

课程按分类进行过滤显示

安装过滤插件

```
pip install django-filter
```

后端配置

```
# settings.py

INSTALLED_APPS = [
    ....
    'django_filters',
    ...
]

REST_FRAMEWORK = {
    ...
    # 过滤组件注册
    'DEFAULT_FILTER_BACKENDS': ('django_filters.rest_framework.DjangoFilterBackend',)
}
```

视图中设置允许进行过滤的字段

courses/views.py

```
from .models import Course
from .serializers import CourseSerializer
class CourseAPIView(ListAPIView):
    queryset = Course.objects.filter(status=0).order_by("-orders", "-students")
    serializer_class = CourseSerializer
    # 设置过滤的字段
    filter_fields = ('course_category',)
```

客户端实现请求根据分类过滤显示课程

组件html代码:

```

<div class="filter">
  <!-- 注意,新增了一个空的div标签包裹分类的内容 -->
  <div @click="get_course">
    <el-row class="filter-el-row1">
      <el-col :span="2" class="filter-text">课程分类:</el-col>
      <el-col :span="2" :data-id="0" class="current">全部</el-col>
      <el-col :span="2" :data-id="item.id" v-for="item in cate_list">
{{item.name}}</el-col>
    </el-row>
  </div>

```

组件js代码:

```

methods:{
  addclass(ele,className){
    let class_array = ele.className.split(" ");
    // 判断元素本身是否已经有了一个同名的class, indexOf返回查询出来的成员的下标
    let index = class_array.indexOf(className);
    // 如果返回值是-1,则表示找不到同名的class
    if(index===-1){
      ele.className+=" "+className
    }
  },
  removeclass(ele,className){
    // 去除元素的class样式
    // 回头修复bug,类似 <div class="col-2 current_activate">
    ele.className = ele.className.replace("current","");
  },
  get_course(event){
    // 给当前被点击的分类元素添加外框高亮
    let ele_list = event.target.parentElement.children;
    for(let i = 0;i<ele_list.length;i++){
      this.removeclass(ele_list[i],"current")
    }
    this.addclass(event.target,"current");
    // 获取标签元素的属性,分类ID
    let id = event.target.getAttribute("data-id");
    let url = "http://127.0.0.1:8000/courses/";
    // 如果分类ID不是0,则表示按分类显示课程
    if(id!=0){
      url+="?course_category="+id
    }
    this.$axios.get(url).then(response=>{
      this.course_list = response.data
    }).catch(error=>{
      console.log(error.response);
    })
  }
},

```

排序显示课程

后端提供排序课程的接口

把原来views.py中的CoursesAPIView新增两句代码:

```
from .models import Course
from .serializers import CourseSerializer
from rest_framework.filters import OrderingFilter
class CourseAPIView(ListAPIView):
    queryset = Course.objects.filter(status=0).order_by("-orders", "-students")
    serializer_class = CourseSerializer
    # 设置过滤的字段
    filter_fields = ('course_category',)
    filter_backends = [OrderingFilter]
    ordering_fields = ('id', 'students', 'price')
```

前端根据排序字段对应的课程顺序

组件html代码

[illegible]

组件js代码;

```
methods:{
  orders(field,type){
    let url = "http://127.0.0.1:8000/courses/";
    if(field=='students'){
      url+="?ordering=-students";
    }else if(field="price"){
      if(type=="+"){
```

```
url+="?ordering=price";
    }else{
        url+="?ordering=-price";
    }
}

this.$axios.get(url).then(response=>{
    this.course_list = response.data
}).catch(error=>{
    console.log(error.response);
})
},
```

分页显示数据

```
from rest_framework.pagination import PageNumberPagination

class StandardPageNumberPagination(PageNumberPagination):
    page_size_query_param = 'page_size'
    max_page_size = 1

class CourseAPIView(ListAPIView):
    queryset = Course.objects.filter(status=0).order_by("-orders", "-students")
    # 设置过滤的字段
    filter_fields = ('course_category',)
    serializer_class = CourseSerializer
    filter_backends = [OrderingFilter]
    ordering_fields = ('id', 'students', 'price', 'course_category')
    pagination_class = StandardPageNumberPagination
```

