- 自我介绍

接触Python：2009
我与PyCon China：2011～2016

- 13+ Coding
- Life is short, let’s Pythonic
- 彪洋科技研发总监
- 2010~2014年
  - 大众点评搜索负责人

目　录

- 1.分享目标

听众： 能使用Python编码的技术人员

分享目标

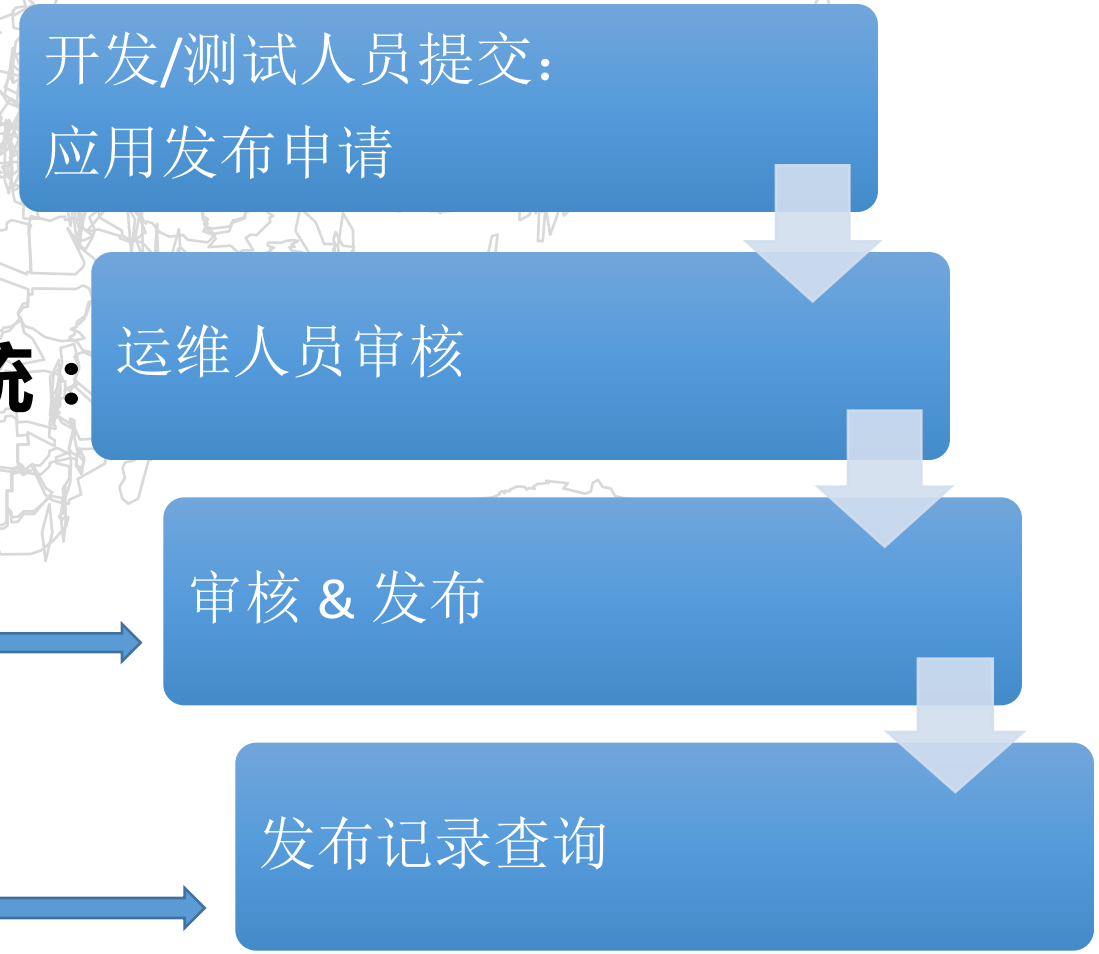
- 一窥Django之美
- 掌握Django Admin的使用
- 现场完成一个Django应用开发

