Messages消息框架

> 在网页应用中，你经常需要在处理完表单或其它类型的用户输入后。显示一个通知消息（也叫做`flash message`给用户
>
> 对于这个功能，`Django`提供基于`Cookie`和会话的消息，无论是匿名用户还是认证的用户。
>
> 其消息框架允许你临时将消息存储在请求中，并在接下来的请求（通常就是下一个请求）中提取它们并显示。每个消息都带有一个特定`level`标签，表示其优先级（例如`info`、`warning`或`error`）

> `django-admin startproject`创建的默认`settings.py`已经包含启用消息框架功能需要的所有的设置

> - `INSTALLED_APPS`中的`'django.contrib.messages'`。
> - `MIDDLEWARE_CLASSES`中的`'django.contrib.sessions.middleware.SessionMiddleware'`和`'django.contrib.messages.middleware.MessageMiddleware'`
>
> 默认的后端存储 依赖`[sessions]`
>
> 所以`MIDDLEWARE_CLASSES`中必须启用`SessionMiddleware`并出现在`MessageMiddleware`之前
>
> - `TEMPLATES`设置中定义的`DjangoTemplates`的`'context_processors'`选项包含`'django.contrib.messages.context_processors.messages'`

消息级别

> `from django.contrib import messages`

- `messages.debug`
- `messages.info`
- `messages.success`
- `messages.warning`
- `messages.error`

使用消息框架

> 视图函数只需要创建`messages`消息对象即可

```
messages.warning(request,'登陆失败，用户名或密码无效')
return render(request,'login.html',locals())
```

> 前端模板中判断是否含有`messages`消息，遍历取出即可
>
> 也可以结合`bootstrap`框架让提示消息变得更加美丽

```
{% if messages %}
    {% for message in messages %}
        <div class="alert alert-{{ message.tags }} fade in">
            {{ message }}
        </div>
    {% endfor %}
{% endif %}
```

> **注意**：`messages`对象是一个数据集，并不是单独的一条消息，需要我们在使用时，必须通过`for`循环进行访问

Paginator分页组件

> `from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger`

- `Paginator`：创建分页对象

分页对象内置属性

```
all_ = models.objects.all()
p = Paginator(all_, 10)
# 分页all_数据，每页显示10条数据
```

```
p.count # 总数据量
p.num_pages() # 分页数
p.page_range() # 列表形式返回当前可有的页数 [1,2,3]
```

```
page_1 = p.page(1) # 选择第一页，返回第一页数据对象
page_1.object_list # 返回第一页所有数据
for var in page_1:
    print(var)
A
B
C
...
```

某一页内置属性

```
  page_1.number # 当前页的页码
  page_1.has_next() # 是否有下一页
  page_1.has_previous() # 是否有上一页
  page_1.has_other_pages() # 是否含有其他页
```

```
  page_1.next_page_number() # 下一页的页码
  page_1.previous_page_number() # 上一页的页码
```

```
page_1.start_index() # 该页第一个数据的索引
page_1.end_index() # 该页最后一个数据的索引
```

- `EmptyPage`：取不到页面数据，抛出该异常

```
all_ = models.objects.all()
p = Paginator(all_, 10)
try:
    list_ = p.page(page_num)
except EmptyPage:
    #没有第page_num页
    list_ = paginator.page(1) # 取不到该也数据，直接返回第一页数据
```

- `PageNotAnInteger`：当页数是一个非整数类型时，抛出该异常

模板页面基本使用方式

```
{% if topic_list.has_previous %}
    <!-- 当前页是否含有上一页 -->
    <a href="?page={{ list_.previous_page_number }}">上一页</a>
    <!-- 连接传参形式传递上一页的页码ID -->
{% endif %}

{% if topic_list.has_next %}
    <!-- 当前页是否含有下一页 -->
    <a href="?page={{ list_.next_page_number }}">下一页</a>
    <!-- 连接传参形式传递下一页的页码ID -->
{% endif %}
```