M:\My-Documents\programming\java\source\Python Django Practise\mysite\course  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*admin.py\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
from django.contrib import admin  
  
# Register your models here.  
  
  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*apps.py\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
from django.apps import AppConfig  
  
  
class CourseConfig(AppConfig):  
 name = 'course'  
  
  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*fields.py\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
from django.db import models  
from django.core.exceptions import ObjectDoesNotExist  
  
class OrderField(models.PositiveIntegerField):   
 def \_\_init\_\_(self, for\_fields=None, \*args, \*\*kwargs):  
 self.for\_fields = for\_fields  
 super(OrderField, self).\_\_init\_\_(\*args, \*\*kwargs)  
  
 def pre\_save(self, model\_instance, add):   
 if getattr(model\_instance, self.attname) is None:   
 try:  
 qs = self.model.objects.all()   
 if self.for\_fields:  
 query = {field: getattr(model\_instance, field) for field in self.for\_fields}   
 qs = qs.filter(\*\*query)   
 last\_item = qs.latest(self.attname)   
 value = last\_item.order + 1   
 except ObjectDoesNotExist:  
 value = 0  
 setattr(model\_instance, self.attname, value)   
 return value  
 else:  
 return super(OrderField, self).pre\_save(model\_instance, add)  
  
  
  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*forms.py\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
from django import forms  
from .models import Course, Lesson  
  
class CreateCourseForm(forms.ModelForm):  
 class Meta:  
 model = Course  
 fields = ("title", "overview")  
  
class CreateLessonForm(forms.ModelForm):  
 class Meta:  
 model = Lesson  
 fields = ['course', 'title', 'video', 'description', 'attach']  
   
 def \_\_init\_\_(self, user, \*args, \*\*kwargs):   
 super(CreateLessonForm, self).\_\_init\_\_(\*args, \*\*kwargs)  
 self.fields['course'].queryset = Course.objects.filter(user=user)   
  
  
  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*models.py\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
from django.db import models  
from django.contrib.auth.models import User  
from .fields import OrderField  
  
from slugify import slugify  
  
class Course(models.Model):  
 user = models.ForeignKey(User, related\_name='courses\_user')  
 title = models.CharField(max\_length=200)  
 slug = models.SlugField(max\_length=200, unique=True)  
 overview = models.TextField()  
 created = models.DateTimeField(auto\_now\_add=True)  
 student = models.ManyToManyField(User, related\_name="courses\_joined", blank=True)  
  
 class Meta:  
 ordering = ('-created',)  
  
 def save(self, \*args, \*\*kargs):  
 self.slug = slugify(self.title)  
 super(Course, self).save(\*args, \*\*kargs)  
  
 def \_\_str\_\_(self):  
 return self.title  
  
  
def user\_directory\_path(instance, filename):  
 return "courses/user\_{0}/{1}".format(instance.user.id, filename)  
  
class Lesson(models.Model):  
 user = models.ForeignKey(User, related\_name='lesson\_user')  
 course = models.ForeignKey(Course, related\_name='lesson')  
 title = models.CharField(max\_length=200)  
 video = models.FileField(upload\_to=user\_directory\_path)  
 description = models.TextField(blank=True)  
 attach = models.FileField(blank=True, upload\_to=user\_directory\_path)  
 created = models.DateTimeField(auto\_now\_add=True)  
 order = OrderField(blank=True, for\_fields=['course'])  
  
 class Meta:  
 ordering = ['order']   
  
 def \_\_str\_\_(self):  
 return '{}.{}'.format(self.order, self.title)  
  
  
  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*tests.py\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
from django.test import TestCase  
  
# Create your tests here.  
  
  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*urls.py\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
from django.conf.urls import url  
from django.views.generic import TemplateView   
from .views import AboutView, CourseListView, ManageCourseListView, CreateCourseView, DeleteCourseView, CreateLessonView, ListLessonsView, DetailLessonView  
from .views import StudentListLessonView  
  
urlpatterns = [  
 #url(r'about/$', TemplateView.as\_view(template\_name="course/about.html")),   
 url(r'about/$', AboutView.as\_view(), name="about"),   
 url(r'course-list/$', CourseListView.as\_view(), name="course\_list"),  
 url(r'manage-course/$', ManageCourseListView.as\_view(), name="manage\_course"),  
 url(r'create-course/$', CreateCourseView.as\_view(), name="create\_course"),  
 url(r'delete-course/(?P<pk>\d+)/$', DeleteCourseView.as\_view(), name="delete\_course"),  
 url(r'create-lesson/$', CreateLessonView.as\_view(), name="create\_lesson"),  
 url(r'list-lessons/(?P<course\_id>\d+)/$', ListLessonsView.as\_view(), name="list\_lessons"),  
 url(r'detail-lesson/(?P<lesson\_id>\d+)/$', DetailLessonView.as\_view(), name="detail\_lesson"),  
 url(r'lessons-list/(?P<course\_id>\d+)/$', StudentListLessonView.as\_view(), name="lessons\_list"),  
]  
  