- 掌握如何基于Django Admin进行快速开发
  - 如何3分钟写出增、删、改、查、排序、统计/过滤
  - 如何快速从数据库得到可运行的Admin web
  - 如何一行配置使得Admin web支持移动端
  - 如何3分钟搭建出Rest Service

- 现场练习

背景：
开发人员跟运维提交应用的发布，
走邮件或者QQ沟通，没有记录，
不系统，无法跟踪

目标：开发一个应用发布的管理系统：



运维人员可以审核一个申请，
审核后数据进到“已审核”申请单

申请单可以搜索，排序，条件过滤、统计

- Django是什么

The web framework for perfectionists with deadlines.

For perfectionists

For project with deadline

- 谁在用Django？

- Pinterest,
- Instagram,
- Mozilla,
- The Washington Times,
- Disqus,
- Public Broadcasting Service,
- itbucket,
- Nextdoor.

Why Django

- **基于Python：代码之美，自然语言之美**
- **Django Admin：自带管理后台**，可从DB生成、易定制

- **有大量轮子**：第三方类库（应有尽有，不用在重复造轮子投入）

- **开发框架中的火箭推进器：唯快不破**
  - 10分钟为已有系统快速搭建管理后台
  - AD域集成，权限控制；
  - Rest API；邮件；
  - 流程审批；异步作业；流量控制

- Django适合做什么？

适合的场景

- 偏内容的网站：
  - 如博客，聊天系统，如：Dropbox的zulip群聊

- 偏管理功能的内部系统：
  - 周报管理系统；KPI/OKR考核系统；会议室管理。。。
  - 合同管理；财务管理，采购管理，销售管理等
  - 数据管理：遗留系统的表数据管理/内容管理

- 流程处理：比如线上运维服务器申请，应用发布，SQL审核

- 产品MVP：产品探索，验证产品价值，如 招聘管理工具

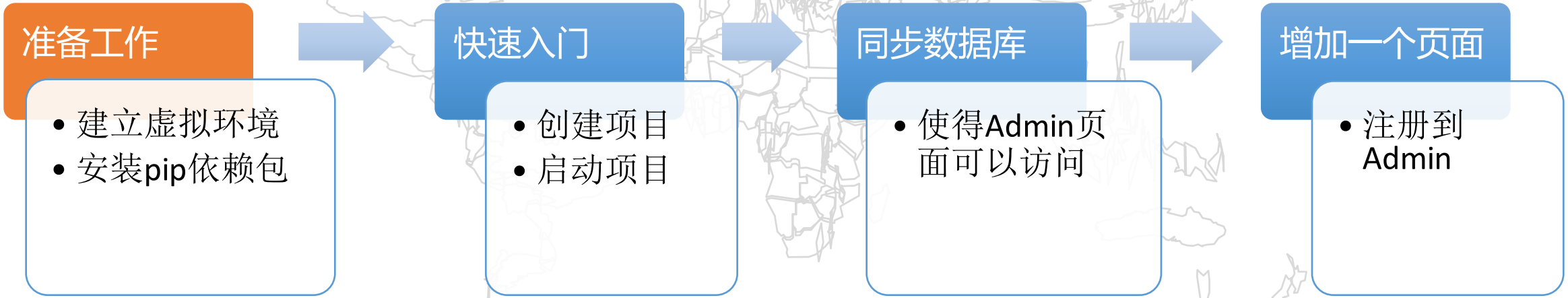
- Django不适合做什么？

不推荐的场景

- 有大量计算逻辑的后台
- 面向C端的复杂互联网产品

目　录

2.三分钟快速入门


准备工作
建立虚拟环境
安装pip依赖包
快速入门
创建项目
启动项目
同步数据库
使得Admin页面可以访问
增加一个页面
注册到Admin

- 准备工作：创建隔离的包管理环境

安装virtualenv & virtualenvwrapper

```
sudo pip install virtualenv
sudo pip install virtualenvwrapper
```

