django$_{dicom}$

Release 0.1.0

Jul 19, 2022

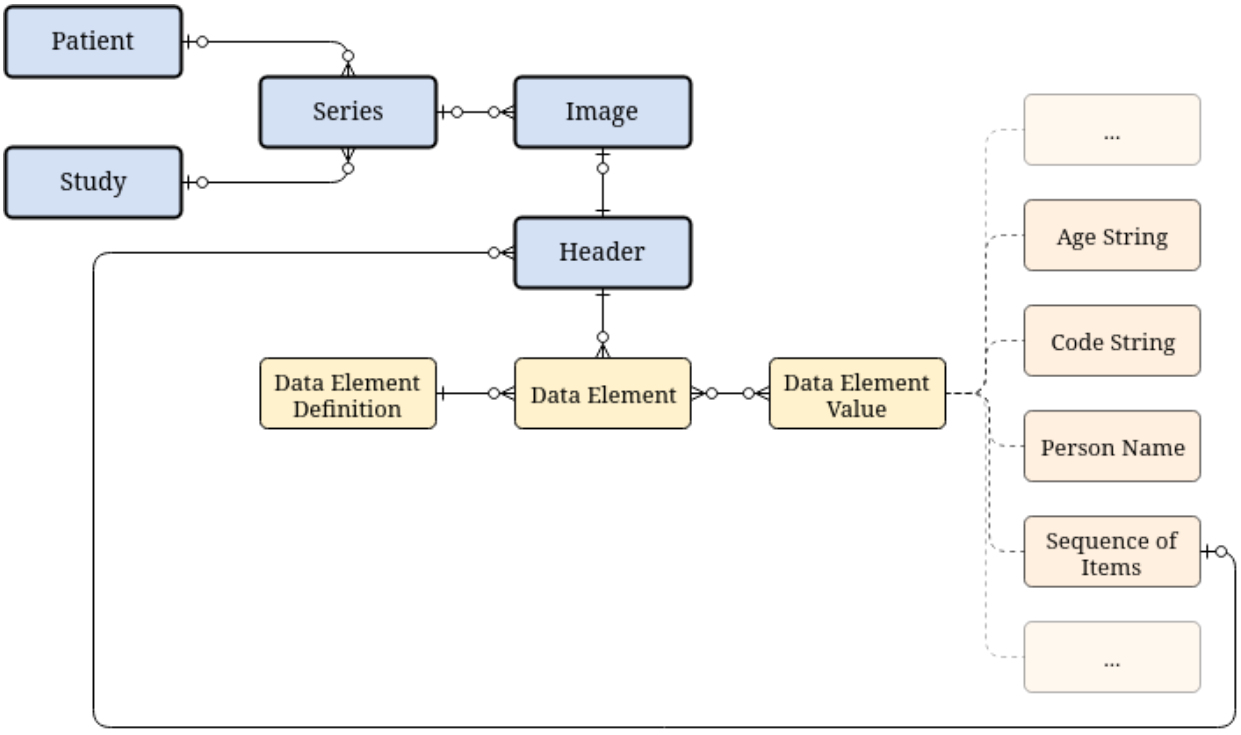
Contents:

CHAPTER 1

Overview

django_dicom is a reusable Django application built to maintain a database of DICOM data. It was created to support the pylabber project, but does not depend on it.

DICOM header information is represented using `Model` subclasses that represent the various entities it may contain (see the DICOM standard specification and this blog post for more information) and provides utility methods to import data and easily maintain the DICOM entities and their relationship.



The fundamental entities (colored in blue) will be created automatically whenever data is *imported to the database* by reading the required header information using dicom_parser.

The *import mode* configuration will determine which data elements will be serialized to the database under that `Header`.

CHAPTER 2

Installation

To install the latest version of *django_dicom*, simply run:

```
pip install django_dicom
```

Once installed:

- Add *django_dicom* to the INSTALLED_APPS:

Listing 1: settings.py

```
INSTALLED_APPS = [
    ...
    'django_dicom',
]
```

- Include *django_dicom*'s URLs:

Listing 2: urls.py

```
urlpatterns = [
    ...
    path("api/", include("django_dicom.urls", namespace="dicom")),
]
```

- Create the database tables:

```
python manage.py migrate django_dicom
```

Warning: *django_dicom* can only integrate with a PostgreSQL database. In order to set-up your project with PostgreSQL follow this DjangoGirls tutorial.

- Start the development server and visit http://127.0.0.1:8000/admin/ or http://127.0.0.1:8000/dicom/.

CHAPTER 3

User Guide

Once *django_dicom* is properly *installed* into your Django project, enter your project's shell by running `./manage.py shell` from within your project's root directory and with your virtual environment activated.

3.1 Import Modes

Currently, three import modes are available:

- Minimal: Do not create *`Header`* or *`DataElement`* instances at all.
- Normal: Create *`Header`*s and save only standard data elements to the database.
- Full: Create *`Header`*s and save all data elements to the database.

By default, *django_dicom*'s import mode will be set to `Normal`.

3.1.1 Changing Import Mode

In order to change the import mode, in your project settings add:

```
DICOM_IMPORT_MODE = "minimal" # or "full"
```

3.2 Importing Data

To import a local repository of *.dcm* files, use the *`ImageManager`*'s *`import_path()`* method:

```
>>> from django_dicom.models import Image

>>> path = '/path/to/local/dcm/repo/'
>>> results = Image.objects.import_path(path)
Importing DICOM data: 4312image [34:24,  2.09image/s]
```

(continues on next page)

(continued from previous page)

```
Successfully imported DICOM data from '/path/to/local/dcm/repo/'!
Created:        4312

>>> results
{'created': 4312, 'existing': 0}
```

You can verify the addition of new data to the database by querying the desired DICOM entity:

```
>>> from django_dicom.models import Image, Patient

>>> Image.objects.count()
4312
>>> Patient.objects.count()
3
```

3.3 Reading Header Information

django_dicom uses the dicom_parser package to read header information and store it as an *Header* instance. Some information is already available to us directly through the values stored in the associated *Image*, *Series*, *Patient*, and *Study* instances. However, *Header* instances are collections of *DataElement* instances, each storing a queryable reference to both the raw and parsed values of a given data element.

By defualt, *django_dicom* will store all official DICOM tags, skipping any private data elements. These will still be easily readable, however, we will not be able to query them from the database. For more information about import modes, see *Import Modes*.

3.3.1 Reading a Single Image's Header

To read a single *Image* instance's header, we can simply use the *instance* property to retrieve dicom_parser's representation of a DICOM image (Image):

```
>>> from django_dicom.models import Image

>>> image = Image.objects.first()
>>> image.instance.header
            Keyword                      VR                VM  Value
Tag
(0008, 0005)  SpecificCharacterSet         Code String       1   ISO_IR 100
(0008, 0008)  ImageType                    Code String       5   ['ORIGINAL', 'PRIMARY
↪', ...
(0008, 0012)  InstanceCreationDate         Date              1   2019-12-18
(0008, 0013)  InstanceCreationTime         Time              1   08:54:41.479000
(0008, 0016)  SOPClassUID                  Unique Identifer  1   1.2.840.10008.5.1.4.
↪1.1.4
(0008, 0018)  SOPInstanceUID               Unique Identifer  1   1.3.12.2.1107.5.2.43.
↪660...
(0008, 0020)  StudyDate                    Date              1   2019-12-18
...
Private Data Elements
=====================
            Keyword                      VR           VM  Value
Tag
```

(continues on next page)

(continued from previous page)

```
(0019, 0010)                                Long String  1    SIEMENS MR HEADER
(0019, 1008)  CsaImageHeaderType            Unknown      1    b'IMAGE NUM 4 '
(0019, 1009)  CsaImageHeaderVersion??       Unknown      1    b'1.0 '
(0019, 100b)  SliceMeasurementDuration      Unknown      1
↪3535
...

>>> image.instance.header.get('PixelSpacing')
[0.48828125, 0.48828125]
```

For more information on dicom_parser's classes and functionality, see the documentation.

3.3.2 Querying the Database

According to the chosen *import mode*, no, some, or all of the saved images' data elements will be serialized to the database as *DataElement* instances. Each *DataElement* represents a single tag within a single header, however, the actual information is available its *definition* and *value* attributes. These attributes associate the *DataElement* instance with reusable *DataElementDefinition* and *DataElementValue* instances, thereby preventing data duplication simplifying queries.

To better understand the way header information is serialized in the database, let's query all the *Series* instances in which the underlying image headers contain a data element named *ImageType* which contains a value of *MOSAIC*.

First, we'll have a look at the relevant *DataElementDefinition* instance:

```
>>> from django_dicom.models import DataElementDefinition
>>> definition = DataElementDefinition.objects.get(keyword="ImageType")
>>> definition
<DataElementDefinition:
Tag                      (0008, 0008)
Keyword                     ImageType
Value Representation               CS
Description               Image Type>
```

Now, let's select any *DataElementValue* instances satisfying our condition:

```
>>> from django_dicom.models import CodeString
>>> values = CodeString.objects.filter(value="MOSAIC")
>>> values.count()
9
```

This means there are 9 different distinct *ImageType* values that contain the string *MOSAIC*. If we want all the related *DataElement* instances, we could:

```
>>> from django_dicom.models import DataElement
>>> data_elements = DataElement.objects.filter(definition=definition, _values__
↪in=values)
>>> data_elements.count()
42236
```

Two important things to note here:

- We used the `_values` attribute to access the raw relationship between the *DataElement* and its associated *DataElementValue*s.

- We used *__in* to query this ManyToMany relationship and retrieve data elements containing any of the desired values. This is necessary because DICOM elements may have multiple values (for more information see value multiplicity).

