Python Django入門 (5)

Python　Django

子モデルのCRUD

子の読み方

親の書籍に対して、子の感想の一覧を表示してみます。

`cms/models.py` で、感想は書籍を外部キーとして持つよう定義しました。

```
class Impression(models.Model):
    """感想"""
    book = models.ForeignKey(Book, verbose_name='書籍', relatec
    comment = models.TextField('コメント', blank=True)
```

よって、書籍に紐づく子供の感想は、related_nameを使って、以下のように読み出すことができます。

ここでも SQL は書きません。

```
impressions = book.impressions.all().order_by('id')    # 書籍の子
```

子の一覧のビュー

今回は、 汎用ビュー の ListView を使って書いてみます。

これを使うと、ページングなどが簡単に実現できます。

cms/views.py に以下の記述を追加します。

```
from django.views.generic.list import ListView
  :

class ImpressionList(ListView):
    """感想の一覧"""
    context_object_name='impressions'
    template_name='cms/impression_list.html'
    paginate_by = 2  # １ページは最大2件ずつでページングする
```

```
    def get(self, request, *args, **kwargs):
        book = get_object_or_404(Book, pk=kwargs['book_id'])
        impressions = book.impressions.all().order_by('id')
        self.object_list = impressions

        context = self.get_context_data(object_list=self.objec
        return self.render_to_response(context)
```