客户端请求后端发送数据

[illegible]

```

        <el-col :span="2" class="current"><span @click="orders('id','')">默认
</span></el-col>
        <el-col :span="2"><span @click="orders('students','')">人气</span></el-
col>

        <el-col :span="2">
            <span @click="orders('price','+')">价格</span>
            <div class="filter-price">
                <span class="up" @click="orders('price','+')"
:~class="filter_price==true?'active':''"><i class="el-icon-caret-top"></i></span>
                <span class="down" @click="orders('price','-')"
:~class="filter_price==false?'active':''"><i class="el-icon-caret-bottom"></i></span>
            </div>
        </el-col>
    </el-row>
</div>
<div class="courses_list">
    <el-row v-for="course in course_list" class="course-item">
        <el-col :span="24" class="course-item-box">
            <el-row>
                <el-col :span="12" class="course-item-left"></el-col>
                <el-col :span="12" class="course-item-right">
                    <div class="course-title">
                        <p class="box-title">{{course.name}}</p>
                        <p class="box-number">{{course.students}}人已加入学习</p>
                    </div>
                    <div class="author">
                        <p class="box-author">{{course.teacher.name}}
{{course.teacher.title}}</p>
                        <p class="lesson">共{{course.lessons}}课时
<span>/{{course.pub_lessons==course.lessons?'更新完成':course.pub_lessons}}</span></p>
                    </div>
                    <el-row class="course-content">
                        <el-col :span="12"><i class="el-icon-caret-right"></i>01 |
常用模块学习-模块的种类和... <span class="free">免费</span> </el-col>
                        <el-col :span="12"><i class="el-icon-caret-right"></i>02 |
常用模块学习-模块的种类和... <span class="free">免费</span> </el-col>
                        <el-col :span="12"><i class="el-icon-caret-right"></i>03 |
三元运算符... <span class="free">免费</span> </el-col>
                        <el-col :span="12"><i class="el-icon-caret-right"></i>04 |
常用模块学习-模块的种类和... <span class="free">免费</span> </el-col>
                    </el-row>
                    <div class="course-price">
                        <p class="course-price-left">
                            <span class="discount">限时免费</span>
                            <span class="count">¥0.00</span>
                            <span class="old_count">原价: ¥{{course.price}}元</span>
                        </p>
                        <button class="buy">立即购买</button>
                    </div>
                </el-col>
            </el-row>
        </el-col>
    </div>

```

```

        </el-row>
      </div>
    <el-pagination
      @current-change="handleCurrentChange"
      background
      layout="prev, pager, next"
      :page-size="page_size"
      :total="total">
    </el-pagination>
  </div>
  <Footer/>
</div>
</template>

<script>
import Header from "../common/Header"
import Footer from "../common/Footer"
export default {
  name: "Courses",
  data() {
    return {
      current_page: 1,
      filter_price: false,
      cate_list: [], // 课程分类列表
      course_list: [], // 专题课程列表
      page_size: 1, // 默认每一页显示的数据量
      total: 10,
    }
  },
  components: {
    Header,
    Footer,
  },
  methods: {
    handleCurrentChange(val) {
      // 名字必须固定
      let url = "http://127.0.0.1:8000/courses/";
      url += "?page=" + val + "&page_size=" + this.page_size;
      this.$axios.get(url).then(response => {
        this.course_list = response.data.results;
        this.curser_page_cout = response.data.count
      }).catch(error => {
        console.log(error.response);
      })
    },
    orders(field, type) {
      let url = "http://127.0.0.1:8000/courses/";
      if (field === 'students') {
        url += "?ordering=-students";
      } else if (field === "price") {
        if (type === "+") {
          url += "?ordering=price";
        } else {

```

```

        url+="?ordering=-price";
    }
}
this.$axios.get(url).then(response=>{
    this.course_list = response.data
}).catch(error=>{
    console.log(error.response);
})
},
addClass(ele,className){
    let class_array = ele.className.split(" ");
    // 判断元素本身是否已经有了一个同名的class,    indexOf返回查询出来的成员的下标
    let index = class_array.indexOf(className);
    // 如果返回值是-1,则表示找不到同名的class
    if(index===-1){
        ele.className+=" "+className
    }
},
removeClass(ele,className){
    // 去除元素的class样式
    // 回头修复bug,类似 <div class="col-2 current_activate ">
    ele.className = ele.className.replace("current", "");
},
get_course(event){
    // 给当前被点击的分类元素添加外框高亮
    let ele_list = event.target.parentElement.children;
    for(let i = 0;i<ele_list.length;i++){
        this.removeClass(ele_list[i], "current")
    }
    this.addClass(event.target, "current");
    // 获取标签元素的属性,分类ID
    let id = event.target.getAttribute("data-id");
    let url = "http://127.0.0.1:8000/courses/";
    // 如果分类ID不是0,则表示按分类显示课程
    if(id!=0){
        url+="?course_category="+id
    }
    this.$axios.get(url).then(response=>{
        this.course_list = response.data
    }).catch(error=>{
        console.log(error.response);
    })
}
},
created(){
    // 获取课程分类信息
    this.$axios.get(`http://127.0.0.1:8000/courses/cate/`).then(response=>{
        this.cate_list = response.data
    }).catch(error=>{
        console.log(error.response);
    });
}

