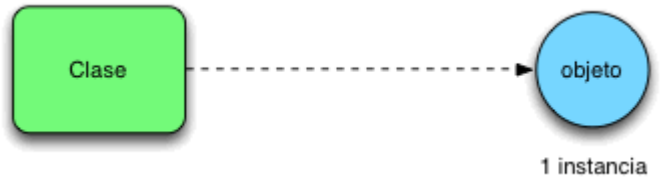
El patrón Singleton es uno de los más utilizados y se encarga de construir una clase de la cual únicamente se pueda construir un objeto. La construcción el patrón no es complicada pero a la gente que empieza le cuesta hacerse una idea de cual pueden ser sus usos. Muchas veces no parece tener mucho sentido crear una clase que solo pueda tener un objeto.



Java Singleton Properties

Vamos a usar un fichero de Properties para clarificar el concepto. Recordemos que un fichero de Properties es un fichero Java que contiene clases/valores. En este caso contendrá los parámetros de conexión a una base de datos.

url=localhost
usuario=root
password=123456

Este fichero puede ser leído desde Java y acceder la información que contiene utilizando la clase Properties de java.util.

```
package com.arquitecturajava.singleton;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.Properties;
```

```
public class Principal {

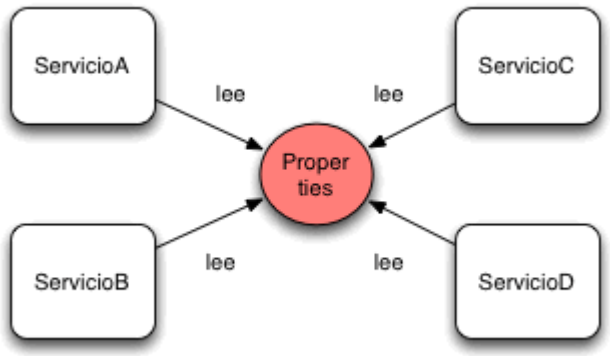
 public static void main(String[] args) {

 Properties p= new Properties();
 try {
 p.load(new FileInputStream(new File("basedatos.properties")));
 System.out.println(p.getProperty("url"));
 } catch (IOException e) {
 // TODO Auto-generated catch block
 e.printStackTrace();

 }
 }

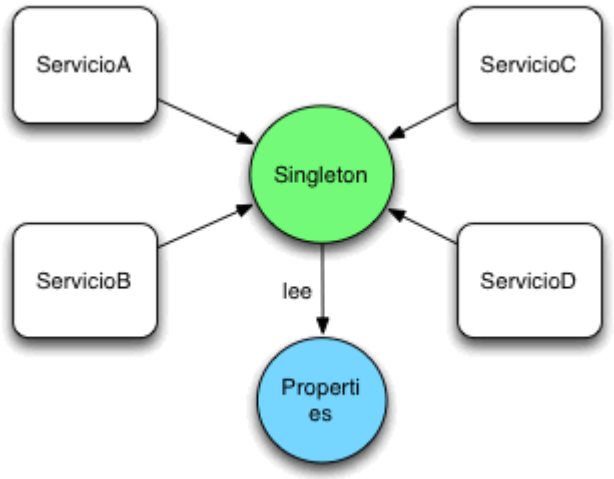
}
```

Esta clase lee el fichero basedatos.properties y nos imprime por consola “localhost”. El problema fundamental es que cada vez que queramos utilizar el fichero de propiedades deberemos leer el fichero de disco. Esto no es muy práctico ya que los parámetros nunca cambian.



La solución pasa por crear una clase Singleton y que esta lea la primera vez el fichero y lo

mantenga en memoria para el resto de ciclo de vida de la aplicación.



El código de la clase podría ser similar a lo siguiente:

```
package com.arquitecturajava.singleton;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.Properties;

public class SingletonProperties {

private static SingletonProperties instancia=null;
private Properties p;
private SingletonProperties() {

p= new Properties();
try {
```

```
p.load(new FileInputStream(new File("basedatos.properties")));
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}

public static SingletonProperties getInstancia() {

if (instancia==null) {

instancia=new SingletonProperties();
}
return instancia;
}

public String getPropiedad(String clave) {

return p.getProperty(clave);
}
}
```

Creada la clase Singleton el programa Main quedaría de la siguiente forma:

```
package com.arquitecturajava.singleton;

public class PrincipalSingleton {
```

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

SingletonProperties sp=SingletonProperties.getInstancia();
System.out.println(sp.getPropiedad("url"));

}

}
```

Los Java Singleton Properties os pueden ayudar en muchas situaciones:

Otros artículos relacionados:

Singleton ClassLoaders

Spring Singleton vs Prototype

EJB Singleton