子の一覧のテンプレート

BootStrapを応用して、

- ページングありの時、「前へ、1、2、次へ」などのページ番号を表すリンクを置く。
- 削除する時、いきなり消すのではなく、確認のモーダルダイアログを出す。

ということをしています。ちょっと長いです。

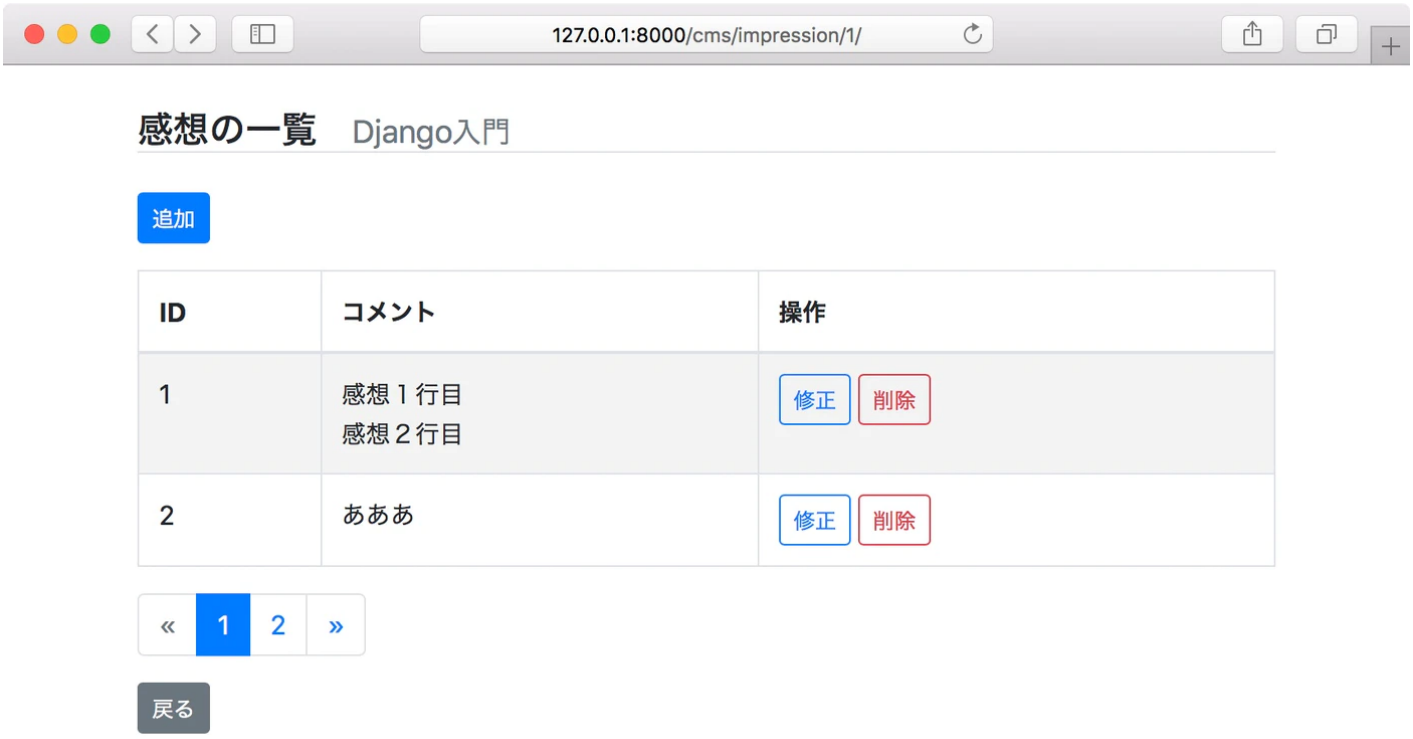
`mybook/cms/templates/cms/base_html` を継承して `mybook/cms/templates/cms/impression_list.html` を作成します。

```
{% extends "cms/base.html" %}

{% block title %}感想の一覧{% endblock title %}

{% block content %}
    <h4 class="mt-4 border-bottom">感想の一覧 <small class="tex
    <a href="{% url 'cms:impression_add' book_id=book.id %}" c
    <table class="table table-striped table-bordered">
      <thead>
        <tr>
          <th>ID</th>
          <th>コメント</th>
          <th>操作</th>
        </tr>
      </thead>
      <tbody>
        {% for impression in impressions %}
        <tr>
          <td>{{ impression.id }}</td>
          <td>{{ impression.comment|linebreaksbr }}</td>
          <td>
            <a href="{% url 'cms:impression_mod' book_id=book.
            <button class="btn btn-outline-danger btn-sm del_c
          </td>
        </tr>
        {% endfor %}
      </tbody>
    </table>
```

```
{% if is_paginated %}
<nav aria-label="Page navigation">
  <ul class="pagination">
    {% if page_obj.has_previous %}
      <li class="page-item"><a class="page-link" href="?pa
    {% else %}
      <li class="page-item disabled"><a class="page-link" 
    {% endif %}
    {% for linkpage in page_obj.paginator.page_range %}
      {% ifequal linkpage page_obj.number %}
        <li class="page-item active"><a class="page-link" 
      {% else %}
        <li class="page-item"><a class="page-link" href="?
    {% endifequal %}
    {% endfor %}
    {% if page_obj.has_next %}
      <li class="page-item"><a class="page-link" href="?pa
    {% else %}
      <li class="page-item disabled"><a class="page-link" 
    {% endif %}
  </ul>
</nav>
{% endif %}

<a href="{% url 'cms:book_list' %}" class="btn btn-seconda

{# 削除を確認するモーダル ダイアログ #}
<div class="modal fade" id="deleteModal" tabindex="-1" rol
```

```
      <div class="modal-dialog" role="document">
        <div class="modal-content">
          <div class="modal-header">
            <h5 class="modal-title" id="deleteModalLabel">確認<
            <button type="button" class="close" data-dismiss="
          </div>
          <div class="modal-body">
            <p>ID: <span id="del_pk"></span> を削除しますか？</p
          </div>
          <div class="modal-footer">
            <a href="#" class="btn btn-primary" id="del_url">O
            <button type="button" class="btn btn-secondary" da
          </div>
        </div>
      </div>
    </div>
{% endblock content %}

{% block extra_js %}
<script>
$(function() {
  $('.del_confirm').on('click', function () {
     $("#del_pk").text($(this).data("pk"));
     $('#del_url').attr('href', $(this).data("url"));
  });
});
</script>
{% endblock %}
```

出来上がる一覧ページは、以下のようなものとなります。
（まだ、この先のことをしないと動かないので、こういうイメージになると考えて下さい）



子の追加、修正のフォーム

`cms/forms.py` に以下のように追加します。

ここでは、 `cms/models.py` の Impression モデルを追加、修正するための元となるフォームを作成します。

```
  :
from cms.models import Book, Impression
  :

class ImpressionForm(ModelForm):
    """感想のフォーム"""
    class Meta:
        model = Impression
        fields = ('comment', )
```

子の追加、修正のビュー

cms/views.py に以下のように追加します。

```
  :
from cms.models import Book, Impression
from cms.forms import BookForm, ImpressionForm
  :

def impression_edit(request, book_id, impression_id=None):
    """感想の編集"""
    book = get_object_or_404(Book, pk=book_id)  # 親の書籍を読む
    if impression_id:   # impression_id が指定されている (修正時)
        impression = get_object_or_404(Impression, pk=impressi
    else:               # impression_id が指定されていない (追加時
```

```
        impression = Impression()

    if request.method == 'POST':
        form = ImpressionForm(request.POST, instance=impressio
        if form.is_valid():    # フォームのバリデーション
            impression = form.save(commit=False)
            impression.book = book  # この感想の、親の書籍をセット
            impression.save()
            return redirect('cms:impression_list', book_id=boo
    else:    # GET の時
        form = ImpressionForm(instance=impression)  # impressi

    return render(request,
                  'cms/impression_edit.html',
                  dict(form=form, book_id=book_id, impression_
```

