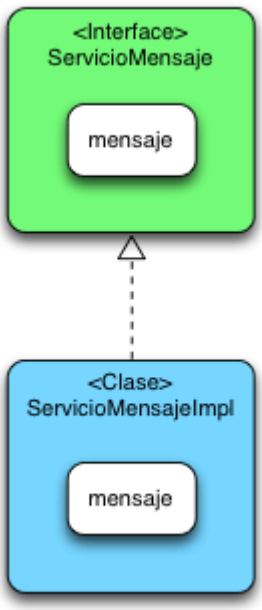
Java Proxy Pattern es uno de los patrones de diseño que más se usa en Java. Lamentablemente a veces es difícil de entender como funciona y sobre todo de en que frameworks se utiliza para solventar los diferentes problemas del desarrollo de aplicaciones.¿ Qué es exactamente un Proxy? .Vamos a intentar explicarlo de forma breve.

Java Proxy Pattern y responsabilidades

En Java las clases implementan una funcionalidad concreta a través de sus distintos métodos. Vamos a suponer que tenemos una clase de servicio con un método que nos devuelve el mensaje “hola amigo pepito”, recibiendo como parámetro el nombre que deseemos. Para ello construiremos una interface y una clase que la implemente.



Vamos a ver el resultado a nivel de código:

```
package com.arquitecturajava.proxy;

public interface ServicioMensaje {
public String mensaje(String persona);
}
```

```
package com.arquitecturajava.proxy2;

public class ServicioMensajeImpl implements ServicioMensaje {

 public String mensaje(String persona)
{

 return "hola amigo "+ persona;
 }
}
```

Ahora usaremos un programa principal para crear el servicio y ejecutarlo.

```
package com.arquitecturajava.proxy;

public class Principal {

public static void main(String[] args) {

ServicioMensaje sm= new ServicioMensajeImpl();
```

```
System.out.println(sm.mensaje("pepito"));

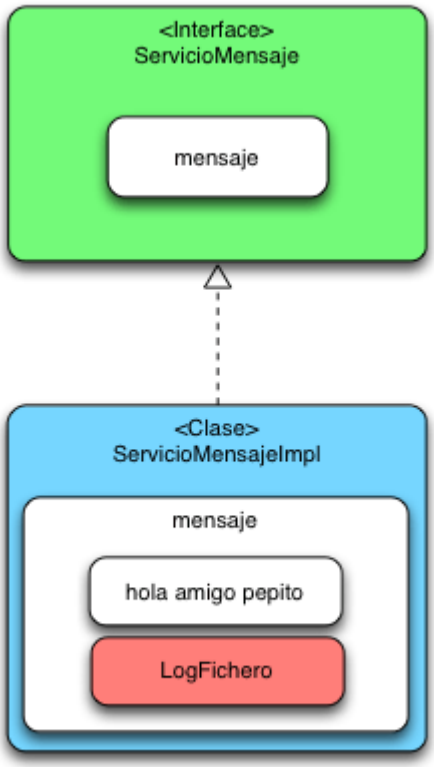
}

}
```

El resultado aparecerá en la consola:



Todo funciona correctamente y es un enfoque clásico. Ahora bien que sucede si necesitamos que cada vez que invocamos al método mensaje se realice un log de la petición a un fichero. La solución más sencilla es incluir esta funcionalidad en el método mensaje.



Es cierto que es la más sencilla pero también es la más pobre ya que no podremos reutilizar la clase salvo que en el nuevo programa también realicemos un log. Esta claro que tenemos un problema de diseño ya que la clase tiene dos responsabilidades muy dispares y no cumple con el principio SRP. ¿Cómo podemos solventarlo?. Para eso existe el patrón Proxy.

El concepto de Java Proxy Pattern

¿Que es un Proxy a nivel de Design Pattern? . Un proxy no es mas que un intermediario que se diseña para apoyar a otra clase y dotarla de funcionalidad adicional. En nuestro caso vamos a extraer la funcionalidad de Log a otra clase ServicioMensajeProxy. De esta forma no hará falta modificar la clase original sino que el proxy delegará en ella.

```
package com.arquitecturajava.proxy2;

public class ServicioMensajeProxy implements ServicioMensaje {
private ServicioMensaje sm;


@Override
public String mensaje(String persona) {

System.out.println("log del mensaje para"+ persona);
//mensaje delegado
return sm.mensaje(persona);
}
public ServicioMensajeProxy() {
super();
this.sm = new ServicioMensajeImpl();
}

}
```

Ahora será suficiente con crear primero el proxy e invocarle para que todo se ejecute de forma ordenada.

```
package com.arquitecturajava.proxy2;

public class Principal {

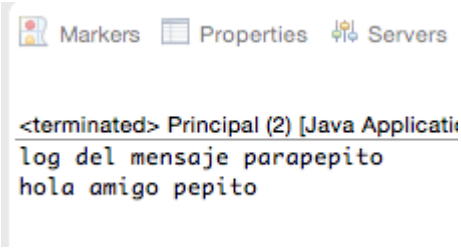
public static void main(String[] args) {
```

```
ServicioMensaje sm= new ServicioMensajeProxy();
System.out.println(sm.mensaje("pepito"));

}

}
```

El resultado se muestra por consola:



Acabamos de construir un Proxy ,uno de los patrones que más se usa en frameworks como Spring o standards como EJB. La clave es entender que se trata de un intermediario que añade funcionalidad y modifica el comportamiento por defecto de nuestras clases.

Otros artículos relacionados: Patrón Factoría , Patrón Singleton