Django Messages Framework

Cruz Arredondo Jose David, UTT. November 7, 2023

Summary - The Django messages framework is a built-in feature that enables developers to display notifications or messages to users based on their actions within a web application. These messages can be used to communicate various types of information such as success messages, error alerts, or informative notes. The framework simplifies the process of sending messages to users and provides an easy way to handle different types of messages. Messages can be stored in different levels such as DEBUG, INFO, SUCCESS, WARNING, and ERROR, allowing developers to categorize messages based on their importance. The messages framework also supports different storage backends, allowing developers to choose where to store the messages, such as in cookies or the session. This feature is particularly useful for providing feedback to users after form submissions or other user interactions in Django web applications

I. Introduction

This document serves as a comprehensive guide to understanding and harnessing the full potential of Django's Messages Framework. Whether you're a novice web developer eager to enhance user interaction or an experienced coder looking to streamline your messaging system, this guide will walk you through the fundamentals, best practices, and advanced techniques of utilizing Django's Messages Framework effectively.

II. Enabling Messages of Django

Messages are implemented through a middleware class and corresponding context processor.
The default settings.py created by django-admin startproject already contains all the settings required to enable message functionality:

'django.contrib.messages' is in INSTALLED_APPS.
MIDDLEWARE contains

'django.contrib.sessions.middleware.SessionMiddleware' and 'django.contrib.messages.middleware.MessageMiddleware'.

The default storage backend relies on sessions. That's why SessionMiddleware must be enabled and appear before MessageMiddleware in MIDDLEWARE.

The 'context_processors' option of the DjangoTemplates backend defined in your TEMPLATES setting contains 'django.contrib.messages.context_processors.messages'.

If you don't want to use messages, you can remove 'django.contrib.messages' from your INSTALLED_APPS, the MessageMiddleware line from MIDDLEWARE, and the messages context processor from TEMPLATES.

III. Configuring the Message Engine

The messages framework can use different backends to store temporary messages.

Django provides three built-in storage classes in django.contrib.messages:

class storage.session.SessionStorage:
This class stores all messages inside of the request's session. Therefore it requires Django's contrib.sessions application.

class storage.cookie.CookieStorage:
This class stores the message data in a cookie (signed with a secret hash to prevent manipulation) to persist notifications across requests. Old messages are dropped if the cookie data size would exceed 2048 bytes.

class storage.fallback.FallbackStorage:
This class first uses CookieStorage, and falls back to using SessionStorage for the messages that could not fit in a single cookie. It also requires Django's contrib.sessions application.[1]

```
MESSAGE_STORAGE =
"django.contrib.messages.storage.cookie.CookieStorage"
```

IV. Using Messages in Views and Templates

To add a message, call:

```
from django.contrib import messages

messages.add_message(request, messages.INFO, "Hello world.")
```

V. Characteristics

1) *Integration with Templates:*

Django's messages framework seamlessly integrates with templates, allowing developers to display messages in the HTML pages. Template tags and filters are available to render messages in a user-friendly format, enhancing the visual presentation of messages to end-users.

```
{% if messages %}
<ul class="messages">
    {% for message in messages %}
    <li{% if message.tags %} class="{{ message.tags }}"{%
endif %}>
        {% if message.level == DEFAULT_MESSAGE_LEVELS.ERROR
%}Important: {% endif %}
        {{ message }}
    </li>
    {% endfor %}
</ul>
{% endif %}
```

2) *Automatic Message Expiration:*

Messages can be set to automatically expire after a certain period, ensuring that outdated messages do not clutter the user interface. This automatic expiration mechanism helps in maintaining a clean and efficient messaging system.

```
storage = messages.get_messages(request)
for message in storage:
    do_something_with(message)
storage.used = False
```

3) *Security Best Practices:*

Django emphasizes security best practices such as secure password hashing, protection against common attacks like Cross-Site Request Forgery (CSRF), and secure session management.
Developers are encouraged to follow these practices to ensure the integrity and confidentiality of user data.

4) *Message Levels:*

Messages can be categorized into different levels like DEBUG, INFO, SUCCESS, WARNING, and ERROR. Each level represents a different type of message, allowing developers to convey the importance or urgency of the information being presented.

Messages levels are nothing more than integers, so you can define your own level constants and use them to create more customized user feedback

When creating custom message levels you should be careful to avoid overloading existing levels.

Level Constant	Value
DEBUG	10
INFO	20
SUCCESS	25
WARNING	30
ERROR	40

For more direct control over message tags, you can optionally provide a string containing extra tags to any of the add methods:

```
messages.add_message(request, messages.INFO, "Over 9000!",
extra_tags="dragonball")
messages.error(request, "Email box full", extra_tags="email")
```

Extra tags are added before the default tag for that level and are space separated. [2]

VI. Conclusion

The Messages Framework, with its simplicity in usage and robust functionality, stands as a testament to Django's commitment to developer convenience and user satisfaction. By providing developers with an easy yet sophisticated way to manage messages, Django enables the creation of applications that not only function flawlessly but also communicate effectively with their users..

VII. References

[1] Django (2022) Django, Django Project. Available at: https://docs.djangoproject.com/en/4.2/ref/contrib/messages/ (Accessed: 07 November 2023).

[2] ordinarycoders (2023) How to use Django Messages Framework, Ordinarycoders.com. Available at: https://ordinarycoders.com/blog/article/django-messages-framework (Accessed: 07 November 2023).