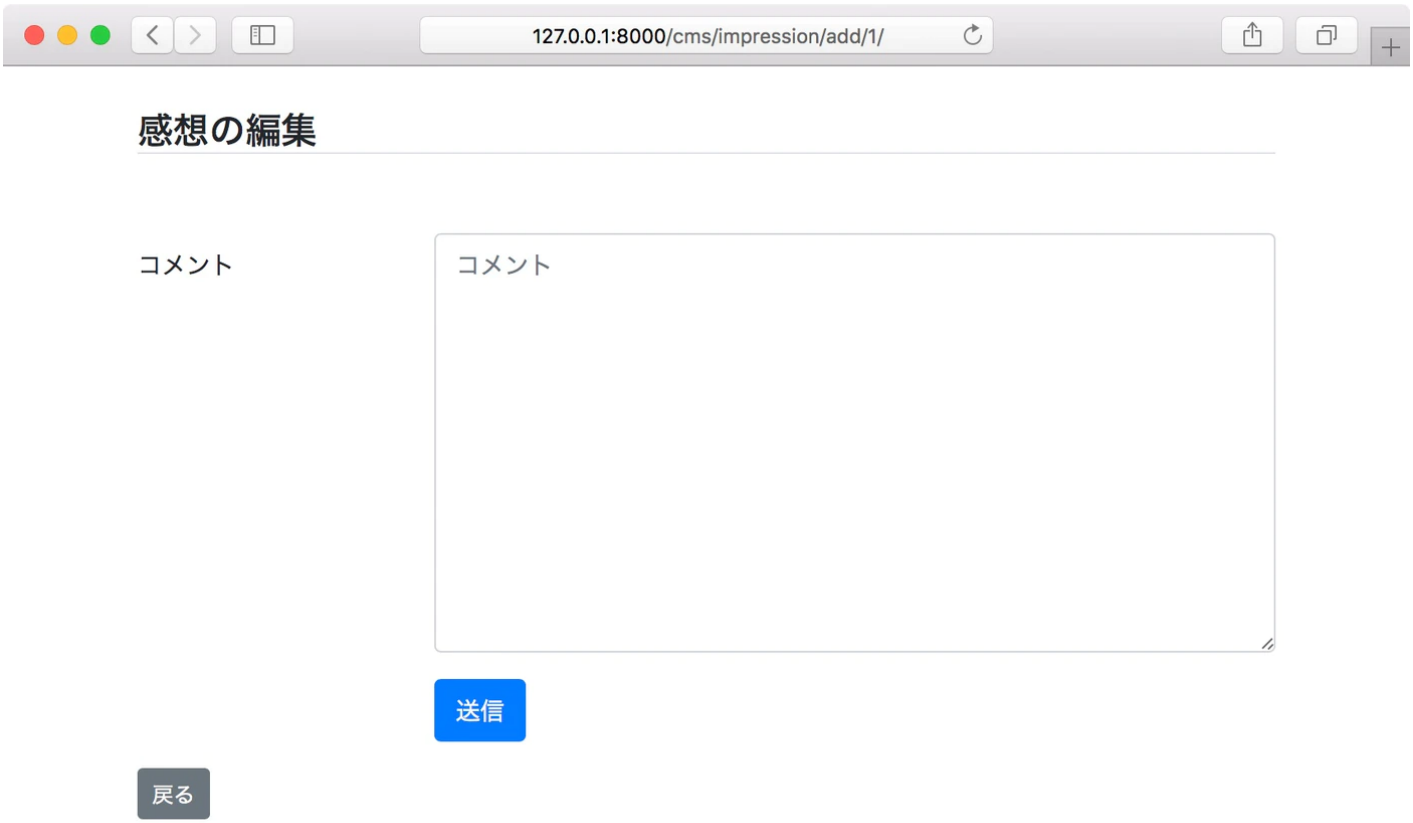
子の追加、修正のテンプレート

mybook/templates/base_html を継承して mybook/cms/templates/cms/impression_edit.html を作成します。

```
{% extends "cms/base.html" %}
{% load bootstrap4 %}
```

```
{% block title %}感想の編集{% endblock title %}

{% block content %}
    <h4 class="mt-4 mb-5 border-bottom">感想の編集</h4>
    {% if impression_id %}
    <form action="{% url 'cms:impression_mod' book_id=book_id
    {% else %}
    <form action="{% url 'cms:impression_add' book_id=book_id
    {% endif %}
      {% csrf_token %}
      {% bootstrap_form form layout='horizontal' %}
      <div class="form-group row">
        <div class="offset-md-3 col-md-9">
          <button type="submit" class="btn btn-primary">送信</b
        </div>
      </div>
    </form>
    <a href="{% url 'cms:impression_list' book_id=book_id %}" 
{% endblock content %}
```

