
pylabber

Release 0.1.0

Jun 08, 2022

Contents:

1	Overview	3
2	Installation	5
3	Existing Apps	7
4	Frequently Asked Questions	9
5	Reference	11
6	Indices and tables	43
	Python Module Index	45
	Index	47

pylabber is a [Django](#) project designed to provide the core functionality required to conduct neuroscientific research.

See also:

For more information about Django applications and the difference between projects and apps see Django's [Getting Started](#) page, as well as the [Projects and Applications](#) section.

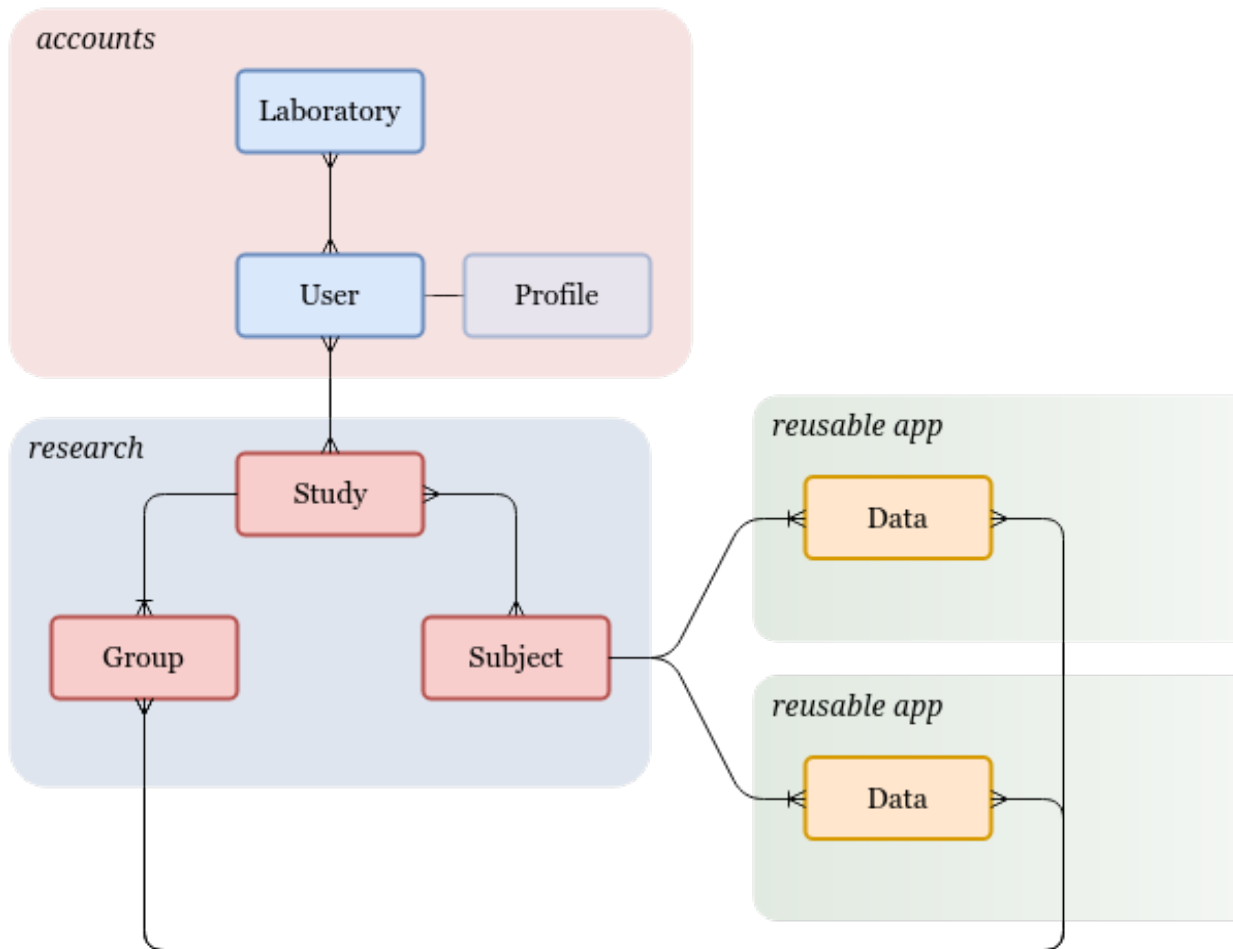
CHAPTER 1

Overview

The project’s root directory contains the *pylabber* module, which holds the applications settings and URL configurations, as well as two native apps:

- *accounts*: Manages *User* (i.e. researcher) and *Laboratory* accounts.
- *research*: Manages the most elementary “research entities”; *Subject*, *Study*, and *Group*.

The basic *models* detailed above are meant to be integrated with external, *reusable apps*, providing domain-specific data models and functionality over them.



For a list of integrated reusable apps see *Existing Apps*.

CHAPTER 2

Installation

Note: *pylabber* requires PostgreSQL. Before starting the installation, make sure you have [PostgreSQL](#) installed and create a new database.

There are many tutorials online providing detailed instructions on how to do this, some recommended ones can be found in [Digital Ocean](#), [DjangoGirls](#), and [Medium](#).

1. Clone *pylabber* to a local directory:

```
$ git clone https://github.com/TheLabbingProject/pylabber
```

2. From within the project's directory, create a virtual environment using *venv*:

```
$ python3 -m venv venv
```

3. Activate the virtual environment:

```
$ source venv/bin/activate
```

4. Install the required dependencies:

```
$ pip install -r requirements.txt
```

5. Create the appropriate environment variables for the *settings.py* file. You can find each of those variables wrapped in an *env*("VAR_NAME") call.

An example for an environment variables definition in development could look like:

Listing 1: *.env*

```
export SECRET_KEY="s0m3-$$/pER-s3CrE7_kEy!"
export DB_NAME=pylabber
export DB_USER=postgres
```

(continues on next page)

(continued from previous page)

```
export DB_PASSWORD=p$q1@ddmmIN
export MEDIA_ROOT="/where/to/save/files/locally/"
```

6. Apply the project's migrations:

```
$ python manage.py migrate
```

7. Run *pylabber* locally:

```
$ python manage.py runserver
```

or, open an interactive Django shell using *django-extensions*:

```
$ python manage.py shell_plus
```

CHAPTER 3

Existing Apps

- `django_analyses`: Database-driven pipeline engine. Allows the creation of a library of analyses and keeps records of configurations and runs.
- `django_mri`: Format-agnostic MRI data management. Provides a `Scan` model that abstractifies data management and retrieval.

Frequently Asked Questions

- *What is The Labbing Project?*
- *Is The Labbing Project only suitable for MRI-based neuroscientific research?*
- *How are different data types integrated into the application's database?*

4.1 What is The Labbing Project?

A collection of reusable Django apps and a single dedicated parent Django project (i.e. pylabber) used to manage and share research data and derivatives across researchers and across labs.

Hint: For more information about the difference between Django apps and projects, see Django's [Projects and Applications](#) documentation.

4.2 Is The Labbing Project only suitable for MRI-based neuroscientific research?

Not at all! The purpose of the `research` app is to provide common, generic models (such as `Subject` and `Study`) that may be associated with any number of data models originating from any number of reusable apps.

4.3 How are different data types integrated into the application's database?

The modular design of the project enables researchers from different fields to create specialized apps that manage the relevant data formats and facilitate commonplace workflows. E.g. the first app created for this purpose was `django_dicom`, which manages `DICOM` data, and the second was `django_mri`, providing a format-agnostic `Scan` abstraction, as well as other useful models and MRI-based research utilities. Each reusable app integrates with pylabber's internal `research` app to associate the data with an instance of the `Subject` model. For more information, see pylabber's [overview](#).

5.1 accounts package

5.1.1 Module contents

Manages researcher (*User*) and *Laboratory* accounts.

5.1.2 Subpackages

accounts.filters package

Module contents

Filters for *the app's models*.

Notes

For more information, see:

- [Django REST Framework filtering documentation](#).
- [django-filter's documentation for Integration with DRF](#).

Submodules

accounts.filters.user module

Definition of the *UserFilter* class.

```
class accounts.filters.user.UserFilter(data=None, queryset=None, *, request=None, pre-  
                                     fix=None)  
    Bases: django_filters.rest_framework.filterset.FilterSet  
    Provides useful filtering options for the User model.  
  
    base_filters = {'email': <django_filters.filters.LookupChoiceFilter object>, 'first_n  
    declared_filters = {'email': <django_filters.filters.LookupChoiceFilter object>, 'fir
```

accounts.models package

Module contents

Definition of the app's [models](#). For an illustration of the relationship between the different models, see the [Overview](#).

Submodules

accounts.models.choices module

Subclasses of the `ChoiceEnum` class used to represent raw and human-readable values for choices within the `accounts.models` module.

```
class accounts.models.choices.Role  
    Bases: pylabber.utils.utils.ChoiceEnum  
    An enumeration.  
  
    AFF = 'Research Affiliate'  
    MAN = 'Lab Manager'  
    MSC = 'M.Sc. Student'  
    PHD = 'Ph.D. Candidate'  
    PI = 'Principle Investigator'  
    POST = 'Postdoctoral Researcher'  
    RA = 'Research Assistant'  
  
class accounts.models.choices.Title  
    Bases: pylabber.utils.utils.ChoiceEnum  
    An enumeration.  
  
    BSC = 'B.Sc.'  
    MSC = 'M.Sc.'  
    PHD = 'Ph.D.'  
    PROF = 'Prof.'
```

accounts.models.laboratory module

Definition of the [Laboratory](#) model.

```
class accounts.models.laboratory.Laboratory(*args, **kwargs)
    Bases: django_extensions.db.models.TitleDescriptionModel, django_extensions.db.models.TimeStampedModel
```

A class to represents a research laboratory.

```
get_absolute_url() → str
    Returns the canonical URL for this instance.
```

References

- [get_absolute_url\(\)](#)

Returns URL

Return type str

image

Just like the FileDescriptor, but for ImageFields. The only difference is assigning the width/height to the width_field/height_field, if appropriate.

laboratorymembership_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

members

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

Pizza.toppings and Topping.pizzas are ManyToManyDescriptor instances.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

accounts.models.profile module

Definition of the *Profile* model.

```
class accounts.models.profile.Profile(*args, **kwargs)
    Bases: django.db.models.base.Model
```

A user profile, associated to each user using a OneToOne relationship and created automatically usings signals.