Now, if we would like to query all the images to which these data elements belong:

```
>>> image_ids = data_elements.values_list('header__image', flat=True)
>>> images = Image.objects.filter(id__in=image_ids)
```

or the series:

```
>>> from django_dicom.models import Series
>>> series_ids = images.values_list('series', flat=True)
>>> series = Series.objects.filter(id__in=series_ids)
>>> series.count()
409
>>> series.first().description
'IR-DTI_30dir_3iso'
```

3.4 Reading Series Data

django_dicom relies on dicom_parser and pydicom's to convert an *Image*'s pixel data from bytes to a NumPy ndarray.

Let's query an anatomical ("MPRAGE"[1]) *Series* instance we've added based on its Series Description:

```
>>> from django_dicom.models import Series
>>> series = Series.objects.filter(description__contains="MPRAGE").first()
```

Great! now all we need to do in order to get a NumPy ndarray of the underlying data would be to use the *data* property:

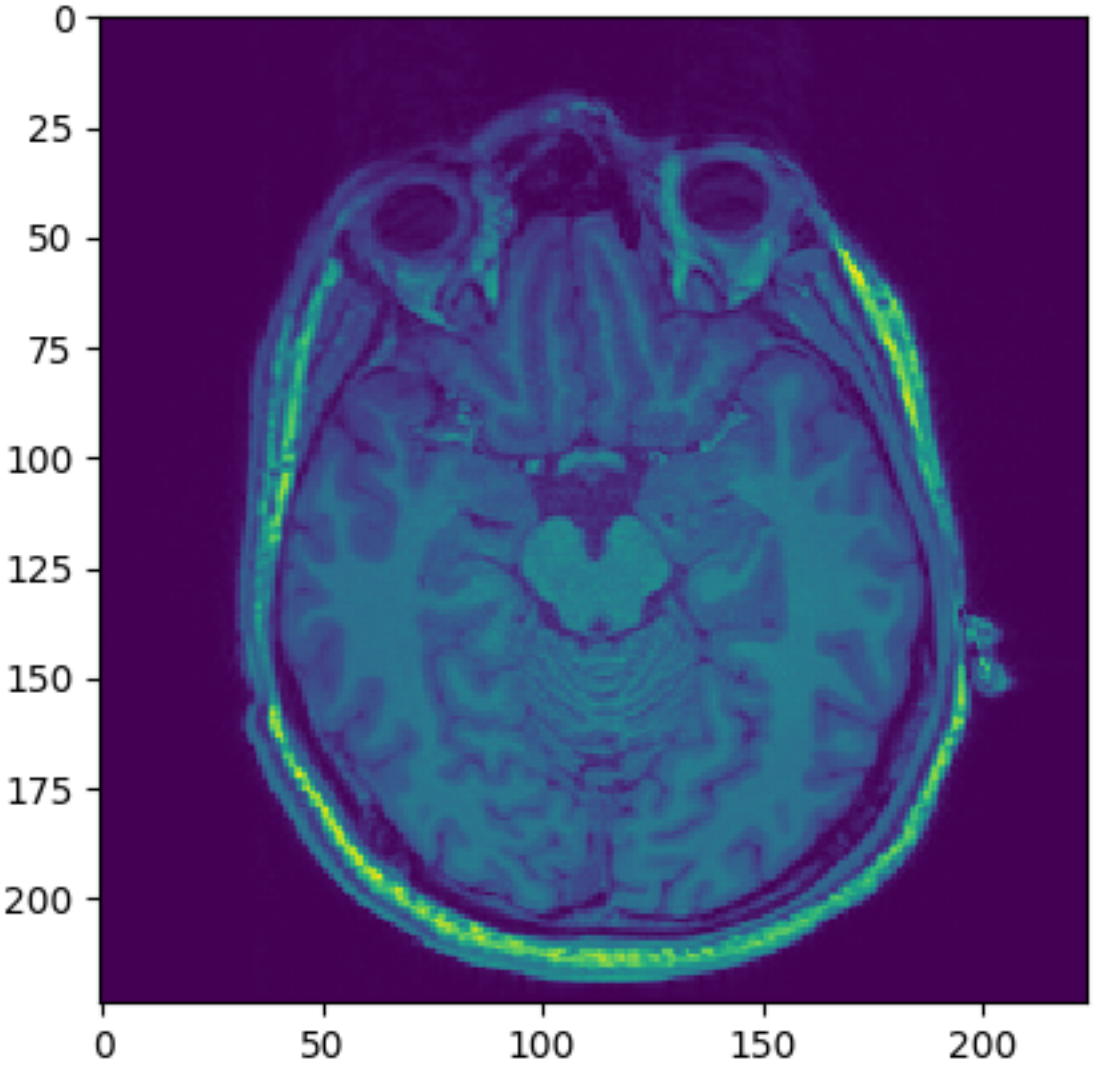
```
>>> data = series.get_data()
>>> series.data.shape
(224, 224, 208)
```

To inspect a particular slice, we could use matplotlib:

```
>>> import matplotlib.pyplot as plt
>>> plt.imshow(series.data[:, :, 100])
>>> plt.show()
```

This should return a figure similar to this:

[1] Brant-Zawadzki, M., Gillan, G. D., & Nitz, W. R. (1992). MP RAGE: a three-dimensional, T1-weighted, gradient-echo sequence–initial experience in the brain. Radiology, 182(3), 769-775.


0
25
50
75
100
125
150
175
200
0
50
100
150
200

CHAPTER 4

Reference

4.1 Module contents

This application manages DICOM files by defining Django models that represent the various DICOM entities.

4.2 Subpackages

4.2.1 Exceptions

Module contents

Definition of custom exceptions that may be raised by *django_dicom*. For more information about error and exceptions see the Python docs.

Submodules

django_dicom.exceptions.import_error module

Definition of the `DicomImportError` class.

exception django_dicom.exceptions.dicom_import_error.**DicomImportError**

Bases: `Exception`

Raised whenever importing new DICOM data to the database fails.

4.2.2 Filters

Module contents

Provides filter classes for the REST API endpoints (for more information see the DRF docs and django-filter's integration with DRF docs).

Submodules

django_dicom.filters.image_filter module

Definition of the *ImageFilter* class.

class django_dicom.filters.image_filter.**ImageFilter**(*data=None*, *queryset=None*, ***, *request=None*, *prefix=None*)

Bases: django_filters.rest_framework.filterset.FilterSet

Provides filtering functionality for the *ImageViewSet*.

Available filters are:

- *id*: Primary key
- *series_uid*: Series instance UID (contains)
- *series_description*: Series description (contains)
- *number*: Series number (exact)
- *created_after_date*: Create after date
- *created_before_date*: Create before date
- *created_after_time*: Create after time
- *created_before_time*: Create before time

class Meta

Bases: object

fields = ('id', 'uid', 'number')

model

alias of *django_dicom.models.image.Image*

base_filters = {'created_after_date': <django_filters.filters.DateFilter object>, 'cr

declared_filters = {'created_after_date': <django_filters.filters.DateFilter object>,

django_dicom.filters.patient_filter module

Definition of the FilterSet subclass that will be assigned to the *PatientViewSet*'s *filter_class* attribute value.

class django_dicom.filters.patient_filter.**PatientFilter**(*data=None*, *queryset=None*, ***, *request=None*, *prefix=None*)

Bases: django_filters.rest_framework.filterset.FilterSet

Provides filtering functionality for the *PatientViewSet*.

Available filters are:

- *id*: Primary key

- *uid*: Patient UID (contains, icontains, or exact)
- *born_after_date*: Earliest date of birth
- *born_before_date*: Latest date of birth
- *name_prefix*: Any of the existing *name_prefix* values in the database
- *given_name*: Given name value (contains, icontains, or exact)
- *middle_name*: Middle name value (contains, icontains, or exact)
- *family_name*: Family name value (contains, icontains, or exact)
- *name_suffix*: Any of the existing *name_prefix* values in the database
- *sex*: Any of the sex options defined in the `Sex Enum`
- *study__id*: Related *Study* ID

class Meta

Bases: `object`

fields = ('id', 'uid')

model

alias of *django_dicom.models.patient.Patient*

base_filters = {'date_of_birth': <django_filters.filters.DateFromToRangeFilter object

declared_filters = {'date_of_birth': <django_filters.filters.DateFromToRangeFilter ob

filter_by_study(*queryset: django.db.models.query.QuerySet*, *name: str*, *value: int*) → django.db.models.query.QuerySet

Returns all *Patient* instances that have *Series* instances belonging to the *Study* with the specified *value* as primary key.

Used by `study__id`.

Parameters

- **queryset** (`QuerySet`) – *Patient* instances
- **name** (*str*) – Name of the queried filter field
- **value** (*int*) – *Study* primary key

Returns Filtered *Patient* instances

Return type `QuerySet`

django_dicom.filters.series_filter module

Definition of the *SeriesFilter* class.

class django_dicom.filters.series_filter.**SeriesFilter**(*data=None*, *queryset=None*, ***, *request=None*, *prefix=None*)

Bases: `django_filters.rest_framework.filterset.FilterSet`

Provides filtering functionality for the *SeriesViewSet*.

