蓝鲸智云 腾讯云

Django开发基础（三）

蓝鲸智云讲师 hongsong

课程内容

- Django复杂查询
- 外键的简单使用
- 页面的后台渲染
  - Template模板渲染机制
  - Mako模板简单使用
- 数据更新与删除
- 日志记录和查看

Django复杂查询

基本查询 – 奖项


```
1. awards = Award.objects.filter(name__contains='qb')
2. awards = Award.objects.filter(name__contains='qb',
                                 organization__name=u'腾讯科技')
3. awards = Award.objects.filter(name__contains='qb', state=1)
4. awards = Award.objects.filter(name__contains='qb',
                                 start_time__gte='2017-12-01',
                                 end_time__lte='2017-12-03')
```

复杂查询 - filter

过滤关键字	含义	SQL
__contains	包含	where field like“%test%”
__startswith	以..开头	where field like“test%”
__gt, __gte	大于，大于等于	where field > 10
__lt, __lte	小于，小于等于	where field < 10
__range=[a,b]	a,b两者之间	where field between a and b
__in=(a, b)	值属于列表中的值	where field in (a,b)

```
1.awards = Award.objects.filter(name__contains='qb')
2.awards = Award.objects.filter(name__startswith='qb')
3.awards = Award.objects.filter(name__endswith='qb')
4.awards = Award.objects.filter(applicant_num__range=[10,20])
```

复杂查询 – 排除、排序、去重、总数

1.排除

```
Award.objects.filter(name__contains="qb").exclude(is_delete=1)
```

2.排序

```
Award.objects.all().order_by('start_time')
```

3.去重

```
Award.objects.all().values('name').distinct()
```

4.总数

```
Award.objects.all().count()
```

复杂查询 – Q查询

1. 在filter中使用Q表达式进行更复杂的查询

2. 可通过操作符 |（或），&（与），~（非）来组合Q表达式

```
1. Award.objects.filter(Q(name__contains="qb") & Q(level=2))
2. Award.objects.filter(Q(name__istartswith="qb")|
3.                      Q(applicant_num__range=(1,5)))
4. Award.objects.filter(~Q(name__startswith="qb"))
```

外键的简单使用

Model外键在数据库中的表现

"Persons" 表：

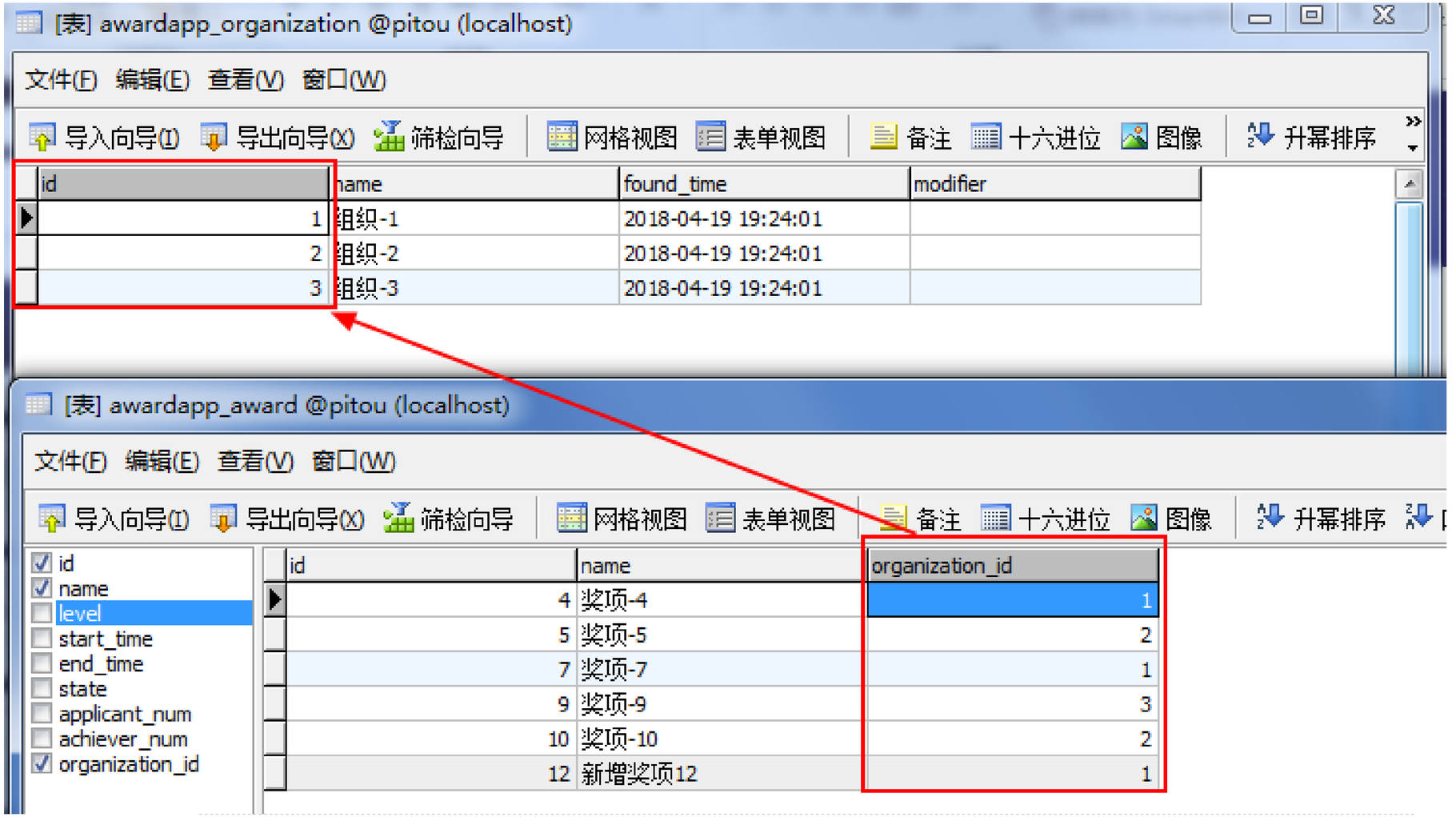
Id_P	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing

"Orders" 表：

Id_O	OrderNo	Id_P
1	77895	3
2	44678	3
3	22456	1
4	24562	1

- Orders表中的ForeignKey(外键, Id_P) 指向Persons表中的 PrimaryKey(主键, Id_P)
- 外键的好处：
  - 满足了现实世界的对象间关系的描述需求（买家和订单的关系）
  - 保护数据间的关系完整性（订单中必须存在买家）
  - 把较大的数据对象拆分成关联的对象，方便管理

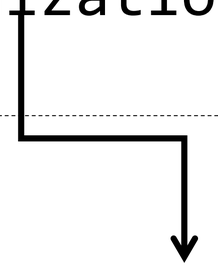
Model外键申明



Model外键查询

- 当成普通字段使用

```
1. # 遍历所有奖品所属的组织，并打印组织名称
2. award_queryset =Award.objects.all()
3. for item in award_queryset:
4.     organization = item.organization
5.     print organization.name
```



organization是关联的组织对象

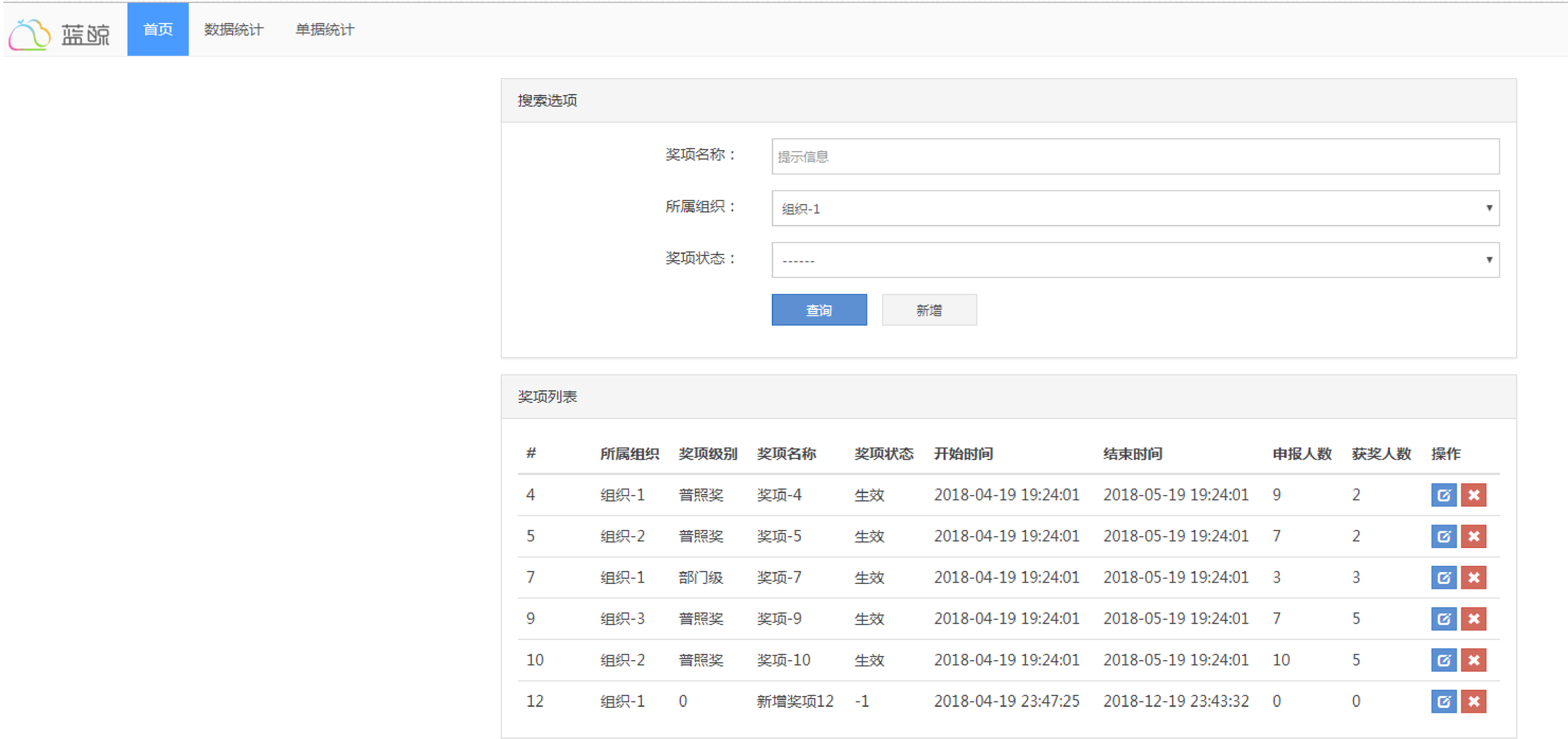
Model外键作为查询条件

- 字段名+双下划线（__）+foreignKey表的某个字段过滤

```
1.# 查询数据
2.awards =Award.objects.filter(organization__name__contains="组织")
3.for item in awards:
4.    organization = item.organization
5.    print organization.name
```