References

- [Extending the User model.](#)

bio

Short user biography.

date_of_birth

User's date of birth.

get_absolute_url() → str

Returns the canonical URL for this instance.

References

- [get_absolute_url\(\)](#)

Returns URL

Return type str

get_full_name(include_title: bool = True) → str

Returns the full name of the user, including a title if any.

Parameters **include_title** (*bool*, *optional*) – Whether to include title (academic credentials etc.) if specified

Returns User's full name

Return type str

get_title_display(*, field=<django.db.models.fields.CharField: title>)

get_title_repr() → str

Returns the verbose title of the user as defined in the *Title* Enum.

Returns Verbose title representation

Return type str

image

Profile image.

institute

The institute to which a user belongs, if any.

title

Academic or any other kind of title.

user

OneToOne relationship with the user model.

accounts.models.user module

Definition of the *User* model.

```
class accounts.models.user.User(*args, **kwargs)
```

Bases: `django.contrib.auth.models.AbstractUser`

Custom User model definition.

References

- Using a custom user model when starting a project.

exportdestination_set

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

groups

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

laboratory_set

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

laboratorymembership_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

logentry_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

mri_uploads

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

profile

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

Place.restaurant is a ReverseOneToOneDescriptor instance.

run_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

study_set

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

Pizza.toppings and Topping.pizzas are ManyToManyDescriptor instances.

Most of the implementation is delegated to a dynamically defined manager class built by create_forward_many_to_many_manager() defined below.

user_permissions

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

accounts.serializers package

Module contents

Django REST Framework `serializers` module for the `accounts` package.

Submodules

accounts.serializers.group module

Definition of the `GroupSerializer` class.

```
class accounts.serializers.group.GroupSerializer (instance=None, data=<class
                                                'rest_framework.fields.empty'>,
                                                **kwargs)
```

Bases: `rest_framework.serializers.HyperlinkedModelSerializer`

`Serializer` class for the `Group` model.

accounts.serializers.laboratory module

Definition of the `LaboratorySerializer` class.

```
class accounts.serializers.laboratory.LaboratorySerializer (instance=None,
                                                            data=<class
                                                            'rest_framework.fields.empty'>,
                                                            **kwargs)
```

Bases: `rest_framework.serializers.HyperlinkedModelSerializer`

`Serializer` class for the `Laboratory` model.

accounts.serializers.profile module

Definition of the `ProfileSerializer` class.

```
class accounts.serializers.profile.ProfileSerializer (instance=None, data=<class
                                                    'rest_framework.fields.empty'>,
                                                    **kwargs)
```

Bases: `rest_framework.serializers.HyperlinkedModelSerializer`

`Serializer` class for the `Profile` model.

accounts.serializers.user module

Definition of the *UserSerializer* class.

```
class accounts.serializers.user.UserSerializer (instance=None, data=<class
                                             'rest_framework.fields.empty'>,
                                             **kwargs)
    Bases: rest_auth.serializers.UserDetailsSerializer
    Serializer class for the User model.
```

References

- <https://www.django-rest-framework.org/api-guide/serializers/>

update (*username*, *data*: *dict*)

Update a user's personal information, including profile data.

Parameters

- **username** (*User*) – User to be updated
- **data** (*dict*) – User information

Returns Updated user instance

Return type *User*

accounts.views package

Module contents

Django REST Framework *ViewSet*s for the *accounts* package.

Submodules

accounts.views.group module

Definition of the *GroupViewSet* class.

```
class accounts.views.group.GroupViewSet (**kwargs)
    Bases: pylabber.views.defaults.DefaultsMixin, rest_framework.viewsets.
    ModelViewSet
    API endpoint that allows Group instances to be viewed or edited.

    basename = None
    description = None
    detail = None
    name = None
    queryset
    serializer_class
        alias of accounts.serializers.group.GroupSerializer
```

suffix = None

accounts.views.laboratory module

Definition of the *LaboratoryViewSet* class.

```
class accounts.views.laboratory.LaboratoryViewSet (**kwargs)
    Bases:      pylabber.views.defaults.DefaultsMixin,      rest_framework.viewsets.
                ModelViewSet

    API endpoint that allows Laboratory instances to be viewed or edited.

    basename = None
    description = None
    detail = None
    name = None
    queryset
    serializer_class
        alias of accounts.serializers.laboratory.LaboratorySerializer
    suffix = None
```

accounts.views.profile module

Definition of the *ProfileViewSet* class.

```
class accounts.views.profile.ProfileViewSet (**kwargs)
    Bases:      pylabber.views.defaults.DefaultsMixin,      rest_framework.viewsets.
                ModelViewSet

    API endpoint that allows Profile instances to be viewed or edited.

    basename = None
    description = None
    detail = None
    name = None
    queryset
    serializer_class
        alias of accounts.serializers.profile.ProfileSerializer
    suffix = None
```

accounts.views.user module

Definition of the *UserViewSet* class.

```
class accounts.views.user.UserViewSet (**kwargs)
    Bases:      pylabber.views.defaults.DefaultsMixin,      rest_framework.viewsets.
                ModelViewSet

    API endpoint that allows User instances to be viewed or edited.
```

```
basename = None
description = None
detail = None
filter_class
    alias of accounts.filters.user.UserFilter
filter_queryset (queryset) → django.db.models.query.QuerySet
    Filter the returned users according to the requesting user's permissions.

    Parameters queryset (QuerySet) – Base queryset

    Returns User instances

    Return type QuerySet
get_institutions (request)
name = None
queryset
serializer_class
    alias of accounts.serializers.user.UserSerializer
suffix = None
```

5.1.3 Submodules

5.1.4 accounts.admin module

```
class accounts.admin.ExportDestinationAdmin (model, admin_site)
    Bases: django.contrib.admin.options.ModelAdmin

    fieldsets = ((None, {'fields': ('id', 'title', 'description', 'username', 'password',
    form
        alias of accounts.forms.export_destination.ExportDestinationForm
    list_display = ('id', 'title', 'description', 'ip', 'username', 'destination', 'sftp')
    media
    readonly_fields = ('id',)
    sftp (destination: accounts.models.export_destination.ExportDestination) → bool

class accounts.admin.LaboratoryAdmin (model, admin_site)
    Bases: django.contrib.admin.options.ModelAdmin

    class Media
        Bases: object

        css = {'all': ('accounts/css/hide_admin_original.css',)}
    fields = ('title', 'description', 'image', 'image_tag')
    image_tag (instance: accounts.models.laboratory.Laboratory) → str
    inlines = (<class 'accounts.admin.LaboratoryMembershipInLine'>,)
    list_display = ('id', 'title', 'description', 'created', 'modified')
    media
```

```

    readonly_fields = ('image_tag',)

class accounts.admin.LaboratoryMembershipInline (parent_model, admin_site)
    Bases: django.contrib.admin.options.TabularInline

    extra = 0

    media

    model
        alias of accounts.models.laboratory_membership.LaboratoryMembership

class accounts.admin.ProfileInline (parent_model, admin_site)
    Bases: django.contrib.admin.options.StackedInline

    can_delete = False

    fields = ('title', 'image', 'image_tag', 'date_of_birth', 'institute', 'bio')

    fk_name = 'user'

    image_tag (instance: accounts.models.profile.Profile) → str

    media

    model
        alias of accounts.models.profile.Profile

    readonly_fields = ('image_tag',)

    verbose_name_plural = 'Profile'

class accounts.admin.UserAdmin (model, admin_site)
    Bases: django.contrib.auth.admin.UserAdmin

    get_inline_instances (request, obj=None)

    get_institute (instance)

    inlines = (<class 'accounts.admin.ProfileInline'>, <class 'accounts.admin.LaboratoryMembershipInline'>)

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

    list_select_related = ('profile',)

    media

```

5.1.5 accounts.apps module

```

class accounts.apps.AccountsConfig (app_name, app_module)
    Bases: django.apps.config.AppConfig

    name = 'accounts'

    ready ()
        Override this method in subclasses to run code when Django starts.

```

5.1.6 accounts.signals module

`accounts.signals.create_auth_token (sender, instance=None, created=False, **kwargs)`
 Automatically creates an authentication token for the user, as described in [the DRF documentation](#).

Parameters `created (bool, optional)` – Whether the instance is being created or updated, by default False

```
accounts.signals.post_save_user_model_receiver(sender, instance, created, *args,
                                                **kwargs)
```

Standard implementation for user profile creation.

Parameters

- **sender** (*django.db.models.Model*) – The class of the instance being saved
- **instance** (*django.db.models.Model*) – The instance being saved
- **created** (*bool*) – Whether the instance is being created or updated

5.1.7 accounts.urls module

```
accounts.urls.path(route, view, kwargs=None, name=None, *, Pattern=<class
                    'django.urls.resolvers.RoutePattern'>)
```

5.2 pylabber package

5.2.1 Module contents

5.2.2 Submodules

5.2.3 pylabber.settings module

Django settings for pylabber project.