指定Python解释器为python3.5的虚拟环境

```
mkvirtualenv -p python3.5 python3
```

在~/.zshrc或~/.bashrc中加入初始化脚本

```
export WORKON_HOME=$HOME/.virtualenvs
source /usr/local/bin/virtualenvwrapper.sh
```

激活虚拟环境

```
workon
workon python3
```

- 准备工作：安装Django和依赖项

安装Django和相关依赖包

```
pip install django
pip install django-admin-bootstrapped
```

保存依赖包到*requirements.txt*文件

```
pip freeze > requirements.txt
```

以便日后方便地批量安装项目需要的所有依赖

```
pip install -r requirements.txt
```

- Read time out: 当pip遇到墙

$pip install xlrd

ReadTimeoutError: HTTPSConnectionPool(host='pypi.python.org', port=443): Read timed out.

解决方法一：使用国内mirror

$mkdir ~/.pip/, vim ~/.pip/pip.conf

[global]

http://pypi.douban.com/simple

trusted-host = pypi.douban.com

Honor & Thanks Goes To 豆瓣 douban

- Read time out: 当pip遇到墙 – 科学上网之道

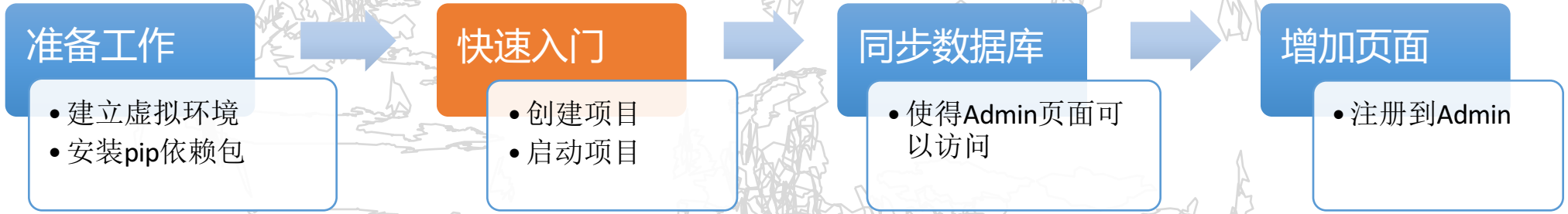
解决方法二：搭建内部的PIP仓库

pypi server
devpi-server
artifactory/nexus

解决方法三： VPN （SS & Proxifier）

还有什么更好的方法？

三分钟快速入门 – 开始入门（创建/启动项目）



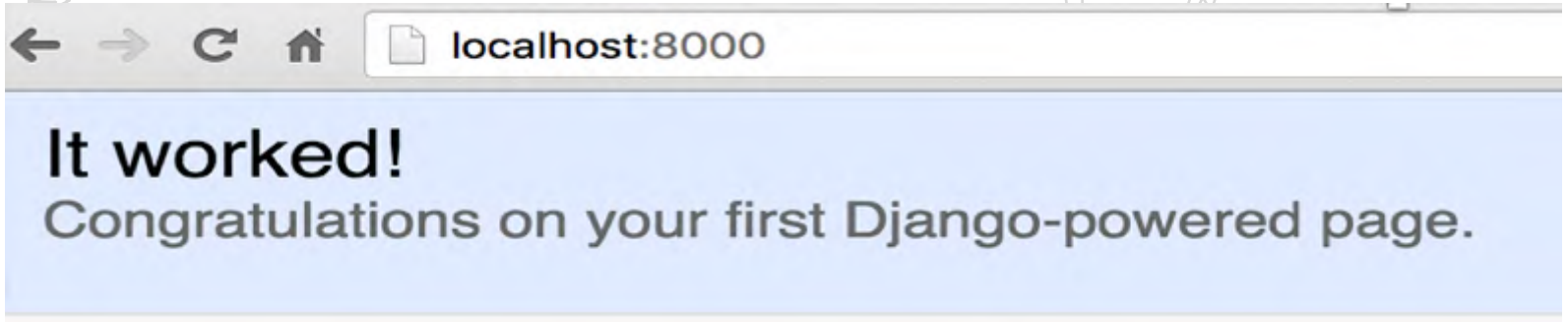
创建项目

```
django-admin startproject opsrecorder
```

