The Admin Site and the Model API

Reindert-Jan Ekker

@rjekker http://nl.linkedin.com/in/rjekker

Overview

Django Admin Site

- Auto-generated UI to edit your data
- Registering Models with Admin Site
- Creating a super user
- Very customizable

Model API

- Work with data from Python
- Query data
- Save/update/delete
- Relations (foreign keys)

Demo

The Django Admin Interface

- Create a superuser
- Logging in
- Registering our Models
- Entering data
- Displaying Models

Demo

Customizing the Admin Site

- Adding a dropdown for Game status
- Creating a ModelAdmin class
- Customizing the list view
- Inline editing in the list view

Registering Models with the Admin Site

```
# in admin.py
from django.contrib import admin
from .models import Game
admin.site.register(Game)

# Don't forget to create a superuser (in the terminal)
python manage.py createsuperuser
```

Adding __str__ to Models

```
from __future__ import unicode_literals
from django.utils.encoding import python_2_unicode_compatible
# other imports


@python_2_unicode_compatible
class Game(models.Model):
    def __str__(self):
        # return a string representation
```

```python
@admin.register(Game)
class GameAdmin(admin.ModelAdmin):
    list_display = (...)
    # and many other options
```

Admin Site Customization

The admin is extremely customizable

Documentation: https://goo.gl/TacNMS

Demo

The Model API

```
Game.objects
```

```
Game.objects.get(pk=5)
```

```
Game.objects.all()
```

```
Game.objects.filter(
    status='A')
```

- **Model classes have a Manager**
  Class attribute `objects`
  `Game.objects` not `g.objects`
- `get()` **returns a single instance**
  Throws exception when no matches
  Or more than 1 match
- `all()` **returns all rows**
- `filter()` **returns matching objects**
  exclude() does the opposite

Docs: https://goo.gl/OD7B06

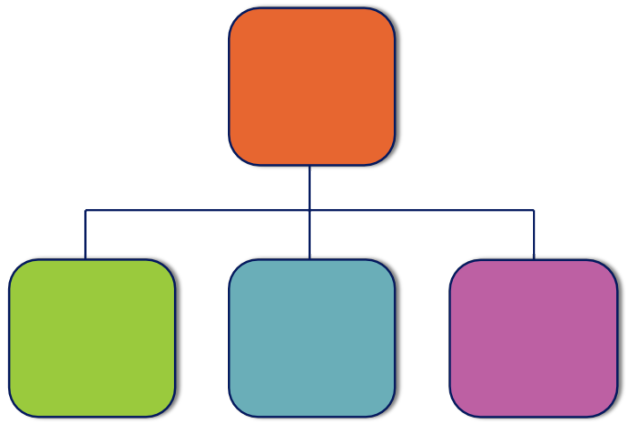
Save and Delete

```
# Create a new instance with keyword arguments
m = Move(x=1, y=2, game=g)

# save() does SQL INSERT for a new object and sets id field
# or UPDATE for existing object with changes
m.save()

# delete() removes an object from the database
g = Game.objects.get(pk=1); g.delete()
```

One-to-Many Relations



Defined by a ForeignKey field

- On the “many” side of the relation

“One” side gets a xxx_set attribute

- xxx is name of related model
- This is a “related manager” object
- Works just like “objects” manager

Set relation from Move m to Game g:

- m.game=g, or
- g.move_set.add(m)

Django also offers OneToOne and ManyToMany fields (http://goo.gl/rgqWZu)

Overview

Django Admin Site

- `admin.site.register()`
- `manage.py createsuperuser`
- Customize using `ModelAdmin` class
- `__str__`, Unicode and python 2

Model API

- Managers and QuerySets
- Save/update/delete
- Relations (foreign keys)