Available filters are:

- *id*: Primary key
- *uid*: Series Instance UID

- *patient_id*: Related `Patient` instance's primary key
- *study_uid*: Related `Study` instance's `uid` value
- *study_description*: Related `Study` instance's `description` value (in-icontains)
- *modality*: Any of the values defined in `Modality`
- *description*: Series description value (contains, icontains, or exact)
- *number*: Series number value
- *protocol_name*: Protocol name value (contains)
- *scanning_sequence*: Any combination of the values defined in `ScanningSequence`
- *sequence_variant*: Any combination of the values defined in `SequenceVariant`
- *echo_time*: `echo_time` value
- *inversion_time*: `inversion_time` value
- *repetition_time*: `repetition_time` value
- *flip_angle*: Any of the existing `flip_angle` in the database
- *date_after*: Exact `date` value
- *date_before*: Create before date
- *time_after*: Create after time
- *time_before*: Create before time
- *manufacturer*: Any of the existing `manufacturer` in the database
- *manufacturer_model_name*: Any of the existing `manufacturer_model_name` in the database
- *device_serial_number*: Any of the existing `device_serial_number` in the database
- *institution_name*: Any of the existing `institution_name` in the database
- *pulse_sequence_name*: `pulse_sequence_name` value (in-icontains)
- *sequence_name*: `sequence_name` value (in-icontains)

class Meta

Bases: `object`

fields = ('id', 'uid', 'number', 'patient__id')

model

alias of `django_dicom.models.series.Series`

base_filters = {'date': <django_filters.filters.DateRangeFilter object>, 'description

declared_filters = {'date': <django_filters.filters.DateRangeFilter object>, 'descrip

django_dicom.filters.series_filter.**filter_array**(*queryset: django.db.models.query.QuerySet*, *field_name: str*, *value: list*)

Returns an exact lookup for a PostgreSQL ArrayField.

Parameters

- **queryset** (`QuerySet`) – The filtered queryset
- **field_name** (*str*) – The name of the field the queryset is being filtered by
- **value** (*list*) – The values to filter by

django_dicom.filters.series_filter.**filter_header**(*queryset: django.db.models.query.QuerySet*, *field_name: str*, *values: str*)

Returns a desired lookup for a DicomHeader field.

Parameters

- **queryset** (QuerySet) – The filtered queryset
- **field_name** (*str*) – The name of the field the queryset is being filtered by
- **values** (*dict*) – The fields and values to filter by

django_dicom.filters.series_filter.**filter_in_string**(*queryset: django.db.models.query.QuerySet*, *field_name: str*, *values: list*)

Returns a in-icontains mixed lookup with 'or' between values for a CharField.

Parameters

- **queryset** (QuerySet) – The filtered queryset
- **field_name** (*str*) – The name of the field the queryset is being filtered by
- **values** (*str*) – The values to filter by

django_dicom.filters.study_filter module

Definition of the *StudyFilter* class.

class django_dicom.filters.study_filter.**StudyFilter**(*data=None*, *queryset=None*, ***, *request=None*, *prefix=None*)

Bases: django_filters.rest_framework.filterset.FilterSet

Provides filtering functionality for the *StudyViewSet*.

Available filters are:

- *id*: Primary key
- *uid*: Study instance UID
- *description*: Study description (contains, icontains, or exact)
- *created_after_date*: Create after date
- *created_before_date*: Create before date
- *created_after_time*: Create after time
- *created_before_time*: Create before time

class Meta

Bases: object

fields = ('id',)

model

alias of *django_dicom.models.study.Study*

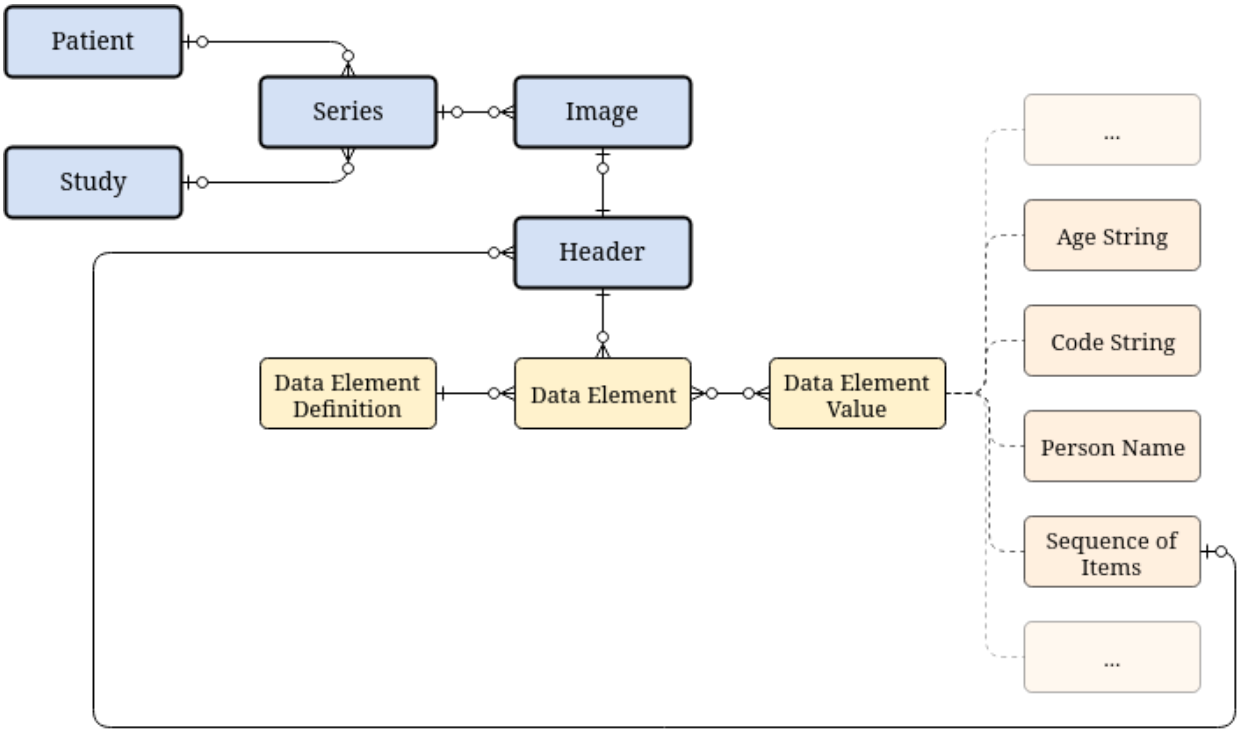
base_filters = {'date': <django_filters.filters.DateFromToRangeFilter object>, 'descr

declared_filters = {'date': <django_filters.filters.DateFromToRangeFilter object>, 'd

4.2.3 Models

Module contents

Creates `Model` subclasses to represent the various DICOM entities.



Subpackages

django_dicom.models.managers package

Module contents

`Manager` subclasses for some of the app's models.

References

- Django's documentation on managers.

Subpackages

django_dicom.models.managers.values package

Module contents

Custom managers for specific *`DataElementValue`* subclasses.

Submodules

django_dicom.models.managers.values.sequence_of_items module

Definition of the *SequenceOfItemsManager* class.

class django_dicom.models.managers.values.sequence_of_items.**SequenceOfItemsManager**
Bases: *django_dicom.models.managers.data_element_value.DataElementValueManager*

Custom manager for the *SequenceOfItems* model.

from_dicom_parser(*data_element: dicom_parser.data_element.DataElement*) → tuple
Create a sequence of items by reading the included headers and populating the database accordingly.

Parameters **data_element** (*DicomDataElement*) – Sequence of items data element

Returns The data element and whether is was created or not

Return type Tuple[*DataElementValue*, bool]

Submodules

django_dicom.models.managers.data_element module

Definition of a custom Manager for the *DataElement* model.

class django_dicom.models.managers.data_element.**DataElementManager**
Bases: django.db.models.manager.Manager

Custom Manager for the *DataElement* model.

create_from_dicom_parser(*header*, *definition*, *data_element: dicom_parser.data_element.DataElement*)
Creates a new instance under *header* using the provided *definition* and *data_element*.

Parameters

- **header** (*Header*) – The header instance with which the created data element should be associated.
- **definition** (*DataElementDefinition*) – The data element definition of the created data element
- **data_element** (dicom_parser.data_element.DataElement) – Object representing a single data element in memory

Returns The created instance

Return type *DataElement*

from_dicom_parser(*header*, *data_element: dicom_parser.data_element.DataElement*)
Creates a new instance under *header* using the provided dicom_parser.data_element.DataElement instance.

Parameters

- **header** (*Header*) – The header instance with which the created data element should be associated.
- **data_element** (dicom_parser.data_element.DataElement) – Object representing a single data element in memory

Returns The created instance

Return type *DataElement*

django_dicom.models.managers.data_element_definition module

Definition of a custom `Manager` for the *DataElementDefinition* model.

class django_dicom.models.managers.data_element_definition.**DataElementDefinitionManager**
Bases: `django.db.models.manager.Manager`

Custom `Manager` for the *DataElementDefinition* model.

from_dicom_parser(*data_element: dicom_parser.data_element.DataElement*) → tuple
Gets or creates a *DataElementDefinition* instance from a dicom_parser `DataElement`.

Parameters **data_element** (`dicom_parser.data_element.DataElement`) – Object representing a single data element in memory

Returns data_element_definition, created

Return type tuple

django_dicom.models.managers.data_element_definition.**data_element_to_definition**(*data_element: dicom_parser.dat* → dict
Converts a dicom_parser `DataElement` to a dictionary of keyword arguments that may be used to instantiate a *DataElementDefinition* instance.

Parameters **data_element** (`dicom_parser.data_element.DataElement`) – Object representing a single data element in memory

Returns *DataElementDefinition* instantiation keyword arguments

Return type dict

django_dicom.models.managers.data_element_value module

Definition of a custom `InheritanceManager` for the *DataElementValue* model.

For more information about the `InheritanceManager` class, see django-model-utils's InheritanceManager documentation.

class django_dicom.models.managers.data_element_value.**DataElementValueManager**
Bases: `model_utils.managers.InheritanceManager`

Custom `InheritanceManager` for the *DataElementValue* model.

from_dicom_parser(*data_element: dicom_parser.data_element.DataElement*) → Tuple
Get or create some *DataElementValue* subclass instances from a dicom_parser `DataElement`.