运行项目

```
cd opsrecorder && django-admin startapp releaserecorder
python manage.py runserver 0.0.0.0:8080
```

访问这里: http://localhost:8080/

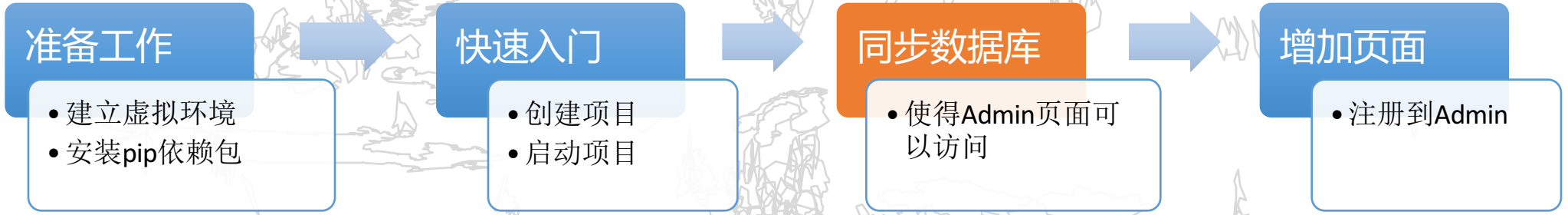


- 咦， 传说中的Admin管理台页面哪里去了？

浏览器访问项目: http://localhost:8080/admin

Ooooooooops

同步数据库 & 创建Admin管理后台的账号



创建数据库，从model同步到数据库：

```
./manage.py migrate
```

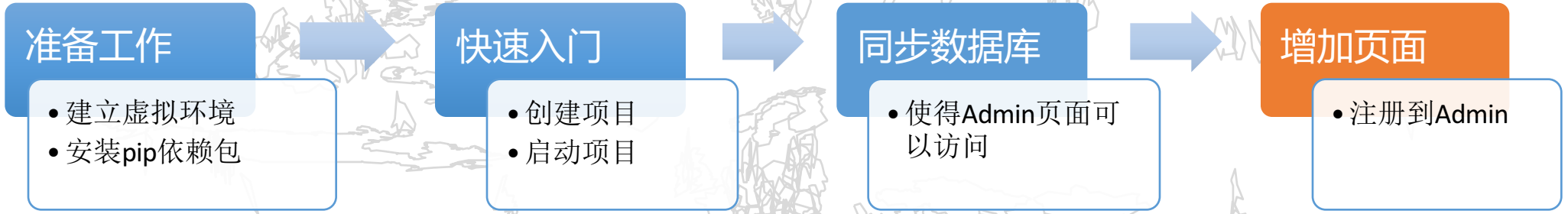
创建超级管理员（输入用户名/密码，如: admin/admin123）：

```
./manage.py createsuperuser
```

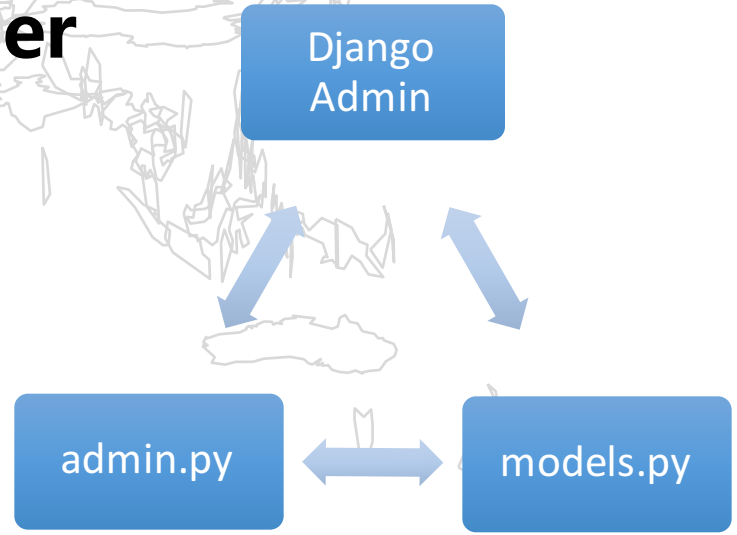
访问 http://localhost:8080/admin （不需要重启）

看到登陆页，恭喜你!

