Ask Django

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

EP 5. 아임포트 연동하기 (2)

jsonfield를 통해
아임포트 결제 메타정보 저장

jsonfield 설치

```
pip install jsonfield
```

- `jsonfield.JSONField`를 통해, 모델에서 JSON필드를 사용할 수 있습니다.
- 이는 CharField필드에 대한 Wrapper입니다.
- JSON필드 내 필드에 대한 쿼리는 불가합니다.

shop/models.py

```
from jsonfield import JSONField

class Item(models.Model):
    # 중략
    meta = JSONField(blank=True, default={})
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

Order에 대한 Admin.list_display 구현

shop/models.py

```
import pytz
from datetime import datetime
from django.contrib.humanize.templatetags.humanize import intcomma
from django.utils.safestring import mark_safe

def named_property(name):
    def wrap(fn):
        fn.short_description = name
        return property(fn)
    return wrap


def timestamp_to_datetime(timestamp):
    if timestamp:
        tz = pytz.timezone(settings.TIME_ZONE)
        return datetime.utcfromtimestamp(timestamp).replace(tzinfo=tz)
    return None
```

```python
class Item(models.Model):
    # 중략
    is_ready = property(lambda self: self.status == 'ready')
    is_paid = property(lambda self: self.status == 'paid')
    is_paid_ok = property(lambda self: self.status == 'paid' and self.amount == self.meta.get('amount'))
    is_cancelled = property(lambda self: self.status == 'cancelled')
    is_failed = property(lambda self: self.status == 'failed')

    receipt_url = named_property('영수증')(lambda self: self.meta.get('receipt_url'))
    cancel_reason = named_property('취소이유')(lambda self: self.meta.get('cancel_reason'))
    fail_reason = named_property('실패이유')(lambda self: self.meta.get('fail_reason', ''))

    paid_at = named_property('결제일시')(lambda self: timestamp_to_datetime(self.meta.get('paid_at')))
    failed_at = named_property('실패일시')(lambda self: timestamp_to_datetime(self.meta.get('failed_at')))
    cancelled_at = named_property('취소일시')(lambda self: timestamp_to_datetime(self.meta.get('cancelled_at')))
```

```
    # 중략

    @named_property('결제금액')
    def amount_html(self):
        return mark_safe('<div style="float: right;">{0}</div>'.format(intcomma(self.amount)))

    @named_property('처리결과')
    def status_html(self):
        cls, text_color = '', ''
        help_text = ''
        if self.is_ready:
            cls, text_color = 'fa fa-shopping-cart', '#ccc'
        elif self.is_paid_ok:
            cls, text_color = 'fa fa-check-circle', 'green'
        elif self.is_cancelled:
            cls, text_color = 'fa fa-times', 'gray'
            help_text = self.cancel_reason
        elif self.is_failed:
            cls, text_color = 'fa fa-ban', 'red'
            help_text = self.fail_reason
        html = '''
            <span style="color: {text_color};" title="this is title">
                <i class="{class_names}"></i>
                {label}
            </span>'''.format(class_names=cls, text_color=text_color, label=self.get_status_display())
        if help_text:
            html += '<br/>' + help_text
        return mark_safe(html)

    @named_property('영수증 링크')
    def receipt_link(self):
        if self.is_paid_ok and self.receipt_url:
            return mark_safe('<a href="{0}" target="_blank">영수증</a>'.format(self.receipt_url))
```

shop/admin.py

```
class OrderAdmin(admin.ModelAdmin):
    list_display = ['imp_uid', 'user', 'name', 'amount_html', 'status_html', 'paid_at', 'receipt_link']
```

주문 갱신 명령 구현

shop/models.py

```python
from django.http import Http404

class Order(models.Model):
    def update(self, commit=True, meta=None):
        '결재내역 갱신'
        if self.imp_uid:
            # self.meta = meta or self.api.find(imp_uid=self.imp_uid)
            try:
                self.meta = meta or self.api.find(imp_uid=self.imp_uid)
            except Iamport.HttpError:
                raise Http404('Not found {}'.format(self.imp_uid))
            # merchant_uid는 반드시 매칭되어야 합니다.
            assert str(self.merchant_uid) == self.meta['merchant_uid']
            self.status = self.meta['status']
```

shop/admin.py

```python
@admin.register(Order)
class OrderAdmin(admin.ModelAdmin):
    list_display = ['imp_uid', 'user', 'name', 'amount_html', 'status_html', 'paid_at', 'receipt_link']
    actions = ['do_update']

    def do_update(self, request, queryset):
        '주문 정보를 갱신합니다.'
        total = queryset.count()
        if total > 0:
            for order in queryset:
                order.update()
            self.message_user(request, '주문 {}건의 정보를 갱신했습니다.'.format(total))
        else:
            self.message_user(request, '갱신할 주문이 없습니다.')
    do_update.short_description = '선택된 주문들의 아임포트 정보 갱신하기'
```

Admin에서 결제 취소 구현

shop/models.py

```
class Order(models.Model):
    # 중략
    def cancel(self, reason=None, commit=True):
        '결제내역 취소'
        try:
            meta = self.api.cancel(reason, imp_uid=self.imp_uid)
            assert str(self.merchant_uid) == self.meta['merchant_uid']
            self.update(commit=commit, meta=meta)
        except Iamport.ResponseError as e:  # 취소시 오류 예외처리(이미 취소된 결제는 에러가 발생함)
            self.update(commit=commit)
        if commit:
            self.save()
```

shop/admin.py

```python
@admin.register(Order)
class OrderAdmin(admin.ModelAdmin):
    list_display = ['imp_uid', 'user', 'name', 'amount_html', 'status_html', 'paid_at', 'receipt_link']
    actions = ['do_update', 'do_cancel']

    def do_cancel(self, request, queryset):
        '선택된 주문에 대해 결제취소요청을 합니다.'
        queryset = queryset.filter(status='paid')
        total = queryset.count()
        if total > 0:
            for order in queryset:
                order.cancel()
            self.message_user(request, '주문 {}건을 취소했습니다.'.format(total))
        else:
            self.message_user(request, '취소할 주문이 없습니다.')
    do_cancel.short_description = '선택된 주문에 대해 결제취소요청하기'
```

Profile 주문내역에 처리결과/결과일시/영수증 필드 추가

accounts/profile.html

```
<tr>
    <th>상품명</th>
    <th>결제가격</th>
    <th>처리결과</th>
    <th>결과일시</th>
    <th>영수증</th>
</tr>

<tr>
    <td>{{ order.name }}</td>
    <td class="text-right">{{ order.amount|intcomma }}</td>
    <td>{{ order.amount_html }}</td>
    <td>{{ order.status_html }}</td>
    <td>{{ order.paid_at|default:"-" }}</td>
    <td>{{ order.receipt_link|default:"-" }}</td>
</tr>
```