Parameters **data_element** (`dicom_parser.data_element.DataElement`) – Object representing a single data element in memory

Returns data_element_value or data_element_values, created

Return type Tuple[Union[*DataElementValue*, List[*DataElementValue*, ..]], bool]

get_or_create_from_nonsequence(*data_element: dicom_parser.data_element.DataElement*) → Tuple

Get or create some `DataElementValue` subclass instances from a non-*Sequence of Items* dicom_parser `DataElement`.

Parameters **data_element** (`dicom_parser.data_element.DataElement`) – Object representing a single data element in memory

Returns data_element_value or data_element_values, created

Return type Tuple[Union[*DataElementValue*, List[*DataElementValue*, ..]], bool]

handle_invalid_data(*ValueModel*, *data_element: dicom_parser.data_element.DataElement*, *error: Exception*) → Tuple

If reading the value from the data element using dicom_parser raises an exception, this method is called to create an "empty" instance of the *ValueModel* (i.e. with `raw` and `value` set to *None*) and log the exception in the `warnings` field.

Parameters

- **ValueModel** (`DataElementValue`) – Some `DataElementValue` subclass used to instatiate values
- **data_element** (`dicom_parser.data_element.DataElement`) – Object representing a single data element in memory
- **error** (`Exception`) – The raised exception

Returns data_element_value, created

Return type tuple

handle_multiple_values(*ValueModel*, *data_element: dicom_parser.data_element.DataElement*) → Tuple[list, bool]

Handles data elements with a `value_multiplicity` greater than 1.

Parameters

- **ValueModel** (`DataElementValue`) – Some `DataElementValue` subclass used to instatiate values
- **data_element** (`dicom_parser.data_element.DataElement`) – Object representing a single data element in memory

Returns data_element_values, any_created

Return type Tuple[List[*DataElementValue*, ..], bool]

handle_no_value(*ValueModel*) → Tuple

Handles data elements with a `value_multiplicity` of 0. Returns an "empty" instance of the *ValueModel* (i.e. with `raw` and `value` set to *None*).

Parameters **ValueModel** (`DataElementValue`) – Some `DataElementValue` subclass used to instatiate values

Returns dicom_data_element, created

Return type Tuple[DicomDataElement, bool]

handle_single_value(*ValueModel*, *data_element: dicom_parser.data_element.DataElement*) → tuple

Handles data elements with a `value_multiplicity` of 1.

Parameters

- **ValueModel** (*DataElementValue*) – Some *DataElementValue* subclass used to instatiate values
- **data_element** (dicom_parser.data_element.DataElement) – Object representing a single data element in memory

Returns data_element_value, created

Return type Tuple[*DataElementValue*, bool]

handle_value_multiplicity(*ValueModel*, *data_element: dicom_parser.data_element.DataElement*) → Tuple

Handles the creation of the *DataElementValue* subclass instances according to the dicom_parser DataElement's value_multiplicity attribute.

Parameters

- **ValueModel** (*DataElementValue*) – Some *DataElementValue* subclass used to instatiate values
- **data_element** (dicom_parser.data_element.DataElement) – Object representing a single data element in memory

Returns data_element_value or data_element_values, created

Return type Tuple[Union[*DataElementValue*, List[*DataElementValue*, ..]], bool]

django_dicom.models.managers.dicom_entity module

Definition of a custom Manager for the *DicomEntity* model.

class django_dicom.models.managers.dicom_entity.**DicomEntityManager**

Bases: django.db.models.manager.Manager

Custom Manager for the *DicomEntity* model.

from_header(*header*) → Tuple

Get or create an instance using the provided header.

Parameters **header** (*django_dicom.models.header.Header*) – Header instance to query this entity's information from

Returns dicom_entity, created

Return type Tuple[*DicomEntity*, bool]

django_dicom.models.managers.header module

Definition of a custom Manager for the *Header* model.

class django_dicom.models.managers.header.**HeaderManager**

Bases: django.db.models.manager.Manager

Custom Manager for the *Header* model.

from_dicom_parser(*header: dicom_parser.header.Header*, ***kwargs*)

Creates a new instance from a dicom_parser dicom_parser.header.Header.

Parameters **header** (dicom_parser.header.Header) – Object representing an entire DICOM header in memory

Returns Created instance

Return type *django_dicom.models.header.Header*

Raises DicomImportError – DICOM header read error

django_dicom.models.managers.image module

Definition of the *ImageManager* class.

class django_dicom.models.managers.image.**ImageManager**

Bases: *django_dicom.models.managers.dicom_entity.DicomEntityManager*

Custom Manager for the *Image* model.

TEMP_DCM_FILE_NAME = 'tmp.dcm'

Name given to DICOM files that need to be saved locally in order to be read.

create_from_dcm(*path: pathlib.Path*, *autoremove: bool = True*)

Creates an *Image* instance from a given path.

Parameters

- **path** (pathlib.Path) – Local *.dcm* file path
- **autoremove** (*bool, optional*) – Whether to remove the local copy of the *.dcm* file under MEDIA_ROOT if creation fails, by default True

Returns The created image

Return type *Image*

get_or_create(**args*, ***kwargs*) → Tuple

Overrides get_or_create() to call *get_or_create_from_dcm()* in case the *dcm* keyword argument is provided.

Returns image, created

Return type Tuple[*Image*, bool]

get_or_create_from_dcm(*path: pathlib.Path*, *autoremove: bool = True*) → Tuple

Gets or creates an *Image* instance based on the contents of the provided *.dcm* path.

Parameters

- **path** (pathlib.Path) – Local *.dcm* file path
- **autoremove** (*bool, optional*) – Whether to remove the local copy of the *.dcm* file under MEDIA_ROOT if creation fails, by default True

Returns image, created

Return type Tuple[*Image*, bool]

import_path(*path: pathlib.Path*, *progressbar: bool = True*, *report: bool = True*, *persistent: bool = True*, *pattern: bool = '*.dcm'*, *autoremove: bool = True*) → django.db.models.query.QuerySet

Iterates the given directory tree and imports any *.dcm* files found within it.

Parameters

- **path** (pathlib.Path) – Base path for recursive *.dcm* import
- **progressbar** (*bool, optional*) – Whether to display a progressbar or not, by default True

- **report** (*bool, optional*) – Whether to print out a summary report when finished or not, by default True
- **persistent** (*bool, optional*) – Whether to continue and raise a warning or to raise an exception when failing to read a DICOM file's header
- **pattern** (*str, optional*) – Globbing pattern to use for file import

Returns The created *Image* instances

Return type QuerySet

report_import_path_results(*path: pathlib.Path*, *counter: dict*) → None
Reports the result of a recursive path import.

Parameters

- **path** (pathlib.Path) – Base path of DICOM data import
- **counter** (*dict*) – Dictionary containing *created* and *existing* keys containing the number of files which fit in each category.

store_image_data(*image_data: _io.BufferedReader*) → pathlib.Path
Stores binary image data to a temporary local path under the project's MEDIA_ROOT.

Parameters **image_data** (io.BufferedReader) – Binary DICOM image data

Returns Path of the created file

Return type pathlib.Path

django_dicom.models.utils package

Module contents

Utilities for the *models* module.

Submodules

django_dicom.models.utils.fields module

Custom Field subclasses.

class django_dicom.models.utils.fields.**ChoiceArrayField**(*base_field*, *size=None*, ***kwargs*)
Bases: django.contrib.postgres.fields.array.ArrayField

A field that allows us to store an array of choices. Uses Django 1.9's postgres ArrayField and a MultipleChoiceField for its formfield.

formfield(***kwargs*)
Return a django.forms.Field instance for this field.

django_dicom.models.utils.help_text module

help_text string constants for the various fields.

django_dicom.models.utils.logs module

Log messages to be used by models.

django_dicom.models.utils.meta module

django_dicom.models.utils.meta.**get_model**(*model_name: str*) → django.db.models.base.Model

django_dicom.models.utils.utils module

class django_dicom.models.utils.utils.**ImportMode**

Bases: enum.Enum

An enumeration.

FULL = 'Full'

MINIMAL = 'Minimal'

NORMAL = 'Normal'

django_dicom.models.utils.utils.**check_element_inclusion**(*data_element*) → bool

django_dicom.models.utils.utils.**get_dicom_root**() → pathlib.Path

django_dicom.models.utils.utils.**get_group_model**()

django_dicom.models.utils.utils.**get_import_configuration**() → dict

django_dicom.models.utils.utils.**get_import_mode**() → django_dicom.models.utils.utils.ImportMode

django_dicom.models.utils.utils.**get_mri_root**() → pathlib.Path

django_dicom.models.utils.utils.**get_subject_model**()

django_dicom.models.utils.utils.**snake_case_to_camel_case**(*string: str*) → str

django_dicom.models.utils.validators module

Definition of custom field validations.

django_dicom.models.utils.validators.**validate_file_extension**(*value*)

django_dicom.models.values package

Module contents

Definition of *DataElementValue* subclasses for every type of value representation (VR).

Hint: For more information about value representations see the official DICOM standard (part 05, section 6.2).

Submodules

django_dicom.models.values.age_string module

Definition of the *AgeString* model.

class django_dicom.models.values.age_string.**AgeString**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single *AgeString* data element value.

raw
Overrides *raw* to assign a CharField.

value
Overrides *value* to assign a FloatField.

django_dicom.models.values.application_entity module

Definition of the *ApplicationEntity* model.

class django_dicom.models.values.application_entity.**ApplicationEntity**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single *ApplicationEntity* data element value.

raw
Overrides *raw* to assign a CharField.

value
Overrides *value* to assign a CharField.

django_dicom.models.values.code_string module

Definition of the *CodeString* model.

class django_dicom.models.values.code_string.**CodeString**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single *CodeString* data element value.

raw
Overrides *raw* to assign a CharField.

value
Overrides *value* to assign a CharField.

django_dicom.models.values.csa_header module

Definition of the *CsaHeader* model.

class django_dicom.models.values.csa_header.**CsaHeader**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single Siemens' CSA header value.

