蓝鲸智云 腾讯云

Django基础（二）

蓝鲸智云讲师 jeremy

课程内容

- Model介绍
  - 创建、字段说明、migrate指令说明
  - model设计
  - CURD
- Admin管理后台
- 文件上传下载

一、Django Model

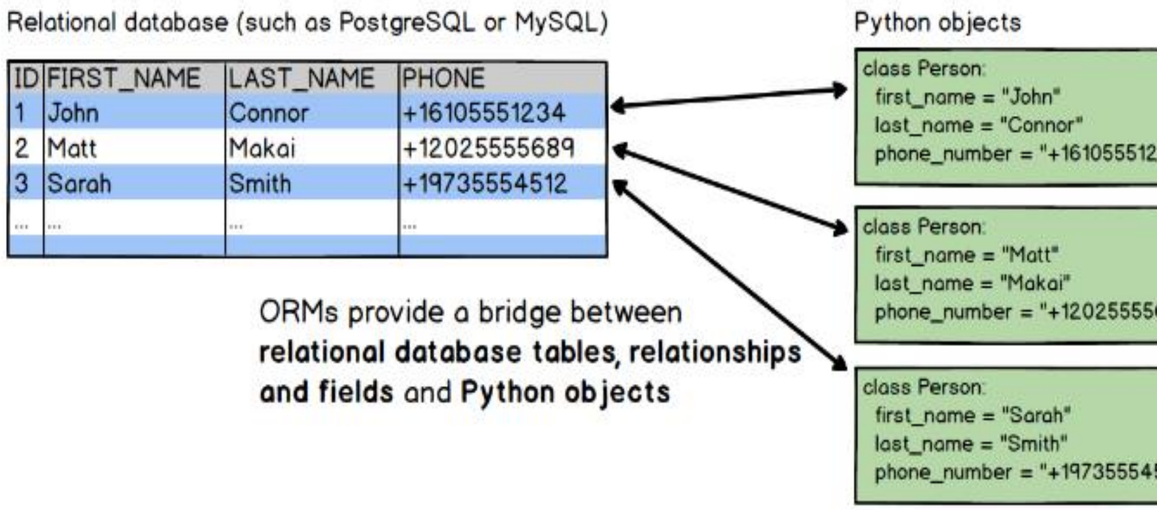
Model介绍

- 什么是 **Model ?**

A model is the single, definitive source of information about your data.
It contains the essential fields and behaviors of the data you're storing.
Generally, each model maps to a single database table.

- **Basics**

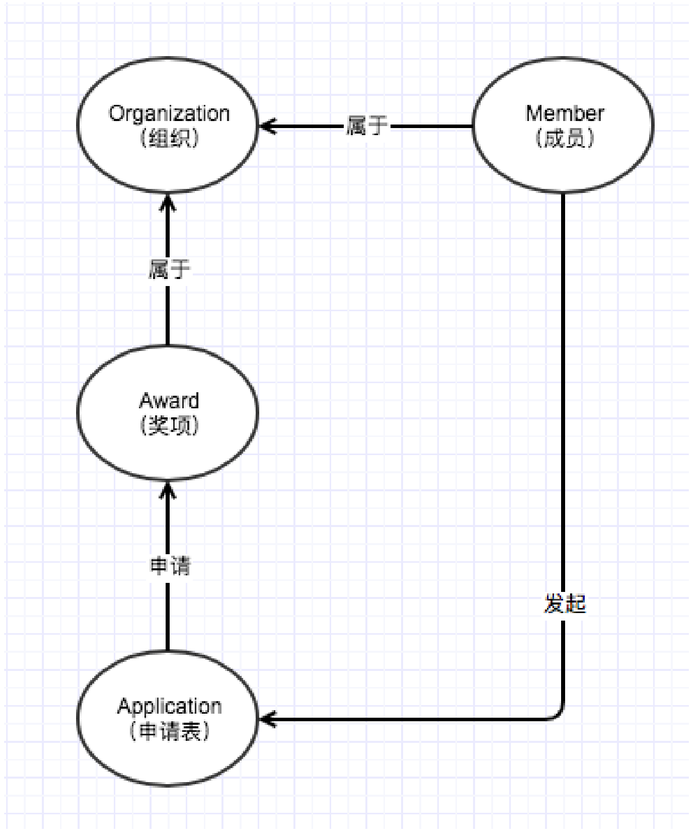
1. 每一个Model代表了数据库中的一张表
2. 每一个Model都是[django.db.models.Model]的子类
3. Model的每个属性代表了数据库表的每个字段



