

Secretos y Estrategias del Logging en Django

Pycun #22



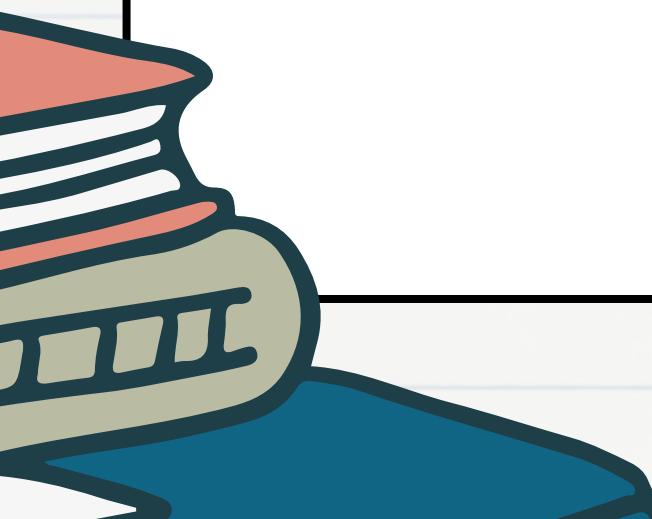
¿Que es el logging?

- Proceso de registrar mensajes que documentan eventos.
- Los mensajes son conocidos como logs.
- Los logs sirven para monitorear el comportamiento de una aplicación



Logging en Django

- Es una abstraccion de la biblioteca “logging” de python.
- Utiliza una estructura de diccionario para poder configurarse.
- Se maneja por niveles.



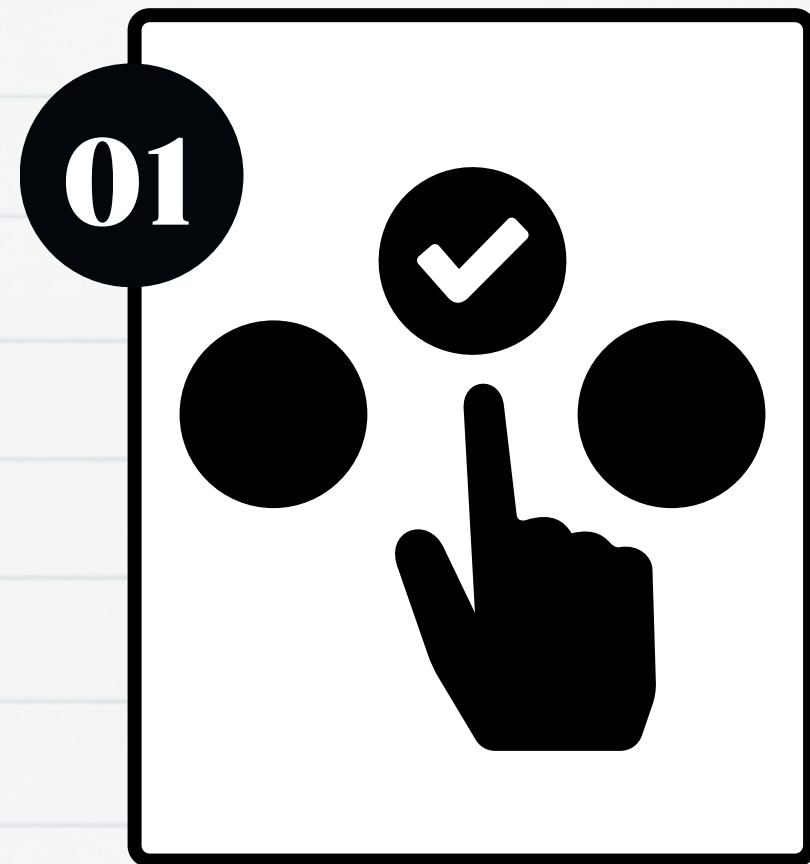


```
LOGGING = {  
    "version": 1,  
    "disable_existing_loggers": False,  
    "formatters": {  
        "simple": {  
            "format": "{levelname} {message}",  
            "style": "{}",  
        },  
    },  
    "filters": {  
        "require_debug_true": {  
            "()": "django.utils.log.RequireDebugTrue",  
        },  
    },  
    "handlers": {  
        "console": {  
            "level": "INFO",  
            "filters": ["require_debug_true"],  
            "class": "logging.StreamHandler",  
            "formatter": "simple",  
        },  
    },  
    "loggers": {  
        "django": {  
            "handlers": ["console"],  
            "propagate": True,  
        },  
    },  
}
```