Hint: For more information about CSA headers, see dicom_parser's CSA headers documentation.

raw
Overrides *raw* to assign a `TextField`.

to_html(*verbose: bool = False*, ***kwargs*) → str
Returns the HTML representation of this instance.

If the *verbose* keyword argument is passed as *True*, returns the entire header as JSON. Otherwise, returns an HTML link to this instance in the admin site.

Parameters **verbose** (*bool, optional*) – Whether to return all of the header information or just a link, by default False

Returns HTML text containing either JSON encoded header information or a link to the admin site

Return type str

value
Overrides *value* to assign a `JSONField`.

django_dicom.models.values.data_element_value module

Definition of the *DataElementValue* model.

class django_dicom.models.values.data_element_value.**DataElementValue**(**args*, ***kwargs*)

Bases: `django.db.models.base.Model`

A parent `Model` representing a single value contained by some *DataElement* instance. If the data element has value multiplicity greater than 1, it will have multiple instances of this model associated with it, each with its own *index* value.

admin_link
Creates an HTML tag to link to this instance within the admin site.

Returns Link to this instance in the admin site

Return type str

agestring
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.

applicationentity
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.

codestring
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.

csaheader
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.

data_element_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.

date
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.

datetime
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.

decimalstring
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.

floatingpointdouble
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.

floatingpointsingle
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.

get_raw_peek (*size: int = 100*) → str
Returns a truncated string of the raw data element's value (appended with "... " if changed).

Parameters **size** (*int, optional*) – Maximal string length, by default 100

Returns Truncated string

Return type str

index
If the value is one of a number of values within a DataElement (a DataElement with a value multiplicity that is greater than 1), this field keeps the index of this value.

integerstring
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.

longstring
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.

longtext
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.

objects = <django_dicom.models.managers.data_element_value.DataElementValueManager obj

otherword
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.

personname
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.

raw = None
Raw data element value (meant to be overridden by child models).

sequenceofitems
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.

shortstring
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.

shorttext
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.

signedlong
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.

signedshort
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.

time
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.

to_html(***kwargs*) → str
Returns the HTML representation of this instance. This method simlpy returns the *value*, child models should override to provide an appropriate representation.

Returns HTML representation of this instance

Return type str

uniqueidentifier
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.

unknown
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.

unlimitedtext
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.

unsignedlong
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.

unsignedshort
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.

value = None
Interpreted data element value (meant to be overridden by child models).

warnings
If any warnings were raised by *dicom_parser*, log them in the database.

django_dicom.models.values.date module

Definition of the *Date* model.

class django_dicom.models.values.date.**Date**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A `Model` representing a single *Date* data element value.

raw
Overrides *raw* to assign a `CharField`.

value
Overrides *value* to assign a `CharField`.

django_dicom.models.values.datetime module

Definition of the *DateTime* model.

class django_dicom.models.values.datetime.**DateTime**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A `Model` representing a single *DateTime* data element value.

get_next_by_value(*, *field=<django.db.models.fields.DateTimeField: value>*, *is_next=True*, ***kwargs*)

get_previous_by_value(*, *field=<django.db.models.fields.DateTimeField: value>*, *is_next=False*, ***kwargs*)

raw
Overrides *raw* to assign a `CharField`.

value
Overrides *value* to assign a `CharField`.

django_dicom.models.values.decimal_string module

Definition of the *DecimalString* model.

class django_dicom.models.values.decimal_string.**DecimalString**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single *DecimalString* data element value.

raw
Overrides *raw* to assign a CharField.

value
Overrides *value* to assign a FloatField.

django_dicom.models.values.floating_point_double module

Definition of the *FloatingPointDouble* model.

class django_dicom.models.values.floating_point_double.**FloatingPointDouble**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single *FloatingPointDouble* data element value.

raw
Overrides *raw* to assign a CharField.

value
Overrides *value* to assign a FloatField.

django_dicom.models.values.floating_point_single module

Definition of the *FloatingPointSingle* model.

class django_dicom.models.values.floating_point_single.**FloatingPointSingle**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single *FloatingPointSingle* data element value.

raw
Overrides *raw* to assign a CharField.

value
Overrides *value* to assign a FloatField.

django_dicom.models.values.integer_string module

Definition of the *IntegerString* model.

class django_dicom.models.values.integer_string.**IntegerString**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single *IntegerString* data element value.

raw
Overrides *raw* to assign a CharField.

value
Overrides *value* to assign a IntegerField.

django_dicom.models.values.integer_string.**MAX_VALUE = 2147483647**
Maximal *IntegerString* value.

django_dicom.models.values.integer_string.**MIN_VALUE = -2147483648**
Minimal *IntegerString* value.

django_dicom.models.values.long_string module

Definition of the *LongString* model.

class django_dicom.models.values.long_string.**LongString**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single *LongString* data element value.

raw
Overrides *raw* to assign a CharField.

value
Overrides *value* to assign a CharField.

django_dicom.models.values.long_text module

Definition of the *LongText* model.

class django_dicom.models.values.long_text.**LongText**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single *LongText* data element value.

raw
Overrides *raw* to assign a CharField.

value
Overrides *value* to assign a TextField.

django_dicom.models.values.other_word module

Definition of the *OtherWord* model.

class django_dicom.models.values.other_word.**OtherWord**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single *OtherWord* data element value.

raw
Overrides *raw* to assign a BinaryField.

value
Overrides *value* to assign a IntegerField.

django_dicom.models.values.person_name module

Definition of the *PersonName* model.

class django_dicom.models.values.person_name.**PersonName**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single *PersonName* data element value.

raw
Overrides *raw* to assign a BinaryField.

to_html(***kwargs*) → str
Returns the HTML representation of this instance.

Returns HTML representation of this instance

Return type str

value
Overrides *value* to assign a JSONField.

django_dicom.models.values.sequence_of_items module

Definition of the *SequenceOfItems* model.

class django_dicom.models.values.sequence_of_items.**SequenceOfItems**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A django.db.models.Model representing the value stored in a single *DataElement* with a value representation (VR) of *SQ*.

Sequence of Items data elements are arrays of nested headers, and therefore this model does not override the *raw* and *value* field definitions (and so they remain *None*).

header_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.

objects = <django_dicom.models.managers.values.sequence_of_items.SequenceOfItemsManage

to_html(*verbose: bool = False*, ***kwargs*) → str
Returns the HTML representation of this instance. This method simlpy returns the *value*, child models should override to provide an appropriate representation.

Returns HTML representation of this instance

Return type str

django_dicom.models.values.short_string module

Definition of the *ShortString* model.

class django_dicom.models.values.short_string.**ShortString**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single *ShortString* data element value.

raw
Overrides *raw* to assign a CharField.

value
Overrides *value* to assign a Charield.

django_dicom.models.values.short_text module

Definition of the ShortString model.

class django_dicom.models.values.short_text.**ShortText**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single *ShortText* data element value.

raw
Overrides *raw* to assign a CharField.

value
Overrides *value* to assign a Charield.

django_dicom.models.values.signed_long module

Definition of the *SignedLong* model.

django_dicom.models.values.signed_long.**MAX_VALUE = 2147483647**
Maximal *SignedLong* value.

django_dicom.models.values.signed_long.**MIN_VALUE = -2147483648**
Minimal *SignedLong* value.

class django_dicom.models.values.signed_long.**SignedLong**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single *SignedLong* data element value.

raw
Overrides *raw* to assign an IntegerField.

value
Overrides *value* to assign an IntegerField.

django_dicom.models.values.signed_short module

Definition of the *SignedShort* model.

django_dicom.models.values.signed_short.**MAX_VALUE = 32767**
Maximal *SignedShort* value.

django_dicom.models.values.signed_short.**MIN_VALUE = -32768**
Minimal *SignedShort* value.

class django_dicom.models.values.signed_short.**SignedShort**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single *SignedShort* data element value.

raw
Overrides *raw* to assign an IntegerField.

value
Overrides *value* to assign an IntegerField.

django_dicom.models.values.time module

Definition of the *Time* model.

class django_dicom.models.values.time.**Time**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single *Time* data element value.

raw
Overrides *raw* to assign a CharField.

to_html(***kwargs*) → str
Returns the HTML representation of this instance.