For more information on this file, see <https://docs.djangoproject.com/en/dev/topics/settings/>

For the full list of settings and their values, see <https://docs.djangoproject.com/en/dev/ref/settings/>

```
pylabber.settings.DICOM_IMPORT_MODE = 'minimal'
```

Prevent the creation of DICOM data elements in the database.

```
pylabber.settings.ENABLE_COUNT_FILTERING = False
```

Do not annotate querysets with related model counts which drastically slows down queries.

5.2.4 pylabber.test_settings module

5.2.5 pylabber.urls module

pylabber URL Configuration

The `urlpatterns` list routes URLs to views.

Examples: * Function views

1. Add an import: from my_app import views
2. Add a URL to urlpatterns: `path('', views.home, name='home')`
 - **Class-based views**
 1. Add an import: from other_app.views import Home
 2. Add a URL to urlpatterns: `path('', Home.as_view(), name='home')`

- Including another URLconf

1. Import the include() function: `from django.urls import include, path`
2. Add a URL to urlpatterns: `path('blog/', include('blog.urls'))`

```
pylabber.urls.path(route, view, kwargs=None, name=None, *, Pattern=<class
'django.urls.resolvers.RoutePattern'>)
```

5.2.6 pylabber.utils module

5.2.7 pylabber.wsgi module

WSGI config for pylabber project.

It exposes the WSGI callable as a module-level variable named `application`.

For more information on this file, see <https://docs.djangoproject.com/en/2.0/howto/deployment/wsgi/>

5.3 research package

5.3.1 Module contents

A reusable Django app to manage research information.

5.3.2 Subpackages

research.filters package

Module contents

Filters for *the app's models*.

Submodules

research.filters.subject_filter module

Definition of the *SubjectFilter* class.

```
class research.filters.subject_filter.SubjectFilter(data=None, queryset=None, *,
                                                    request=None, prefix=None)
```

Bases: `django_filters.rest_framework.filterset.FilterSet`

Provides useful filtering options for the *Subject* model.

```
base_filters = {'born_after_date': <django_filters.filters.DateFilter object>, 'born_
```

```
declared_filters = {'born_after_date': <django_filters.filters.DateFilter object>, 'b
```

```
filter_by_dicom_patient (queryset, name, value)
```

Find the subject that represents a particular DICOM *Patient* instance.

Parameters

- **queryset** (*django.db.models.QuerySet*) – The *Subject* queryset to filter.
- **name** (*str*) – The name of the model field to filter on.
- **value** (*int*) – DICOM *Patient* ID.

filter_by_measurement (*queryset, name, value*)

filter_by_procedure (*queryset, name, value*)

filter_by_questionnaire_id (*queryset, name, value*)

filter_by_studies (*queryset, name, value*)

filter_by_study_group (*queryset, name, value*)

filter_nullable_charfield (*queryset, name, values*)

research.models package

Module contents

Definition of the app's *models*. For an illustration of the relationship between the different models, see the *Overview*.

Submodules

research.models.choices module

Subclasses of the *ChoiceEnum* class used to represent raw and human-readable values for choices within the *research.models* module.

class `research.models.choices.DominantHand`

Bases: `pylabber.utils.utils.ChoiceEnum`

An *Enum* representing supported dominant hand options.

A = `'Ambidextrous'`

L = `'Left'`

R = `'Right'`

class `research.models.choices.Gender`

Bases: `pylabber.utils.utils.ChoiceEnum`

An *Enum* representing supported gender options.

CIS = `'Cisgender'`

OTHER = `'Other'`

TRANS = `'Transgender'`

class `research.models.choices.Sex`

Bases: `pylabber.utils.utils.ChoiceEnum`

An *Enum* representing supported sex options.

F = `'Female'`

M = `'Male'`

U = `'Other'`

research.models.group module

Definition of the *Group* model.

```
class research.models.group.Group(*args, **kwargs)
    Bases: django_extensions.db.models.TitleDescriptionModel, django_extensions.db.models.TimeStampedModel
```

Represents a unique study group (i.e. a grouping of subjects according to some experimental design in the context of a study).

mri_scan_set

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

study

The study associated with this experimental group.

research.models.study module

Definition of the *Study* model.

```
class research.models.study.Study(*args, **kwargs)
    Bases: django_extensions.db.models.TitleDescriptionModel, django_extensions.db.models.TimeStampedModel
```

Represents a single study in the database.

collaborators

Researchers collaborating on this study.

get_absolute_url()

Returns the canonical URL for this instance.

References

- `get_absolute_url()`

Returns URL

Return type `str`

group_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

image

An optional image to supplement the description.

n_subjects

Returns the number of subjects associated with this study.

Returns Number of subjects associated with this study

Return type `int`

procedures

The experimental procedures associated with this study.

query_associated_subjects () → `django.db.models.query.QuerySet`

Returns a queryset of subjects associated with this study.

See also:

- `subject_set()`

Returns Subjects associated with this study

Return type `models.QuerySet`

subject_set

Returns a queryset of subjects associated with this study.

See also:

- `query_associated_subjects()`

Returns Subjects associated with this study

Return type `models.QuerySet`

subjects

Subjects associated with this study. This field is currently not used, but kept because in the future it might be used for “caching” associated subjects to save queries.

research.models.subject module

Definition of the `Subject` model.

class `research.models.subject.Subject` (*args, **kwargs)

Bases: `django_extensions.db.models.TimeStampedModel`

Represents a single research subject. Any associated data model should be associated with this model.

BIDS_DIR_TEMPLATE = `'sub-{pk}'`

build_bids_directory (*force: bool = False, persistent: bool = True, progressbar: bool = False, progressbar_position: int = 0*)

custom_attributes

Custom attributes dictionary.

date_of_birth

Subject's date of birth.

dominant_hand

Subject's dominant hand.

first_name

Subject's first name.

gender

Subject's gender.

get_absolute_url()

Returns the canonical URL for this instance.

References

- [get_absolute_url\(\)](#)

Returns URL

Return type `str`

get_bids_directory() → `pathlib.Path`

get_dominant_hand_display(*, field=<django.db.models.fields.CharField: dominant_hand>)

get_full_name() → `str`

Returns a formatted string with the subject's full name (first name and then last name).

Returns Subject's full name

Return type `str`

get_gender_display(*, field=<django.db.models.fields.CharField: gender>)

get_personal_information() → `pandas.core.series.Series`

Temporary method to use an external table to retrieve subject personal information.

Returns Subject personal information

Return type `pd.Series`

get_questionnaire_data()

A method to link between a subject to it's questionnaire data.

Returns Subject and Questionnaire information.

Return type `pd.Series`

get_raw_information() → `pandas.core.series.Series`

Temporary method to use an external table to retrieve subject information.

Returns Subject information

Return type `pd.Series`

get_sex_display(*, field=<django.db.models.fields.CharField: sex>)

id_number

Some representative ID number unique to this subject.

last_name

Subject's last name.

mri_session_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

query_measurements () → `django.db.models.query.QuerySet`

query_procedures () → `django.db.models.query.QuerySet`

query_run_set () → `django.db.models.query.QuerySet`

query_scores (*analysis*: `Union[django_analyses.models.analysis.Analysis, Iterable[django_analyses.models.analysis.Analysis]]` = `None`, *analysis_title*: `Union[str, Iterable[str]]` = `None`, *analysis_version*: `Union[django_analyses.models.analysis_version.AnalysisVersion, Iterable[django_analyses.models.analysis_version.AnalysisVersion]]` = `None`, *analysis_version_title*: `Union[str, Iterable[str]]` = `None`, *atlas*=`None`, *atlas_title*: `Union[str, Iterable[str]]` = `None`, *metric*=`None`, *metric_title*: `Union[str, Iterable[str]]` = `None`, *region*=`None`, *region_title*: `Union[str, Iterable[str]]` = `None`, *region_index*: `Union[int, Iterable[int]]` = `None`, *hemisphere*: `str` = `None`) → `django.db.models.query.QuerySet`

query_studies () → `django.db.models.query.QuerySet`

Returns a queryset of `Study` instances this subject has data associated with.

Returns Associated studies

Return type `models.QuerySet`

query_study_groups () → `django.db.models.query.QuerySet`

save (*args, **kwargs)

Overrides the model's `save()` method to process custom attributes.

Hint: For more information, see Django's documentation on [overriding model methods](#).

sex

Subject's sex.

study_set

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

research.models.procedure module

Definition of the *Procedure* model.

```
class research.models.procedure.Procedure (*args, **kwargs)
    Bases: django_extensions.db.models.TitleDescriptionModel

    Represents an experimental procedure.

    add_event (event: research.models.event.Event, index: int = None)
        Performs an event addition.

    events
        Represents an ordered list of events in a procedure.

    get_absolute_url ()
        Returns the canonical URL for this instance.
```

References

- `get_absolute_url()`

Returns URL

Return type `str`

max_index

Returns the maximal `index` field value of any associated `ProcedureStep` instances. If there aren't any, returns -1.

Returns Maximal step index, or -1

Return type `int`

step_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

study_set

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

research.models.event module

Definition of the *Event* model.

```
class research.models.event.Event (*args, **kwargs)
    Bases: django_extensions.db.models.TitleDescriptionModel

    Represents an event as a part of a procedure.

    get_absolute_url()
        Returns the canonical URL for this instance.
```

References

- `get_absolute_url()`

Returns URL

Return type `str`

measurementdefinition

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

procedure_set

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

procedurestep_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

task

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant (Model) :
    place = OneToOneField (Place, related_name='restaurant')
```

Place.restaurant is a ReverseOneToOneDescriptor instance.

research.models.task module

Definition of the *Task* model.

```
class research.models.task.Task (*args, **kwargs)
    Bases: research.models.event.Event
```

Represents an experimental task.

event_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant (Model) :
    place = OneToOneField (Place, related_name='restaurant')
```

Restaurant.place is a ForwardOneToOneDescriptor instance.

event_ptr_id

get_absolute_url()

Returns the canonical URL for this instance.