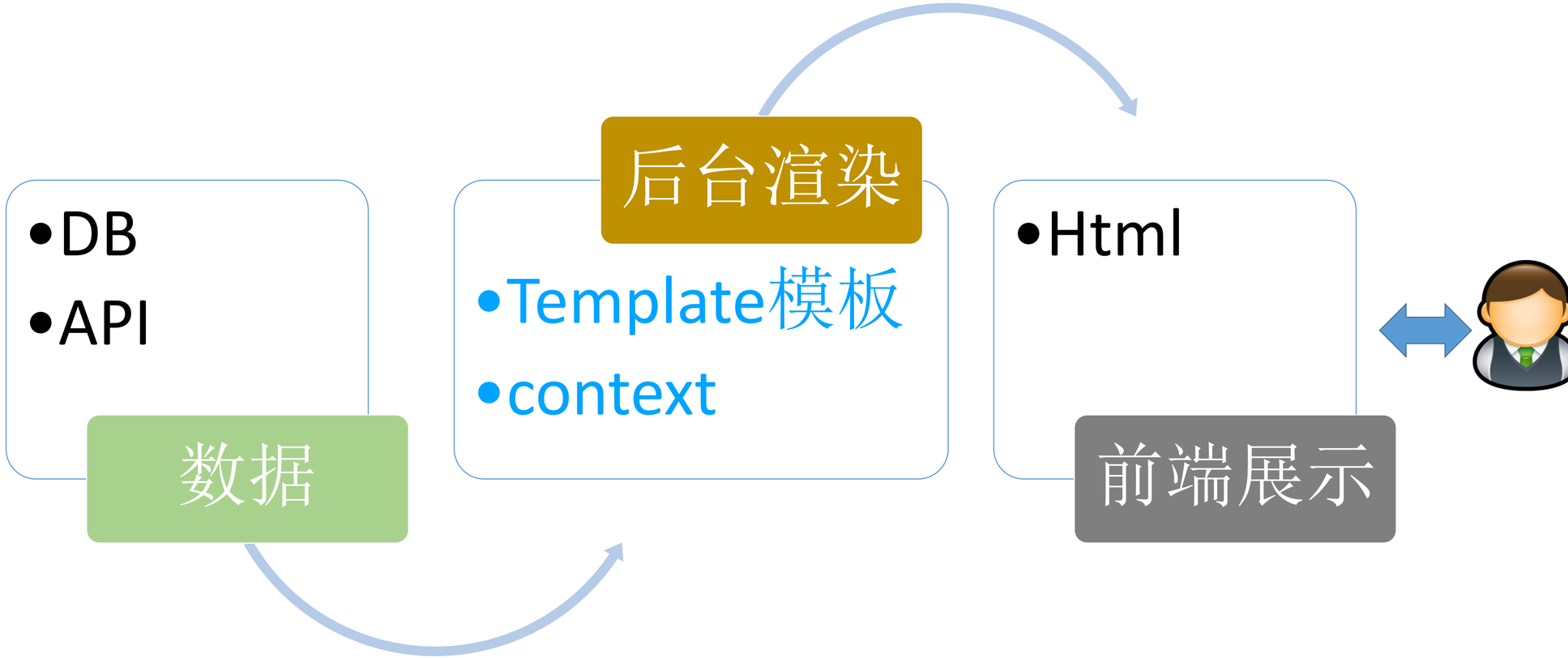
页面的后台渲染

实战 - 奖项管理页面


蓝鲸
首页
数据统计
单据统计
搜索选项
奖项名称：
提示信息
所属组织：
组织-1
奖项状态：

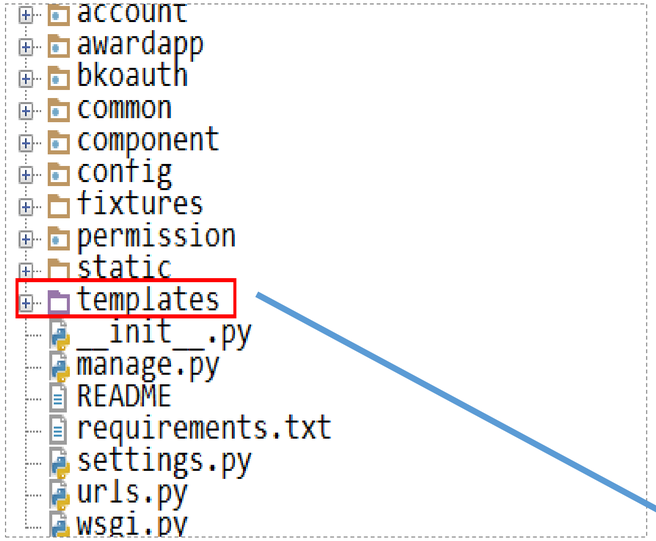
查询
新增
奖项列表
所属组织 奖项级别 奖项名称 奖项状态 开始时间 结束时间 申报人数 获奖人数 操作
4 组织-1 普照奖 奖项-4 生效 2018-04-19 19:24:01 2018-05-19 19:24:01 9 2
5 组织-2 普照奖 奖项-5 生效 2018-04-19 19:24:01 2018-05-19 19:24:01 7 2
7 组织-1 部门级 奖项-7 生效 2018-04-19 19:24:01 2018-05-19 19:24:01 3 3
9 组织-3 普照奖 奖项-9 生效 2018-04-19 19:24:01 2018-05-19 19:24:01 7 5
10 组织-2 普照奖 奖项-10 生效 2018-04-19 19:24:01 2018-05-19 19:24:01 10 5
12 组织-1 0 新增奖项12 -1 2018-04-19 23:47:25 2018-12-19 23:43:32 0 0

Template后台渲染机制



Template 配置

- **默认templates目录**



- **Settings配置**

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': list(TEMPLATE_DIRS),
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': list(TEMPLATE_CONTEXT_PROCESSORS),
        },
    },
]
```

```
MAKO_TEMPLATE_DIR = os.path.join(PROJECT_ROOT, 'templates')
MAKO_TEMPLATE_MODULE_DIR = os.path.join(
                                    PROJECT_DIR,
                                    'templates_module',
                                    APP_CODE
                                )
```

Mako Template模板

