove_empty_vos.**Command**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Bases: `django.core.management.base.BaseCommand`

Django command to migrate the database.

Methods:

add_arguments(parser)	Add connection arguments to the parser.
handle(*args[, database, exclude, ...])	Migrate the database.

Attributes:

help

add_arguments(*parser*)
Add connection arguments to the parser.

handle(**args*, *database: str = 'default'*, *exclude: list[str] = []*, *without_confirmation: bool = False*, ***options*)
Migrate the database.

help = 'Remove virtual organization of the helmholtz AAI without users.'

1.4.7 django_helmholtz_aai package

Django Helmholtz AAI

Generic Django app for connecting with the Helmholtz AAI.

Functions:

login(request, user, userinfo)	Login the helmholtz user into django.

django_helmholtz_aai.**login**(*request*, *user:* models.HelmholtzUser, *userinfo: dict[str, Any]*)
Login the helmholtz user into django.

Notes

Emits the *aai_user_logged_in* signal

Subpackages

django_helmholtz_aai.management package

Subpackages

django_helmholtz_aai.tests package

Tests for the *django_helmholtz_aai* app.

Submodules

Submodules

Admin interfaces

This module defines the django Helmholtz AAI Admin interfaces, based upon the interfaces from `django.contrib.auth.admin`.

Classes:

HelmholtzAAIUserAdmin(model, admin_site)	
HelmholtzVirtualOrganizationAdmin(model, ...)	

class django_helmholtz_aai.admin.**HelmholtzAAIUserAdmin**(*model*, *admin_site*)
Bases: `django.contrib.auth.admin.UserAdmin`

Attributes:

list_display

media

list_display = ('username', 'first_name', 'last_name', 'email', 'eduperson_unique_id', 'is_staff')

property media

class django_helmholtz_aai.admin.**HelmholtzVirtualOrganizationAdmin**(*model*, *admin_site*)

Bases: django.contrib.auth.admin.GroupAdmin

Attributes:

list_display

media

list_display = ('name', 'eduperson_entitlement')

property media

App config

App config for the django_helmholtz_aai app.

Classes:

DjangoHelmholtzAaiConfig(app_name, app_module)	

class django_helmholtz_aai.apps.**DjangoHelmholtzAaiConfig**(*app_name*, *app_module*)

Bases: django.apps.config.AppConfig

Attributes:

default_auto_field

name

default_auto_field = 'django.db.models.BigAutoField'

name = 'django_helmholtz_aai'

1.5 Contribution and development hints

The django-helmholtz-aai project is developed by the Helmholtz Coastal Data Center (HCDC) of the Helmholtz-Zentrum Hereon. It is open-source as we believe that this package can be helpful for multiple other django applications, and we are looking forward for your feedback, questions and especially for your contributions.

- If you want to ask a question, are missing a feature or have comments on the docs, please open an issue at the source code repository
- If you have suggestions for improvement, please let us know in an issue, or fork the repository and create a merge request. See also *Contributing in the development*.

1.5.1 Contributing in the development

Thanks for your wish to contribute to this app!! The source code of the django-helmholtz-aai package is hosted at https://gitlab.hzdr.de/hcdc/django/django-helmholtz-aai. It's an open gitlab where you can register via GitHub, or via the Helmholtz AAI. Once you created an account, you can fork this repository to your own user account and implement the changes. Afterwards, please make a merge request into the main repository. If you have any questions, please do not hesitate to create an issue on gitlab and contact the developers.

Once you created you fork, you can clone it via

```
git clone https://gitlab.hzdr.de/<your-user>/django-helmholtz-aai.git
```

and install it in development mode with the *[dev]* option via:

```
pip install -e ./django-helmholtz-aai/[dev]
```

Once you installed the package, run the migrations:

```
cd django-helmholtz-aai/
python manage.py migrate
```

which will create an sqlite-database for you.

Fixing the docs

The documentation for this package is written in restructured Text and built with sphinx and deployed on readthedocs.

If you found something in the docs that you want to fix, head over to the `docs` folder and build the docs with *make html* (or *make.bat* on windows). The docs are then available in `docs/_build/html/index.html` that you can open with your local browser.

Implement your fixes in the corresponding `.rst`-file and push them to your fork on gitlab.

Contributing to the code

We use automated formatters (see their config in `pyproject.toml` and `setup.cfg`), namely

- Black for standardized code formatting
- blackdoc for standardized code formatting in documentation
- Flake8 for general code quality
- isort for standardized order in imports.
- mypy for static type checking on type hints

We highly recommend that you setup pre-commit hooks to automatically run all the above tools every time you make a git commit. This can be done by running:

```
pre-commit install
```

from the root of the repository. You can skip the pre-commit checks with `git commit --no-verify` but note that the CI will fail if it encounters any formatting errors.

You can also run the pre-commit step manually by invoking:

```
pre-commit run --all-files
```

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[client-registration] https://hifis.net/doc/helmholtz-aai/howto-services/

PYTHON MODULE INDEX

d

INDEX

G

H

I

L