Model介绍

- **我们拿我们的课后练习《奖项申报》思考下**

1. 这里涉及到哪些对象？
2. 这些对象有哪些属性？
3. 用Django怎么去表示这些对象和属性



Model介绍

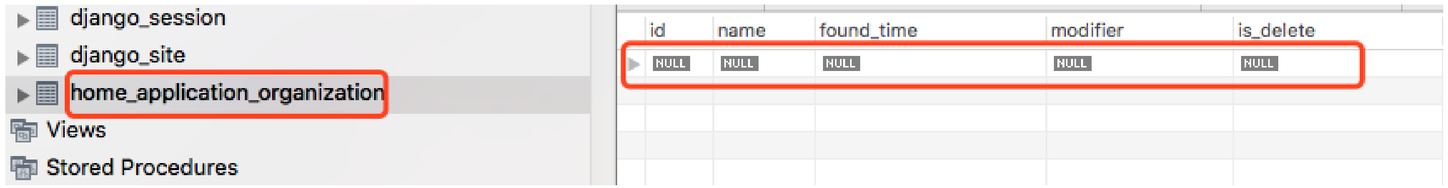
- 以 **Organization** 为例演示

Models.py

```python
class Organization(models.Model):
    """
    组织
    """
    name = models.CharField(verbose_name=u"组织名称", max_length=64, unique=True)
    found_time = models.DateTimeField(verbose_name=u"建立时间", auto_now_add=True)
    modifier = models.CharField(verbose_name=u"建立用户", max_length=64)
    is_delete = models.BooleanField(verbose_name=u"是否删除", default=False)

    def __str__(self):
        return "%s" % self.name
```

MySQL



Model介绍

- 以 **Member** 为例演示

Models.py

```
class Member(models.Model):
    """
    组织成员表
    """
    username = models.CharField(verbose_name=u"用户", max_length=32)
    organization = models.ForeignKey(verbose_name=u"所属组织", to=Organization)
    role = models.IntegerField(verbose_name=u"级别",
                               choices=[(1, '负责人'), (2, '参评人')])

    def __str__(self):
        return "%s-%s-%s" % (self.username, self.organization, self.role)
```

Model介绍

- **字段类型（Field types）**

Django提供的常见的字段类型有：

- IntegerField　整数类型字段
- CharField　字符类型字段
- DateTimeField　时间类型字段
- 其他：BooleanField、DateField、TextField
- 自定义

- OneToOneField：　一对一的关系
- ForeignKey：　多对一的关系
- ManyToManyField：　多对多的关系

More info：　https://docs.djangoproject.com/en/1.8/ref/models/fields/#model-field-types

Model介绍

- **字段选项配置（Field Options）**
  - null: true | false
  - blank： true | false
  - choices:

```
YEAR_IN_SCHOOL_CHOICES = (
    ('FR', 'Freshman'),
    ('SO', 'Sophomore'),
    ('JR', 'Junior'),
    ('SR', 'Senior'),
)
```

  - default: 默认值
  - primary_key: 是否为主键
  - unique: 是否值唯一
  - help_text：

Model介绍

- **Model的元信息（Meta Options）**

可以通过在Model的内部定义class Meta来提供Model的一些元信息，比如下面的例子：

```
1.from django.db import models
2.
3.class Example(models.Model):
4.
5.    horn_length = models.IntegerField()
6.
7.    class Meta:
8.        ordering = ["horn_length"]
9.        verbose_name_plural = "oxen"
```

常用的元信息

- ordering
- db_table
- verbose_name
- verbose_name_plural

Moreinfo: https://docs.djangoproject.com/en/1.8/ref/models/options/

Model介绍

- **字段取名时的一些限制（Field name restrictions）**

  - 字段名不能是Python保留的关键字

```
1.class Example(models.Model):
2.    pass = models.IntegerField() # 'pass' is a reserved word!
```

  - 字段名不能包含双下划线

```
1.class Example(models.Model):
2.    foo__bar = models.IntegerField() # 'foo__bar' has two underscores!
```