Returns HTML representation of this instance

Return type str

value
Overrides *value* to assign a TimeField.

django_dicom.models.values.unique_identifier module

Definition of the *UniqueIdentifier* model.

class django_dicom.models.values.unique_identifier.**UniqueIdentifier**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single *UniqueIdentifier* data element value.

raw
Overrides *raw* to assign a CharField.

value
Overrides *value* to assign a CharField.

django_dicom.models.values.unknown module

Definition of the *Unknown* model.

class django_dicom.models.values.unknown.**Unknown**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single *Unknown* data element value.

raw
Overrides *raw* to assign a TextField.

value
Overrides *value* to assign a TextField.

django_dicom.models.values.unlimited_text module

Definition of the *UnlimitedText* model.

class django_dicom.models.values.unlimited_text.**UnlimitedText**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single *Unknown* data element value.

raw
Overrides *raw* to assign a TextField.

value
Overrides *value* to assign a TextField.

django_dicom.models.values.unsigned_long module

Definition of the *UnsignedLong* model.

django_dicom.models.values.unsigned_long.**MAX_VALUE = 4294967296**
Maximal *UnsignedLong* value.

class django_dicom.models.values.unsigned_long.**UnsignedLong**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single *UnsignedLong* data element value.

raw
Overrides *raw* to assign a PositiveIntegerField.

value
Overrides *value* to assign a PositiveIntegerField.

django_dicom.models.values.unsigned_short module

Definition of the *UnsignedShort* model.

django_dicom.models.values.unsigned_short.**MAX_VALUE = 65536**
Maximal *UnsignedShort* value.

class django_dicom.models.values.unsigned_short.**UnsignedShort**(**args*, ***kwargs*)
Bases: *django_dicom.models.values.data_element_value.DataElementValue*

A Model representing a single *UnsignedShort* data element value.

raw
Overrides *raw* to assign a PositiveIntegerField.

value
Overrides *value* to assign a PositiveIntegerField.

django_dicom.models.values.vr_to_model module

A utility module used to get the appropriate *DataElementValue* subclass (*"ValueModel"*) for a given data element.

django_dicom.models.values.vr_to_model.**TAG_TO_MODEL = {('0029', '1010'): 'CsaHeader', ('00**
Special cases might require a custom *DataElementValue* subclass based on the element's tag rather than its VR.

django_dicom.models.values.vr_to_model.**VR_TO_MODEL = {<ValueRepresentation.AE: 'Applicatio**
Dictionary with ValueRepresentation items as keys and strings representing the appropriate *DataElementValue* subclass as values.

django_dicom.models.values.vr_to_model.**get_value_model**(*data_element: dicom_parser.data_element.DataElement*)
Returns the *DataElementValue* subclass matching the given data element.

Parameters **data_element** (dicom_parser.data_element.DataElement) – Object representing a single data element in memory

Returns Value model

Return type *DataElementValue*

django_dicom.models.values.vr_to_model.**get_value_model_name**(*data_element: dicom_parser.data_element.DataElement*) → str
Returns the name of the *DataElementValue* subclass matching the given data element.

Parameters **data_element** (dicom_parser.data_element.DataElement) – Object representing a single data element in memory

Returns Name of the appropriate *"ValueModel"*

Return type str

Submodules

django_dicom.models.data_element module

Definition of the *DataElement* class.

class django_dicom.models.data_element.**DataElement**(**args*, ***kwargs*)
Bases: django.db.models.base.Model

A model representing a single DICOM data element.

Each *DataElement* instance belongs to a *Header*, and each *Header* belongs to an *Image* or *SequenceOfItems*.

While the *DataElement* instance holds the reference to the associated models, the defining characteristics of the data element are saved as a *DataElementDefinition* instance and the values are saved as *DataElementValue* subclass instances in order to prevent data duplication.

admin_link
Returns an HTML tag linking to this instance in the admin site.

Returns HTML link to this instance

Return type str

definition
The *DataElementDefinition* # noqa: E501 instance holding information about this element's DICOM tag.

header
The *Header* instance to which this data element belongs.

objects = <django_dicom.models.managers.data_element.DataElementManager object>

to_html(***kwargs*) → str
Returns an HTML representation of this instance.

Any keyword arguments will be passed to the associated *DataElementValue* subclass instances.

Returns HTML representaion of this instance

Return type str

to_verbose_dict() → dict
Returns a dictionary representation of this instance.

Returns This instance's information

Return type dict

to_verbose_series() → pandas.core.series.Series
Returns a `Series` representation of this instance.

Returns This instance's information

Return type `pandas.Series`

value
Returns the value or values (according to the value multiplicity) of the associated *DataElementValue* instances.

Returns Data element value

Return type Any

value_multiplicity
Returns the number of *DataElementValue* related to this instance.

Returns Value multiplicity

Return type int

Hint: For more information see the DICOM standard's definition of value multiplicity.

---

django_dicom.models.data_element_definition module

Definition of the *DataElementDefinition* class.

class django_dicom.models.data_element_definition.**DataElementDefinition**(**args*, ***kwargs*)

Bases: `django.db.models.base.Model`

A model representing a single DICOM data element definition.

Notes

For more information see the DICOM standard's documentation as well as registry of data elements.

admin_link
Returns an HTML tag linking to this instance in the admin site.

Returns HTML link to this instance

Return type str

data_element_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.

description
A short description of this data element.

get_value_representation_display(*, *field=<django.db.models.fields.CharField: value_representation>*)

keyword
Most data elements have some keyword that facilitates querying header information.

objects = <django_dicom.models.managers.data_element_definition.DataElementDefinitionM

tag
Data element tags are an ordered pair of 16-bit unsigned integers representing the *Group Number* followed by *Element Number* and they are represented in the database using an array (list) of two four character strings.

to_dict() → dict
Returns a dictionary representation of this instance.

Returns This instance's dictionary representation

Return type dict

to_series() → pandas.core.series.Series
Returns a `Series` representation of this instance.

Returns Series representation of this instance

Return type `pandas.Series`

value_representation
The value representation (VR) defines the type of information stored in the data element. This will be used to determine which *DataElementValue* subclass is instantiated to save this information to the database.

django_dicom.models.dicom_entity module

Definition of the *DicomEntity* class. This abstract model serves as a base class for the various DICOM entities (e.g. *Study* and *Patient*).

class django_dicom.models.dicom_entity.**DicomEntity**(**args*, ***kwargs*)
Bases: django_extensions.db.models.TimeStampedModel

Abstract model providing common functionality for DICOM entities. For more information about DICOM entites, see this introduction.

FIELD_TO_HEADER = {}
A dictionary used to convert field names to header keywords.

class Meta
Bases: object

abstract = False

admin_link
Calls *get_admin_link()* to return a link to this instance's page in the admin site.

Returns HTML element

Return type str

get_admin_link(*text: str = None*) → str
Returns an HTML link to this instance's page in the admin site.

Parameters **text** (*str, optional*) – Text to display, by default None

Returns HTML element

Return type str

get_header_fields() → list
Returns a list of the derived model's fields which represent DICOM header information.

Returns Fields representing DICOM header information

Return type list

classmethod get_header_keyword(*field_name: str*) → str
Returns the data element keyword to return the requested field's value from header data. Relies on the derived model's *FIELD_TO_HEADER* class attribute. If no matching key is found, will simply return the field's name in CamelCase formatting (the formatting of pydicom's header keywords).

Returns pydicom header keyword

Return type str

is_header_field(*field: django.db.models.fields.Field*) → bool
Returns a boolean indicating whether the provided field represents DICOM header information or not.

Parameters **field** (Field) – Field in question

Returns Whether this field represent DICOM header information or not

Return type bool

log_creation() → None
Logs the creation of a new instance of the calling Entity.

objects

save(**args*, ***kwargs*)
Overrides save() to create logs and update the instance's fields from header information if provided.

update_fields_from_header(*header*, *exclude: list = None*) → None
Update fields from header data.

Parameters

- **header** (*dicom_parser.header.Header*) – DICOM header data
- **exclude** (*list, optional*) – Field names to exclude, default is []

django_dicom.models.header module

Definition of the *Header* class.

class django_dicom.models.header.**Header**(**args*, ***kwargs*)
Bases: model_utils.models.TimeStampedModel

A model representing a single DICOM Data Set.

admin_link
Creates an HTML tag to link to this instance within the admin site.

Returns Link to this instance in the admin site

Return type str

data_element_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.

get_entity_uid(*entity: django_dicom.models.dicom_entity.DicomEntity*) → str
Returns the UID of the provided DICOM entity from the header information.

Parameters **entity** (*DicomEntity*) – The entity for which a UID is desired

Returns The provided DICOM entity's unique identifier

Return type str

get_or_create_entity(*entity: django_dicom.models.dicom_entity.DicomEntity*) → tuple
Get or create a DICOM entity's instance from this header.

Parameters **entity** (*DicomEntity*) – The desired DICOM entity's model

Returns instance, creatd

Return type tuple

get_or_create_patient() → tuple
Get or create the *Patient* instance corresponding to this header.

Returns instance, created

Return type tuple

get_or_create_series() → tuple
Get or create the *Series* instance corresponding to this header.

Returns instance, created

Return type tuple

get_or_create_study() → tuple
Get or create the *Study* instance corresponding to this header.

Returns instance, created

Return type tuple

get_value_by_keyword(*keyword: str*) → Any
Returns a data element's value by keyword **from the database** (not from the DICOM header itself, ergo only elements that are saved to the database may be queried).

Parameters **keyword** (*str*) – Data element keyword

Returns Data element value

Return type Any

image
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.

index
This Data Set's index in the sequence (if *parent* is not None).

instance
Caches the created `dicom_parser.header.Header` instance to prevent multiple reades.

Returns Header information

Return type `dicom_parser.header.Header`

objects = <django_dicom.models.managers.header.HeaderManager object>

parent
Data Set <http://dicom.nema.org/medical/dicom/current/output/chtml/part05/chapter_7.html>'_s 'may be nested. If this header (Data Set) is an item of some sequence, this field holds that reference.

to_html(*verbose: bool = False*, ***kwargs*) → str
Returns an HTML representation of this instance.

Parameters **verbose** (*bool, optional*) – Whether to return a JSON with all of the header information or just a link to this instance's admin page, by default False.

Returns HTML representaion of this instance

Return type str

to_verbose_list() → List[dict]
Returns a list of dictionaries containing the information from the data elements in this header.

Returns Header information as a list of dictionaries

Return type List[dict]

django_dicom.models.image module

Definition of the *Image* class.

class django_dicom.models.image.**Image**(**args*, ***kwargs*)
Bases: *django_dicom.models.dicom_entity.DicomEntity*

A model to represent a single instance of the Image entity. This model is normally instantiated with the *dcm* field set to some *.dcm* file from which the header information is read.

