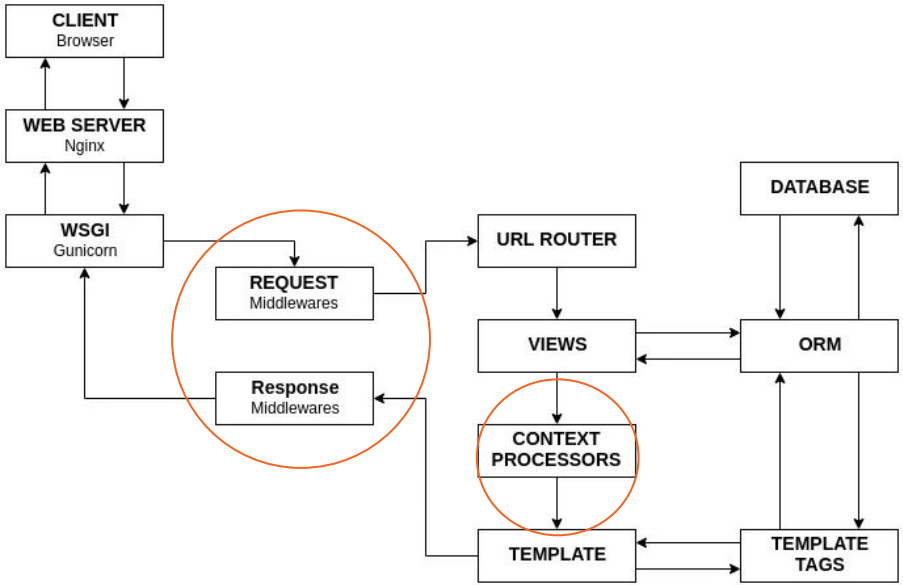
Middlewares y context processors

Cómo modificar la respuesta fuera de la vista en Django

¿Quien soy?

- Desarrollador desde el 2015
- Líder técnico / Project manager en Swapps hasta el 2023
- Actualmente senior software developer en SnapEDA

Ciclo de vida de la app Django



Fuente: https://goutomroy.medium.com/request-and-response-cycle-in-django-338518096640

Capas de la aplicación Django:

1. *Request* Middlewares
2. URL Router or URL Dispatcher
3. Views
4. Context Processors
5. Template Renderers
6. *Response* Middlewares

Fuente: https://learnbatta.com/course/django/request-response-lifecycle/

¿Qué son los middlewares?

"Middleware is a framework of hooks into Django's request/response processing. It's a light, low-level 'plugin' system for globally altering Django's input or output." - Django official documentation

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

Consideraciones de los middlewares

- Para activar un middleware, debes agregarlo a la lista MIDDLEWARE en los settings
- El orden de los middlewares importa; algunos pueden modificar o depender del resultado de otros. Siempre lee la documentación antes de usarlos

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

¿Cómo definir un middleware?


```python
class SimpleMiddleware:
    def __init__(self, get_response):
        self.get_response = get_response
        # One-time configuration and initialization.

    def __call__(self, request):
        # Code to be executed for each request before
        # the view (and later middleware) are called.

        response = self.get_response(request)

        # Code to be executed for each request/response after
        # the view is called.

        return response

    def process_view(self, request, view_func, view_args, view_kwargs):

        # called just before Django calls the view

        return None

    def process_exception(self, request, exception):

        # called when a view raises an exception.

        return None

    def process_template_response(self, request, response):

        # called just after the view has finished executing

        return None
```

Middlewares más conocidos

- **SecurityMiddleware:** Habilita mejoras de seguridad al ciclo petición/respuesta (HTTPS, Cross Origin Policies, etc).
- **SessionMiddleware:** Habilita el uso de sesiones de usuario, tanto autenticado como anónimo.
- **CommonMiddleware:** Habilita algunas funcionalidades básicas como redirecciones a www. , agregar el slash final a la url, etc.
- **CsrfViewMiddleware :** Habilita la protección contra falsificaciones de petición (Cross Site Request Forgeries CSRF) en los forms.
- **AuthenticationMiddleware:** Agrega el atributo “user” al request junto con información del mismo.

Middleware Demo

Repository:

https://github.com/jlariza/django-middlewares-context-processors-example

¿Qué son los context processors?

"A context processor has a simple interface: It's a Python function that takes one argument, an HttpRequest object, and returns a dictionary that gets added to the template context." - Django official documentation

```
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',
            ],
        },
    },
]
```

Consideraciones de los context processors

- Para activar un context processor, debes agregarlo a la lista “context_processors” en la key “OPTIONS” del setting TEMPLATES
- El orden de los CP importa; si un CP agrega una variable al contexto y otro posterior reutiliza su nombre, la variable será sobreescrita.

```
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',
            ],
        },
    },
]
```

¿Cómo definir un context processor?


```python
def my_context_processor(request):

    # process variables from request if required

    return {
        "MY_KEY": "my_value"
    }
```

Context processors más conocidos

- **Debug:** Si **DEBUG=True**, Añade la variable “debug” que puede ser usada para mostrar información de debugueo. También añade la variable “sql_queries” que contiene información de las queries realizadas en la vista.
- **Request:** Agrega el objeto HttpRequest de la vista a la variable “request”.
- **Auth:** Agrega información del usuario y los permisos relacionados a su cuenta en las variables “user” y “perms” respectivamente.

Context processor Demo

Repository:

https://github.com/jlariza/django-middlewares-context-processors-example

Caso de uso común: Django messages framework

Configurar el framework de mensajes de Django requiere:

1. Agregar `'django.contrib.sessions.middleware.SessionMiddleware'` y `'django.contrib.messages.middleware.MessageMiddleware'` a los middlewares instalados

2. Agregar `'django.contrib.messages.context_processors.messages'` a los context processors instalados

Caso de uso común: Django messages framework

```python
class MessageMiddleware(MiddlewareMixin):
    """
    Middleware that handles temporary messages.
    """

    def process_request(self, request):
        request._messages = default_storage(request)

    def process_response(self, request, response):
        """
        Update the storage backend (i.e., save the messages).

        Raise ValueError if not all messages could be stored and DEBUG is True.
        """
        # A higher middleware layer may return a request which does not contain
        # messages storage, so make no assumption that it will be there.
        if hasattr(request, "_messages"):
            unstored_messages = request._messages.update(response)
            if unstored_messages and settings.DEBUG:
                raise ValueError("Not all temporary messages could be stored.")
        return response
```

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


def messages(request):
    """
    Return a lazy 'messages' context variable as well as
    'DEFAULT_MESSAGE_LEVELS'.
    """
    return {
        "messages": get_messages(request),
        "DEFAULT_MESSAGE_LEVELS": DEFAULT_LEVELS,
    }
```

Fuente: https://github.com/django/django/tree/main/django/contrib/messages

¿Preguntas?