
django**tutor**

Выпуск 0.0.1

Arsen Sardaryan

мар. 19, 2025

Contents:

1	django tutor	3
	Содержание модулей Python	11
	Алфавитный указатель	13

Add your content using `reStructuredText` syntax. See the [reStructuredText](#) documentation for details.


```
class polls.models.Choice(id, question, choice_text, votes)
```

Базовые классы: `Model`

Параметры

- `id` (*`BigAutoField`*) – Primary key: ID
- `choice_text` (*`CharField`*) – Choice text
- `votes` (*`IntegerField`*) – Votes

Relationship fields:

Параметры

`question` (*`ForeignKey`* to *`Question`*) – Question (related name: choice)

```
exception DoesNotExist
```

Базовые классы: `ObjectDoesNotExist`

```
exception MultipleObjectsReturned
```

Базовые классы: `MultipleObjectsReturned`

```
_meta = <Options for Choice>
```

```
choice_text
```

Type: `CharField`

Choice text

```
id
```

Type: `BigAutoField`

Primary key: ID

```
objects = <django.db.models.Manager object>
```

question

Type: `ForeignKey` to *Question*

Question (related name: choice)

question_id

Internal field, use *question* instead.

votes

Type: `IntegerField`

Votes

```
class polls.models.Monitoring(monitoring_id, cpu_usage, memory_usage, network_traffic,
                             monitoring_time, virtual_machines)
```

Базовые классы: `Model`

Параметры

- *monitoring_id* (`UUIDField`) – Primary key: Monitoring id
- *cpu_usage* (`FloatField`) – Cpu usage
- *memory_usage* (`FloatField`) – Memory usage
- *network_traffic* (`TimeField`) – Network traffic
- *monitoring_time* (`DateTimeField`) – Monitoring time

Relationship fields:

Параметры

virtual_machines (`ForeignKey` to *Virtuals*) – Virtual machines (related name: monitoring)

exception `DoesNotExist`

Базовые классы: `ObjectDoesNotExist`

exception `MultipleObjectsReturned`

Базовые классы: `MultipleObjectsReturned`

`_meta` = <Options for Monitoring>

cpu_usage

Type: `FloatField`

Cpu usage

```
get_next_by_monitoring_time(*, field=<django.db.models.DateTimeField: monitoring_time>,
                             is_next=True, **kwargs)
```

Finds next instance based on *monitoring_time*. See `get_next_by_F00()` for more information.

```
get_previous_by_monitoring_time(*, field=<django.db.models.DateTimeField:
                                     monitoring_time>, is_next=False, **kwargs)
```

Finds previous instance based on *monitoring_time*. See `get_previous_by_F00()` for more information.

memory_usage

Type: `FloatField`

Memory usage


```

monitoring_id
    Type: UUIDField
    Primary key: Monitoring id
monitoring_time
    Type: DateTimeField
    Monitoring time
network_traffic
    Type: TimeField
    Network traffic
objects = <django.db.models.Manager object>
virtual_machines
    Type: ForeignKey to Virtuals
    Virtual machines (related name: monitoring)
virtual_machines_id
    Internal field, use virtual_machines instead.
class polls.models.Product(id, name, price)
    Базовые классы: Model

    Параметры
    • id (BigAutoField) – Primary key: ID
    • name (CharField) – Name
    • price (DecimalField) – Price
exception DoesNotExist
    Базовые классы: ObjectDoesNotExist
exception MultipleObjectsReturned
    Базовые классы: MultipleObjectsReturned
__init__()
_meta = <Options for Product>

id
    Type: BigAutoField
    Primary key: ID
name
    Type: CharField
    Name
objects = <django.db.models.Manager object>
price
    Type: DecimalField
    Price