References

- [get_absolute_url\(\)](#)

Returns URL

Return type str

research.models.measurement_definition module

Definition of the *MeasurementDefinition* model.

```
class research.models.measurement_definition.MeasurementDefinition (*args,
                                                                    **kwargs)
    Bases: research.models.event.Event
```

Represents an experimental measurement definition.

content_type

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child (Model) :
    parent = ForeignKey (Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

content_type_id

event_ptr

Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

event_ptr_id

get_absolute_url()

Returns the canonical URL for this instance.

References

- `get_absolute_url()`

Returns URL

Return type `str`

get_instance_set() → `django.db.models.query.QuerySet`

Returns a queryset of collected measurements associated with this measurements definition.

Returns Collected data instances

Return type `models.QuerySet`

instance_set

Returns a queryset of collected measurements associated with this measurements definition.

Returns Collected data instances

Return type `models.QuerySet`

See also:

- `get_instance_set()`

mri_session_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

query_associated_studies() → `django.db.models.query.QuerySet`

research.models.validators module

Validators for django fields within the `research.models` module.

`research.models.validators.not_future(value)`

research.serializers package

Module contents

Django REST Framework serializers module for the `research` package.

Submodules

research.serializers.group module

Definition of the `GroupSerializer` class.

```
class research.serializers.group.GroupReadSerializer (instance=None, data=<class
                                                    'rest_framework.fields.empty'>,
                                                    **kwargs)
```

Bases: `rest_framework.serializers.HyperlinkedModelSerializer`

`HyperlinkedModelSerializer` for the `Group` model to be used in GET requests.

```
class research.serializers.group.GroupSerializer (instance=None, data=<class
                                                    'rest_framework.fields.empty'>,
                                                    **kwargs)
```

Bases: `rest_framework.serializers.HyperlinkedModelSerializer`

`HyperlinkedModelSerializer` for the `Group` model to be used in POST requests.

```
class research.serializers.group.MiniStudySerializer (instance=None, data=<class
                                                    'rest_framework.fields.empty'>,
                                                    **kwargs)
```

Bases: `rest_framework.serializers.ModelSerializer`

A minimized `Study ModelSerializer` used in the `GroupReadSerializer` class.

research.serializers.study module

Definition of the `StudySerializer` class.

```
class research.serializers.study.MiniStudySerializer (instance=None, data=<class
                                                    'rest_framework.fields.empty'>,
                                                    **kwargs)
```

Bases: `rest_framework.serializers.HyperlinkedModelSerializer`

Minified serializer for the `Study` model.

```
class research.serializers.study.StudySerializer (instance=None, data=<class
                                                    'rest_framework.fields.empty'>,
                                                    **kwargs)
```

Bases: `rest_framework.serializers.HyperlinkedModelSerializer`

Serializer for the `Study` model.

research.serializers.subject module

Definition of the *SubjectSerializer* class.

```
class research.serializers.subject.AdminSubjectSerializer (instance=None,
                                                         data=<class
                                                         'rest_framework.fields.empty'>,
                                                         **kwargs)
```

Bases: *research.serializers.subject.SubjectSerializer*

```
class research.serializers.subject.SubjectSerializer (instance=None, data=<class
                                                         'rest_framework.fields.empty'>,
                                                         **kwargs)
```

Bases: *rest_framework.serializers.ModelSerializer*

Base serializer for the *Subject* model.

research.views package

Module contents

Submodules

research.views.group module

Definition of the *GroupViewSet* class.

```
class research.views.group.GroupViewSet (**kwargs)
    Bases:      pylabber.views.defaults.DefaultsMixin,      rest_framework.viewsets.
    ModelViewSet
```

API endpoint that allows *Group* instances to be viewed or edited.

basename = None

description = None

detail = None

filter_class

alias of *research.filters.group_filter.GroupFilter*

get_serializer_class ()

Return the class to use for the serializer. Defaults to using *self.serializer_class*.

You may want to override this if you need to provide different serializations depending on the incoming request.

(Eg. admins get full serialization, others get basic serialization)

name = None

queryset

suffix = None

research.views.study module

Definition of the *StudyViewSet* class.

```
class research.views.study.StudyViewSet (**kwargs)
    Bases:          pylabber.views.defaults.DefaultsMixin,      rest_framework.viewsets.
                    ModelViewSet

    API endpoint that allows Study instances to be viewed or edited.

    aggregate (request) → rest_framework.response.Response
        Returns related model counts if count filtering is enabled.

        Parameters request (Request) – API request

        Returns Aggregated queryset or informational message

        Return type Response

basename = None

description = None

detail = None

filter_class
    alias of research.filters.study_filter.StudyFilter

filter_queryset (queryset)
    Filters the returned study queryset according to the user's collaborations.

        Parameters queryset (QuerySet) – Base Study queryset

        Returns Studies in which the user is a collaborator

        Return type QuerySet

get_queryset () → django.db.models.query.QuerySet
    Overrides the parent get_queryset () method to apply aggregated annotation if count filtering is enabled.

        Returns Patient queryset

        Return type QuerySet

name = None

queryset

serializer_class
    alias of research.serializers.study.StudySerializer

suffix = None
```

research.views.subject module

Definition of the *SubjectViewSet* class.

```
class research.views.subject.SubjectViewSet (**kwargs)
    Bases:          pylabber.views.defaults.DefaultsMixin,      rest_framework.viewsets.
                    ModelViewSet

    API endpoint that allows Subject instances to be viewed or edited.

basename = None
```

```
description = None
detail = None
export_files(request)
filter_class
    alias of research.filters.subject_filter.SubjectFilter
filter_queryset(queryset)
    Given a queryset, filter it with whichever filter backend is in use.

    You are unlikely to want to override this method, although you may need to call it either from a list view,
    or from a custom get_object method if you want to apply the configured filtering backend to the default
    queryset.

get_serializer_class()
    Return the class to use for the serializer. Defaults to using self.serializer_class.

    You may want to override this if you need to provide different serializations depending on the incoming
    request.

    (Eg. admins get full serialization, others get basic serialization)

name = None
ordering_fields = ('id', 'id_number', 'first_name', 'last_name', 'date_of_birth', 'sex')
plot(request, *args, **kwargs)
plot_script(request, *args, **kwargs)
plot_summary(request, *args, **kwargs)
queryset
suffix = None
to_csv(request, *args, **kwargs)
```

5.3.3 Submodules

5.3.4 *research.admin* module

```
class research.admin.CollaboratorsInline(parent_model, admin_site)
    Bases: django.contrib.admin.options.TabularInline

    class Media
        Bases: object

        css = {'all': ('research/css/hide_admin_original.css',)}

    email(instance) → str

    extra = 0

    fields = ('user_id', 'title', 'first_name', 'last_name', 'username', 'email')

    first_name(instance) → str

    has_add_permission(request, instance)
        Return True if the given request has permission to add an object. Can be overridden by the user in sub-
        classes.

    last_name(instance) → str
```

```

media
model
    alias of research.models.study.Study_collaborators
readonly_fields = ('user_id', 'title', 'first_name', 'last_name', 'username', 'email')
title(instance) → str
username(instance) → str
verbose_name_plural = 'Collaborators'
class research.admin.DecadeBornListFilter(request, params, model, model_admin)
    Bases: django.contrib.admin.filters.SimpleListFilter
    DECADES = ('40s', '50s', '60s', '70s', '80s', '90s', '00s')
    lookups(request, model_admin)
        Returns a list of tuples. The first element in each tuple is the coded value for the option that will appear
        in the URL query. The second element is the human-readable name for the option that will appear in the
        right sidebar.
    parameter_name = 'decade'
    queryset(request, queryset)
        Returns the filtered queryset based on the value provided in the query string and retrievable via self.value().
    title = 'decade born'
class research.admin.GroupAdmin(model, admin_site)
    Bases: django.contrib.admin.options.ModelAdmin
    inlines = (<class 'research.admin.ScanInline'>,)
    list_display = ('id', 'study_', 'title', 'description', 'mri_scan_count')
    list_filter = ('title', ('study__title', <class 'research.admin.custom_titled_filter.<
media
mri_scan_count(instance: research.models.group.Group) → int
readonly_fields = ('study_', 'mri_scan_count')
search_fields = ('study__title', 'title', 'description')
study_(instance: research.models.group.Group) → str
class research.admin.MeasurementDefinitionAdmin(model, admin_site)
    Bases: django.contrib.admin.options.ModelAdmin
    list_display = ('id', 'title', 'description', 'content_type', 'n_collected')
media
n_collected(instance: research.models.measurement_definition.MeasurementDefinition) → int
class research.admin.ProcedureAdmin(model, admin_site)
    Bases: django.contrib.admin.options.ModelAdmin
    inlines = (<class 'research.admin.StudyInline'>, <class 'research.admin.ProcedureStepI
    list_display = ('id', 'title', 'description', 'step_count')
media
step_count(instance: research.models.procedure.Procedure) → int