FIELD_TO_HEADER = {'date': 'InstanceCreationDate', 'number': 'InstanceNumber', 'time
A dictionary of DICOM data element keywords to be used to populate a created instance's fields.

create_header_instance() → django_dicom.models.header.Header
Creates a *Header* instance from a dicom_parser.header.Header.

Returns Created instance

Return type *Header*

data
Facilitates access to the Image instance's data.

Returns The image's pixel data

Return type np.ndarray

date
Instance Creation Date value.

dcm
A reference to the DICOM image file.

default_path
Default unique path for this image based on its header information.

Returns Default image location

Return type pathlib.Path

get_absolute_url() → str
Returns the absolute URL for this instance. For more information see the Django documentation.

Returns This instance's absolute URL path

Return type str

get_default_path() → pathlib.Path
Returns a unique default path under MEDIA_ROOT for this instance based on its header information.

Returns This instance's default location

Return type pathlib.Path

header
The associated *Header* instance representing this image's header information.

instance
Caches the created dicom_parser.image.Image instance to prevent multiple reads.

Returns Image information

Return type dicom_parser.image.Image

logger = <Logger data.dicom.image (WARNING)>

number
Instance Number value.

objects = <django_dicom.models.managers.image.ImageManager object>

patient
Returns the *Patient* associated with this image.

Returns Associated *Patient* instance

Return type *Patient*

rename(*target: pathlib.Path*) → None
Move the *.dcm* file this instance references to some target destination.

Parameters **target** (pathlib.Path) – Destination path

save(**args*, *rename: bool = True*, ***kwargs*)
Overrides *save()* to check for missing header information or associated DICOM entities and create them if a *.dcm* file is provided.

Parameters **rename** (*bool, optional*) – Whether to move the file this instance is a reference to to a default path under MEDIA_ROOT or not, by default True

sequence_type
Returns the MRI sequence type detected by *dicom_parser*.

Returns Sequence type identifier

Return type str

series
The *Series* instance to which this image belongs.

time
Instance Creation Time value.

uid
SOP Instance UID value.

warnings
In case any warnings are raised by dicom_parser upon reading the image's header information, they are stored in this field.

django_dicom.models.patient module

Definition of the *Patient* class.

class django_dicom.models.patient.**Patient**(**args*, ***kwargs*)
Bases: *django_dicom.models.dicom_entity.DicomEntity*

A model to represent a single instance of the Patient entity.

FIELD_TO_HEADER = {'date_of_birth': 'PatientBirthDate', 'sex': 'PatientSex', 'uid':
A dictionary of DICOM data element keywords to be used to populate a created instance's fields.

date_of_birth
Patient Birth Date value.

family_name
Patient's Name value.

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

Returns This instance's absolute URL path

Return type str

get_full_name() → str
Returns the first and last names of the patient.

Returns Patient's first and last names.

Return type str

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

given_name
Patient's Name value.

logger = <Logger data.dicom.patient (WARNING)>

middle_name
Patient's Name value.

n_images
Returns the number of associated images.

See also:

- *query_n_images()*

Returns Number of associated images

Return type int

n_series
Returns the number of associated series.

Returns Number of associated series

Return type int

n_studies
Returns the number of associated studies.

See also:

- *query_n_studies()*

Returns Number of associated studies

Return type int

name_prefix
Patient's Name value.

name_suffix
Patient's Name value.

objects = <django.db.models.manager.DicomEntityManagerFromPatientQuerySet object>

query_n_images() → int
Returns the number of associated images.

See also:

n_images()

Returns Number of associated images

Return type int

query_n_studies() → int
Returns the number of associated studies.

See also:

n_studies()

Returns Number of associated studies

Return type int

query_research_subject()
Returns the associated research subject, if such a model is registered and a matching instance exists.

See also:

- *research_subject()*

Returns Associated research subject

Return type Subject

research_subject
Returns the associated research subject, if such a model is registered and a matching instance exists.

See also:

- *query_research_subject()*

Returns Associated research subject

Return type Subject

series_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.

sex
Patient's Sex value.

uid
Patient ID value.

update_fields_from_header(*header*, *exclude: list = None*) → None
Overrides `update_fields_from_header()` to handle setting the name parts.

Parameters

- **header** (`Header`) – DICOM header information.
- **exclude** (*list, optional*) – Field names to exclude (the default is [], which will not exclude any header fields).

update_patient_name(*header*) → None
Parses the patient's name from the DICOM header and updates the instance's fields.

Parameters **header** (Header) – A DICOM image's Header instance.

django_dicom.models.series module

Definition of the *Series* class.

class django_dicom.models.series.**Series**(**args*, ***kwargs*)
Bases: *django_dicom.models.dicom_entity.DicomEntity*

A model to represent a single instance of the Series entity.

FIELD_TO_HEADER = {'date': 'SeriesDate', 'description': 'SeriesDescription', 'mr_acq
A dictionary of DICOM data element keywords to be used to populate a created instance's fields.

MR_ACQUISITION_2D = '2D'
MR Acquisition Type value.

MR_ACQUISITION_3D = '3D'

MR_ACQUISITION_TYPE_CHOICES = (('2D', '2D'), ('3D', '3D'))

body_part_examined
Body Part Examined value.

data
Returns the dicom_parser.series.Series.data property's value.

Returns Series data

Return type np.ndarray

date
Series Date value.

datetime
Returns a datetime.datetime object by combining the values of the *date* and *time* fields.

Returns Series datetime

Return type datetime.datetime

description
Series Description value.

device_serial_number
Device Serial Number value.

echo_time
Echo Time value.

echo_train_length
Echo Train Length value.

flip_angle
Flip Angle value.

get_absolute_url() → str
Returns the absolute URL for this instance. For more information see the Django documentation.

Returns This instance's absolute URL path

Return type str

get_file_paths() → Tuple[pathlib.Path]

get_modality_display(*, *field=<django.db.models.fields.CharField: modality>*)

get_mr_acquisition_type_display(*, *field=<django.db.models.fields.CharField: mr_acquisition_type>*)

get_path() → pathlib.Path
Returns the base directory containing the images composing this series.

Returns This series's base directory path

Return type str

get_patient_position_display(*, *field=<django.db.models.fields.CharField: patient_position>*)

get_sample_header() → dicom_parser.header.Header
Return a sample `Header` instance for this series.

See also:

- *sample_header()*

Returns Sample image header information

Return type DicomHeader

get_scanning_sequence_display() → list
Returns the display valuse of this instance's *scanning_sequence* attribute.

Returns Verbose scanning sequence values

Return type list

get_sequence_type_display(*, *field=<django.db.models.fields.CharField: sequence_type>*)

get_sequence_variant_display() → list
Returns the display valuse of this instance's *sequence_variant* attribute.

Returns Verbose sequence variant values

Return type list

image_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.

instance
Caches the created `dicom_parser.series.Series` instance to prevent multiple reades.

Returns Series information

Return type `dicom_parser.series.Series`

institution_name
Institution Name value.

inversion_time
Inversion Time value.

logger = <Logger data.dicom.series (WARNING)>

magnetic_field_strength
Magnetic Field Strength value.

manufacturer
Manufacturer value.

manufacturer_model_name
Manufacturer's Model Name value.

missing_relation
Returns whether this instance misses an associated *Patient* or *Study*.

Returns Whether this instance has missing relationships

Return type bool

modality
Modality value.

mr_acquisition_type
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

number
Series Number value.

operators_name
Operator's Name value.

path
Returns the base path of this series' images.

Returns Series directory path

Return type pathlib.Path

patient
The *Patient* instance to which this series belongs.

patient_position
Patient Position value.

pixel_spacing
Pixel Spacing value.

protocol_name
Protocol Name value.

pulse_sequence_name
Pulse Sequence Name value.

repetition_time
Repetition Time value.

sample_header
Return a sample Header instance for this series.

See also:

- *get_sample_header()*

Returns Sample image header information

Return type DicomHeader

save(**args*, ***kwargs*) → None
Overrides *save()* to create any missing related DICOM entities if required.

scanning_sequence
Scanning Sequence value.

sequence_name
Sequence Name value.

sequence_type
Scanning sequence type identifier, as detected by *dicom_parser*.

sequence_variant
Sequence Variant value.

slice_thickness
Slice Thickness value.

spatial_resolution
Returns the 3D spatial resolution of the instance by combining the values of the *pixel_spacing* and *slice_thickness* fields.

Returns (x, y, z) resolution in millimeters

Return type tuple

study
The *Study* instance to which this series belongs.

time
Series Time value.

uid
Series Instance UID value.

update_sequence_type(*save: bool = True*)
Checks the sequence type identifier detected by *dicom_parser* and updates the serialized value if required.

Parameters **save** (*bool*) – Whether to save changes or not, default is True

django_dicom.models.study module

Definition of the *Study* class.

class django_dicom.models.study.**Study**(**args*, ***kwargs*)
Bases: *django_dicom.models.dicom_entity.DicomEntity*

A model to represent a single instance of the Study entity.

FIELD_TO_HEADER = {'date': 'StudyDate', 'description': 'StudyDescription', 'time':
A dictionary of DICOM data element keywords to be used to populate a created instance's fields.

date
Study Date value.

description
Study Description value.

get_absolute_url()
Returns the absolute URL for this instance. For more information see the Django documentation.

Returns This instance's absolute URL path

Return type str

logger = <Logger data.dicom.study (WARNING)>

n_images
Returns the number of associated images.

See also:

query_n_images()

Returns Number of associated images

Return type int

n_patients
Returns the number of associated patients.

See also:

query_n_patients()

Returns Number of associated patients

Return type int

n_series
Returns the number of associated series.