```

```

        // 获取课程信息
        this.$axios.get(`http://127.0.0.1:8000/courses/?
page=${this.current_page}&page_size=${this.page_size}`).then(response=>{
            this.course_list = response.data.results
            this.total = response.data.count
        }).catch(error=>{
            console.log(error.response);
        });
    }
}
</script>

```

```

<style scoped>
.courses{
    padding-top: 80px;
}
.main{
    width: 1100px;
    height: auto;
    margin: 0 auto;
    padding-top: 35px;
}
.main .filter{
    width: 100%;
    height: auto;
    margin-bottom: 35px;
    padding: 25px 0px 25px 0px;
    background: #fff;
    border-radius: 4px;
    box-shadow: 0 2px 4px 0 #f0f0f0;
}
.filter .el-col{
    text-align: center;
    padding: 6px 0px;
    line-height: 16px;
    margin-left: 14px;
    position: relative;
    transition: all .3s ease;
    cursor: pointer;
    color: #4a4a4a;
}
.filter-el-row1{
    padding-bottom: 18px;
    margin-bottom: 17px;
}
.filter .filter-text{
    text-align: right;
    font-size: 16px;
    color: #888;
}
.filter .filter-text2{

```



```

}
.filter .filter-el-row1 .current{
    color: #ffc210;
    border: 1px solid #ffc210!important;
    border-radius: 30px;
}
.filter .filter-el-row2 .current{
    color: #ffc210;
}
.filter-price{
    display:inline-block;
    vertical-align: middle;
}
.filter-price .up, .filter-price .down{
    display: block;
    line-height: 8px;
    font-size: 13px;
    margin: -4px;
    color: #d8d8d8;
}
.current .filter-price .active{
    color: #ffc210;
}
.course-item{
    margin-bottom: 35px;
}
.course-item .course-item-box{
    padding: 20px 30px 20px 20px;
}
.course-item{
    box-shadow: 2px 3px 16px rgba(0,0,0,.1);
    transition: all .2s ease;
}
.course-item .course-item-left{
    width: 423px;
    height: 210px;
    margin-right: 30px;
}
.course-title{
    overflow: hidden; /* 在父元素中使用可以清除子元素的浮动影响 */
}
.course-title .box-title{
    font-size: 26px;
    color: #333333;
    float: left;
    margin-bottom: 8px;
}
.course-title .box-number{
    font-size: 14px;
    color: #9b9b9b;
    font-family: PingFangSC-Light;
    float: right;
    padding-top: 12px;
}

```

```
}
.course-item-right{
  width: 56.6%;
}
.author{
  font-size: 14px;
  color: #9b9b9b;
  margin-bottom: 14px;
  padding-bottom: 14px;
  overflow: hidden;
}
.author .box-author{
  float:left;
}
.author .lesson{
  float: right;
}
.course-content .el-icon-caret-right{
  border: 1px solid #000;
  border-radius: 50%;
  margin-right: 6px;
}
.course-content .el-col{
  font-size: 14px;
  color: #666;
  width: 50%;
  margin-bottom: 15px;
  cursor: pointer;
}
.course-content .el-col:hover{
  color: #ffc210;
}
.course-content .el-col:hover .el-icon-caret-right, .course-content .el-col:hover .free{
  border-color: #ffc210;
  color: #ffc210;
}
.course-content .el-col .free{
  width: 34px;
  height: 20px;
  color: #fd7b4d;
  margin-left: 10px;
  border: 1px solid #fd7b4d;
  border-radius: 2px;
  text-align: center;
  font-size: 13px;
  white-space: nowrap;
}
.course-price{
  overflow: hidden;
}
.course-price .course-price-left{
  float: left;
}
```

```

.course-price .discount{
    padding: 6px 10px;
    display: inline-block;
    font-size: 16px;
    color: #fff;
    text-align: center;
    margin-right: 8px;
    background: #fa6240;
    border: 1px solid #fa6240;
    border-radius: 10px 0 10px 0;
}
.course-price .course-price-left{
    line-height: 22px;
}
.course-price .count{
    font-size: 24px;
    color: #fa6240;
}
.course-price .old_count{
    font-size: 14px;
    color: #9b9b9b;
    text-decoration: line-through;
    margin-left: 10px;
}
.course-price .buy{
    float: right;
    width: 120px;
    height: 38px;
    font-size: 16px;
    border-radius: 3px;
    border: 1px solid #fd7b4d;
    background: transparent; /* 透明 */
    color: #fa6240;
    cursor: pointer;
    transition: all .2s ease-in-out; /* css3新版本的样式中支持支持 jQuery里面的动画预设效果 */
    /* all表示当前元素的所有样式 .2s表示改变样式完成的时间 ease-in-out */
}
.course-price .buy:hover{
    color: #fff;
    background: #ffc210;
    border: 1px solid #ffc210;
}
</style>