```

```
class research.admin.ProcedureInline (parent_model, admin_site)
    Bases: django.contrib.admin.options.TabularInline

    class Media
        Bases: object

        css = {'all': ('research/css/hide_admin_original.css',)}

    can_delete = False

    description (instance) → str

    extra = 0

    fields = ('id_link', 'title', 'description')

    has_add_permission (request, instance)
        Return True if the given request has permission to add an object. Can be overridden by the user in sub-
        classes.

    id_link (instance) → str

    media

    model
        alias of research.models.study.Study_procedures

    readonly_fields = ('id_link', 'title', 'description')

    title (instance) → str

    verbose_name_plural = 'Procedures'

class research.admin.ProcedureStepInline (parent_model, admin_site)
    Bases: django.contrib.admin.options.TabularInline

    class Media
        Bases: object

        css = {'all': ('research/css/hide_admin_original.css',)}

    can_delete = False

    event_description (instance: research.models.procedure_step.ProcedureStep) → str

    event_title (instance: research.models.procedure_step.ProcedureStep) → str

    event_type (instance: research.models.procedure_step.ProcedureStep) → str

    extra = 0

    fields = ('index_link', 'event_type', 'event_title', 'event_description')

    has_add_permission (request, instance: research.models.procedure_step.ProcedureStep)
        Return True if the given request has permission to add an object. Can be overridden by the user in sub-
        classes.

    index_link (instance: research.models.procedure_step.ProcedureStep) → str

    media

    model
        alias of research.models.procedure_step.ProcedureStep

    readonly_fields = ('index_link', 'event_type', 'event_title', 'event_description')

    verbose_name_plural = 'Steps'
```

```

class research.admin.ScanInline (parent_model, admin_site)
    Bases: django_admin_inline_paginator.admin.TabularInlinePaginated

    class Media
        Bases: object

        css = {'all': ('research/css/hide_admin_original.css',)}

    comments (instance) → str
    description (instance) → str
    download (instance) → str
    echo_time (instance) → float
    extra = 0
    fields = ('id_link', 'subject', 'session', 'number', 'time', 'description', 'echo_time')
    has_add_permission (request, instance)
        Return True if the given request has permission to add an object. Can be overridden by the user in sub-
        classes.
    id_link (instance) → str
    inversion_time (instance) → float
    media
    model
        alias of django_mri.models.scan.Scan_study_groups
    number (instance) → int
    readonly_fields = ('id_link', 'subject', 'session', 'number', 'time', 'description', 'echo_time')
    repetition_time (instance) → float
    session (instance) → str
    spatial_resolution (instance) → str
        Returns a nicely formatted representation of the scan's spatial resolution.

        Parameters instance (django_mri.Scan_study_groups) – Scan_study_groups in-
            stance

        Returns Formatted spatial resolution representation

        Return type str
    subject (instance) → str
    time (instance) → datetime.datetime
    verbose_name_plural = 'Scans'

class research.admin.SessionInline (parent_model, admin_site)
    Bases: django.contrib.admin.options.TabularInline

    class Media
        Bases: object

        css = {'all': ('research/css/hide_admin_original.css',)}

    can_delete = False
    download (instance: django_mri.models.session.Session) → str

```

```
extra = 0

fields = ('id_link', 'time', 'measurement', 'scan_count', 'comments', 'download')

has_add_permission(request, instance: django_mri.models.session.Session)
    Return True if the given request has permission to add an object. Can be overridden by the user in sub-
    classes.

id_link(instance: django_mri.models.session.Session) → str

media

model
    alias of django_mri.models.session.Session

readonly_fields = ('id_link', 'time', 'scan_count', 'download')

scan_count(instance: django_mri.models.session.Session) → int

verbose_name_plural = 'MRI Sessions'

class research.admin.StudyAdmin(model, admin_site)
    Bases: django.contrib.admin.options.ModelAdmin

    exclude = ('subjects', 'collaborators', 'procedures')

    fields = ('title', 'description', 'image', 'participant_list')

    inlines = (<class 'research.admin.CollaboratorsInline'>, <class 'research.admin.Proced

    list_display = ('title', 'description', 'created')

    media

    participant_list(instance: research.models.study.Study) → str

    readonly_fields = ('participant_list',)

class research.admin.StudyAssociationFilter(request, params, model, model_admin)
    Bases: django.contrib.admin.filters.SimpleListFilter

    lookups(request, model_admin)
        Must be overridden to return a list of tuples (value, verbose value)

    parameter_name = 'study participation'

    queryset(request, queryset)
        Return the filtered queryset.

    title = 'study participation'

class research.admin.StudyInline(parent_model, admin_site)
    Bases: django.contrib.admin.options.TabularInline

    class Media
        Bases: object

        css = {'all': ('research/css/hide_admin_original.css',)}

    can_delete = False

    description(instance) → str

    extra = 0

    fields = ('id_link', 'title', 'description')
```

```

has_add_permission (request, instance: research.models.procedure_step.ProcedureStep)
    Return True if the given request has permission to add an object. Can be overridden by the user in sub-
    classes.

id_link (instance) → str

media

model
    alias of research.models.study.Study_procedures

readonly_fields = ('id_link', 'title', 'description')

title (instance) → str

verbose_name_plural = 'Studies'

class research.admin.SubjectAdmin (model, admin_site)
    Bases: django.contrib.admin.options.ModelAdmin

    actions = ('export_csv',)

    changelist_view (request, extra_context=None)
        The 'change list' admin view for this model.

    export_csv (request, queryset)

    inlines = (<class 'research.admin.SessionInLine'>,)

    list_display = ('id', 'id_number', 'first_name', 'last_name', 'sex', 'date_of_birth',
    list_filter = ('sex', 'dominant_hand', <class 'research.admin.DecadeBornListFilter'>,
    media

    n_mri_sessions (instance: research.models.subject.Subject) → int

    readonly_fields = ('n_mri_sessions',)

    search_fields = ('id', 'id_number', 'first_name', 'last_name', 'date_of_birth__year')

class research.admin.SubjectsInline (parent_model, admin_site)
    Bases: django.contrib.admin.options.TabularInline

    date_of_birth (instance)

    extra = 0

    first_name (instance)

    id_number (instance)

    last_name (instance)

    media

    model
        alias of research.models.study.Study_subjects

    readonly_fields = ('id_number', 'first_name', 'last_name', 'sex', 'date_of_birth')

    sex (instance)

    verbose_name_plural = 'Subjects'

class research.admin.TaskAdmin (model, admin_site)
    Bases: django.contrib.admin.options.ModelAdmin

    list_display = ('id', 'title', 'description')

```

media

`research.admin.custom_titled_filter(title: str)`
Copied from SO: <https://stackoverflow.com/a/21223908/4416932>

5.3.5 research.apps module

```
class research.apps.ResearchConfig(app_name, app_module)
    Bases: django.apps.config AppConfig
    name = 'research'
    ready()
        Loads the app's signals.
```

References

- `ready()`

5.3.6 research.urls module

```
research.urls.path(route, view, kwargs=None, name=None, *, Pattern=<class
'django.urls.resolvers.RoutePattern'>)
```

CHAPTER 6

Indices and tables

- `genindex`
- `modindex`
- `search`

a

- `accounts`, 11
- `accounts.admin`, 20
- `accounts.apps`, 21
- `accounts.filters`, 11
- `accounts.filters.user`, 11
- `accounts.models`, 12
- `accounts.models.choices`, 12
- `accounts.models.laboratory`, 12
- `accounts.models.profile`, 13
- `accounts.models.user`, 14
- `accounts.serializers`, 17
- `accounts.serializers.group`, 17
- `accounts.serializers.laboratory`, 17
- `accounts.serializers.profile`, 17
- `accounts.serializers.user`, 18
- `accounts.signals`, 21
- `accounts.urls`, 22
- `accounts.views`, 18
- `accounts.views.group`, 18
- `accounts.views.laboratory`, 19
- `accounts.views.profile`, 19
- `accounts.views.user`, 19

p

- `pylabber`, 22
- `pylabber.settings`, 22
- `pylabber.test_settings`, 22
- `pylabber.urls`, 22
- `pylabber.utils`, 23
- `pylabber.wsgi`, 23

r

- `research`, 23
- `research.admin`, 36
- `research.apps`, 42
- `research.filters`, 23
- `research.filters.subject_filter`, 23
- `research.models`, 24

- `research.models.choices`, 24
- `research.models.event`, 30
- `research.models.group`, 25
- `research.models.measurement_definition`, 31
- `research.models.procedure`, 29
- `research.models.study`, 25
- `research.models.subject`, 26
- `research.models.task`, 31
- `research.models.validators`, 33
- `research.serializers`, 33
- `research.serializers.group`, 33
- `research.serializers.study`, 33
- `research.serializers.subject`, 34
- `research.urls`, 42
- `research.views`, 34
- `research.views.group`, 34
- `research.views.study`, 35
- `research.views.subject`, 35

A

A (*research.models.choices.DominantHand* attribute), 24
accounts (module), 11
accounts.admin (module), 20
accounts.apps (module), 21
accounts.filters (module), 11
accounts.filters.user (module), 11
accounts.models (module), 12
accounts.models.choices (module), 12
accounts.models.laboratory (module), 12
accounts.models.profile (module), 13
accounts.models.user (module), 14
accounts.serializers (module), 17
accounts.serializers.group (module), 17
accounts.serializers.laboratory (module), 17
accounts.serializers.profile (module), 17
accounts.serializers.user (module), 18
accounts.signals (module), 21
accounts.urls (module), 22
accounts.views (module), 18
accounts.views.group (module), 18
accounts.views.laboratory (module), 19
accounts.views.profile (module), 19
accounts.views.user (module), 19
AccountsConfig (class in *accounts.apps*), 21
actions (*research.admin.SubjectAdmin* attribute), 41
add_event() (*research.models.procedure.Procedure* method), 29
AdminSubjectSerializer (class in *research.serializers.subject*), 34
AFF (*accounts.models.choices.Role* attribute), 12
aggregate() (*research.views.study.StudyViewSet* method), 35