```
<div class="clearfix">
    % for item in my_award_list:
        <div class="col-xs-4">
            <div class="panel panel-default">
                <div class="panel-body">
                    % if item.picture_url:
                        <a href="${SITE_URL}personal/application/add.html?award_id=${item.id}"><img
                                class="award-img" src="${item.picture_url}"/></a>
                    %else:
                        <a href="${SITE_URL}personal/application/add.html?award_id=${item.id}"><img
                                class="award-img"
                                src="${STATIC_URL}awardapp/award/default-award.jpg"/></a>
                    %endif
                    <p>${item.organization.name}</p>
                    <p>${item.name}</p>
                    <p>提报人数${item.applicant_num}人</p>
                </div>
            </div>
        </div>
    %endfor
</div>
```

- 变量引用
  - ${var}
- 逻辑控制：
  - %if ... %endif
  - %for...%endfor

Mako模板语法

```
1. <!- base.html-->
2. <html>
3.
4. <head>
5.    <title>我是标题</title>
6. </head>
7.
8. <body>
9.
10.  <%block name="content"/>
11.     <p>我是个占位块，名字叫content</p>
12.  <%block/>
13.
14.</body>
15.
16.</html>
```

```
1. <!- test-block.html-->
2.
3. <% inherit 'base.html' />
4.
5.
6. <%block name="content"/>
7.   <p>我会继承并覆盖base.html中
8.      名字叫content的占位块中的内容
9.   </p>
10.<%block/>
```

占坑: <%block>
填坑: inherit

```
1. <!- test-block.html-->
2. <html>

3. <head>
4.    <title>我是标题</title>
5. </head>
6.
7. <body>
8.
9.   <%block name="content"/>
10.   <p>我会继承并覆盖base.html中
11.      名字叫content的占位块中的内容
12.   </p>
13.   <%block/>
14.
15.</body>
16.
17.</html>
```

Mako模板渲染

```python
from common.mymako import render_mako_context

def index(request):
    '''
    首页
    '''
    my_award_list = models.Award.objects.filter(
        organization__member__username=request.session['user_qq'],
        organization__member__role=2,
        organization__is_delete=False,
        is_delete=False
    )

    context = {'my_award_list': my_award_list}

    return render_mako_context(request, 'awardapp/index.html', context)
```

用来渲染的数据

Mako模板渲染函数

Mako模板文件

数据更新与删除

表单渲染主键设置

```
<tbody>
% for item in award_list:
    <tr>
        <td>${item.achiever_num}</td>
          …
        <td>
            <a href=“${SITE_URL}/backstage/award/detail/${item.id}/” >查看</a>
            <a href=“${SITE_URL}/backstage/award/edit/${item.id}/” >编辑</a>
            <a onclick=“open_modal('${SITE_URL}/backstage/award/del/${item.id}/','是否删除？')">删除</a>
        </td>
    </tr>
%endfor
</tbody>
```

更新数据

```
// ajax请求
$.ajax({
    url: '/award/edit/${id}/',
    type: 'POST',
    data: {'data':update_data},
    success: function (data) {
        // do something after edit finished
    }
})
```

发起请求

```
## url配置
url(r'^award/edit/(\d+)/$', 'award_edit',name='award_edit')
```

处理请求

```
#后台更新代码
def award_edit(request, award_id):
    award_queryset = models.Award.objects.filter(
                                id=award_id, is_delete=False
                            )
    update_data=request.POST.get("data")
    award_queryset.update(** update_data)
```

删除记录

```
// 前端发起请求
$.ajax({
    url: ‘/award/delete/${id}/',
    type: 'POST',
    data: {},
    success: function (data) {
        // do something after delete finished
    }
})
```

发起请求

```
## 路由配置 - urls.py
url(r'^award/delete/(\d+)/$', 'award_delete',name='award_delete')
```

处理请求

```
## 后台删除记录
def award_delete(request, award_id):
    award_queryset = models.Award.objects.filter(
                                id=award_id
                               ).delete()
```

日志记录logger

Logger日志记录

- Settings配置