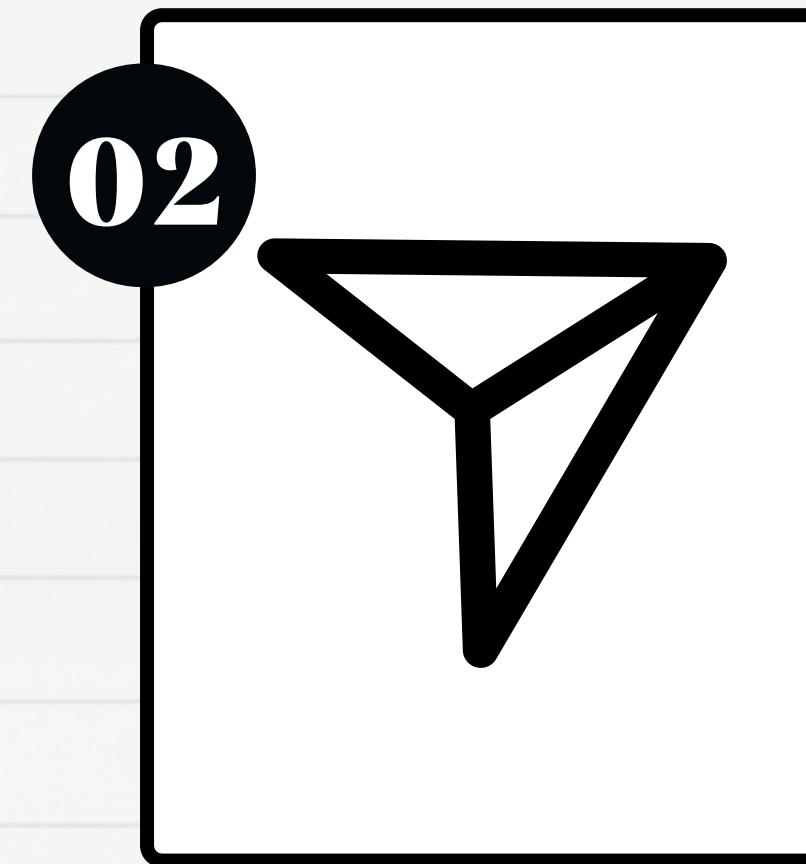
Configuración en el settings de Django

Logger



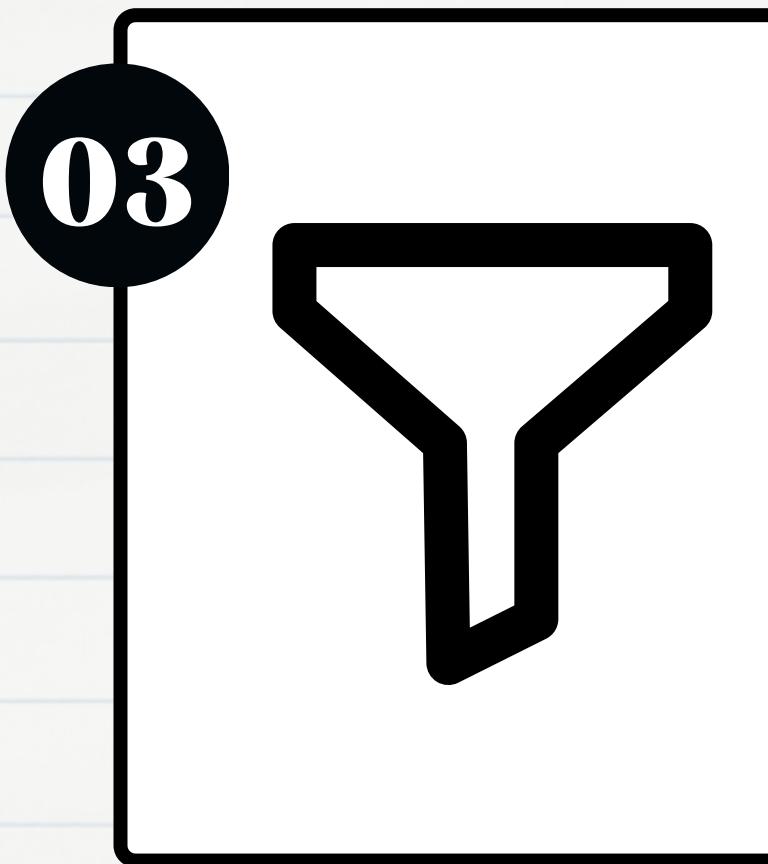
