Resource Sharing

Release Beta 1.0

Radhesh Sarma, Simran Sahni, Nikhil Krishna, Abhirath Singh

Apr 10, 2020

CONTENTS:

1 All about us 1

2 Models 2

3 Views 5

4 Forms 6

5 Serializers 8

Python Module Index 9

Index 10

CHAPTER ONE

ALL ABOUT US

We are Django developers from BITS Pilani Hyderabad Campus

CHAPTER TWO

MODELS

class blog.models.**Task**(*id*, *author*, *content*, *date_created*)

exception DoesNotExist

exception MultipleObjectsReturned

author

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

In the example:

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

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

author_id

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

content

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

date_created

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

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

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

id

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

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

class users.models.**CustomUser**(*id*, *password*, *last_login*, *is_superuser*, *username*, *first_name*, *last_name*, *is_staff*, *is_active*, *date_joined*, *mobile_number*, *address*, *email*)

exception DoesNotExist

exception MultipleObjectsReturned

address

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

emailaddress_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_next_by_date_joined(*, *field=<django.db.models.fields.DateTimeField: date_joined>*, *is_next=True*, ***kwargs*)

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

groups

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

In the example:

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

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

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

id

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

logentry_set

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

In the example:

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

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

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

mobile_number

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

task_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.

user_permissions

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

In the example:

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

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

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

class users.models.**userdata**(*id*, *name*, *email*, *content*)

exception DoesNotExist

exception MultipleObjectsReturned

content

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

email

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

id

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

name

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

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

CHAPTER

THREE

VIEWS

class users.views.**UserDetail**(***kwargs*)

queryset = <QuerySet [<CustomUser: radheshsarma2299@outlook.com>]>

serializer_class
alias of *users.serializers.UserSerializer*

class users.views.**UserList**(***kwargs*)

queryset = <QuerySet [<CustomUser: radheshsarma2299@outlook.com>]>

serializer_class
alias of *users.serializers.UserSerializer*

class blog.views.**BlogDetail**(***kwargs*)

queryset = <QuerySet [<Task: Demo 1>]>

serializer_class
alias of *blog.serializers.BlogSerializer*

class blog.views.**BlogList**(***kwargs*)

queryset = <QuerySet [<Task: Demo 1>]>

serializer_class
alias of *blog.serializers.BlogSerializer*

blog.views.**home**(*request*)

blog.views.**remove**(*request, item_id*)

blog.views.**task_list**(*request*)

class pages.views.**AboutPageView**(***kwargs*)

template_name = 'pages/about.html'

class pages.views.**HomePageView**(***kwargs*)

template_name = 'pages/home.html'

CHAPTER

FOUR

FORMS

class users.forms.**CustomUserChangeForm**(**args*, ***kwargs*)

class Meta

fields = ('email', 'username', 'mobile_number', 'address')

model

alias of `users.models.CustomUser`

base_fields = {'address': <django.forms.fields.CharField object>, 'captcha': <captcha

declared_fields = {'captcha': <captcha.fields.ReCaptchaField object>, 'password': <d

property media

class users.forms.**CustomUserCreationForm**(**args*, ***kwargs*)

class Meta

fields = ('email', 'username', 'mobile_number', 'address')

model

alias of `users.models.CustomUser`

base_fields = {'address': <django.forms.fields.CharField object>, 'captcha': <captcha

declared_fields = {'captcha': <captcha.fields.ReCaptchaField object>, 'password1': <

property media

class users.forms.**PostForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)

class Meta

fields = ('name', 'email', 'content')

model

alias of `users.models.userdata`

base_fields = {'content': <django.forms.fields.CharField object>, 'email': <django.f

declared_fields = {}

property media

class blog.forms.**PostForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)

class Meta

fields = ('content',)

model

alias of `blog.models.Task`

base_fields = {'captcha': <captcha.fields.ReCaptchaField object>, 'content': <django

declared_fields = {'captcha': <captcha.fields.ReCaptchaField object>}

property media

CHAPTER

FIVE

SERIALIZERS

class blog.serializers.**BlogSerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)

class Meta

fields = 'content'

model
alias of *blog.models.Task*

class users.serializers.**UserSerializer**(*instance=None*, *data=<class 'rest_framework.fields.empty'>*, ***kwargs*)

class Meta

fields = ('mobile_number', 'address', 'email')

model
alias of *users.models.CustomUser*

PYTHON MODULE INDEX

b

blog.forms, 7
blog.models, 2
blog.serializers, 8
blog.views, 5

p

pages.views, 5

u

users.forms, 6
users.models, 2
users.serializers, 8
users.views, 5

INDEX

A

AboutPageView (*class in pages.views*), 5
address (*users.models.CustomUser attribute*), 3
author (*blog.models.Task attribute*), 2
author_id (*blog.models.Task attribute*), 2

