Practical TastyPie

For the Modern Djangonaut
Here’s the plan.
1. Model resourcefully.
2. URLs are queries.
3. Non-modeled data.
4. When things go awry.
Let’s get this party started.
1. Model resourcefully.
models.py

class Team(models.Model):
    name = models.CharField(max_length=255)
    city = models.CharField(max_length=255)
    ...

class Player(models.Model):
    team = models.ForeignKey(Team)
    first_name = models.CharField(max_length=255)
    last_name = models.CharField(max_length=255)
    ...

class Game(models.Model):
    teams = models.ManyToManyField(Team)
    date_and_time = models.DateTimeField()
    ...

Friday, February 24, 2012
models.py

class Team(models.Model):
    name = models.CharField(max_length=255)
    city = models.CharField(max_length=255)
    ...

class Player(models.Model):
    team = models.ForeignKey(Team)
    first_name = models.CharField(max_length=255)
    last_name = models.CharField(max_length=255)
    ...

class Game(models.Model):
    teams = models.ManyToManyField(Team)
    date_and_time = models.DateTimeField()
from tastypie.resources import ModelResource
from tastypie import fields
from tastypie.api import Api
from tastypie.constants import ALL
from tastypie.constants import ALL_WITH_RELATIONS
class TeamResource(ModelResource):
    class Meta:
        queryset = Team.objects.all()
        resource_name = 'teams'
        allowed_methods = ['get']
        filtering = {
            'name': ALL,
            'city': ALL,
        }
        ordering = filtering

...
class PlayerResource(ModelResource):
    team = fields.ForeignKey(my_app.TeamResource, 'team')

class Meta:
    queryset = Player.objects.all()
    resource_name = 'players'
    allowed_methods = ['get']
    filtering = {
        'first_name': ALL,
        'last_name': ALL,
        'team': ALL_WITH_RELATIONS
    }
    ordering = filtering
/api/v1/players/
/api/v1/teams/
CONGRATULATIONS ON YOUR API.

NEED TO GO. HEAR THE CAN OPENER.
2. URLs are queries.
class PlayerResource(ModelResource):
    team = fields.ForeignKey(my_app.TeamResource, 'team')

class Meta:
    queryset = Player.objects.all()
    resource_name = 'players'
    allowed_methods = ['get']
    filtering = {
        'first_name': ALL,
        'last_name': ALL,
        'team': ALL_WITH_RELATIONS
    }
    ordering = filtering
All players on a team?
/api/v1/players/?team__name=Redskins
api.py

...

class PlayerResource(ModelResource):
    team = fields.ForeignKey(my_app.TeamResource, 'team')

class Meta:
    queryset = Player.objects.all()
    resource_name = 'players'
    allowed_methods = ['get']
    filtering = {
        'first_name': ALL,
        'last_name': ALL,
        'team': ALL_WITH_RELATIONS
    }

    ordering = filtering

...
Order alphabetically?
/api/v1/players/?order_by=last_name
NINE AND NINETY QUANDARIES I DO POSSESS

BUT ARBITRARILY FILTERED DATA DOES NOT FIGURE AMONGST THEM
3. Non-modeled data.
class GameResource(ModelResource):
    teams = fields.ManyToMany(my_app.TeamResource, 'team')

class Meta:
    queryset = Player.objects.all()
    resource_name = 'players'
    allowed_methods = ['get']
    filtering = {
        'game_date_time': ALL,
        'teams': ALL_WITH_RELATIONS
    }
    ordering = filtering

...
api.py

... 

class GameResource(ModelResource):
    teams = fields.ManyToManyField(my_app.TeamResource, 'team')
    day_of_the_week = fields.CharField()  

class Meta:
    queryset = Player.objects.all()
    resource_name = 'players'
    allowed_methods = ['get']
    filtering = {
        'game_date_time': ALL,
        'teams': ALL_WITH_RELATIONS
    }
    ordering = filtering 

...
api.py

...

class GameResource(ModelResource):
    teams = fields.ManyToMany(my_app.TeamResource, ‘team’)
    day_of_the_week = fields.CharField()

...

def dehydrate_day_of_the_week(self, bundle):
    return bundle.obj.game_date_time.strftime(‘%a’)

Friday, February 24, 2012
DEHYDRATE

ALL OF THE THINGS
4. When things go awry.
Or, when you code like @jasonbartz.
Avoid

/foo/?limit=0
Avoid

/foo/?non_indexed_field__icontains=foo
Avoid

/foo/?callback=a_long-ever_changing_number
May the power of Bradlee compel you.