Que información debemos ver.

Handler



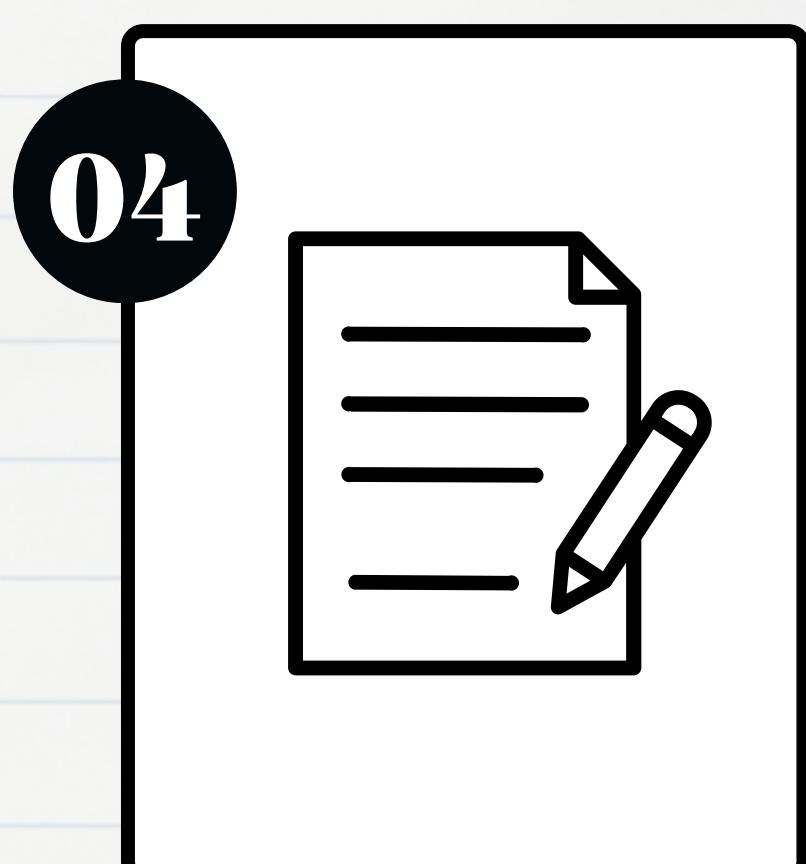
Que hacemos con la información.