增加一个页面



1. **./manage.py startapp releaserecorder**
2. **models.py中增加对象定义**



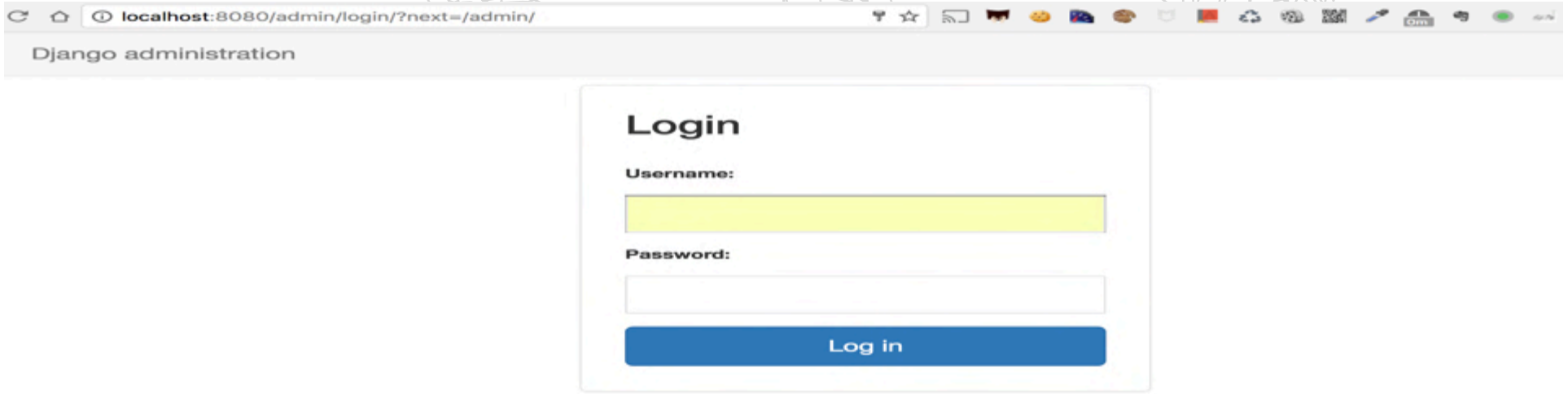
3. **admin.site.register()**
4. **settings.py中增加apps**

5. **./manage.py makemigrations releaserecorder ；./manage migrate**
6. **刷新管理台页面**

- 如何支持移动设备的访问(Bootstrapped)

$pip install django-admin-bootstrapped

添加 **django_admin_bootstrapped** 到settings.py文件中 **INSTALLED_APPS**的最前面:
INSTALLED_APPS = ('django_admin_bootstrapped',
'django.contrib.admin', ...)