```

```

    qux
        Doc comment for instance attribute qux.

    spam
        Docstring for instance attribute spam.
class polls.models.Protocol(*values)
    Базовые классы: TextChoices

    RDP = 'rdp'

    SSH = 'ssh'

    VNC = 'vnc'

    static _generate_next_value_(name, start, count, last_values)
        Generate the next value when not given.

        name: the name of the member start: the initial start value or None count: the number of existing
        members last_values: the list of values assigned
class polls.models.Question(id, question_text, pub_date)
    Базовые классы: Model

    Параметры

        • id (BigAutoField) – Primary key: ID

        • question_text (CharField) – Question text

        • pub_date (DateTimeField) – Date published

    Reverse relationships:

    Параметры
        choice (Reverse ForeignKey from Choice) – All choices of this question (related name
        of question)

exception DoesNotExist
    Базовые классы: ObjectDoesNotExist

exception MultipleObjectsReturned
    Базовые классы: MultipleObjectsReturned

_meta = <Options for Question>

choice_set
    Type: Reverse ForeignKey from Choice

    All choices of this question (related name of question)

get_next_by_pub_date(*, field=<django.db.models.DateTimeField: pub_date>, is_next=True,
                    **kwargs)

    Finds next instance based on pub_date. See get_next_by_FOO() for more information.

get_previous_by_pub_date(*, field=<django.db.models.DateTimeField: pub_date>,
                        is_next=False, **kwargs)

    Finds previous instance based on pub_date. See get_previous_by_FOO() for more information.

```

```

id
    Type: BigAutoField
    Primary key: ID
objects = <django.db.models.Manager object>
pub_date
    Type: DateTimeField
    Date published
question_text
    Type: CharField
    Question text
was_published_recently()
class polls.models.Virtuals(id, hostname, protocol, user_id, address, port, user_vm, password_vm,
                             ignore_cert)

```

Базовые классы: `Model`

Параметры

- `id` (`UUIDField`) – Primary key: Id
- `hostname` (`CharField`) – Hostname
- `protocol` (`CharField`) – Protocol
- `address` (`CharField`) – Address
- `port` (`IntegerField`) – Port
- `user_vm` (`CharField`) – User vm
- `password_vm` (`CharField`) – Password vm
- `ignore_cert` (`BooleanField`) – Ignore cert

Relationship fields:

Параметры

`user_id` (`ForeignKey` to `User`) – User id (related name: `virtuals`)

Reverse relationships:

Параметры

`monitoring` (Reverse `ForeignKey` from `Monitoring`) – All monitorings of this virtuals (related name of `virtual_machines`)

exception `DoesNotExist`

Базовые классы: `ObjectDoesNotExist`

exception `MultipleObjectsReturned`

Базовые классы: `MultipleObjectsReturned`

`_meta` = <Options for Virtuals>

`address`

Type: `CharField`

Address

```

get_protocol_display(*, field=<django.db.models.CharField: protocol>)
    Shows the label of the protocol. See get_FOO_display() for more information.

hostname
    Type: CharField
    Hostname

id
    Type: UUIDField
    Primary key: Id

ignore_cert
    Type: BooleanField
    Ignore cert

monitoring_set
    Type: Reverse ForeignKey from Monitoring
    All monitorings of this virtuals (related name of virtual_machines)

objects = <django.db.models.Manager object>

password_vm
    Type: CharField
    Password vm

port
    Type: IntegerField
    Port

protocol
    Type: CharField
    Protocol
    Choices:
        • rdp
        • ssh
        • vnc

user_id
    Type: ForeignKey to User
    User id (related name: virtuals)

user_id_id
    Internal field, use user_id instead.

user_vm
    Type: CharField
    User vm

class polls.views.MonitoringDetailView(**kwargs)
    Базовые классы: APIView

```

```
    delete(request, monitoring_id)

    get(request, monitoring_id)

class polls.views.MonitoringListView(**kwargs)
    Базовые классы: APIView

    get(request)

    post(request)

class polls.views.VirtualsDetailView(**kwargs)
    Базовые классы: APIView

    delete(request, vm_id)

    get(request, vm_id)

class polls.views.VirtualsListView(**kwargs)
    Базовые классы: APIView

    get(request)

    post(request)
```


p

`polls.models`, 3

`polls.views`, 8

СИМВОЛЫ

`__init__()` (метод `polls.models.Product`), 5
`_generate_next_value_()` (статический метод `polls.models.Protocol`), 6
`_meta` (ампубум `polls.models.Choice`), 3
`_meta` (ампубум `polls.models.Monitoring`), 4
`_meta` (ампубум `polls.models.Product`), 5
`_meta` (ампубум `polls.models.Question`), 6
`_meta` (ампубум `polls.models.Virtuals`), 7

A

`address` (ампубум `polls.models.Virtuals`), 7

C

`Choice` (класс в `polls.models`), 3
`Choice.DoesNotExist`, 3
`Choice.MultipleObjectsReturned`, 3
`choice_set` (ампубум `polls.models.Question`), 6
`choice_text` (ампубум `polls.models.Choice`), 3
`cpu_usage` (ампубум `polls.models.Monitoring`), 4