B

base_fields (*blog.forms.PostForm attribute*), 7
base_fields (*users.forms.CustomUserChangeForm attribute*), 6
base_fields (*users.forms.CustomUserCreationForm attribute*), 6
base_fields (*users.forms.PostForm attribute*), 6
blog.forms
    module, 7
blog.models
    module, 2
blog.serializers
    module, 8
blog.views
    module, 5
BlogDetail (*class in blog.views*), 5
BlogList (*class in blog.views*), 5
BlogSerializer (*class in blog.serializers*), 8
BlogSerializer.Meta (*class in blog.serializers*), 8

C

content (*blog.models.Task attribute*), 2
content (*users.models.userdata attribute*), 4
CustomUser (*class in users.models*), 2
CustomUser.DoesNotExist, 2
CustomUser.MultipleObjectsReturned, 2
CustomUserChangeForm (*class in users.forms*), 6
CustomUserChangeForm.Meta (*class in users.forms*), 6
CustomUserCreationForm (*class in users.forms*), 6
CustomUserCreationForm.Meta (*class in users.forms*), 6

D

date_created (*blog.models.Task attribute*), 2
declared_fields (*blog.forms.PostForm attribute*), 7
declared_fields (*users.forms.CustomUserChangeForm attribute*), 6
declared_fields (*users.forms.CustomUserCreationForm attribute*), 6
declared_fields (*users.forms.PostForm attribute*), 6

E

email (*users.models.userdata attribute*), 4
emailaddress_set (*users.models.CustomUser attribute*), 3

F

fields (*blog.forms.PostForm.Meta attribute*), 7
fields (*blog.serializers.BlogSerializer.Meta attribute*), 8
fields (*users.forms.CustomUserChangeForm.Meta attribute*), 6
fields (*users.forms.CustomUserCreationForm.Meta attribute*), 6
fields (*users.forms.PostForm.Meta attribute*), 6
fields (*users.serializers.UserSerializer.Meta attribute*), 8

G

get_next_by_date_created() (*blog.models.Task method*), 2
get_next_by_date_joined() (*users.models.CustomUser method*), 3
get_previous_by_date_created() (*blog.models.Task method*), 2
get_previous_by_date_joined() (*users.models.CustomUser method*), 3
groups (*users.models.CustomUser attribute*), 3

H

home() (*in module blog.views*), 5
HomePageView (*class in pages.views*), 5

I

id (*blog.models.Task attribute*), 2
id (*users.models.CustomUser attribute*), 3

id (*users.models.userdata attribute*), 4

L

logentry_set (*users.models.CustomUser attribute*), 3

M

media() (*blog.forms.PostForm property*), 7
media() (*users.forms.CustomUserChangeForm property*), 6
media() (*users.forms.CustomUserCreationForm property*), 6
media() (*users.forms.PostForm property*), 6
mobile_number (*users.models.CustomUser attribute*), 3
model (*blog.forms.PostForm.Meta attribute*), 7
model (*blog.serializers.BlogSerializer.Meta attribute*), 8
model (*users.forms.CustomUserChangeForm.Meta attribute*), 6
model (*users.forms.CustomUserCreationForm.Meta attribute*), 6
model (*users.forms.PostForm.Meta attribute*), 6
model (*users.serializers.UserSerializer.Meta attribute*), 8
module
    blog.forms, 7
    blog.models, 2
    blog.serializers, 8
    blog.views, 5
    pages.views, 5
    users.forms, 6
    users.models, 2
    users.serializers, 8
    users.views, 5

N

name (*users.models.userdata attribute*), 4

O

objects (*blog.models.Task attribute*), 2
objects (*users.models.userdata attribute*), 4

P

pages.views
    module, 5
PostForm (*class in blog.forms*), 7
PostForm (*class in users.forms*), 6
PostForm.Meta (*class in blog.forms*), 7
PostForm.Meta (*class in users.forms*), 6

Q

queryset (*blog.views.BlogDetail attribute*), 5
queryset (*blog.views.BlogList attribute*), 5
queryset (*users.views.UserDetail attribute*), 5
queryset (*users.views.UserList attribute*), 5

R

remove() (*in module blog.views*), 5

S

serializer_class (*blog.views.BlogDetail attribute*), 5
serializer_class (*blog.views.BlogList attribute*), 5
serializer_class (*users.views.UserDetail attribute*), 5
serializer_class (*users.views.UserList attribute*), 5

T

Task (*class in blog.models*), 2
Task.DoesNotExist, 2
Task.MultipleObjectsReturned, 2
task_list() (*in module blog.views*), 5
task_set (*users.models.CustomUser attribute*), 3
template_name (*pages.views.AboutPageView attribute*), 5
template_name (*pages.views.HomePageView attribute*), 5

U

user_permissions (*users.models.CustomUser attribute*), 4
userdata (*class in users.models*), 4
userdata.DoesNotExist, 4
userdata.MultipleObjectsReturned, 4
UserDetail (*class in users.views*), 5
UserList (*class in users.views*), 5
users.forms
    module, 6
users.models
    module, 2
users.serializers
    module, 8
users.views
    module, 5
UserSerializer (*class in users.serializers*), 8
UserSerializer.Meta (*class in users.serializers*), 8