```
if RUN_MODE == "DEVELOP":
    LOG_LEVEL = LOG_LEVEL_DEVELOP
    LOG_CLASS = 'logging.handlers.RotatingFileHandler'
elif RUN_MODE == "TEST":
    LOGGING_DIR = LOGGING_DIR_ENV    # 使用环境相关的LOGGING_DIR
    LOG_CLASS = 'logging.handlers.RotatingFileHandler'
    LOG_LEVEL = LOG_LEVEL_TEST
elif RUN_MODE == "PRODUCT":
    LOGGING_DIR = LOGGING_DIR_ENV    # 使用环境相关的LOGGING_DIR
   LOG_LEVEL = LOG_LEVEL_PRODUCT
   LOG_CLASS = 'logging.handlers.RotatingFileHandler'
```

Logger日志记录

- 日志级别
  - Critical
  - Exception – 包含堆栈信息
  - Error—错误日志
  - Warning – 告警
  - Info – 普通
  - Debug – 调试日志，test和现网不适合打印

```
# 开发框架日志级别设置

# 本地开发环境日志级别
LOG_LEVEL_DEVELOP = 'DEBUG'
# 测试环境日志级别
LOG_LEVEL_TEST = 'INFO'
# 正式环境日志级别
LOG_LEVEL_PRODUCT = 'ERROR'
```

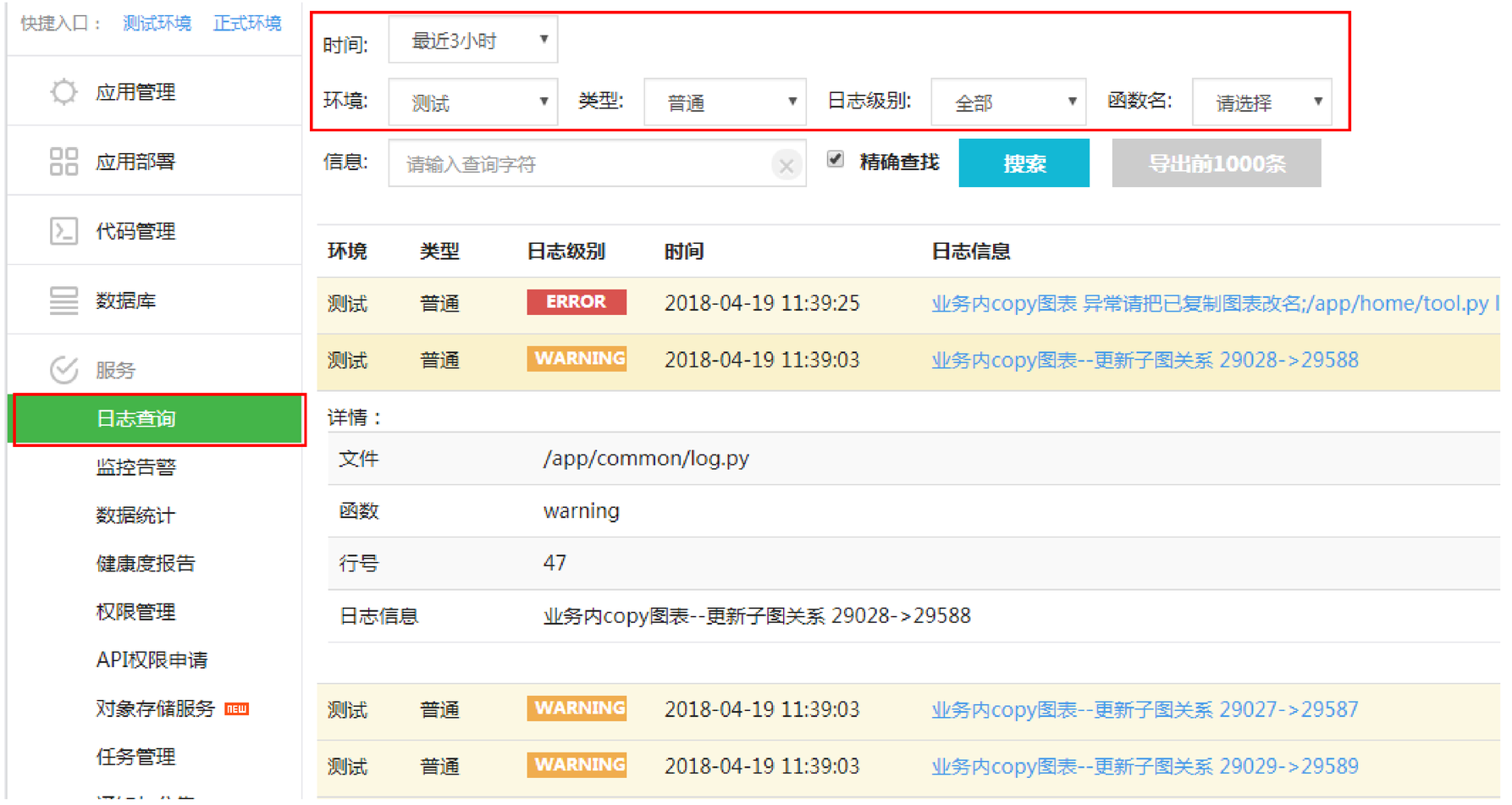
Logger日志打印方法