Filter



Purgamos la información

Formatter



Como vemos la información

Loggers

```
LOGGING = {  
    # ...  
    "loggers": {  
        "django": { # Del cual parten los siguientes loggers  
            # ...  
        }  
        "root": {},  
        "django.request": {}, # (Solicitudes HTTP 4XX y 5XX)  
        "django.db.backends": {} # (Consultas y Operaciones en SQL)  
        "django.template": {}, # (Sistema de Plantillas de Django)  
        "django.security": {},  
        "django.server": {}, # (Solicitudes de desarrollo al ejecutar 'runserver')  
        "django.admin": {}, # (Acciones en el admin de Django)  
        "django.signals": {}, # (Pre-save y post-save)  
        "django.mail": {}, # (Correos electronicos dentro de Django: 'send_mail')  
        "django.template.context_processors": {},  
        "mail_admins": {}, # (Correos a los administradores en caso de Error)  
        "myapp": {},  
        "myapp.view": {},  
        "third_party_apps": {}, # (Como asyncio, requests, urllib3)  
    }  
}
```

Handlers



```
LOGGING = {  
    # ...  
  
    "handlers": {  
        "StreamHandler": { # Envía los mensajes de logging a la salida estándar (como la consola)  
            # ...  
        }  
        "FileHandler": {}, # Escribe los mensajes de logging en un archivo.  
        "WatchedFileHandler": {} # Similar a FileHandler, diseñado para reabrir el archivo de log si cambia  
        "RotatingFileHandler": {}, # Cambia de archivo cuando este alcanza cierto tamaño  
        "TimedRotatingFileHandler": {}, # Es útil para organizar los logs en archivos por intervalo.  
        "SocketHandler": {}, # Envía los mensajes de logging a un socket de red  
        "SMTPHandler": {}, # Envía los mensajes de logging por correo electrónico  
        "NullHandler": {}, # (Pre-save y post-save)  
        "HttpHandler": {}, # (Correos electronicos dentro de Django: 'send_mail')  
        "django.template.context_processors": {},  
        "AdminEmailHandler": {}, # (Correos a los administradores en caso de Error)  
    }  
}
```



Filters



```
class ImportantFilter(logging.Filter):
    def filter(self, record):
        # Verifica si la palabra "importante" está en el mensaje del log
        return 'importante' in record.getMessage()

LOGGING = {
    # ...
    'filters': {
        'important_filter': {
            '()': 'ImportantFilter',
        }
    },
}
```



Formatters

```
LOGGING = {
    # ...
    'formatters': {
        'verbose': {
            'format': '%(levelname)s %(asctime)s %(module)s %(message)s',
            'style': '%',
        },
        'simple': {
            'format': '%(levelname)s: %(message)s',
            'style': '%',
        },
    },
}
```



Niveles de Logging y su Aplicación en Django



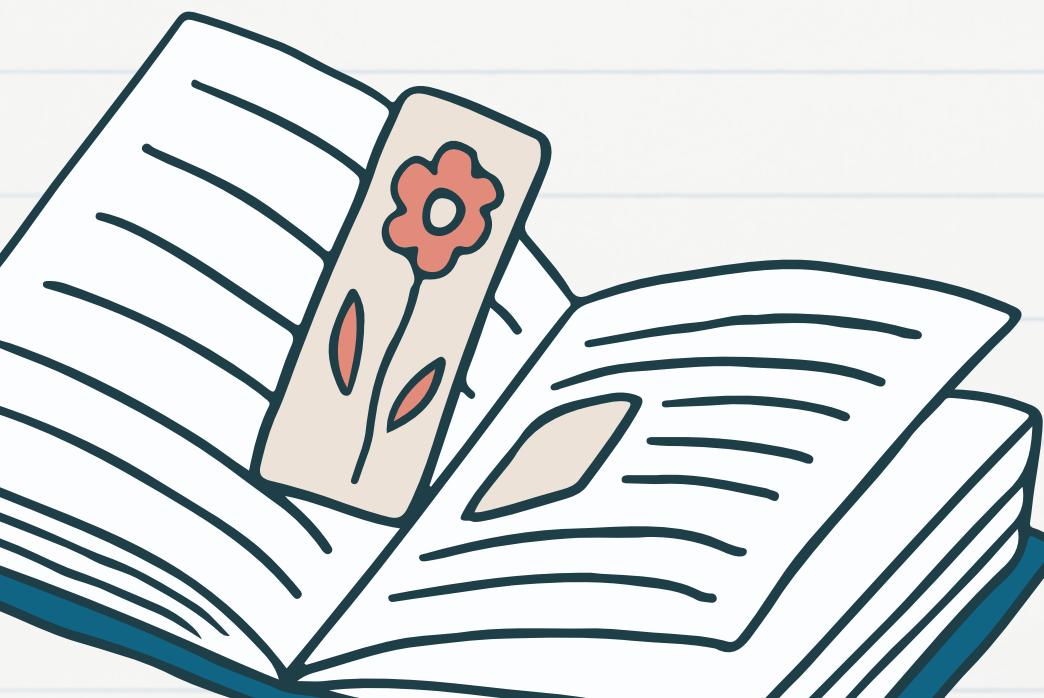
- DEBUG
- INFO
- WARNING
- ERROR
- CRITICAL