```

客户端的课程列表显示当前课程的章节信息

对于显示课程以外的一些附加信息,需要在序列化器进行序列化器的时候调整.

```
from .models import Course
class CourseSerializer(serializers.ModelSerializer):
    # 这里调用的序列化器,必须事先在前面已经声明好的,否则报错
    teacher = TeacherSerializer()
    # 课程和课程章节的外键,因为需要获取具体的数据而不是ID,所以我们需要自定义序列化器
    coursechapters=CourseChaptersSerializer(many=True)
    class Meta:
        model= Course
        fields =
("coursechapters", "id", "name", "course_img", "students", "lessons", "pub_lessons", "price", "
teacher")
```

前端显示课程章节信息

```
<el-row class="course-content">
    <el-col v-for="chapters in course.coursechapters" :span="12"><i class="el-icon-
caret-right"></i>{{chapters.chapter<10?"0"+chapters.chapter:chapters.chapter}} |
{{chapters.name}} <span class="free">免费</span> </el-col>
    <!--<el-col :span="12"><i class="el-icon-caret-right"></i>02 | 常用模块学习-模块的种类
和... <span class="free">免费</span> </el-col>-->
    <!--<el-col :span="12"><i class="el-icon-caret-right"></i>03 | 三元运算符... <span
class="free">免费</span> </el-col>-->
    <!--<el-col :span="12"><i class="el-icon-caret-right"></i>04 | 常用模块学习-模块的种类
和... <span class="free">免费</span> </el-col>-->
</el-row>
```

显示当前课程所属的价格

价格策略模型

价格服务类型名称: 限时免费, 限时折扣, 限时减免, 积分抵扣, 满减
公式: 服务价格

限时免费	原价-原价
限时折扣	原价*0.8
限时减免	原价-减免价
积分抵扣	原价-(积分计算后换算价格)
满减	原价-(满减计算后换算价格)

模型代码:

```

from django.db import models

# Create your models here.
class CourseCategory(models.Model):
    """
    课程分类
    """
    name = models.CharField(max_length=64, unique=True, verbose_name="分类名称")
    orders = models.IntegerField(verbose_name="课程排序", null=True)
    is_show = models.BooleanField(default=True, verbose_name="是否上线")
    is_delete = models.BooleanField(default=False, verbose_name="逻辑删除")
    class Meta:
        db_table = "ly_course_category"
        verbose_name = "课程分类"
        verbose_name_plural = "课程分类"

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


class Course(models.Model):
    """
    专题课程
    """
    course_type = (
        (0, '付费'),
        (1, 'VIP专享'),
        (2, '学位课程')
    )
    level_choices = (
        (0, '初级'),
        (1, '中级'),
        (2, '高级')
    )
    status_choices = (
        (0, '上线'),
        (1, '下线'),
        (2, '预上线')
    )
    name = models.CharField(max_length=128, verbose_name="课程名称")
    course_img = models.ImageField(max_length=255, verbose_name="封面图片", blank=True,
null=True)
    course_type = models.SmallIntegerField(choices=course_type, verbose_name="付费类型")
    # 使用这个字段的原因
    brief = models.TextField(max_length=2048, verbose_name="课程概述", null=True,
blank=True)
    level = models.SmallIntegerField(choices=level_choices, default=1, verbose_name="难
度等级划分")
    pub_date = models.DateField(verbose_name="发布日期", auto_now_add=True)
    period = models.IntegerField(verbose_name="建议学习周期(day)", default=7)
    orders = models.IntegerField(verbose_name="课程排序")