追加、修正のページは、以下のようになります。（まだ、この先のことをしないと動かないので、こういうイメージになると考えて下さい）



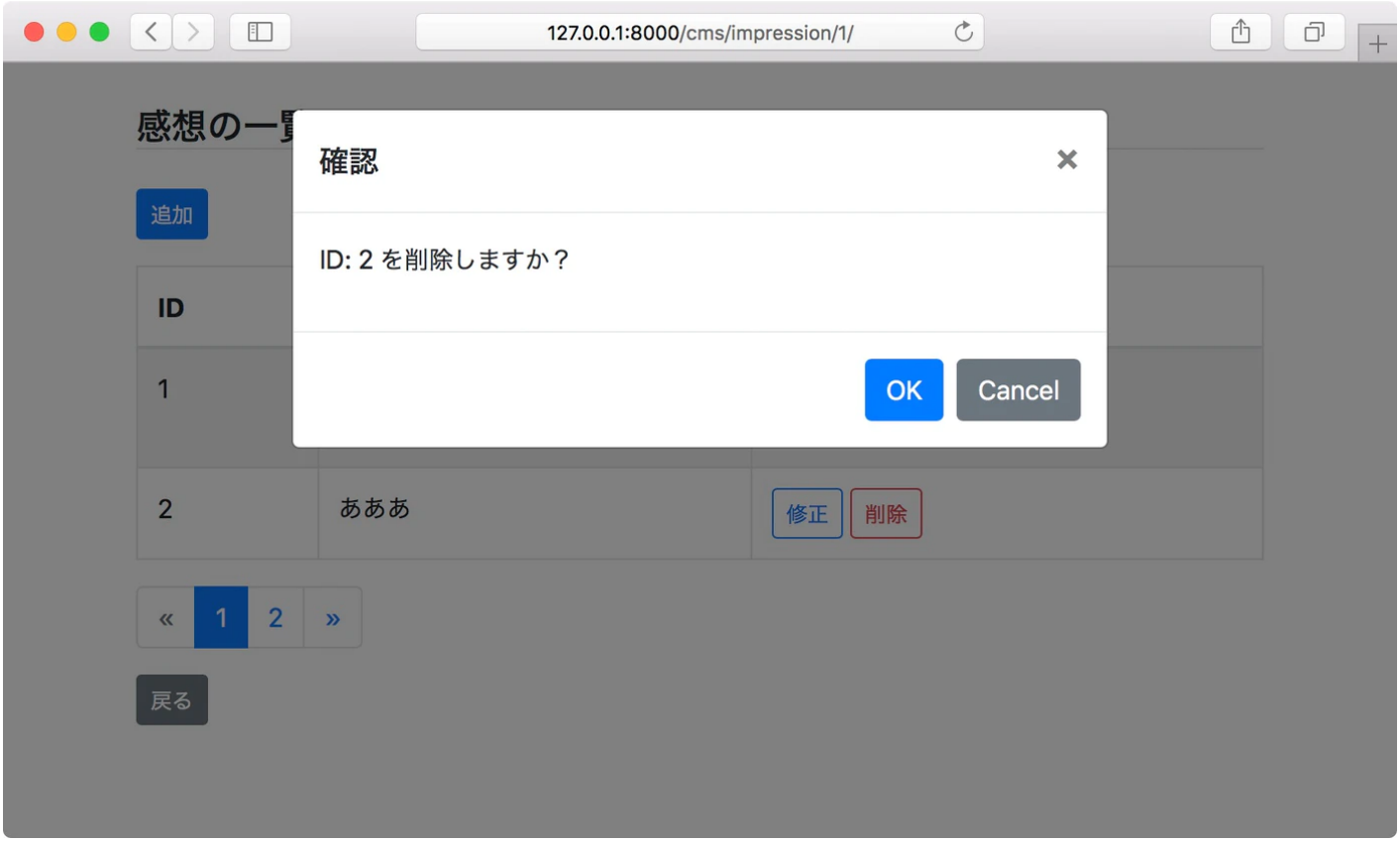
子の削除のビュー

`cms/views.py` に以下のように追加します。

今回は、いきなり消さずに、Bootstrap の モーダルダイアログを出して、確認メッセージを出しています。
ただ、ビューの中身は、親の書籍の時と変わりません。

```
def impression_del(request, book_id, impression_id):
    """感想の削除"""
```

```
    impression = get_object_or_404(Impression, pk=impression_i
    impression.delete()
    return redirect('cms:impression_list', book_id=book_id)
```



子のURLスキーム

cms/urls.py に以下のように追加します。

```
urlpatterns = [
    :
```

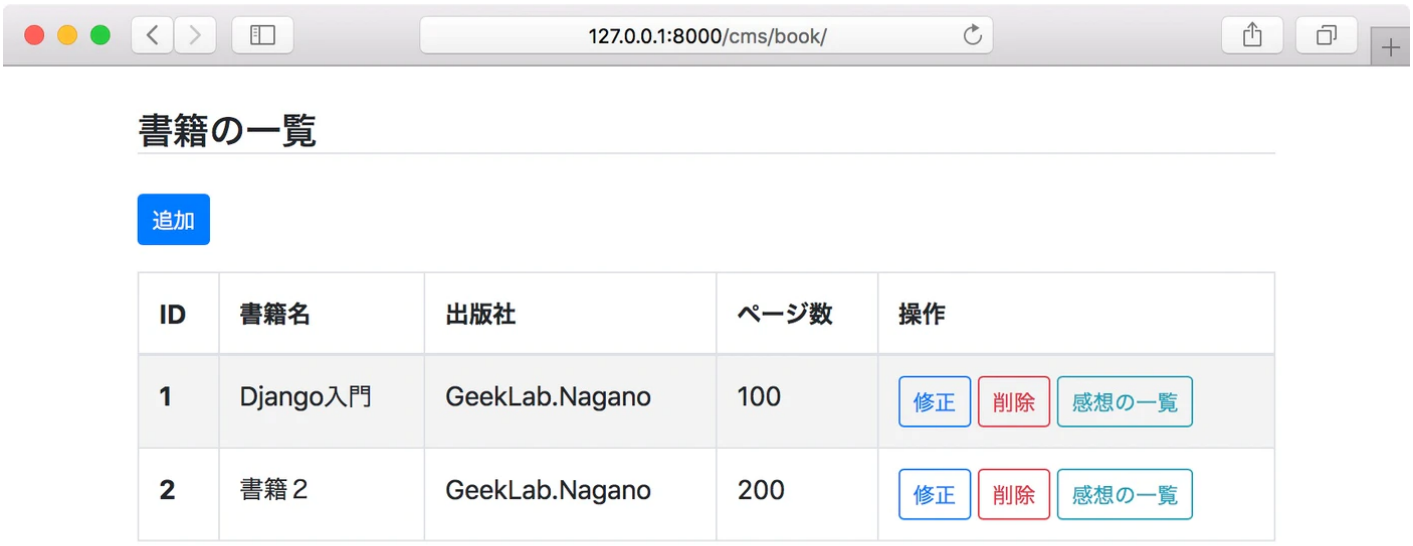
```
    # 感想
    path('impression/<int:book_id>/', views.ImpressionList.as_
    path('impression/add/<int:book_id>/', views.impression_edi
    path('impression/mod/<int:book_id>/<int:impression_id>/',
    path('impression/del/<int:book_id>/<int:impression_id>/',
]
```

親の一覧の修正

親の書籍の一覧から、該当する書籍の「感想一覧」が出せるよう、リンクを追加します。

mybook/cms/templates/cms/book_list.html に1行追加します。

```
        <td>
          <a href="{% url 'cms:book_mod' book_id=book.id %}"
          <a href="{% url 'cms:book_del' book_id=book.id %}"
          <a href="{% url 'cms:impression_list' book_id=book
        </td>
```



それでは、ローカルサーバを起動し、「書籍の一覧」から「感想の一覧」をたどって、感想の登録／修正／削除を行ってみましょう。

```
http://127.0.0.1:8000/cms/book/
```

ここまでの説明のように親子関係のあるモデルのCRUDができれば、後はモデルをどう設計するかということなので、応用して色々なものが作れると思います。

Python Django入門 (6) に続きます。