Messages Framework

The messages framework allows you to temporarily store messages in one request and retrieve them for display in a subsequent request.

Django provides full support for cookie- and session-based messaging, for both anonymous and authenticated users.

INSTALLED_APPS = ['django.contrib.messages']

MIDDLEWARE = ['django.contrib.sessions.middleware.SessionMiddleware', 'django.contrib.messages.middleware.MessageMiddleware']

‘context_processors’: ['django.contrib.messages.context_processors.messages’]

Messages Levels and Tags

The messages framework is based on a configurable level architecture similar to that of the Python logging module.

Message Level – Message levels allow you to group messages by type so they can be filtered or displayed differently in views and templates.

Message Tag – Message tags are a string representation of the message level plus any extra tags that were added directly in the view. Tags are stored in a string and are separated by spaces. Typically, message tags are used as CSS classes to customize message style based on message type.

Level	Tag	Value	Purpose

How to use

Write Message:-

add_message(request, level, message, extra_tags='', fail_silently=False) – This method is used to add/write messages.

Setting fail_silently=True only hides the MessageFailure that would otherwise occur when the messages framework disabled and one attempts to use one of the add_message family of methods. It does not hide failures that may occur for other reasons.

```
from django.contrib import messages
messages.add_message(request, messages.INFO, ‘Info le lo info’)
```

How to use

Write Message by Shortcut Methods:-

```
from django.contrib import messages
messages.debug(request, '%s SQL statements were executed.' % count)
messages.info(request, 'Three credits remain in your account.')
messages.success(request, 'Profile details updated.')
messages.warning(request, 'Your account expires in three days.')
messages.error(request, 'Document deleted.')
```

How to use

Display Message:-

```
{% if messages %}
    {% for message in messages %}
        {% if message.tags %} {{ message.tags }} {% endif %}
        {{ message }}
    {% endfor %}
{% endif %}
```

Methods

get_level ( ) - This method is used to retrieved the current effective level.

from django.contrib import messages

current_level = messages.get_level(request)

set_level ( ) - This method is used to set minimum recorded level.

from django.contrib import messages

messages.set_level(request, messages.DEBUG)

This will record messages with a level of DEBUG and higher.

Display Message:-

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

Change Tags

Open **settings.py**

```
from django.contrib.messages import constants as messages
MESSAGE_TAGS = {
    messages.ERROR: 'danger',
}
```

get_messages(request)

get_messages(request) – This method is used to get message. If you need to get message out side template you can use get_messages(request).

```
from django.contrib.messages import get_messages
storage = get_messages(request)
for message in storage:
    do_something_with_the_message(message)
```