B

base_filters (*accounts.filters.user.UserFilter* attribute), 12

base_filters (*research.filters.subject_filter.SubjectFilter* attribute), 23
basename (*accounts.views.group.GroupViewSet* attribute), 18
basename (*accounts.views.laboratory.LaboratoryViewSet* attribute), 19
basename (*accounts.views.profile.ProfileViewSet* attribute), 19
basename (*accounts.views.user.UserViewSet* attribute), 19
basename (*research.views.group.GroupViewSet* attribute), 34
basename (*research.views.study.StudyViewSet* attribute), 35
basename (*research.views.subject.SubjectViewSet* attribute), 35
BIDS_DIR_TEMPLATE (*research.models.subject.Subject* attribute), 26
bio (*accounts.models.profile.Profile* attribute), 14
BSC (*accounts.models.choices.Title* attribute), 12
build_bids_directory() (*research.models.subject.Subject* method), 26

C

can_delete (*accounts.admin.ProfileInline* attribute), 21
can_delete (*research.admin.ProcedureInline* attribute), 38
can_delete (*research.admin.ProcedureStepInline* attribute), 38
can_delete (*research.admin.SessionInLine* attribute), 39
can_delete (*research.admin.StudyInline* attribute), 40
changelist_view() (*research.admin.SubjectAdmin* method), 41
CIS (*research.models.choices.Gender* attribute), 24
collaborators (*research.models.study.Study* attribute), 25

CollaboratorsInline (class in *research.admin*), 36
 CollaboratorsInline.Media (class in *research.admin*), 36
 comments() (*research.admin.ScanInline* method), 39
 content_type (*research.models.measurement_definition.MeasurementDefinition* attribute), 31
 content_type_id (*research.models.measurement_definition.MeasurementDefinition* attribute), 31
 create_auth_token() (in module *accounts.signals*), 21
 css (*accounts.admin.LaboratoryAdmin.Media* attribute), 20
 css (*research.admin.CollaboratorsInline.Media* attribute), 36
 css (*research.admin.ProcedureInline.Media* attribute), 38
 css (*research.admin.ProcedureStepInline.Media* attribute), 38
 css (*research.admin.ScanInline.Media* attribute), 39
 css (*research.admin.SessionInLine.Media* attribute), 39
 css (*research.admin.StudyInline.Media* attribute), 40
 custom_attributes (*research.models.subject.Subject* attribute), 26
 custom_titled_filter() (in module *research.admin*), 42

D

date_of_birth (*accounts.models.profile.Profile* attribute), 14
 date_of_birth (*research.models.subject.Subject* attribute), 26
 date_of_birth() (*research.admin.SubjectsInline* method), 41
 DecadeBornListFilter (class in *research.admin*), 37
 DECADES (*research.admin.DecadeBornListFilter* attribute), 37
 declared_filters (*accounts.filters.user.UserFilter* attribute), 12
 declared_filters (*research.filters.subject_filter.SubjectFilter* attribute), 23
 description (*accounts.views.group.GroupViewSet* attribute), 18
 description (*accounts.views.laboratory.LaboratoryViewSet* attribute), 19
 description (*accounts.views.profile.ProfileViewSet* attribute), 19
 description (*accounts.views.user.UserViewSet* attribute), 20
 description (*research.views.group.GroupViewSet* attribute), 34
 description (*research.views.study.StudyViewSet* attribute), 35
 description (*research.views.subject.SubjectViewSet* attribute), 36
 description() (*research.admin.ProcedureInline* method), 38
 description() (*research.admin.ScanInline* method), 39
 description() (*research.admin.StudyInline* method), 40
 detail (*accounts.views.group.GroupViewSet* attribute), 18
 detail (*accounts.views.laboratory.LaboratoryViewSet* attribute), 19
 detail (*accounts.views.profile.ProfileViewSet* attribute), 19
 detail (*accounts.views.user.UserViewSet* attribute), 20
 detail (*research.views.group.GroupViewSet* attribute), 34
 detail (*research.views.study.StudyViewSet* attribute), 35
 detail (*research.views.subject.SubjectViewSet* attribute), 36
 DICOM_IMPORT_MODE (in module *pylabber.settings*), 22
 dominant_hand (*research.models.subject.Subject* attribute), 27
 DominantHand (class in *research.models.choices*), 24
 download() (*research.admin.ScanInline* method), 39
 download() (*research.admin.SessionInLine* method), 39

E

echo_time() (*research.admin.ScanInline* method), 39
 email() (*research.admin.CollaboratorsInline* method), 36
 ENABLE_COUNT_FILTERING (in module *pylabber.settings*), 22
 Event (class in *research.models.event*), 30
 event_description() (*research.admin.ProcedureStepInline* method), 38
 event_ptr (*research.models.measurement_definition.MeasurementDefinition* attribute), 32
 event_ptr (*research.models.task.Task* attribute), 31
 event_ptr_id (*research.models.measurement_definition.MeasurementDefinition* attribute), 32
 event_ptr_id (*research.models.task.Task* attribute), 31
 event_title() (*research.admin.ProcedureStepInline* method), 38

[event_type\(\)](#) (*research.admin.ProcedureStepInline method*), 38
[events](#) (*research.models.procedure.Procedure attribute*), 29
[exclude](#) (*research.admin.StudyAdmin attribute*), 40
[export_csv\(\)](#) (*research.admin.SubjectAdmin method*), 41
[export_files\(\)](#) (*research.views.subject.SubjectViewSet method*), 36
[exportdestination_set](#) (*accounts.models.user.User attribute*), 15
[ExportDestinationAdmin](#) (*class in accounts.admin*), 20
[extra](#) (*accounts.admin.LaboratoryMembershipInline attribute*), 21
[extra](#) (*research.admin.CollaboratorsInline attribute*), 36
[extra](#) (*research.admin.ProcedureInline attribute*), 38
[extra](#) (*research.admin.ProcedureStepInline attribute*), 38
[extra](#) (*research.admin.ScanInline attribute*), 39
[extra](#) (*research.admin.SessionInline attribute*), 39
[extra](#) (*research.admin.StudyInline attribute*), 40
[extra](#) (*research.admin.SubjectsInline attribute*), 41

F

[F](#) (*research.models.choices.Sex attribute*), 24
[fields](#) (*accounts.admin.LaboratoryAdmin attribute*), 20
[fields](#) (*accounts.admin.ProfileInline attribute*), 21
[fields](#) (*research.admin.CollaboratorsInline attribute*), 36
[fields](#) (*research.admin.ProcedureInline attribute*), 38
[fields](#) (*research.admin.ProcedureStepInline attribute*), 38
[fields](#) (*research.admin.ScanInline attribute*), 39
[fields](#) (*research.admin.SessionInline attribute*), 40
[fields](#) (*research.admin.StudyAdmin attribute*), 40
[fields](#) (*research.admin.StudyInline attribute*), 40
[fieldsets](#) (*accounts.admin.ExportDestinationAdmin attribute*), 20
[filter_by_dicom_patient\(\)](#) (*research.filters.subject_filter.SubjectFilter method*), 23
[filter_by_measurement\(\)](#) (*research.filters.subject_filter.SubjectFilter method*), 24
[filter_by_procedure\(\)](#) (*research.filters.subject_filter.SubjectFilter method*), 24
[filter_by_questionnaire_id\(\)](#) (*research.filters.subject_filter.SubjectFilter method*), 24

[filter_by_studies\(\)](#) (*research.filters.subject_filter.SubjectFilter method*), 24
[filter_by_study_group\(\)](#) (*research.filters.subject_filter.SubjectFilter method*), 24
[filter_class](#) (*accounts.views.user.UserViewSet attribute*), 20
[filter_class](#) (*research.views.group.GroupViewSet attribute*), 34
[filter_class](#) (*research.views.study.StudyViewSet attribute*), 35
[filter_class](#) (*research.views.subject.SubjectViewSet attribute*), 36
[filter_nullable_charfield\(\)](#) (*research.filters.subject_filter.SubjectFilter method*), 24
[filter_queryset\(\)](#) (*accounts.views.user.UserViewSet method*), 20
[filter_queryset\(\)](#) (*research.views.study.StudyViewSet method*), 35
[filter_queryset\(\)](#) (*research.views.subject.SubjectViewSet method*), 36
[first_name](#) (*research.models.subject.Subject attribute*), 27
[first_name\(\)](#) (*research.admin.CollaboratorsInline method*), 36
[first_name\(\)](#) (*research.admin.SubjectsInline method*), 41
[fk_name](#) (*accounts.admin.ProfileInline attribute*), 21
[form](#) (*accounts.admin.ExportDestinationAdmin attribute*), 20

G

[Gender](#) (*class in research.models.choices*), 24
[gender](#) (*research.models.subject.Subject attribute*), 27
[get_absolute_url\(\)](#) (*accounts.models.laboratory.Laboratory method*), 13
[get_absolute_url\(\)](#) (*accounts.models.profile.Profile method*), 14
[get_absolute_url\(\)](#) (*research.models.event.Event method*), 30
[get_absolute_url\(\)](#) (*research.models.measurement_definition.MeasurementDefinition method*), 32
[get_absolute_url\(\)](#) (*research.models.procedure.Procedure method*), 29
[get_absolute_url\(\)](#) (*research.models.study.Study method*), 25