  
  
  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*views.py\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
from django.views.generic import TemplateView, ListView, DeleteView  
from django.core.urlresolvers import reverse\_lazy  
from django.views.generic.edit import CreateView  
from django.shortcuts import redirect, render, get\_object\_or\_404, render\_to\_response  
from django.views import View   
from django.views.generic.base import TemplateResponseMixin   
from django.http import HttpResponse  
  
from .forms import CreateCourseForm, CreateLessonForm  
from .models import Course, Lesson   
  
from braces.views import LoginRequiredMixin  
  
class AboutView(TemplateView):  
 template\_name = "course/about.html"  
  
  
class CourseListView(ListView):   
 model = Course   
 context\_object\_name = "courses"   
 template\_name = 'course/course\_list.html'   
  
  
  
  
class UserMixin:   
 def get\_queryset(self):  
 qs = super(UserMixin, self).get\_queryset()  
 return qs.filter(user=self.request.user)  
  
class UserCourseMixin(UserMixin, LoginRequiredMixin):  
 model = Course  
 login\_url = "/account/login/"   
  
  
  
class ManageCourseListView(UserCourseMixin, ListView):   
 context\_object\_name = "courses"   
 template\_name = 'course/manage/manage\_course\_list.html'  
  
  
  
class CreateCourseView(UserCourseMixin, CreateView):   
 fields = ['title', 'overview']   
 template\_name = 'course/manage/create\_course.html'  
  
 def post(self, request, \*args, \*\*kargs):   
 form = CreateCourseForm(data=request.POST)  
 if form.is\_valid():  
 new\_course = form.save(commit=False)  
 new\_course.user = self.request.user  
 new\_course.save()  
 return redirect("course:manage\_course")   
 return self.render\_to\_response({"form":form})  
  
class DeleteCourseView(UserCourseMixin, DeleteView):   
 template\_name = 'course/manage/delete\_course\_confirm.html'  
 success\_url = reverse\_lazy("course:manage\_course")  
  
 def dispatch(self, \*args, \*\*kwargs):   
 resp = super(DeleteCourseView, self).dispatch(\*args, \*\*kwargs)   
 if self.request.is\_ajax():   
 response\_data = {"result": "ok"}  
 return HttpResponse(json.dumps(response\_data), content\_type="application/json")  
 else:  
 return resp  
  
  
  
class CreateLessonView(LoginRequiredMixin, View):   
 model = Lesson  
 login\_url = "/account/login/"  
  
 def get(self, request , \*args, \*\*kwargs):   
 form = CreateLessonForm(user=self.request.user)   
 return render(request, "course/manage/create\_lesson.html", {"form":form})  
  
 def post(self, request, \*args, \*\*kwargs):   
 form = CreateLessonForm(self.request.user, request.POST, request.FILES)   
 if form.is\_valid():  
 new\_lesson = form.save(commit=False)  
 new\_lesson.user = self.request.user  
 new\_lesson.save()  
 return redirect("course:manage\_course")  
  
  
class ListLessonsView(LoginRequiredMixin, TemplateResponseMixin, View):  
 login\_url = "/account/login/"  
 template\_name = 'course/manage/list\_lessons.html'   
  
 def get(self, request, course\_id):   
 course = get\_object\_or\_404(Course, id=course\_id)   
 return self.render\_to\_response({'course':course})   
  
  
class DetailLessonView(LoginRequiredMixin, TemplateResponseMixin, View):  
 login\_url = "/account/login/"  
 template\_name = "course/manage/detail\_lesson.html"  
  
 def get(self, request, lesson\_id):  
 lesson = get\_object\_or\_404(Lesson, id=lesson\_id)  
 return self.render\_to\_response({"lesson":lesson})  
  
class StudentListLessonView(ListLessonsView):   
 template\_name = "course/slist\_lessons.html"   
  
 def post(self, request, \*args, \*\*kwargs):  
 course = Course.objects.get(id=kwargs['course\_id'])  
 course.student.add(self.request.user)  
 return HttpResponse("ok")