Model介绍——CURD

- ORM(对象关系映射)详解
  - 新增数据

```
Group.objects.create(name="group1", leader="wang")
Group(name="group2", leader="li").save()
```

```
Business.objects.create(
    name="biz1",
    creator="page",
    members=8,
    group=Group.objects.get(id=1))
```

  - 更新数据

```
Group.objects.filter(name="group1").update(leader="chen")
```

  - 删除数据

```
Group.objects.filter(name="group2").delete()
```

Model介绍——CURD

- ORM(对象关系映射)详解
  - **get**（不存在或存在多项满足查询条件的数据，都会报错）

```
Group.objects.get(name="group1", leader="chen")
```

  - **filter (** 返回过滤后对象 **)**

```
Group.objects.filter(name="group1", leader="chen")
```

  - **all (** 返回全部对象 **)**

```
Group.objects.all()
```

  - 支持级联

```
Group.objects.filter(name="group1").filter(leader="chen")
```

Model介绍——CURD

- ORM(对象关系映射)详解
  - **__contains**（包含）
  - **__startswith、__endswith** （以…开头、结尾）
  - **__gt、__lt (** 大于、小于 **)**
  - **__range (** 两者之间 **)**
  - **fkey__field （** 外键查询 **)**

```
Group.objects.filter(name__contains="group")
```

```
Business.objects.filter(group__name__endswith="1")
```

Model介绍

- 组织创建举例

```
# ajax 请求
$.ajax({
    'url': '/organization/add/',
    'type': 'POST',
    'data': {
        'data': create_data
    },
    success: function (data) {
        do_something();
    }
});

# url 配置
url(r'^organization/add/$', ’organization_add',name=’organization_add'),


# 后台创建代码
def organization_add(request):
    create_data = request.POST.get('data')
    Organization.objects.create(**create_data)
```

步骤说明

- 前端ajax请求到后台
- 后台获取到创建的数据
- create(**create_data)

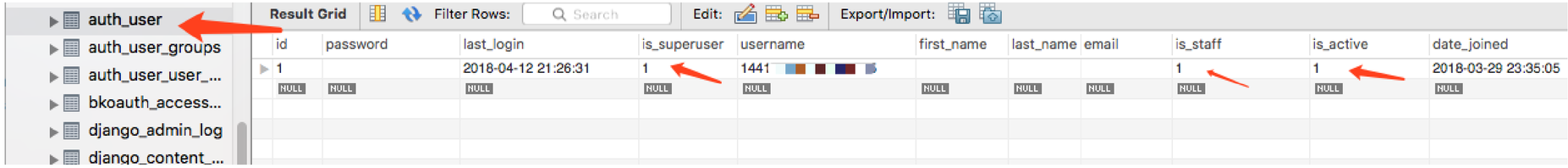
二、Admin管理后台

Admin后台管理

Philosophy

Generating admin sites for your staff or clients to add, change and delete content is tedious work that doesn't require much creativity. For that reason, Django entirely automates creation of admin interfaces for models.