```

```

        attachment_path = models.FileField(max_length=128, verbose_name="课件路径",
blank=True, null=True)
        status = models.SmallIntegerField(choices=status_choices, default=0,
verbose_name="课程状态")
        course_category = models.ForeignKey("CourseCategory", on_delete=models.CASCADE,
null=True, blank=True, verbose_name="课程分类")
        students = models.IntegerField(verbose_name="学习人数", default = 0)
        lessons = models.IntegerField(verbose_name="总课时数量", default = 0)
        pub_lessons = models.IntegerField(verbose_name="课时更新数量", default = 0)
        price = models.DecimalField(max_digits=6, decimal_places=2, verbose_name="课程原
价", default=0)
        teacher = models.ForeignKey("Teacher", on_delete=models.DO_NOTHING, null=True,
blank=True, verbose_name="授课老师")
        price_service_type =
models.ForeignKey("PriceServiceType", on_delete=models.DO_NOTHING, null=True,
blank=True, verbose_name="价格服务")
        class Meta:
            db_table = "ly_course"
            verbose_name = "专题课程"
            verbose_name_plural = "专题课程"

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

        # 自定义显示字段
        # def cate_name(self):
        #     return self.course_category.name

class Teacher(models.Model):
    """讲师、导师表"""
    role_choices = (
        (0, '讲师'),
        (1, '导师'),
    )
    name = models.CharField(max_length=32, verbose_name="讲师title")
    role = models.SmallIntegerField(choices=role_choices, default=0, verbose_name="讲师
身份")
    title = models.CharField(max_length=64, verbose_name="职位、职称")
    signature = models.CharField(max_length=255, help_text="导师签名", blank=True,
null=True)
    image = models.ImageField(blank=True, verbose_name = "讲师封面")
    brief = models.TextField(max_length=1024, verbose_name="讲师描述")

    class Meta:
        db_table = "ly_teacher"
        verbose_name = "讲师导师"
        verbose_name_plural = "讲师导师"

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

class CourseChapter(models.Model):

```

```

    """课程章节"""
    course = models.ForeignKey("Course", related_name='coursechapters',
on_delete=models.CASCADE, verbose_name="课程名称")
    chapter = models.SmallIntegerField(verbose_name="第几章", default=1)
    name = models.CharField(max_length=128, verbose_name="章节标题")
    summary = models.TextField(verbose_name="章节介绍", blank=True, null=True)
    pub_date = models.DateField(verbose_name="发布日期", auto_now_add=True)

    class Meta:
        db_table = "ly_course_chapter"
        verbose_name = "课程章节"
        verbose_name_plural = "课程章节"

    def __str__(self):
        return "%s:(第%s章)%s" % (self.course, self.chapter, self.name)

class CourseLesson(models.Model):
    """课程课时"""
    section_type_choices = (
        (0, '文档'),
        (1, '练习'),
        (2, '视频')
    )
    chapter = models.ForeignKey("CourseChapter", related_name='coursesections',
on_delete=models.CASCADE, verbose_name="课程章节")
    name = models.CharField(max_length=128, verbose_name="课时标题")
    orders = models.PositiveSmallIntegerField(verbose_name="课时排序")
    section_type = models.SmallIntegerField(default=2, choices=section_type_choices,
verbose_name="课时种类")
    section_link = models.CharField(max_length=255, blank=True, null=True,
verbose_name="课时链接", help_text="若是video, 填vid,若是文档, 填link")
    duration = models.CharField(verbose_name="视频时长", blank=True, null=True,
max_length=32) # 仅在前端展示使用
    pub_date = models.DateTimeField(verbose_name="发布时间", auto_now_add=True)
    free_trail = models.BooleanField(verbose_name="是否可试看", default=False)

    class Meta:
        db_table = "ly_course_lesson"
        verbose_name = "课程课时"
        verbose_name_plural = "课程课时"

    def __str__(self):
        return "%s-%s" % (self.chapter, self.name)

class PriceServiceType(models.Model):
    """价格服务类型"""
    name = models.CharField(max_length=32, verbose_name="服务名称")

    class Meta:
        db_table = "ly_price_service_type"
        verbose_name = "价格服务"

