djangotutorial

Выпуск 1.0

alekseyf

мар. 28, 2025

Contents:

Add your content using `reStructuredText` syntax. See the reStructuredText documentation for details.

`class polls.models.User`(*id*, *name*, *email*, *phone*, *date_registered*)

Базовые классы: `Model`

Параметры

- `id` (*BigAutoField*) – Primary key: ID
- `name` (*CharField*) – Name
- `email` (*EmailField*) – Email
- `phone` (*CharField*) – Phone
- `date_registered` (*DateTimeField*) – Date registered

Reverse relationships:

Параметры

`order` (Reverse `ForeignKey` from *Order*) – All orders of this user (related name of *user*)

`name`

Type: `CharField`

Name

A wrapper for a deferred-loading field. When the value is read from this

`email`

Type: `EmailField`

Email

A wrapper for a deferred-loading field. When the value is read from this

`phone`

Type: `CharField`

Phone

A wrapper for a deferred-loading field. When the value is read from this

`date_registered`

Type: `DateTimeField`

Date registered

A wrapper for a deferred-loading field. When the value is read from this

`exception DoesNotExist`

Базовые классы: `ObjectDoesNotExist`

`exception MultipleObjectsReturned`

Базовые классы: `MultipleObjectsReturned`

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

Finds next instance based on *date_registered*. See `get_next_by_FOO()` for more information.

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

Finds previous instance based on *date_registered*. See `get_previous_by_FOO()` for more information.

`id`

Type: `BigAutoField`

Primary key: ID

A wrapper for a deferred-loading field. When the value is read from this

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

`order_set`

Type: Reverse `ForeignKey` from *`Order`*

All orders of this user (related name of *`user`*)

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 polls.models.Product(`*id*, *name*, *category*, *price*, *stock_ quantity*, *description*`)`

Базовые классы: `Model`

Параметры

- `id` (*`BigAutoField`*) – Primary key: ID
- `name` (*`CharField`*) – Name
- `category` (*`CharField`*) – Category
- `price` (*`DecimalField`*) – Price
- `stock_quantity` (*`IntegerField`*) – Stock quantity
- `description` (*`TextField`*) – Description

Reverse relationships:

Параметры

`orderitem` (Reverse `ForeignKey` from *`OrderItem`*) – All order items of this product (related name of *`product`*)

`name`

Type: `CharField`

Name

A wrapper for a deferred-loading field. When the value is read from this

`category`

Type: `CharField`

Category

A wrapper for a deferred-loading field. When the value is read from this `price`

Type: `DecimalField`

Price

A wrapper for a deferred-loading field. When the value is read from this

`stock_quantity`

Type: `IntegerField`

Stock quantity

A wrapper for a deferred-loading field. When the value is read from this

`description`

Type: `TextField`

Description

A wrapper for a deferred-loading field. When the value is read from this

`exception DoesNotExist`

Базовые классы: `ObjectDoesNotExist`

`exception MultipleObjectsReturned`

Базовые классы: `MultipleObjectsReturned`

`id`

Type: `BigAutoField`

Primary key: ID

A wrapper for a deferred-loading field. When the value is read from this

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

`orderitem_set`

Type: Reverse `ForeignKey` from *`OrderItem`*

All order items of this product (related name of *`product`*)

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 polls.models.Order(`*id*, *user*, *date_ordered*, *status*`)`

Базовые классы: `Model`

Параметры

- `id` (*`BigAutoField`*) – Primary key: ID
- `date_ordered` (*`DateTimeField`*) – Date ordered
- `status` (*`CharField`*) – Status

Relationship fields:

Параметры
user (ForeignKey to *User*) – User (related name: order)

Reverse relationships:

Параметры
items (Reverse ForeignKey from *OrderItem*) – All items of this order (related name of *order*)

STATUS_CHOICES = [('new', 'New'), ('processing', 'Processing'), ('shipped', 'Shipped'), ('completed', 'Completed')]

user

Type: ForeignKey to *User*

User (related name: order)

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

In the example:

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

date_ordered

Type: DateTimeField

Date ordered

A wrapper for a deferred-loading field. When the value is read from this

status

Type: CharField

Status

Choices:

- new
- processing
- shipped
- completed

A wrapper for a deferred-loading field. When the value is read from this

exception DoesNotExist

Базовые классы: ObjectDoesNotExist

exception MultipleObjectsReturned

Базовые классы: MultipleObjectsReturned

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

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

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

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

`get_status_display(*, field=<django.db.models.CharField: status>)`

Shows the label of the `status`. See `get_FOO_display()` for more information.

`id`

Type: `BigAutoField`

Primary key: ID

A wrapper for a deferred-loading field. When the value is read from this

`items`

Type: Reverse `ForeignKey` from `OrderItem`

All items of this order (related name of `order`)

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

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

`user_id`

Internal field, use `user` instead.

`class polls.models.OrderItem(id, order, product, quantity)`

Базовые классы: `Model`

Параметры

- `id` (`BigAutoField`) – Primary key: ID
- `quantity` (`PositiveIntegerField`) – Quantity

Relationship fields:

Параметры

- `order` (`ForeignKey` to `Order`) – Order (related name: `items`)
- `product` (`ForeignKey` to `Product`) – Product (related name: `orderitem`)

`order`

Type: `ForeignKey` to `Order`

Order (related name: `items`)

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

In the example:

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

`product`

Type: `ForeignKey` to *`Product`*

Product (related name: `orderitem`)

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

In the example:

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

`quantity`

Type: `PositiveIntegerField`

Quantity

A wrapper for a deferred-loading field. When the value is read from this

`exception DoesNotExist`

Базовые классы: `ObjectDoesNotExist`

`exception MultipleObjectsReturned`

Базовые классы: `MultipleObjectsReturned`

`id`

Type: `BigAutoField`

Primary key: ID

A wrapper for a deferred-loading field. When the value is read from this

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

`order_id`

Internal field, use *`order`* instead.

`product_id`

Internal field, use *`product`* instead.

`class polls.views.UserViewSet(`*`**kwargs`*`)`

Базовые классы: `ModelViewSet`

`queryset = QuerySet`

`serializer_class`

псевдоним для `UserSerializer`

`class polls.views.ProductViewSet(`*`**kwargs`*`)`

Базовые классы: `ModelViewSet`

`queryset = QuerySet`

`serializer_class`

псевдоним для `ProductSerializer`

`class polls.views.OrderViewSet(`*`**kwargs`*`)`

Базовые классы: `ModelViewSet`

`queryset = QuerySet`

`serializer_class`

псевдоним для `OrderSerializer`

`class polls.views.OrderItemViewSet(`***kwargs*`)`

Базовые классы: `ModelViewSet`

`queryset = QuerySet`

`serializer_class`

псевдоним для `OrderItemSerializer`

`polls.views.api_home(`*request*, **args*, ***kwargs*`)`

Возвращает список всех доступных маршрутов API во всем проекте.

`polls.views.collect_routes(`*patterns*, *request*, *prefix=''*`)`

Рекурсивно собирает все маршруты API во всем Django-проекте и корректно формирует URL.

`polls.views.get_base_url(`*request=None*`)`

Возвращает базовый URL в зависимости от окружения

Содержание модулей Python

Алфавитный указатель