- 创建管理员角色
  - 通过 MYSQL 管理端添加超级权限



Admin后台管理

- 在 admin.py 文件中将 model 注册到 admin

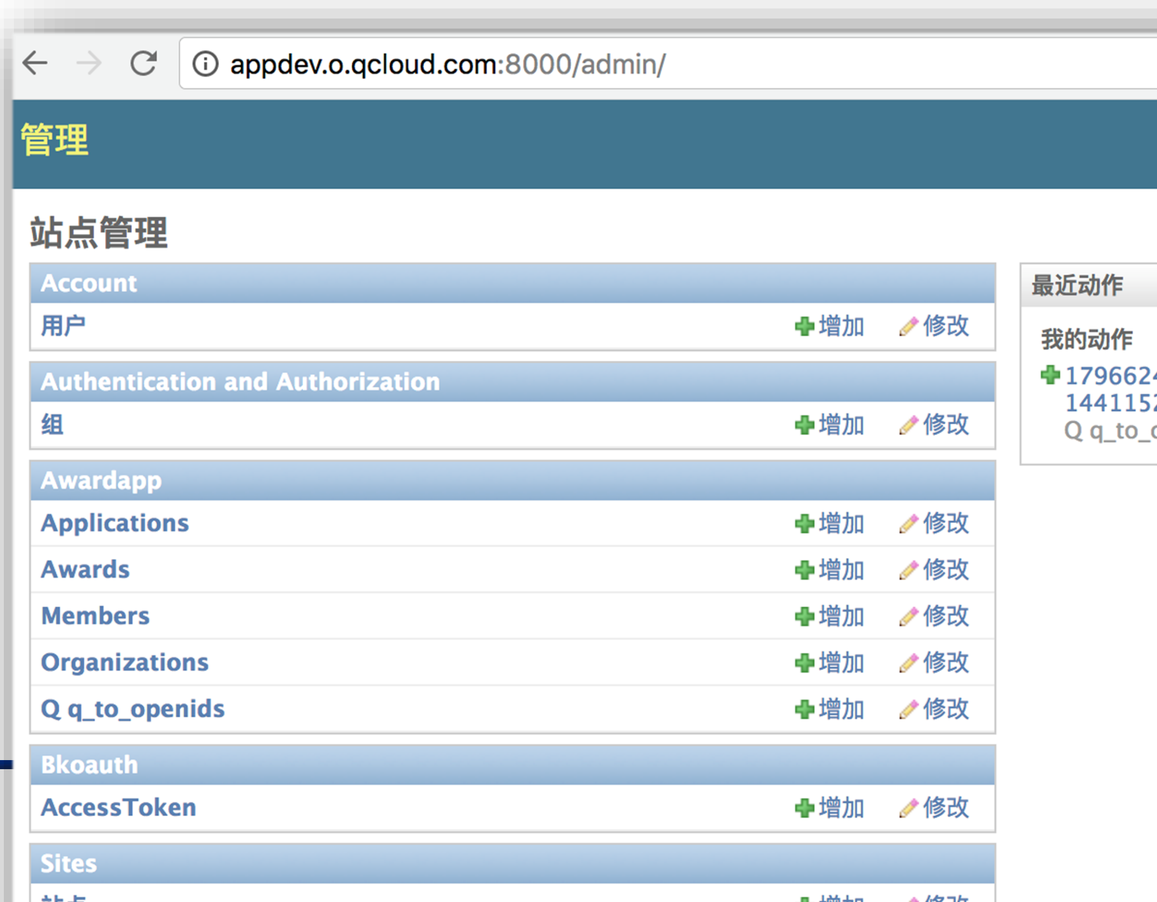
```
from django.contrib import admin

# Register your models here.

from home_application import models

admin.site.register(models.Organization)
admin.site.register(models.Member)
admin.site.register(models.Award)
admin.site.register(models.Application)
```

More info：https://docs.djangoproject.com/en/1.8/ref/contrib/admin/



三、文件上传与下载

文件上传与下载——COS存储

- **COS是什么？**

对象存储服务（Cloud Object Storage，COS）是面向企业和个人开发者提供的高可用，高稳定，强安全的云端存储服务。可将任意数量和形式的非结构化数据放入COS，并在其中实现数据的管理和处理。简单理解就是用来存东西的，包括存储图片、文件等

More info：https://cloud.tencent.com/document/product/436

文件上传与下载——COS存储

- **COS怎么用**

COS的具体使用细节我们已经封装到cos.py文件中，大家可以直接下载下来使用，需要了解里面的FileHandler对象怎么使用，步骤如下：

- **使用步骤**

1. 在requirements文件中添加qcloud_cos_v4包
2. 在settings文件中配置好COS相关的设置
3. 将cos.py文件放到自己的工程下
4. 创建FileHandler对象
5. 使用刚创建的FileHandler对象的upload_file方法上传文件。

```
# cos.py

# 初始化FileHandler对象
def __init__(self, view_request, file_obj):

# 判断文件夹是否在cos中存在
def is_exist(self, folder):


# 判断文件夹是否在cos中存在
def create_folder(self):


# 上传文件 ☆
def upload_file(self, folder, file_type):
```

文件上传

- **HTML**

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title></title>
</head>
<body>

<form enctype="multipart/form-data" method="POST" action="${SITE_URL}uploads/">
    <input type="file" name="file"/><br/>
    <input type="submit" value="上传文件"/>
</form>

</body>
</html>
```

Moreinfo: https://docs.djangoproject.com/en/1.8/topics/http/file-uploads/

文件上传

- 后端处理逻辑

```
def uploads(request):
    file_upload_obj = FileHandler(request, request.FILES[‘file’])
    ret = file_upload_obj.upload_file(u‘/media/’, ‘.jpg$’)
    if not ret[‘status’]:
        return HttpResponse("File uploads error")

