Practica 5. Patrones de diseño

Helena Solano

2025-09-29

```python
class Logger:
    #Creamos un atributo de clase donde se guarda la unica instancia
    _instancia = None

    # __new__ es el metodo que controla la creación del objeto antes de init.
Sirve para asegurarnos de que solo exista una unica
    # instancia de la clase Logger
    def __new__(cls, *args, **kwargs):
        # *args es un argumento posicional que permite reciber multiples
parametros.
        # **kwargs permite cualquier cantidad de parametros con nombre
        #Validar si existe o no la instancia aun:
        if cls._instancia is None:
            cls._instancia = super().__new__(cls) #Creamos instancia de logger
            # Agregando un atributo "archivo" que apunta a un archivo físico
            # "a" significa appened = Todo lo que se escriba se agrega al
final del archivo.
            cls._instancia.archivo = open("app.log", "a")
        return cls._instancia #Devolvermos siempre la misma instancia

    def log(self, mensaje):
        #Simulando un registro de logs
        self.archivo.write(mensaje + "\n")
        self.archivo.flush() #Método para guardar en el disco

logger1 = Logger() #Creamos la primera y unica instancia
logger2 = Logger() #Devolver la misma instancia, sin crear una nueva

logger1.log("Inicio de sesión en la aplicación")
logger2.log("El usuario se utenticó")

# Comprobar que son el mismo objeto de memoria
print(logger1 is logger2) #Devuelve true o false

# Actividad de la práctica
```

```python
class Presidente:
    _instancia = None

    def __new__(cls, nombre):
        if cls._instancia is None:
            cls._instancia = super().__new__(cls)
            cls._instancia.nombre = nombre
            cls._instancia.historial = []
        return cls._instancia

    def accion(self, accion):
        evento = f"{self.nombre} {accion}"
        self.historial.append(evento)
        print(evento)

p1 = Presidente("AMLO")
p2 = Presidente("Peña Nieto")
p3 = Presidente("Fox")

#Todos apuntan al mismo presidente
p1.accion("firmó decreto")
p2.accion("visitó España")
p3.accion("aprobó presupuesto")

print("\nHistorial del presidente:")
print(p1.historial)

# Validación de singleton
print(p1 is p2 is p3) #true o false
```

```
True
AMLO firmó decreto
AMLO visitó España
AMLO aprobó presupuesto

Historial del presidente:
['AMLO firmó decreto', 'AMLO visitó España', 'AMLO aprobó presupuesto']
True
```

1. ¿Qué pasaría si eliminamos la verificación if cls._instancia is None en el metodo new? Cada vez que creemos un objeto se va a generar una instancia diferente.
2. ¿Qué significa el “True” en p1 is p2 is p3 en el contexto del metodo singleton? Significa que solo existe una instancia y # que son el mismo objeto de memoria.
3. ¿Es buena idea usar Singleton para todo lo que sea global? Menciona un ejemplo donde no sería recomendable. No, porque se pueden ocasionar problemas si muchos procesos utilizan

la misma instancia. Ejemplo no recomendable: Un usuario en sesión dentro de una aplicación web. Cada persona que se conecte debe tener su propia instancia con su información.