</

Patrón de diseño Proxy

/>

Leonardo Martin Bacelis Bautista

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

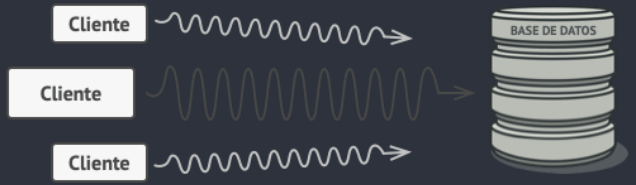
</Proxy

- Proxy es un patrón de diseño estructural que te permite proporcionar un sustituto o marcador de posición para otro objeto. Un proxy controla el acceso al objeto original, permitiéndote hacer algo antes o después de que la solicitud llegue al objeto original.

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</Problema

- ¿Por qué querrías controlar el acceso a un objeto? Imagina que tienes un objeto enorme que consume una gran cantidad de recursos del sistema. Lo necesitas de vez en cuando, pero no siempre.



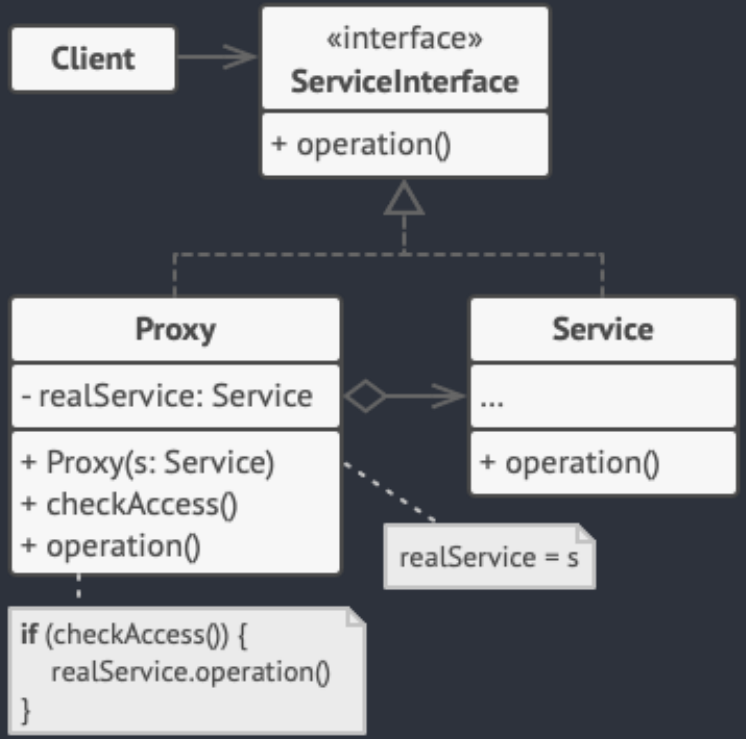
- Puedes llevar a cabo una implementación diferida, es decir, crear este objeto sólo cuando sea realmente necesario. Todos los clientes del objeto tendrán que ejecutar algún código de inicialización diferida. Lamentablemente, esto seguramente generará una gran cantidad de código duplicado.

- En un mundo ideal, querríamos meter este código directamente dentro de la clase de nuestro objeto, pero eso no siempre es posible. Por ejemplo, la clase puede ser parte de una biblioteca cerrada de un tercero.