Secretos y

Trucos

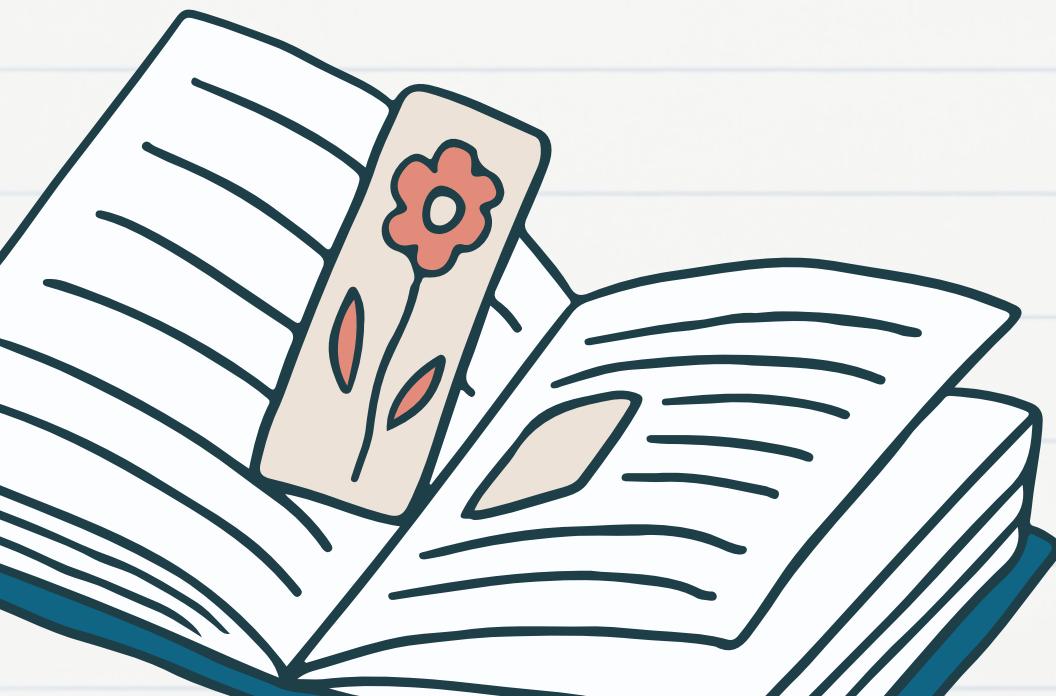
Avanzados en

Logging



- Crear logs por rangos de tiempo y eliminarlos automáticamente.
- “Centralized Logging”
- “Custom Logging”

Prácticas y Recomendaciones



- No guardar información sensible
(contraseñas, tarjetas de crédito/débito)
- No almacenar demasiada información,
pero si la suficiente (Espacio en Disco)
- Almacenar logs en un lugar seguro
(carpeta restringida por permisos, por
ejemplo.)
- Logs por tiempo (Rotating logs)



Gracias