- 日常开发中，如下的功能，需要多少行代码？

1. **列表展示**
2. **模糊搜索**
3. **过滤统计**
4. **排序 & 多维排序**

Home / Releaserecorder / Release applications

Select release application to change + Add release application

Search Search Filter

Action: --------- Go 0 of 2 selected

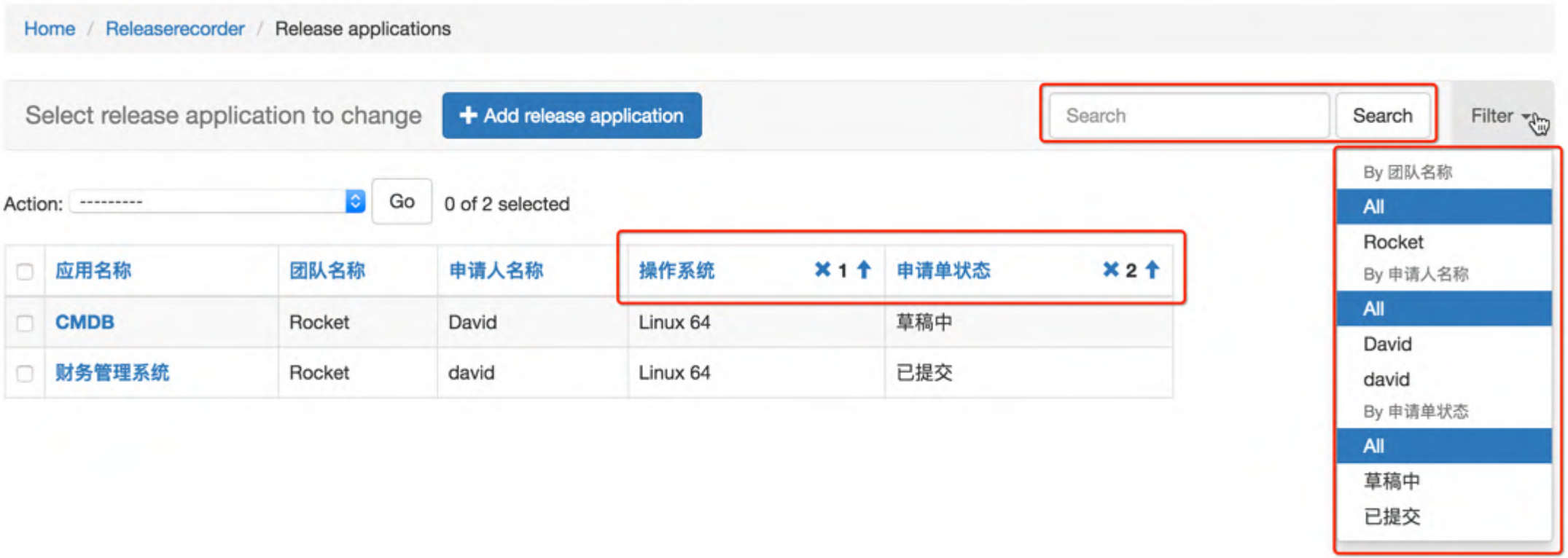
应用名称	团队名称	申请人名称	操作系统 ✖ 1	申请单状态 ✖ 2
CMDB	Rocket	David	Linux 64	草稿中
财务管理系统	Rocket	david	Linux 64	已提交

By 团队名称
All
Rocket
By 申请人名称
All
David
david
By 申请单状态
All
草稿中
已提交

- 3行代码，写出如下的功能页面

1. **列表展示**
2. **模糊搜索**
3. **过滤统计**
4. **排序 & 多维排序**

```
list_display =  ('appname','team_name','application_username','os_type','status',)
search_fields =  ('appname','application_note','deploy_note',)
list_filter =  ('team_name','application_username','status',)
```



列表页的查询

```python
#coding:utf-8
from django.contrib import admin
from releaserecorder.models import ReleaseApplication
from django.contrib.admin import actions

def submit_application(modeladmin, request, queryset):
    queryset.update(status='submitted')

submit_application.short_description = "提交申请"


class ReleaseApplicationAdmin(admin.ModelAdmin):
    actions = [submit_application]

    readonly_fields = ('created_date','updated_date','status',)
    list_display =  ('appname','team_name','application_username','os_type','status',)
    search_fields =  ('appname','application_note','deploy_note',)
    list_filter =  ('team_name','application_username','status',)

# Register your models here.
admin.site.register(ReleaseApplication,ReleaseApplicationAdmin)
```