```
#引用
from common.log import logger

# 根据需要打印不同级别的日志
logger.info(u'普通日志')
logger.warning(u'警告日志')
logger.error(u'错误日志')

# 带参数日志
logger.error(u'错误日志，错误详情：%s' % message)
```

开发者中心日志查看



课后作业

- 奖项申请（前端表单，后端接口）
- 奖项申请审核（前端表单，后端接口）
- 渲染组织信息
- 渲染奖项信息，包括首页、奖项列表、奖项详情
- 渲染申请信息，包括我的申请、我的审批、首页
- 组织信息更新页面，回填信息
- 奖项信息更新页面，回填信息

进阶扩展

表单分页方法

- 前端分页：Datatables, kendo等

- 后端分页：
  - 逻辑实现
  - **paginator分页**

```
from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger
```

Paginator分页使用示例

```
>>> from django.core.paginator import Paginator
>>> objects = ['john', 'paul', 'george', 'ringo']
>>> p = Paginator(objects, 2)
>>> p
<django.core.paginator.Paginator object at 0x03E49DF0>
>>> a = p.page(1)
>>> a
<Page 1 of 2>
>>> a.object_list
['john', 'paul']
```

重要方法：

- Paginator(objects, one_page_nums)
  - Paginator对象
- p.page(current_page)
  - 当前页面的属性
- a.object_list
  - 页内元素列表

Paginator分页属性

Input	Output	Type
paginator.**count**	**53**	`<type 'int'>`
paginator.**num_pages**	**6**	`<type 'int'>`
paginator.**page_range**	**xrange(1, 7)**	`<type 'xrange'>`
paginator.**page(2)**	**<Page 2 of 6>**	`<class 'django.core.paginator.Page'>`

Paginator分页python代码实现

```
application_queryset = models.Application.objects.filter(state=5).order_by('-id')

paginator = Paginator.page(application_queryset, 10)

try:
    award_list = paginator.page(request.GET.get('page', default=1))
except PageNotAnInteger:
    award_list = paginator.page(1)
except EmptyPage:
    logger.info(u"当前页面参数大于总页面数 %s" % paginator.num_pages)
    award_list = paginator.page(paginator.num_pages)

context = {
    'award_list': award_list.object_list,
    'page_index': award_list.num_pages
}
return render_mako_context(request, 'awardapp/award_record.html', context)
```

Paginator分页mako代码

```
<table class="table table-hover">
    <thead>
    <tr>
        <td>所属单位</td>
        <td>申报奖项</td>
        <td>申请时间</td>
        <td>申报人/团队</td>
        <td>申报详情</td>
    </tr>
    </thead>
    <tbody>
    % for item in award_list:
        <tr>
            <td>${item.award.organization.name}</td>
            <td> ${item.award.name} </td>
            <td> ${item.apply_time | date:"Y-m-d H:i:s"} </td>
            <td>${item.applicant_info}</td>
            <td>
                <a href="${SITE_URL}personal/application/detail/${item.id}.html">
                    <span style="margin-right: 5px;" class='glyphicon glyphicon-eye-open'
                          aria-hidden='true'></span>查看</a>
            </td>
        </tr>
    %endfor
    </tbody>
</table>
```

Model外键高级查询 – 扩展

- **select_related 直接一次查询将外键信息查出，减少DB访问次数**

```
1. # 查询数据
2. award_queryset = models.Award.objects.all(
3.                         ).select_relate("organization")
4. for item in award_queryset:
5.     organization = item.organization
6.     print organization.id
```

蓝鲸智云

腾讯云

蓝鲸智云公众号

蓝鲸高校版交流群