

Ask Company



리액트와 함께 장고 시작하기 / 장고 Forms

# Messages Framework

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

# Messages Framework

현재 User를 위한 1회성 메시지를 담는 용도

ex) "저장했습니다.", "로그인되었습니다."

HttpRequest 인스턴스를 통해 메시지를 남깁니다.

즉, View에서만 사용 가능

메시지를 1회 노출이 되고 사라집니다.

View를 통한 템플릿 시스템을 통해 노출을 하며, 템플릿 내에서 JavaScript를 통한 노출도 가능

<https://docs.djangoproject.com/en/3.0/ref/contrib/messages/>

# Message Levels를 통한 메시지 분류

파이썬 로깅 모듈의 Level을 차용

레벨에 따라 로깅 여부 판단

혹은 템플릿에서 다른 스타일로 노출

## 레벨 종류

DEBUG : 디폴트 설정으로 무시되는 레벨

개발관련된 메세지이며, 실서비스에서는 무시

INFO : 해당 유저에 대한 정보성 메세지

SUCCESS : 액션이 성공적으로 수행되었음을 알림.

WARNING : 실패가 아직 발생하진 않았지만, 임박했다.

ERROR : 액션이 수행되지 않았거나, 다른 Failure가 발생했다.

<https://docs.djangoproject.com/en/3.0/ref/contrib/messages/#message-levels>

# messages 등록 코드

```
# blog/views.py
from django.contrib import messages

def post_new(request):
    # 중략
    if form.is_valid():
        post = form.save()
        messages.add_message(request, messages.SUCCESS, '새 글이 등록되었습니다.')
        messages.success(request, '새 글이 등록되었습니다.') # 혹은 shortcut 형태
        return redirect(post)
    # 생략
```

# messages 소비

주로 템플릿을 통해서 소비

messages context\_processors를 통해 messages 목록에 접근

.tags 속성을 통해 레벨을 제공

.message 속성을 통해 내용을 제공 (= str(message))

```
{% if messages %}
  <ul class="messages">
    {% for message in messages %}
      <li>
        [{{ message.tags }}] [{{ message.message }}]
      </li>
    {% endfor %}
  </ul>
{% endif %}
```


# 참고) Context Processors

<https://github.com/django/django/blob/2.1/django/template/context.py#L246>

템플릿 기본 로딩 변수목록을 생성해주는 함수 목록

```
# 프로젝트/settings.py
```

```
TEMPLATES = [{  
    'BACKEND':  
    'django.template.backends.django.DjangoTemplates',  
    'DIRS': [],  
    'APP_DIRS': True,  
    'OPTIONS': {  
        'context_processors': [  
            'django.template.context_processors.debug',  
            'django.template.context_processors.request',  
            'django.contrib.auth.context_processors.auth',  
            'django.contrib.messages.context_processors.messages',  
        ],  
    },  
}]
```



```
from django.contrib.messages.api import get_messages  
from django.contrib.messages.constants import DEFAULT_LEVELS
```

```
def messages(request):  
    return {  
        'messages': get_messages(request),  
        'DEFAULT_MESSAGE_LEVELS': DEFAULT_LEVELS,  
    }
```

# Bootstrap4 alert 스타일과 대응해보기

Bootstrap 4	messages levels 대응해보기
alert-primary	
alert-secondary	<b>debug</b>
alert-success	success
alert-danger	<b>error</b>
alert-warning	warning
alert-info	info
alert-light	
alert-dark	

This is a primary alert—check it out!

This is a secondary alert—check it out!

This is a success alert—check it out!

This is a danger alert—check it out!

This is a warning alert—check it out!

This is a info alert—check it out!

This is a light alert—check it out!

This is a dark alert—check it out!

<https://getbootstrap.com/docs/4.0/components/alerts/>

# Bootstrap4 alert HTML 마크업

```
<div class="alert alert-secondary">  
    This is a secondary alert-check it out!  
</div>
```

```
<div class="alert alert-info">  
    This is a info alert-check it out!  
</div>
```

```
<div class="alert alert-success">  
    This is a success alert-check it out!  
</div>
```

```
<div class="alert alert-warning">  
    This is a warning alert-check it out!  
</div>
```

```
<div class="alert alert-danger">  
    This is a danger alert-check it out!  
</div>
```

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



# 출력 tags 변경하기

[프로젝트/settings.py](#)

```
from django.contrib.messages import constants as messages_constants
```

```
MESSAGE_TAGS = {  
    messages_constants.DEBUG: 'secondary',  
    messages_constants.ERROR: 'danger',  
}
```

# MESSAGE\_LEVEL 변경하기

메세지 노출 최소 레벨 (프로젝트/settings.py)

```
from django.contrib.messages import constants as messages_constants
```

```
MESSAGE_LEVEL = messages_constants.INFO    # 디폴트 설정
```

```
MESSAGE_LEVEL = messages_constants.DEBUG
```

# django-bootstrap4의 템플릿 태그

## bootstrap4/messages.html 활용

```
{% load i18n bootstrap4 %}

{% for message in messages %}
    <div class="{% message|bootstrap_message_classes %}" alert-dismissible fade show" role="alert">
        <button type="button" class="close" data-dismiss="alert" aria-label="{% trans
'close' %}">#215;</button>
        {{ message }}
    </div>
{% endfor %}
```

<https://github.com/zostera/django-bootstrap4/blob/master/bootstrap4/templates/bootstrap4/messages.html>

django-bootstrap4을 활용한

# 기본 Form/Messages 템플릿

```
{# Load the tag library #}  
{% load bootstrap4 %}
```

```
{# Load CSS and JavaScript #}  
{% bootstrap_css %}  
{% bootstrap_javascript jquery='full' %}
```

```
{# Display django.contrib.messages as Bootstrap alerts #}  
{% bootstrap_messages %}
```

```
{# Display a form #}  
<form action="/url/to/submit/" method="post" class="form">  
    {% csrf_token %}  
    {% bootstrap_form form %}  
    {% buttons %}  
        <button type="submit" class="btn btn-primary">  
            Submit  
        </button>  
    {% endbuttons %}  
</form>
```

Life is short.  
You need Python and Django.

I will be your pacemaker.

