Patrón de Diseño Memento: Guardando y Restaurando Estados

En el desarrollo de software, a menudo necesitamos la capacidad de "deshacer" acciones o regresar a un estado anterior de la aplicación. Aquí es donde el Patrón de Diseño Memento brilla, permitiéndonos gestionar estos estados de manera elegante y eficiente.

¿Qué es el Patrón Memento?

Propósito Central

Es un patrón de comportamiento que captura y guarda el estado interno de un objeto sin violar su encapsulación. Esto permite restaurar el objeto a un estado previo, ideal para funciones de deshacer/rehacer.

Componentes Clave

- **Originador:** Objeto cuyo estado se guarda.
- **Memento:** Almacena el estado interno del Originador.
- **Caretaker:** Gestiona los mementos (guarda y recupera) sin modificar su contenido.

El patrón Memento desacopla completamente el almacenamiento del estado del objeto cuyo estado se está guardando, asegurando una gran flexibilidad y mantenibilidad en sistemas complejos.

Ejemplo: Editor de Texto con Deshacer

Imaginemos un editor de texto simple donde queremos implementar la funcionalidad de "deshacer". Aquí, el **Originador** sería el `EditorDeTexto`, el **Memento** sería el `EstadoEditor`, y el **Caretaker** sería el `HistorialEditor`.

Cada vez que el texto cambia, el editor crea un memento de su estado actual y lo añade al historial. Cuando el usuario quiere deshacer, el historial le proporciona el memento anterior para restaurar el estado del editor.

```
class EstadoEditor:
 def __init__(self, contenido):
 self._contenido = contenido

 def get_contenido(self):
 return self._contenido

class EditorDeTexto:
 def __init__(self):
 self._contenido = ""

 def escribir(self, texto):
 self._contenido += texto

 def crear_memento(self):
 return EstadoEditor(self._contenido)

 def restaurar_memento(self, memento):
 self._contenido = memento.get_contenido()

 def get_contenido(self):
 return self._contenido

class HistorialEditor:
 def __init__(self):
 self._mementos = []

 def guardar(self, memento):
 self._mementos.append(memento)

 def deshacer(self):
 if not self._mementos:
 return None
 return self._mementos.pop()

# Uso del patrón
editor = EditorDeTexto()
historial = HistorialEditor()

editor.escribir("Hola")
historial.guardar(editor.crear_memento())

editor.escribir(" mundo")
historial.guardar(editor.crear_memento())

editor.escribir("!")
print(f"Estado actual: {editor.get_contenido()}")

editor.restaurar_memento(historial.deshacer())
print(f"Deshacer 1: {editor.get_contenido()}")

editor.restaurar_memento(historial.deshacer())
print(f"Deshacer 2: {editor.get_contenido()}")
```