[get_absolute_url\(\)](#) (*research.models.subject.Subject method*), 27
[get_absolute_url\(\)](#) (*research.models.task.Task method*), 31
[get_bids_directory\(\)](#) (*research.models.subject.Subject method*), 27
[get_dominant_hand_display\(\)](#) (*research.models.subject.Subject method*), 27
[get_full_name\(\)](#) (*accounts.models.profile.Profile method*), 14
[get_full_name\(\)](#) (*research.models.subject.Subject method*), 27
[get_gender_display\(\)](#) (*research.models.subject.Subject method*), 27
[get_inline_instances\(\)](#) (*accounts.admin.UserAdmin method*), 21
[get_instance_set\(\)](#) (*research.models.measurement_definition.MeasurementDefinition method*), 32
[get_institute\(\)](#) (*accounts.admin.UserAdmin method*), 21
[get_institutions\(\)](#) (*accounts.views.user.UserViewSet method*), 20
[get_personal_information\(\)](#) (*research.models.subject.Subject method*), 27
[get_queryset\(\)](#) (*research.views.study.StudyViewSet method*), 35
[get_questionnaire_data\(\)](#) (*research.models.subject.Subject method*), 27
[get_raw_information\(\)](#) (*research.models.subject.Subject method*), 27
[get_serializer_class\(\)](#) (*research.views.group.GroupViewSet method*), 34
[get_serializer_class\(\)](#) (*research.views.subject.SubjectViewSet method*), 36
[get_sex_display\(\)](#) (*research.models.subject.Subject method*), 27
[get_title_display\(\)](#) (*accounts.models.profile.Profile method*), 14
[get_title_repr\(\)](#) (*accounts.models.profile.Profile method*), 14
[Group](#) (*class in research.models.group*), 25
[group_set](#) (*research.models.study.Study attribute*), 25
[GroupAdmin](#) (*class in research.admin*), 37
[GroupReadSerializer](#) (*class in research.serializers.group*), 33
[groups](#) (*accounts.models.user.User attribute*), 15
[GroupSerializer](#) (*class in accounts.serializers.group*), 17
[GroupSerializer](#) (*class in research.serializers.group*), 33
[GroupViewSet](#) (*class in accounts.views.group*), 18
[GroupViewSet](#) (*class in research.views.group*), 34

H

[has_add_permission\(\)](#) (*research.admin.CollaboratorsInline method*), 36
[has_add_permission\(\)](#) (*research.admin.ProcedureInline method*), 38
[has_add_permission\(\)](#) (*research.admin.ProcedureStepInline method*), 38
[has_add_permission\(\)](#) (*research.admin.ScanInline method*), 39
[has_add_permission\(\)](#) (*research.admin.SessionInLine method*), 40
[has_add_permission\(\)](#) (*research.admin.StudyInline method*), 40

I

[id_link\(\)](#) (*research.admin.ProcedureInline method*), 38
[id_link\(\)](#) (*research.admin.ScanInline method*), 39
[id_link\(\)](#) (*research.admin.SessionInLine method*), 40
[id_link\(\)](#) (*research.admin.StudyInline method*), 41
[id_number](#) (*research.models.subject.Subject attribute*), 27
[id_number\(\)](#) (*research.admin.SubjectsInline method*), 41
[image](#) (*accounts.models.laboratory.Laboratory attribute*), 13
[image](#) (*accounts.models.profile.Profile attribute*), 14
[image](#) (*research.models.study.Study attribute*), 26
[image_tag\(\)](#) (*accounts.admin.LaboratoryAdmin method*), 20
[image_tag\(\)](#) (*accounts.admin.ProfileInline method*), 21
[index_link\(\)](#) (*research.admin.ProcedureStepInline method*), 38
[inlines](#) (*accounts.admin.LaboratoryAdmin attribute*), 20
[inlines](#) (*accounts.admin.UserAdmin attribute*), 21
[inlines](#) (*research.admin.GroupAdmin attribute*), 37
[inlines](#) (*research.admin.ProcedureAdmin attribute*), 37
[inlines](#) (*research.admin.StudyAdmin attribute*), 40
[inlines](#) (*research.admin.SubjectAdmin attribute*), 41
[instance_set](#) (*research.models.measurement_definition.MeasurementDefinition attribute*), 32
[institute](#) (*accounts.models.profile.Profile attribute*), 14
[inversion_time\(\)](#) (*research.admin.ScanInline method*), 39

L

L (*research.models.choices.DominantHand* attribute), 24
 Laboratory (class in *accounts.models.laboratory*), 12
 laboratory_set (*accounts.models.user.User* attribute), 15
 LaboratoryAdmin (class in *accounts.admin*), 20
 LaboratoryAdmin.Media (class in *accounts.admin*), 20
 laboratorymembership_set (*accounts.models.laboratory.Laboratory* attribute), 13
 laboratorymembership_set (*accounts.models.user.User* attribute), 15
 LaboratoryMembershipInline (class in *accounts.admin*), 21
 LaboratorySerializer (class in *accounts.serializers.laboratory*), 17
 LaboratoryViewSet (class in *accounts.views.laboratory*), 19
 last_name (*research.models.subject.Subject* attribute), 27
 last_name() (*research.admin.CollaboratorsInline* method), 36
 last_name() (*research.admin.SubjectsInline* method), 41
 list_display (*accounts.admin.ExportDestinationAdmin* attribute), 20
 list_display (*accounts.admin.LaboratoryAdmin* attribute), 20
 list_display (*accounts.admin.UserAdmin* attribute), 21
 list_display (*research.admin.GroupAdmin* attribute), 37
 list_display (*research.admin.MeasurementDefinitionAdmin* attribute), 37
 list_display (*research.admin.ProcedureAdmin* attribute), 37
 list_display (*research.admin.StudyAdmin* attribute), 40
 list_display (*research.admin.SubjectAdmin* attribute), 41
 list_display (*research.admin.TaskAdmin* attribute), 41
 list_filter (*research.admin.GroupAdmin* attribute), 37
 list_filter (*research.admin.SubjectAdmin* attribute), 41
 list_select_related (*accounts.admin.UserAdmin* attribute), 21
 logentry_set (*accounts.models.user.User* attribute), 15
 lookups() (*research.admin.DecadeBornListFilter* method), 37
 lookups() (*research.admin.StudyAssociationFilter*

method), 40

M

M (*research.models.choices.Sex* attribute), 24
 MAN (*accounts.models.choices.Role* attribute), 12
 max_index (*research.models.procedure.Procedure* attribute), 29
 MeasurementDefinition (class in *research.models.measurement_definition*), 31
 measurementdefinition (*research.models.event.Event* attribute), 30
 MeasurementDefinitionAdmin (class in *research.admin*), 37
 media (*accounts.admin.ExportDestinationAdmin* attribute), 20
 media (*accounts.admin.LaboratoryAdmin* attribute), 20
 media (*accounts.admin.LaboratoryMembershipInline* attribute), 21
 media (*accounts.admin.ProfileInline* attribute), 21
 media (*accounts.admin.UserAdmin* attribute), 21
 media (*research.admin.CollaboratorsInline* attribute), 36
 media (*research.admin.GroupAdmin* attribute), 37
 media (*research.admin.MeasurementDefinitionAdmin* attribute), 37
 media (*research.admin.ProcedureAdmin* attribute), 37
 media (*research.admin.ProcedureInline* attribute), 38
 media (*research.admin.ProcedureStepInline* attribute), 38
 media (*research.admin.ScanInline* attribute), 39
 media (*research.admin.SessionInline* attribute), 40
 media (*research.admin.StudyAdmin* attribute), 40
 media (*research.admin.StudyInline* attribute), 41
 media (*research.admin.SubjectAdmin* attribute), 41
 media (*research.admin.SubjectsInline* attribute), 41
 media (*research.admin.TaskAdmin* attribute), 41
 members (*accounts.models.laboratory.Laboratory* attribute), 13
 MiniStudySerializer (class in *research.serializers.group*), 33
 MiniStudySerializer (class in *research.serializers.study*), 33
 model (*accounts.admin.LaboratoryMembershipInline* attribute), 21
 model (*accounts.admin.ProfileInline* attribute), 21
 model (*research.admin.CollaboratorsInline* attribute), 37
 model (*research.admin.ProcedureInline* attribute), 38
 model (*research.admin.ProcedureStepInline* attribute), 38
 model (*research.admin.ScanInline* attribute), 39
 model (*research.admin.SessionInline* attribute), 40
 model (*research.admin.StudyInline* attribute), 41

- model (*research.admin.SubjectsInline attribute*), 41
 mri_scan_count () (*research.admin.GroupAdmin method*), 37
 mri_scan_set (*research.models.group.Group attribute*), 25
 mri_session_set (*research.models.measurement_definition.MeasurementDefinition attribute*), 32
 mri_session_set (*research.models.subject.Subject attribute*), 27
 mri_uploads (*accounts.models.user.User attribute*), 16
 MSC (*accounts.models.choices.Role attribute*), 12
 MSC (*accounts.models.choices.Title attribute*), 12
- ## N
- n_collected () (*research.admin.MeasurementDefinitionAdmin method*), 37
 n_mri_sessions () (*research.admin.SubjectAdmin method*), 41
 n_subjects (*research.models.study.Study attribute*), 26
 name (*accounts.apps.AccountsConfig attribute*), 21
 name (*accounts.views.group.GroupViewSet attribute*), 18
 name (*accounts.views.laboratory.LaboratoryViewSet attribute*), 19
 name (*accounts.views.profile.ProfileViewSet attribute*), 19
 name (*accounts.views.user.UserViewSet attribute*), 20
 name (*research.apps.ResearchConfig attribute*), 42
 name (*research.views.group.GroupViewSet attribute*), 34
 name (*research.views.study.StudyViewSet attribute*), 35
 name (*research.views.subject.SubjectViewSet attribute*), 36
 not_future () (*in module research.models.validators*), 33
 number () (*research.admin.ScanInline method*), 39
- ## O
- ordering_fields (*research.views.subject.SubjectViewSet attribute*), 36
 OTHER (*research.models.choices.Gender attribute*), 24
- ## P
- parameter_name (*research.admin.DecadeBornListFilter attribute*), 37
 parameter_name (*research.admin.StudyAssociationFilter attribute*), 40
 participant_list () (*research.admin.StudyAdmin method*), 40
- path () (*in module accounts.urls*), 22
 path () (*in module pylabber.urls*), 23
 path () (*in module research.urls*), 42
 PHD (*accounts.models.choices.Role attribute*), 12
 PHD (*accounts.models.choices.Title attribute*), 12
 PI (*accounts.models.choices.Role attribute*), 12
 plot_script () (*research.views.subject.SubjectViewSet method*), 36
 plot_summary () (*research.views.subject.SubjectViewSet method*), 36
 POST (*accounts.models.choices.Role attribute*), 12
 post_save_user_model_receiver () (*in module accounts.signals*), 22
 Procedure (*class in research.models.procedure*), 29
 procedure_set (*research.models.event.Event attribute*), 30
 ProcedureAdmin (*class in research.admin*), 37
 ProcedureInline (*class in research.admin*), 37
 ProcedureInline.Media (*class in research.admin*), 38
 procedures (*research.models.study.Study attribute*), 26
 procedurestep_set (*research.models.event.Event attribute*), 30
 ProcedureStepInline (*class in research.admin*), 38
 ProcedureStepInline.Media (*class in research.admin*), 38
 PROF (*accounts.models.choices.Title attribute*), 12
 profile (*accounts.models.user.User attribute*), 16
 Profile (*class in accounts.models.profile*), 13
 ProfileInline (*class in accounts.admin*), 21
 ProfileSerializer (*class in accounts.serializers.profile*), 17
 ProfileViewSet (*class in accounts.views.profile*), 19
 pylabber (*module*), 22
 pylabber.settings (*module*), 22
 pylabber.test_settings (*module*), 22
 pylabber.urls (*module*), 22
 pylabber.utils (*module*), 23
 pylabber.wsgi (*module*), 23
- ## Q
- query_associated_studies () (*research.models.measurement_definition.MeasurementDefinition method*), 32
 query_associated_subjects () (*research.models.study.Study method*), 26
 query_measurements () (*research.models.subject.Subject method*), 28