</Solución

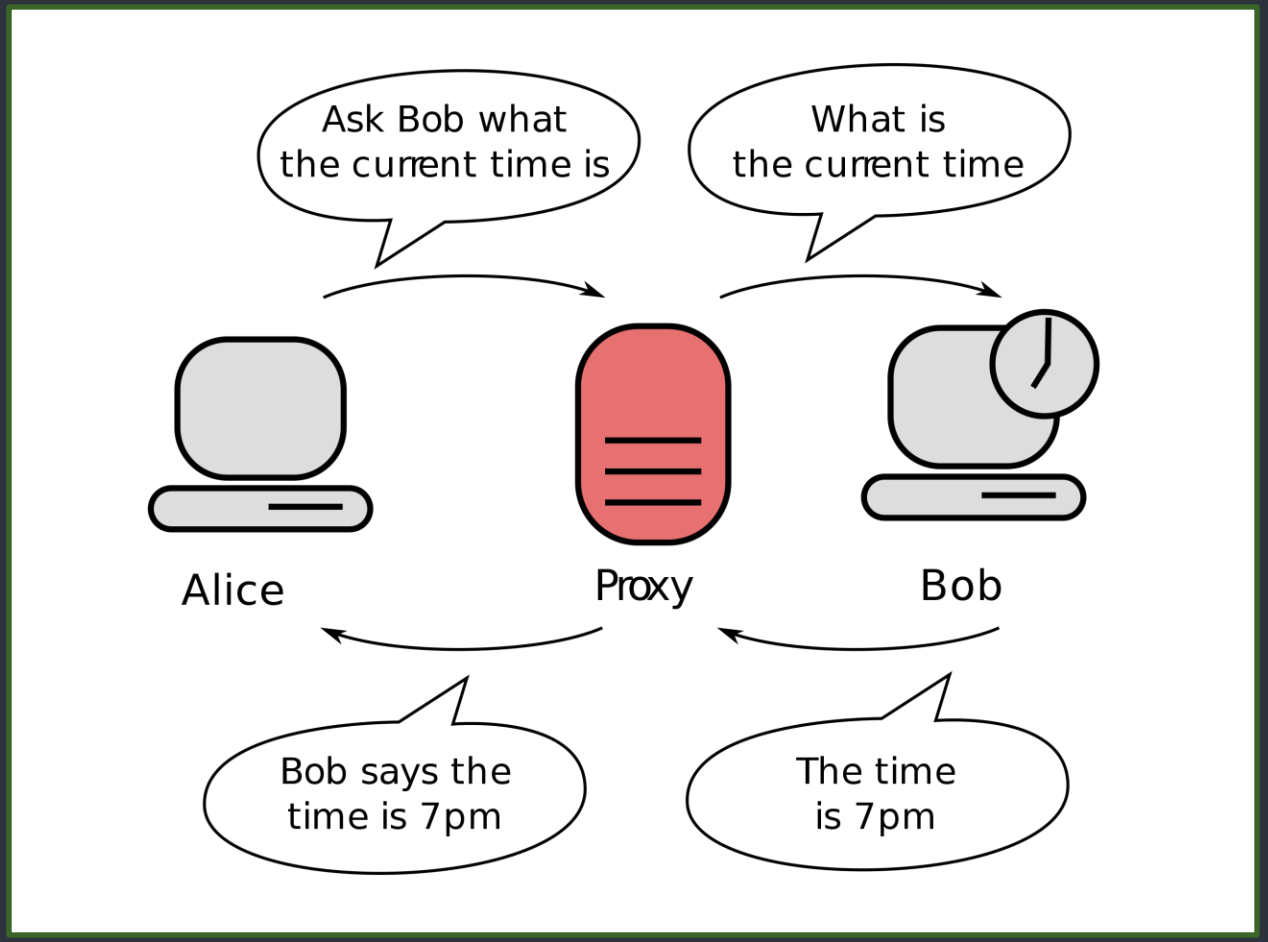
- El patrón Proxy sugiere que crees una nueva clase proxy con la misma interfaz que un objeto de servicio original. Después actualizas tu aplicación para que pase el objeto proxy a todos los clientes del objeto original. Al recibir una solicitud de un cliente, el proxy crea un objeto de servicio real y le delega todo el trabajo.

</Estructura


Client
«interface»
ServiceInterface
+ operation()
Proxy
- realService: Service
+ Proxy(s: Service)
+ checkAccess()
+ operation()
Service
...
+ operation()
realService = s
if (checkAccess()) {
realService.operation()
}

</Aplicabilidad

Proxy	Descripción
Remote proxy	✓ Acceder a objecto en otra dirección
Virtual proxy	✓ Crea objetos que usan muchos recursos cuando sea necesario
Protection proxy	✓ Controla el acceso al objeto
Cache proxy	✓ Almacena resultados usados frecuentenmente
Firewall proxy	✓ Proteger los objetos de ataques
Synchronization proxy	✓ Acceso concurrente seguro
Counting proxy	✓ Conteo de ciertas acciones


Ask Bob what
the current time is
What is
the current time
Alice
Proxy
Bob
Bob says the
time is 7pm
The time
is 7pm

</Imprementación

- Clase Servidor(abstract)
- Clase MiServidor
- Clase ProxyMiServidor

```python
import os
from abc import ABC, abstractmethod

class Servidor(ABC):
    @abstractmethod
    def descargar(self, url):
        pass
```

</Imprementación

- Clase Servidor(abstract)
- Clase MiServidor
- Clase ProxyMiServidor

```python
class MiServidor(Servidor):
    def __init__(self, p, h):
        self.puerto = p
        self.host = h
    def descargar(self, url):
        return "Descargando para " + self.host
```

</Imprementación

- Clase Servidor(abstract)
- Clase MiServidor
- Clase ProxyMiServidor

```
class ProxyMiServidor(Servidor):
    def __init__(self, p, h):
        self.puerto = p
        self.host = h
        self.miservidor = None
    def descargar(self, url):
        if(self.validar(url)):
            if(self.miservidor == None):
                self.miservidor = MiServidor(self.puerto, self.host)
            resultado = self.miservidor.descargar(url)
        else:
            resultado = "Desde " + self.host + " no puedes descargar"
        return resultado
    def validar(self, url):
        if (url == "/home/leonardo/Escritorio/ejemplo/Bob"):
            return True
        return False
```

</Pros y contras

Pros	Contras
✓ Puedes controlar el objeto de servicio sin que los clientes lo sepan.	El código puede complicarse ya que debes introducir gran cantidad de clases nuevas.
✓ Puedes gestionar el ciclo de vida del objeto de servicio cuando a los clientes no les importa.	La respuesta del servicio puede retrasarse.
✓ El proxy funciona incluso si el objeto de servicio no está listo o no está disponible.	
✓ Principio de abierto/cerrado. Puedes introducir nuevos proxies sin cambiar el servicio o los clientes.	

Gracias!

Leonardo Martin Bacelis Bautista