
# Patrón de Diseño Prototype: Clonando Objetos para Crear con Eficiencia

Exploraremos el patrón de diseño Prototype, una herramienta poderosa para la creación eficiente de objetos, especialmente útil cuando el proceso de inicialización es complejo o costoso.

# ¿Qué es el Patrón Prototype?

### Creación por Copia

Es un patrón creacional que permite crear nuevos objetos copiando un objeto existente, conocido como prototipo, en lugar de inicializarlos desde cero.

### Eficiencia y Complejidad

Es ideal cuando la creación de objetos es costosa o compleja, ya que evita repetir procesos intensivos en recursos.

### Copias Superficiales vs. Profundas

La clonación puede ser superficial (shallow copy), copiando solo referencias, o profunda (deep copy), copiando recursivamente todos los objetos anidados.

### Soporte en Python

En Python, se facilita con el módulo `copy`, que ofrece las funciones `copy.copy()` para copias superficiales y `copy.deepcopy()` para copias profundas.

# Ejemplo en Python: Clonando un Prototipo con deepcopy

## Definición del Prototipo

Imaginemos que tenemos un objeto `Documento` con contenido y una lista de autores.

```
import copy

class Documento:
    def __init__(self, nombre, contenido, autores):
        self.nombre = nombre
        self.contenido = contenido
        self.autores = autores

    def __str__(self):
        return f"Documento: {self.nombre}\nContenido:
{self.contenido[:20]}...\nAutores: {', '.join(self.autores)}"

    def clonar(self, tipo_copia="deep"):
        if tipo_copia == "deep":
            return copy.deepcopy(self)
        else:
            return copy.copy(self)
```

## Clonación y Modificación

Creamos un prototipo y luego una copia profunda para asegurarnos de que los cambios en la copia no afecten al original.

```
# Crear un prototipo de documento
doc_original = Documento("Informe Anual", "Detalles
financieros...", ["Alice", "Bob"])
print("Original:", doc_original)

# Clonar el documento
doc_clonado = doc_original.clonar("deep")
doc_clonado.nombre = "Informe Trimestral"
doc_clonado.autores.append("Charlie")
doc_clonado.contenido = "Resultados Q1..."

print("\nClonado (modificado):", doc_clonado)
print("Original (después de clonar y modificar):",
doc_original)
```

Este ejemplo demuestra cómo el uso de `deepcopy` asegura que incluso las listas mutables (como `autores`) y las cadenas de texto se clonen completamente, manteniendo la independencia entre el objeto original y su clon.