`query_procedures()` (*research.models.subject.Subject method*), 28
`query_run_set()` (*research.models.subject.Subject method*), 28
`query_scores()` (*research.models.subject.Subject method*), 28
`query_studies()` (*research.models.subject.Subject method*), 28
`query_study_groups()` (*research.models.subject.Subject method*), 28
`queryset` (*accounts.views.group.GroupViewSet attribute*), 18
`queryset` (*accounts.views.laboratory.LaboratoryViewSet attribute*), 19
`queryset` (*accounts.views.profile.ProfileViewSet attribute*), 19
`queryset` (*accounts.views.user.UserViewSet attribute*), 20
`queryset` (*research.views.group.GroupViewSet attribute*), 34
`queryset` (*research.views.study.StudyViewSet attribute*), 35
`queryset` (*research.views.subject.SubjectViewSet attribute*), 36
`queryset()` (*research.admin.DecadeBornListFilter method*), 37
`queryset()` (*research.admin.StudyAssociationFilter method*), 40

R

`R` (*research.models.choices.DominantHand attribute*), 24
`RA` (*accounts.models.choices.Role attribute*), 12
`readonly_fields` (*accounts.admin.ExportDestinationAdmin attribute*), 20
`readonly_fields` (*accounts.admin.LaboratoryAdmin attribute*), 20
`readonly_fields` (*accounts.admin.ProfileInline attribute*), 21
`readonly_fields` (*research.admin.CollaboratorsInline attribute*), 37
`readonly_fields` (*research.admin.GroupAdmin attribute*), 37
`readonly_fields` (*research.admin.ProcedureInline attribute*), 38
`readonly_fields` (*research.admin.ProcedureStepInline attribute*), 38
`readonly_fields` (*research.admin.ScanInline attribute*), 39
`readonly_fields` (*research.admin.SessionInline attribute*), 40
`readonly_fields` (*research.admin.StudyAdmin attribute*), 40
`readonly_fields` (*research.admin.StudyInline attribute*), 41
`readonly_fields` (*research.admin.SubjectAdmin attribute*), 41
`readonly_fields` (*research.admin.SubjectsInline attribute*), 41
`ready()` (*accounts.apps.AccountsConfig method*), 21
`ready()` (*research.apps.ResearchConfig method*), 42
`repetition_time()` (*research.admin.ScanInline method*), 39
`research` (*module*), 23
`research.admin` (*module*), 36
`research.apps` (*module*), 42
`research.filters` (*module*), 23
`research.filters.subject_filter` (*module*), 23
`research.models` (*module*), 24
`research.models.choices` (*module*), 24
`research.models.event` (*module*), 30
`research.models.group` (*module*), 25
`research.models.measurement_definition` (*module*), 31
`research.models.procedure` (*module*), 29
`research.models.study` (*module*), 25
`research.models.subject` (*module*), 26
`research.models.task` (*module*), 31
`research.models.validators` (*module*), 33
`research.serializers` (*module*), 33
`research.serializers.group` (*module*), 33
`research.serializers.study` (*module*), 33
`research.serializers.subject` (*module*), 34
`research.urls` (*module*), 42
`research.views` (*module*), 34
`research.views.group` (*module*), 34
`research.views.study` (*module*), 35
`research.views.subject` (*module*), 35
`ResearchConfig` (*class in research.apps*), 42
`Role` (*class in accounts.models.choices*), 12
`run_set` (*accounts.models.user.User attribute*), 16

S

`save()` (*research.models.subject.Subject method*), 28
`scan_count()` (*research.admin.SessionInline method*), 40
`ScanInline` (*class in research.admin*), 38
`ScanInline.Media` (*class in research.admin*), 39
`search_fields` (*research.admin.GroupAdmin attribute*), 37
`search_fields` (*research.admin.SubjectAdmin attribute*), 41
`serializer_class` (*accounts.views.group.GroupViewSet attribute*),

- 18
 serializer_class (accounts.views.laboratory.LaboratoryViewSet attribute), 19
 serializer_class (accounts.views.profile.ProfileViewSet attribute), 19
 serializer_class (accounts.views.user.UserViewSet attribute), 20
 serializer_class (research.views.study.StudyViewSet attribute), 35
 session() (research.admin.ScanInline method), 39
 SessionInline (class in research.admin), 39
 SessionInline.Media (class in research.admin), 39
 Sex (class in research.models.choices), 24
 sex (research.models.subject.Subject attribute), 28
 sex() (research.admin.SubjectsInline method), 41
 sftp() (accounts.admin.ExportDestinationAdmin method), 20
 spatial_resolution() (research.admin.ScanInline method), 39
 step_count() (research.admin.ProcedureAdmin method), 37
 step_set (research.models.procedure.Procedure attribute), 29
 Study (class in research.models.study), 25
 study (research.models.group.Group attribute), 25
 study() (research.admin.GroupAdmin method), 37
 study_set (accounts.models.user.User attribute), 16
 study_set (research.models.procedure.Procedure attribute), 29
 study_set (research.models.subject.Subject attribute), 28
 StudyAdmin (class in research.admin), 40
 StudyAssociationFilter (class in research.admin), 40
 StudyInline (class in research.admin), 40
 StudyInline.Media (class in research.admin), 40
 StudySerializer (class in research.serializers.study), 33
 StudyViewSet (class in research.views.study), 35
 Subject (class in research.models.subject), 26
 subject() (research.admin.ScanInline method), 39
 subject_set (research.models.study.Study attribute), 26
 SubjectAdmin (class in research.admin), 41
 SubjectFilter (class in research.filters.subject_filter), 23
 subjects (research.models.study.Study attribute), 26
 SubjectSerializer (class in research.serializers.subject), 34
 SubjectsInline (class in research.admin), 41
 SubjectViewSet (class in research.views.subject), 35
 suffix (accounts.views.group.GroupViewSet attribute), 18
 suffix (accounts.views.laboratory.LaboratoryViewSet attribute), 19
 suffix (accounts.views.profile.ProfileViewSet attribute), 19
 suffix (accounts.views.user.UserViewSet attribute), 20
 suffix (research.views.group.GroupViewSet attribute), 34
 suffix (research.views.study.StudyViewSet attribute), 35
 suffix (research.views.subject.SubjectViewSet attribute), 36
- ## T
- Task (class in research.models.task), 31
 task (research.models.event.Event attribute), 30
 TaskAdmin (class in research.admin), 41
 time() (research.admin.ScanInline method), 39
 title (accounts.models.profile.Profile attribute), 14
 Title (class in accounts.models.choices), 12
 title (research.admin.DecadeBornListFilter attribute), 37
 title (research.admin.StudyAssociationFilter attribute), 40
 title() (research.admin.CollaboratorsInline method), 37
 title() (research.admin.ProcedureInline method), 38
 title() (research.admin.StudyInline method), 41
 to_csv() (research.views.subject.SubjectViewSet method), 36
 TRANS (research.models.choices.Gender attribute), 24
- ## U
- U (research.models.choices.Sex attribute), 24
 update() (accounts.serializers.user.UserSerializer method), 18
 user (accounts.models.profile.Profile attribute), 14
 User (class in accounts.models.user), 14
 user_permissions (accounts.models.user.User attribute), 16
 UserAdmin (class in accounts.admin), 21
 UserFilter (class in accounts.filters.user), 11
 username() (research.admin.CollaboratorsInline method), 37
 UserSerializer (class in accounts.serializers.user), 18
 UserViewSet (class in accounts.views.user), 19
- ## V
- verbose_name_plural (accounts.admin.ProfileInline attribute), 21

`verbose_name_plural` (*research.admin.CollaboratorsInline* attribute),
37

`verbose_name_plural` (*research.admin.ProcedureInline* attribute),
38

`verbose_name_plural` (*research.admin.ProcedureStepInline* attribute),
38

`verbose_name_plural` (*research.admin.ScanInline* attribute), 39

`verbose_name_plural` (*research.admin.SessionInLine* attribute), 40

`verbose_name_plural` (*research.admin.StudyInline* attribute), 41

`verbose_name_plural` (*research.admin.SubjectsInline* attribute), 41