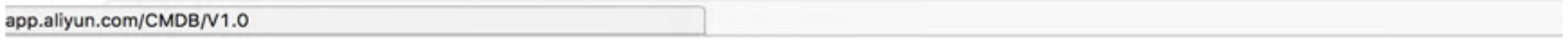
- 等等

你这个例子看起来太简单了，

可是我需要的列有一列包涵复杂的内容，
这一列值不在单个字段中，包涵业务逻辑，django admin能满足我吗？

- 必然是可以的

例： 需要加一列持续集成的“状态”链接，
这个链接URL由应用的名称，版本的信息组成



- Show me the code, please

```python
class ReleaseApplicationAdmin(admin.ModelAdmin):
    actions = [submit_application]

    def get_app_url(self,obj): #显示app的版本持续集成状态页面的链接
        return format_html(u'<a href="http://app.aliyun.com/%s/%s">状态</a>' % (obj.appname, obj.version))
    get_app_url.short_description = '持续集成状态'

    readonly_fields = ('created_date','updated_date','status',)
    list_display =  ('appname','get_app_url','team_name','application_username','os_type','status','version',)
    search_fields =  ('appname','application_note','deploy_note',)
    list_filter =  ('team_name','application_username','status',)

# Register your models here.
admin.site.register(ReleaseApplication,ReleaseApplicationAdmin)
```

这里的 get_app_url()这个方法名可疑随意取，
方法里面可以有任意复杂的逻辑；
实际上， ModelAdmin定义了很多get_xxx_yyy的方法，可当作属性使用
比如get_readonly_fields(), get_fieldsets()等等

- 更多定制逻辑的代码

实现get_xxx()方法，增加逻辑辑代码来控制内容的显示

```
fieldsets = (
    (None, {'fields': ("province","city","laundry_name","boss_name","address","empno","comment",)}),
    ('联系方式', {'fields': ("mobile","email","qq",)}),
    ('其他', {'fields': ("status","created_time","updated_time",)}),
)
#  'fields': ("province","city","laundry_name","boss_name","address","mobile","email","qq","comment",)
ordering = ('city','visit_user','-created_time',)

def get_readonly_fields(self, request, obj):
    return ("created_time","updated_time")

def get_fieldsets(self, request, obj):
    if request.user.is_admin and obj:  # for normal user, show readonly club_name on editor page
        return (
            (None, {'fields': ("province","city","laundry_name","boss_name","address","visit_user","empno","comment",)}),
            ('联系方式', {'fields': ("mobile","email","qq",)}),
            ('其他', {'fields': ("status","created_time","updated_time",)}),
        )

    ## 增加记录时，不显示created_time, updated time
    return (
            (None, {'fields': ("province","city","laundry_name","boss_name","address","empno","comment",)}),
            ('联系方式', {'fields': ("mobile","email","qq",)}),
            ('其他', {'fields': ("status",)}),
        )

def get_queryset(self, request):  #show data only owned by the user
    qs = super(VisitLaundryAdmin, self).get_queryset(request)
    if request.user.is_admin:
        return qs
    return qs.filter(visit_user=request.user)
```

get_readonly_fields()： 会覆盖readonly_fields属性，设置只读属性；
get_fieldsets()：会覆盖fieldsets，定义form页内容的分组显示

目　录

- 遗留系统集成

一个权限管理系统，背景：
公司的一个遗留系统，系统的模块，权限，角色，角色权限，原先是在数据库中直接维护的，容易出错，效率低。要有一个可视化的管理页面来维护；

需求
1). 3分钟生成一个管理后台；

2). 可以灵活定制页面；

3). 可以配置一个权限是否有效；

- 生成项目/从Database反向生成代码

$ pip install psycopg2 -- 安装postgresql驱动

$ django-admin startproject rolemanager

编辑settings.py中的数据库配置

$ vim ~/settings.py

```
DATABASES = {
'default’: {
    'ENGINE': 'django.db.backends.postgresql',
    'NAME': 'mydatabase',
    'USER': 'mydatabaseuser',
    'PASSWORD': 'mypassword',
    'HOST': '127.0.0.1', 'PORT': '5432',
    }
}
```