D

`delete()` (метод `polls.views.MonitoringDetailView`), 8
`delete()` (метод `polls.views.VirtualsDetailView`), 9

G

`get()` (метод `polls.views.MonitoringDetailView`), 9
`get()` (метод `polls.views.MonitoringListView`), 9
`get()` (метод `polls.views.VirtualsDetailView`), 9
`get()` (метод `polls.views.VirtualsListView`), 9
`get_next_by_monitoring_time()` (метод `polls.models.Monitoring`), 4
`get_next_by_pub_date()` (метод `polls.models.Question`), 6
`get_previous_by_monitoring_time()` (метод `polls.models.Monitoring`), 4
`get_previous_by_pub_date()` (метод `polls.models.Question`), 6

`get_protocol_display()` (метод `polls.models.Virtuals`), 7

H

`hostname` (ампубум `polls.models.Virtuals`), 8

I

`id` (ампубум `polls.models.Choice`), 3
`id` (ампубум `polls.models.Product`), 5
`id` (ампубум `polls.models.Question`), 6
`id` (ампубум `polls.models.Virtuals`), 8
`ignore_cert` (ампубум `polls.models.Virtuals`), 8

M

`memory_usage` (ампубум `polls.models.Monitoring`), 4
`module`
 `polls.models`, 3
 `polls.views`, 8
`Monitoring` (класс в `polls.models`), 4
`Monitoring.DoesNotExist`, 4
`Monitoring.MultipleObjectsReturned`, 4
`monitoring_id` (ампубум `polls.models.Monitoring`), 4
`monitoring_set` (ампубум `polls.models.Virtuals`), 8
`monitoring_time` (ампубум `polls.models.Monitoring`), 5
`MonitoringDetailView` (класс в `polls.views`), 8
`MonitoringListView` (класс в `polls.views`), 9

N

`name` (ампубум `polls.models.Product`), 5
`network_traffic` (ампубум `polls.models.Monitoring`), 5

O

`objects` (ампубум `polls.models.Choice`), 3
`objects` (ампубум `polls.models.Monitoring`), 5
`objects` (ампубум `polls.models.Product`), 5

objects (*ампурум polls.models.Question*), 7
 objects (*ампурум polls.models.Virtuals*), 8

P

password_vm (*ампурум polls.models.Virtuals*), 8
 polls.models
 module, 3
 polls.views
 module, 8
 port (*ампурум polls.models.Virtuals*), 8
 post() (*метод polls.views.MonitoringListView*), 9
 post() (*метод polls.views.VirtualsListView*), 9
 price (*ампурум polls.models.Product*), 5
 Product (*класс в polls.models*), 5
 Product.DoesNotExist, 5
 Product.MultipleObjectsReturned, 5
 protocol (*ампурум polls.models.Virtuals*), 8
 Protocol (*класс в polls.models*), 6
 pub_date (*ампурум polls.models.Question*), 7

Q

question (*ампурум polls.models.Choice*), 3
 Question (*класс в polls.models*), 6
 Question.DoesNotExist, 6
 Question.MultipleObjectsReturned, 6
 question_id (*ампурум polls.models.Choice*), 4
 question_text (*ампурум polls.models.Question*), 7
 qux (*ампурум polls.models.Product*), 5

R

RDP (*ампурум polls.models.Protocol*), 6

S

spam (*ампурум polls.models.Product*), 6
 SSH (*ампурум polls.models.Protocol*), 6

U

user_id (*ампурум polls.models.Virtuals*), 8
 user_id_id (*ампурум polls.models.Virtuals*), 8
 user_vm (*ампурум polls.models.Virtuals*), 8

V

virtual_machines (*ампурум polls.models.Monitoring*), 5
 virtual_machines_id (*ампурум polls.models.Monitoring*), 5
 Virtuals (*класс в polls.models*), 7
 Virtuals.DoesNotExist, 7
 Virtuals.MultipleObjectsReturned, 7
 VirtualsDetailView (*класс в polls.views*), 9
 VirtualsListView (*класс в polls.views*), 9
 VNC (*ампурум polls.models.Protocol*), 6
 votes (*ампурум polls.models.Choice*), 4

W

was_published_recently() (*метод polls.models.Question*), 7