```

```

verbose_name_plural = "价格服务"

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

class PriceService(models.Model):
    """价格服务策略"""
    services_type = models.ForeignKey("PriceServiceType", related_name='priceservices',
on_delete=models.CASCADE, verbose_name="服务名称")
    condition = models.IntegerField(verbose_name="满足优惠的价格条件")
    sale=models.CharField(max_length=32, verbose_name="优惠值", help_text="-1表示免费; <br>*号开头表示折扣价, 例如*0.82表示八二折; <br>$号开头表示积分兑换, 例如$50表示可以兑换50积分")
    start_time = models.DateTimeField(verbose_name="优惠策略的开始时间")
    end_time = models.DateTimeField(verbose_name="优惠策略的结束时间")

    class Meta:
        db_table = "ly_price_service"
        verbose_name = "价格服务策略"
        verbose_name_plural = "价格服务策略"

    def __str__(self):
        return "价格服务:%s, 优惠条件:%s, 优惠值:%s" %
(self.services_type.name, self.condition, self.sale)

```

序列化器,代码:

```

from rest_framework import serializers
from .models import CourseCategory
class CourseCategorySerializer(serializers.ModelSerializer):
    class Meta:
        model = CourseCategory
        fields = ("id", "name")

# 开发中一个序列化器 A 中需要同时序列化其他模型 B 的数据返回给客户端,那么直接通过外键默认只会返回主键ID
# 所以我们可以再通过创建一个模型B的序列化器,对模型B的数据进行序列化
# 在序列化器A中直接把模型B的序列化器调用作为字段值来声明即可。

from .models import Teacher
class TeacherSerializer(serializers.ModelSerializer):
    class Meta:
        model = Teacher
        fields = ("id", "name", "title")

from .models import CourseChapter
class CourseChaptersSerializer(serializers.ModelSerializer):
    class Meta:
        model= CourseChapter
        fields = ("id", "chapter", "name")
from .models import PriceService
class PriceServicesSerializer(serializers.ModelSerializer):
    """价格服务策略序列化器"""

```



```

class Meta:
    model = PriceService
    fields = ("condition", "sale", "start_time", "end_time")

from .models import PriceServiceType
class PriceServiceType(serializers.ModelSerializer):
    """价格服务类型序列化器"""
    priceservices = PriceServicesSerializer(many=True)
    class Meta:
        model = PriceServiceType
        # priceservices 价格策略
        fields = ("id", "name", "priceservices")

from .models import Course
class CourseSerializer(serializers.ModelSerializer):
    # 这里调用的序列化器,必须事先在前面已经声明好的,否则报错
    teacher = TeacherSerializer()
    # 课程和课程章节的外键,因为需要获取具体的数据而不是ID,所以我们需要自定义序列化器
    coursechapters=CourseChaptersSerializer(many=True)
    price_service_type=PriceServiceType()
    class Meta:
        model= Course
        fields =
("coursechapters", "id", "name", "course_img", "students", "lessons", "pub_lessons", "price", "
teacher", "price_service_type")

```

adminx.py

```

from .models import PriceServiceType
class PriceServiceTypeModelAdmin(object):
    """价格服务类型模型管理类"""
    pass
xadmin.site.register(PriceServiceType, PriceServiceTypeModelAdmin)

from .models import PriceService
class PriceServiceModelAdmin(object):
    """价格服务策略模型管理类"""
    pass
xadmin.site.register(PriceService, PriceServiceModelAdmin)

```

前端请求数据

```

<template>
  <div class="courses">
    <Header :current_page="current_page"/>
    <div class="main">
      <div class="filter">

```

[illegible]

```

        <span class="discount">
{{course.price_service_type.name}}</span>
        <span class="count">¥{{get_price(course)}}</span>
        <span class="old_count">原价: ¥{{course.price}}元
</span>

        </span>
        <span v-else class="count">¥{{course.price}}</span>
    </p>
    <button class="buy">立即购买</button>
</div>
</el-col>
</el-row>
</el-col>
</el-row>
</div>
<el-pagination
  @current-change="handleCurrentChange"
  background
  layout="prev, pager, next"
  :page-size="page_size"
  :total="total">
</el-pagination>
</div>
<Footer/>
</div>
</template>

<script>
import Header from "../common/Header"
import Footer from "../common/Footer"
export default {
  name: "Courses",
  data() {
    return {
      current_page: 1,
      filter_price: false,
      cate_list: [], // 课程分类列表
      course_list: [], // 专题课程列表
      page_size: 1, // 默认每一页显示的数据量
      total: 10,
    }
  },
  components: {
    Header,
    Footer,
  },
  methods: {
    // 计算价格
    get_price(course) {
      let price = 0;
      course.price = parseInt(course.price);
      // 根据不同的价格服务, 计算对应的价格
      let st = course.price_service_type; // 价格服务类型