- 定制项目

从数据库反向生成model

./manage.py inspectdb > models.py

建立django元数据表

./manage.py makemigrations rolemanager

./manage.py migrate

./manage.py createsuperuser

目　录

- 推荐的Django插件

[django-admin-bootstrapped]
Bootstrapped,自动适应不同屏幕大小，更漂亮的UI

[django-anymail]
发送邮件，集成Mailgun, SendGrid等等

[django-smart_selects]
省市联动库（一行代码指定联动规则，页面自动联动）

[pinax ldap] AD域账号集成

[django-ratelimit]: 流量控制

[django-defender]: 防暴力破解

[django-rest-framework & rest throttling]
为数据提供Rest服务

目　录

- 经验与教训

1. **如何避免重复造轮子，提高效率**
   - 开工前，想想问题是否常见，去Google,Github找现成的实现/类库

2. **通过Migrate，Model中的help_text 未同步到DB表的字段注释中**
   - json中定义表字段&注释，编写工具读取json生成sql & model

3. **没有好用的IDE，推荐开发工具**
   - LiClipse （基于Pydev + Eclipse, 优先推荐）
   - Spyder
   - PyCharm （Community版本不支持Python库如django）
   - VIM, Sublime

- 经验与教训

4. **pip安装软件包出现 timeout错误**
   - 使用 douban的pip镜像

5. **Python的多个版本相互冲突:**
   - 应用使用虚拟环境
   - 使用virtualenvwrapper，workon

6. **Python的多语言问题，常常提示关于编码的错误**
   - 文件头声明编码为utf-8: #coding:utf-8
   - 使用 python3
   - 文件的保存格式设置为 utf-8
   - ~/.vimrc中定义vim编码: set encoding=utf-8

- 经验与教训

7. 如何提升编码效率：model中重复敲大量的models.XXXField效率低

- 定义code template，autocompletion

8. Django这么强大，我怎么样才能了解到还有哪些强大的功能？

- 把 django应用到项目中
- 阅读 django官方文档
- Google

- 附： models.py定义

```python
#coding=utf-8
from django.db import models
from datetime import datetime


OS_TYPE_CHOICES = (('windows','Windows Server'),('linux','Linux 64'),)
APPLICATION_STATUS_CHOICES = (('draft','草稿中'),('submitted','已提交'),)

# Create your models here.
class ReleaseApplication(models.Model):
    id = models.AutoField(primary_key=True)
    appname = models.CharField(max_length=200,verbose_name=u"应用名称",help_text=u"要发布的应用名称")
    version = models.CharField(max_length=200,blank=True,verbose_name=u"应用版本",help_text=u"要发布的应用版本")
    team_name = models.CharField(max_length=200,verbose_name=u"团队名称",help_text=u"负责应用的团队名称")
    responsible_username = models.CharField(max_length=200,verbose_name=u"负责人名称",blank=True,help_text=u"应用的负责人")
    application_username = models.CharField(max_length=200,verbose_name=u"申请人名称",help_text=u"申请人的名称")
    application_note = models.CharField(max_length=200,verbose_name=u"应用描述",blank=True,help_text=u"关于应用的描述")
    deploy_note = models.TextField(verbose_name=u"发布说明",help_text=u"此次发布的步骤说明")


    os_type = models.CharField(max_length=100, choices=OS_TYPE_CHOICES, default='linux', verbose_name=u'操作系统', help_text=
    instance_count = models.IntegerField(default=2,verbose_name=u"应用部署的实例数",help_text=u"申请应用部署的实例数量")
    status = models.CharField(max_length=100, choices=APPLICATION_STATUS_CHOICES,default='draft', verbose_name=u'申请单状态',

    created_date = models.DateTimeField(default=datetime.now, blank=True,verbose_name=u'创建时间')
    updated_date = models.DateTimeField(default=datetime.now, blank=True,verbose_name=u'更新时间')
    remark = models.CharField(max_length=200,blank=True,verbose_name=u"备注",help_text=u"备注")
```

@david_euler

github.com/davideuler