Returns Number of associated series

Return type int

objects = <django.db.models.manager.DicomEntityManagerFromStudyQuerySet object>

query_n_images() → int
Returns the number of associated images.

See also:

n_images()

Returns Number of associated images

Return type int

query_n_patients() → int
Returns the number of associated patients.

See also:

n_patients()

Returns Number of associated patients

Return type int

series_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.

time
Study Time value.

uid
Study Instance UID value.

4.2.4 Serializers

Module contents

Provides serializers for the REST API.

Submodules

django_dicom.serializers.image_serializer module

Definition of the *ImageSerializer* class.

class django_dicom.serializers.image_serializer.**ImageSerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)

Bases: `rest_framework.serializers.ModelSerializer`

A serializer class for the *Image* model.

class Meta
Bases: `object`

fields = ('id', 'series', 'number', 'date', 'time', 'uid')

model
alias of *django_dicom.models.image.Image*

django_dicom.serializers.patient_serializer module

Definition of the *PatientSerializer* class.

class django_dicom.serializers.patient_serializer.**PatientSerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)

Bases: `rest_framework.serializers.HyperlinkedModelSerializer`

A serializer class for the *Patient* model.

class Meta
Bases: object

fields = ('id', 'url', 'uid', 'name_prefix', 'given_name', 'middle_name', 'family_n

model
alias of *django_dicom.models.patient.Patient*

django_dicom.serializers.series_serializer module

Definition of the *SeriesSerializer* class.

class django_dicom.serializers.series_serializer.**SeriesSerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)

Bases: rest_framework.serializers.ModelSerializer

A serializer class for the *Series* model.

class Meta
Bases: object

fields = ('id', 'study', 'patient', 'body_part_examined', 'patient_position', 'numb

model
alias of *django_dicom.models.series.Series*

django_dicom.serializers.study_serializer module

Definition of the *StudySerializer* class.

class django_dicom.serializers.study_serializer.**StudySerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)

Bases: rest_framework.serializers.HyperlinkedModelSerializer

A serializer class for the *Study* model.

class Meta
Bases: object

fields = ('id', 'description', 'date', 'time', 'uid', 'url', 'n_patients', 'n_serie

model
alias of *django_dicom.models.study.Study*

4.2.5 Utils

Module contents

General utilities.

Submodules

django_dicom.utils.html module

Definition of the *Html* class.

`django_dicom.utils.html.`**ADMIN_VIEW_NAMES = {'DataElement': 'admin:django_dicom_dataelement**

Admin site views by model name, used to generate the appropriate URLs.

class `django_dicom.utils.html.`**Html**

Bases: `object`

Automates some HTML generation for the admin site.

**BREAK = '
'**

HORIZONTAL_LINE = '<hr style="background-color: {color};">'

classmethod admin_link(*model_name: str*, *pk: int*, *text: str = None*) → str

Returns a link to the admin site page of the provided model instance.

Parameters

- **model_name** (*str*) – Model to be linked
- **pk** (*int*) – Instance to be linked
- **text** (*str, optional*) – Text to display for the link, by default None (uses the instance's primary key)

Returns HTML link

Return type str

classmethod break_html(*pieces*) → str

Add break (
) tags between an iterable of HTML snippets.

Parameters **pieces** (*Iterable*) – HTML snippets

Returns Joined HTML

Return type str

classmethod horizontal_line(*color: str = 'black'*) → str

Returns an <hr> tag with the desired color.

Parameters **color** (*str, optional*) – Color to style the line with, by default "black"

Returns HTML horizontal line

Return type str

classmethod json(*value*) → str

Formats a JSON string for display in the browser.

Parameters **value** (*str*) – JSON string

Returns Formatted JSON HTML

Return type str

4.2.6 Views

Module contents

Definition of the app's `rest_framework.viewsets.ViewSet` subclasses.

Submodules

django_dicom.views.defaults module

Definition of the *DefaultsMixin* mixin.

class django_dicom.views.defaults.**DefaultsMixin**
Bases: object

Default settings for view authentication, permissions and filtering.

authentication_classes = (<class 'rest_framework.authentication.BasicAuthentication'>,

filter_backends = (<class 'django_filters.rest_framework.backends.DjangoFilterBackend'

permission_classes = (<class 'rest_framework.permissions.IsAuthenticated'>,)

django_dicom.views.image module

Definition of the *ImageViewSet* class.

class django_dicom.views.image.**ImageViewSet**(***kwargs*)
Bases: *django_dicom.views.defaults.DefaultsMixin*, rest_framework.viewsets.ModelViewSet

API endpoint that allows images to be viewed or edited.

filter_class
alias of *django_dicom.filters.image_filter.ImageFilter*

get_queryset()
Get the list of items for this view. This must be an iterable, and may be a queryset. Defaults to using *self.queryset*.

This method should always be used rather than accessing *self.queryset* directly, as *self.queryset* gets evaluated only once, and those results are cached for all subsequent requests.

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

(Eg. return a list of items that is specific to the user)

ordering_fields = ('series', 'number', 'date', 'time')

pagination_class
alias of *django_dicom.views.pagination.StandardResultsSetPagination*

parser_classes = (<class 'rest_framework.parsers.MultiPartParser'>,)

queryset

search_fields = ('number', 'date', 'time', 'uid')

serializer_class
alias of *django_dicom.serializers.image_serializer.ImageSerializer*

django_dicom.views.pagination module

Definition of the *StandardResultsSetPagination* class.

class django_dicom.views.pagination.**StandardResultsSetPagination**
Bases: rest_framework.pagination.PageNumberPagination

Default pagination parameters. This didn't work as part of the DefaultsMixin and therefore has to be defined separately in the 'pagination_class' configuration.

max_page_size = 10000

page_size = 25

page_size_query_param = 'page_size'

django_dicom.views.patient module

Definition of the *PatientViewSet* class.

class django_dicom.views.patient.**PatientViewSet**(***kwargs*)
Bases: *django_dicom.views.defaults.DefaultsMixin*, rest_framework.viewsets.ModelViewSet

API endpoint that allows patients 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

download_series_set(*request*, *uid*)

filter_class
alias of *django_dicom.filters.patient_filter.PatientFilter*

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

ordering_fields = ('id', 'uid', 'family_name', 'given_name', 'sex', 'date_of_birth', '

pagination_class
alias of *django_dicom.views.pagination.StandardResultsSetPagination*

queryset

serializer_class
alias of *django_dicom.serializers.patient_serializer.PatientSerializer*

django_dicom.views.series module

Definition of the *SeriesViewSet* class.

class django_dicom.views.series.**SeriesViewSet**(***kwargs*)

Bases: *django_dicom.views.defaults.DefaultsMixin*, rest_framework.viewsets.ModelViewSet

API endpoint that allows series to be viewed or edited.

filter_class

alias of *django_dicom.filters.series_filter.SeriesFilter*

get_csv(*request: rest_framework.request.Request*) → rest_framework.response.Response

get_manufacturers(*request*)

get_queryset()

Get the list of items for this view. This must be an iterable, and may be a queryset. Defaults to using *self.queryset*.

This method should always be used rather than accessing *self.queryset* directly, as *self.queryset* gets evaluated only once, and those results are cached for all subsequent requests.

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

(Eg. return a list of items that is specific to the user)

listed_zip(*request: rest_framework.request.Request*, *series_ids: str*) → django.http.response.FileResponse

ordering_fields = ('study', 'patient', 'number', 'date', 'time', 'scanning_sequence',

pagination_class

alias of *django_dicom.views.pagination.StandardResultsSetPagination*

queryset

search_fields = ('study', 'patient', 'body_part_examined', 'number', 'description', 'd

serializer_class

alias of *django_dicom.serializers.series_serializer.SeriesSerializer*

to_zip(*request: rest_framework.request.Request*, *pk: int*) → django.http.response.FileResponse

django_dicom.views.study module

Definition of the *StudyViewSet* class.

class django_dicom.views.study.**StudyViewSet**(***kwargs*)

Bases: *django_dicom.views.defaults.DefaultsMixin*, rest_framework.viewsets.ModelViewSet

API endpoint that allows studies 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

filter_class

alias of *django_dicom.filters.study_filter.StudyFilter*

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

Returns Study queryset

Return type QuerySet

pagination_class
alias of *django_dicom.views.pagination.StandardResultsSetPagination*

queryset

serializer_class
alias of *django_dicom.serializers.study_serializer.StudySerializer*

4.3 Submodules

4.4 django_dicom.urls module

Definition of the app's URLs.

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

CHAPTER 5

Resources

5.1 General

- Wikipedia.
- The official DICOM standard website.
- NiBabel's introduction to DICOM.
- DICOM is Easy - wonderful blog dedicated to the DICOM file format and software programming for medical applications.
- DICOM Library - free online medical images, signals or video files sharing and anonymizing service for educational and scientific purposes. Includes a good introduction to the DICOM standard as well as a handy table of official DICOM tags.
- DICOM Standard Browser - an online DICOM encyclopedia.

5.2 Other

- *The first step for neuroimaging data analysis: DICOM to NIfTI conversion*[1] - contains valuable information about the way different vendors store data in the DICOM format (PDF).
- Siemens mosaic format - Excellent NiBabel article.
- Information on GE private tags can be found in the Discovery VCT DICOM Conformance Statement as well as in the GE MR DICOM Conformance Statement.
- NA-MIC Wiki page on DICOM for DWI and DTI - vendor-specific information about DICOM and DWI/DTI data encoding.

[1] Li, X., Morgan, P. S., Ashburner, J., Smith, J., & Rorden, C. (2016). The first step for neuroimaging data analysis: DICOM to NIfTI conversion. Journal of neuroscience methods, 264, 47-56.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index

E

F

G

H

N

O

P

Q

R

S

T

U

V

W