    UploadFile.objects.create(name=request.FILES['file'].name,
                              file_url=ret['data']['file_path'])

    return redirect("test_model_app.views.index")
```

- Model定义

```
class UploadFile(models.Model):
    name = models.CharField(u"文件名", max_length=255)
    file_url = models.TextField(u"文件url")

    def __unicode__(self):
        return self.name
```

文件下载

文件下载主要是页面上的代码，后台不需要写代码处理逻辑。我们将cos上的提供url展示到页面上，当成一个超链接，用户可以直接点击链接进行文件下载。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title></title>
</head>
<body>

<li><a href="http://award-1251020309.cosgz.myqcloud.com/media/xxx.jpg">xxx.jpg</a></li>


</body>
</html>
```

Moreinfo: https://docs.djangoproject.com/en/1.8/topics/http/file-uploads/

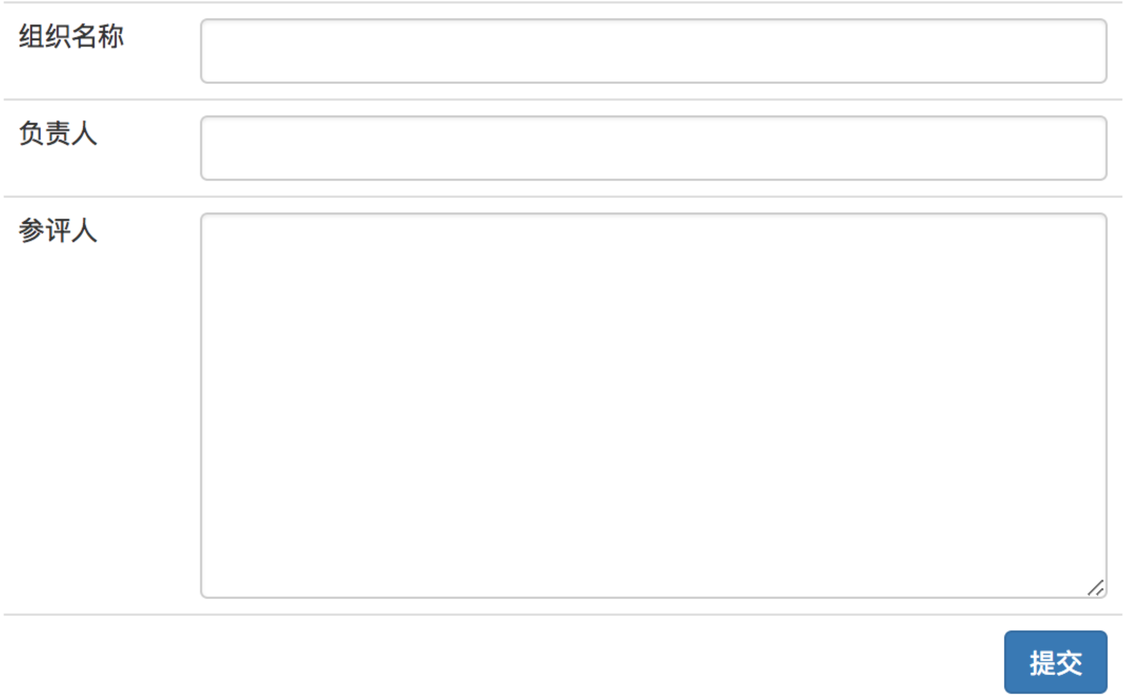
Django基础（二）课后作业

课后作业

- 基础model设计（组织、奖项、申报记录）
- model admin 配置
- 组织信息 CUR 接口
- 奖项 CUR 接口
- 申报记录 CU 接口

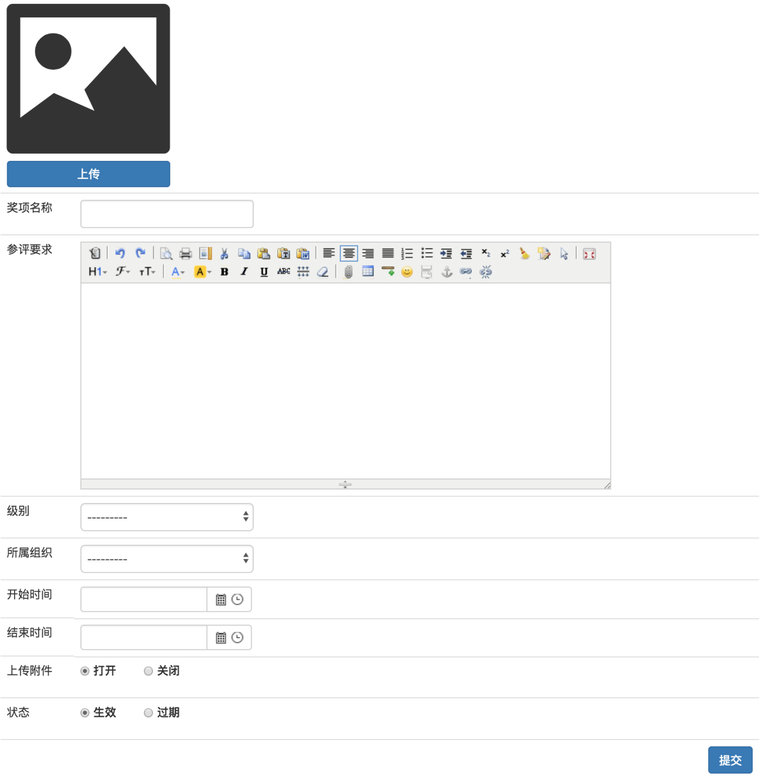
课后作业

- 组织信息新增



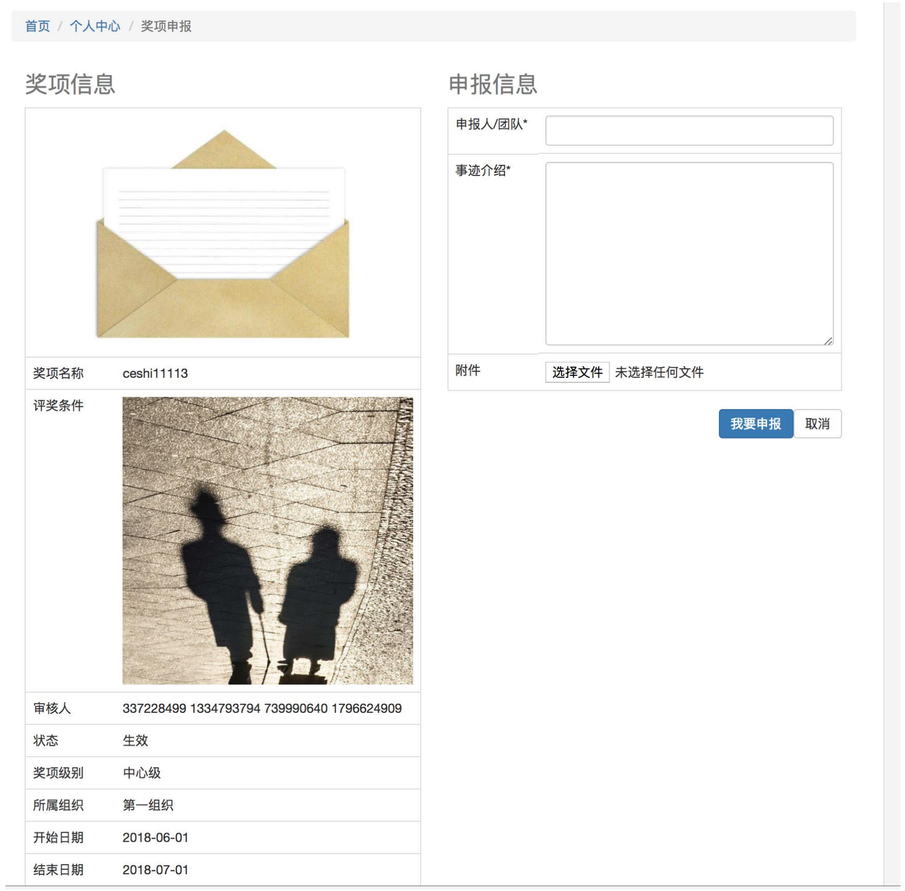
课后作业

- 奖项信息新增（支持图片上传）



课后作业

- 奖项申报页面（支持附件上传）



下集预告

Django基础（三）

- Template模板渲染机制
- 第三方模板引擎Mako语法
- 表格排序分页
- logger日志记录
- 实例演示更新&删除数据

THANK YOU