```

```

    if(st != null && st.priceservices){ // 是否有设置了价格服务,没有设置价格服务的课程,服务
    为值null
        if(st.priceservices[0].condition>0){
            // 1. 优惠条件值大于0,则属于满减
            let list = st.priceservices; // 满减策略
            // 进行满减价格计算
            var real_sale = 0; // 保存满足条件的优惠值
            for(let i = 0;i<list.length;i++){
                list[i].condition = parseInt(list[i].condition);
                list[i].sale = parseInt(list[i].sale);
                if( course.price >= list[i].condition ){
                    // 后端返回的数字也是字符串,所以需要转换
                    if(real_sale<=list[i].sale){
                        real_sale = list[i].sale;
                    }
                }
            }
            console.log(real_sale);
            price = course.price - real_sale;
        }else{
            // 优惠条件值为0,则表示是限时折扣或者限时免费
            if(st.priceservices[0].sale=="-1"){
                // 2. 限时免费
                price = 0;

            }else{
                // 3. 限时折扣
                // 把优惠值sale中的*去掉
                let sale = st.priceservices[0].sale.replace("*","");
                price = course.price * parseFloat(sale)
            }
        }

    }else{
        price = course.price;
    }
    return price.toFixed(2);
},
handleCurrentChange(val){
    // 名字必须固定
    let url = "http://127.0.0.1:8000/courses/";
    url+="?page="+val+"&page_size="+this.page_size;
    this.$axios.get(url).then(response=>{
        this.course_list = response.data.results;
        this.curser_page_cout =response.data.count
    }).catch(error=>{
        console.log(error.response);
    })
},
orders(field,type){
    let url = "http://127.0.0.1:8000/courses/";
    if(field=='students'){
        url+="?ordering=-students";
    }
}

```

```

    }else if(field="price"){
      if(type=="+"){
        url+="?ordering=price";
      }else{
        url+="?ordering=-price";
      }
    }
  }
  this.$axios.get(url).then(response=>{
    this.course_list = response.data
  }).catch(error=>{
    console.log(error.response);
  })
},
addClass(ele,className){
  let class_array = ele.className.split(" ");
  // 判断元素本身是否已经有了一个同名的class,   indexOf返回查询出来的成员的下标
  let index = class_array.indexOf(className);
  // 如果返回值是-1,则表示找不到同名的class
  if(index===-1){
    ele.className+=" "+className
  }
},
removeClass(ele,className){
  // 去除元素的class样式
  // 回头修复下bug,类似 <div class="col-2 current_activate ">
  ele.className = ele.className.replace("current", "");
},
get_course(event){
  // 给当前被点击的分类元素添加外框高亮
  let ele_list = event.target.parentElement.children;
  for(let i = 0;i<ele_list.length;i++){
    this.removeClass(ele_list[i], "current")
  }
  this.addClass(event.target, "current");
  // 获取标签元素的属性,分类ID
  let id = event.target.getAttribute("data-id");
  let url = "http://127.0.0.1:8000/courses/";
  // 如果分类ID不是0,则表示按分类显示课程
  if(id!=0){
    url+="?course_category="+id
  }
  this.$axios.get(url).then(response=>{
    this.course_list = response.data
  }).catch(error=>{
    console.log(error.response);
  })
}
},
created(){
  // 获取课程分类信息
  this.$axios.get(`http://127.0.0.1:8000/courses/cate/`).then(response=>{
    this.cate_list = response.data
  }).catch(error=>{

```

```

        console.log(error.response);
    });

    // 获取课程信息
    this.$axios.get(`http://127.0.0.1:8000/courses/?
page=${this.current_page}&page_size=${this.page_size}`).then(response=>{
        this.course_list = response.data.results
        this.total = response.data.count
    }).catch(error=>{
        console.log(error.response);
    });

    }
}
</script>

```

```

<style scoped>
.courses{
    padding-top: 80px;
}
.main{
    width: 1100px;
    height: auto;
    margin: 0 auto;
    padding-top: 35px;
}
.main .filter{
    width: 100%;
    height: auto;
    margin-bottom: 35px;
    padding: 25px 0px 25px 0px;
    background: #fff;
    border-radius: 4px;
    box-shadow: 0 2px 4px 0 #f0f0f0;
}
.filter .el-col{
    text-align: center;
    padding: 6px 0px;
    line-height: 16px;
    margin-left: 14px;
    position: relative;
    transition: all .3s ease;
    cursor: pointer;
    color: #4a4a4a;
}
.filter-el-row1{
    padding-bottom: 18px;
    margin-bottom: 17px;
}
.filter .filter-text{
    text-align: right;

```

```
font-size: 16px;
color: #888;
}
.filter .filter-text2{
}
.filter .filter-el-row1 .current{
    color: #ffc210;
    border: 1px solid #ffc210!important;
    border-radius: 30px;
}
.filter .filter-el-row2 .current{
    color: #ffc210;
}
.filter-price{
    display:inline-block;
    vertical-align: middle;
}
.filter-price .up, .filter-price .down{
    display: block;
    line-height: 8px;
    font-size: 13px;
    margin: -4px;
    color: #d8d8d8;
}
.current .filter-price .active{
    color: #ffc210;
}
.course-item{
    margin-bottom: 35px;
}
.course-item .course-item-box{
    padding: 20px 30px 20px 20px;
}
.course-item{
    box-shadow: 2px 3px 16px rgba(0,0,0,.1);
    transition: all .2s ease;
}
.course-item .course-item-left{
    width: 423px;
    height: 210px;
    margin-right: 30px;
}
.course-title{
    overflow: hidden; /* 在父元素中使用可以清除子元素的浮动影响 */
}
.course-title .box-title{
    font-size: 26px;
    color: #333333;
    float: left;
    margin-bottom: 8px;
}
.course-title .box-number{
    font-size: 14px;
```

```
        color: #9b9b9b;
        font-family: PingFangSC-Light;
        float: right;
        padding-top: 12px;
    }
    .course-item-right{
        width: 56.6%;
    }
    .author{
        font-size: 14px;
        color: #9b9b9b;
        margin-bottom: 14px;
        padding-bottom: 14px;
        overflow: hidden;
    }
    .author .box-author{
        float:left;
    }
    .author .lession{
        float: right;
    }
    .course-content .el-icon-caret-right{
        border: 1px solid #000;
        border-radius: 50%;
        margin-right: 6px;
    }
    .course-content .el-col{
        font-size: 14px;
        color: #666;
        width: 50%;
        margin-bottom: 15px;
        cursor: pointer;
    }
    .course-content .el-col:hover{
        color: #ffc210;
    }
    .course-content .el-col:hover .el-icon-caret-right,.course-content .el-col:hover .free{
        border-color: #ffc210;
        color: #ffc210;
    }
    .course-content .el-col .free{
        width: 34px;
        height: 20px;
        color: #fd7b4d;
        margin-left: 10px;
        border: 1px solid #fd7b4d;
        border-radius: 2px;
        text-align: center;
        font-size: 13px;
        white-space: nowrap;
    }
    .course-price{
        overflow: hidden;
```



```
}
.course-price .course-price-left{
  float: left;
}
.course-price .discount{
  padding: 6px 10px;
  display: inline-block;
  font-size: 16px;
  color: #fff;
  text-align: center;
  margin-right: 8px;
  background: #fa6240;
  border: 1px solid #fa6240;
  border-radius: 10px 0 10px 0;
}
.course-price .course-price-left{
  line-height: 22px;
}
.course-price .count{
  font-size: 24px;
  color: #fa6240;
}
.course-price .old_count{
  font-size: 14px;
  color: #9b9b9b;
  margin-left: 10px;
  text-decoration: line-through;
}
.course-price .buy{
  float: right;
  width: 120px;
  height: 38px;
  font-size: 16px;
  border-radius: 3px;
  border: 1px solid #fd7b4d;
  background: transparent; /* 透明 */
  color: #fa6240;
  cursor: pointer;
  transition: all .2s ease-in-out; /* css3新版本的样式中支持支持 jQuery里面的动画预设效果 */
  /* all表示当前元素的所有样式 .2s表示改变样式完成的时间 ease-in-out */
}
.course-price .buy:hover{
  color: #fff;
  background: #ffc210;
  border: 1px solid #ffc210;
}
</style>
```