El uso de ficheros de Java Properties es una de las cosas más comunes que tenemos que abordar en el día a día de la programación en Java . Siempre hay cosas de un programa que deseamos parametrizar de forma rápida y externalizarlo . Los ficheros de Java Properties existen desde la primera versión del lenguaje y nos permiten simplificar este tipo de situaciones de forma rápida . Vamos a verlo:

Supongamos que tenemos que parametrizar el acceso a una base de datos usando Java . Normalmente nos encontraremos con un código Java similar a este:

```
package com.arquitecturajava;

public class Principal {

	public static void main(String[] args) {

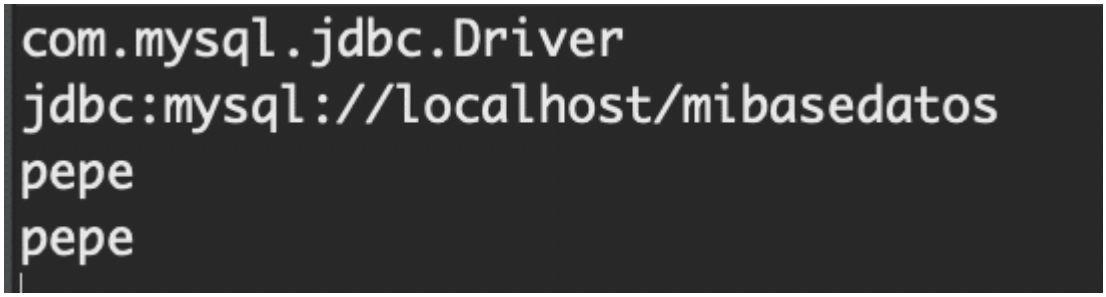
		final String DRIVER = "com.mysql.jdbc.Driver";
		final String URL =
"jdbc:mysql://localhost/mibasedatos";

		final String USUARIO = "pepe";
		final String CLAVE = "pepe";
		System.out.println(DRIVER);
		System.out.println(URL);
		System.out.println(USUARIO);
		System.out.println(CLAVE);

	}

}
```

Si ejecutamos el código veremos el resultado por la consola:

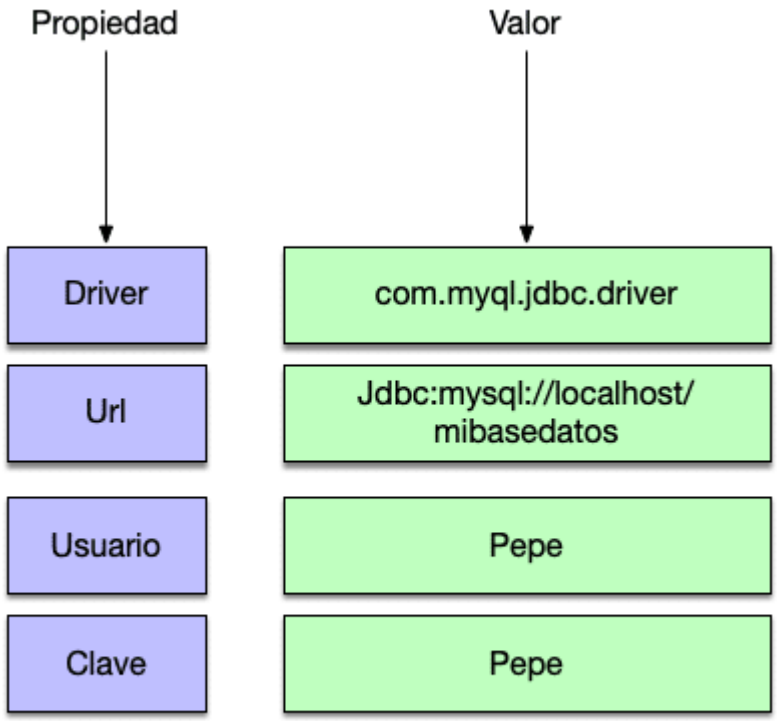


Java Properties

Es evidente que no queremos tener integrado en el código la cadena de conexion , el driver usuario y password . Sino que preferiremos tenerlo en un fichero aparte de tal forma que podamos cambiar a voluntad la base de datos de destino y el usuario que se conecta. Para solventar este problema disponemos de los ficheros de propiedades. Los ficheros de propiedades disponen de una pareja clave valor por linea en la cual se definen las diferentes propiedades.

```
DRIVER=com.mysql.jdbc.Driver
jdbc=mysql://localhost/mibasedatos
USUARIO=pepe
CLAVE=pepe
```

La estructura es muy sencilla :



configuracion.properties

Una vez que tenemos el fichero de propiedades , el siguiente paso es construir el código que nos lea los valores que están almacenados en él.

```
package com.arquitecturajava;

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

public class Principal {

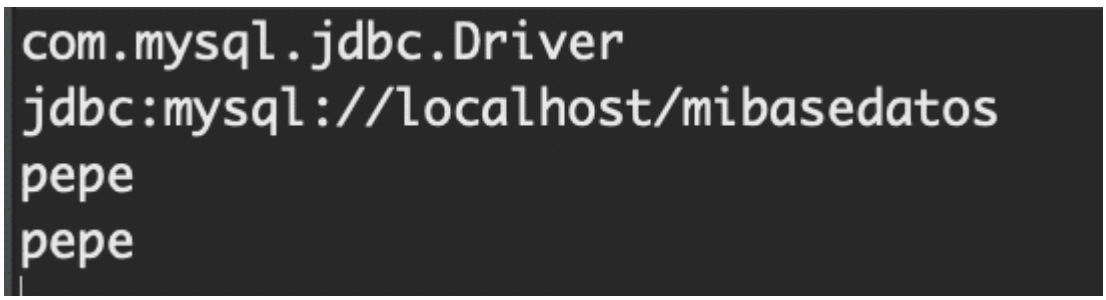
        public static void main(String[] args) {
```

```
                Properties properties= new Properties();
                try {
                        properties.load(new FileInputStream(new
File("configuracion.properties")));
                        System.out.println(properties.get("DRIVER"));
                        System.out.println(properties.get("URL"));
                        System.out.println(properties.get("USUARIO"));
                        System.out.println(properties.get("CLAVE"));
                } catch (FileNotFoundException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

        }

}
```

El resultado será el mismo :



Solamente que ahora hemos conseguido externalizar las propiedades y que no haga falta recompilar el programa para que funcione. El manejo de properties siempre ha sido algo bastante flexible y permite gestionar de forma directa ficheros XML.

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
    <entry key="DRIVER">com.mysql.jdbc.Driver</entry>
    <entry key="URL">jdbc:mysql://localhost/mibasedatos</entry>
    <entry key="USUARIO">pepe</entry>
     <entry key="CLAVE">pepe</entry>
</properties>
```

En este caso la estructura es algo más compleja . Evidentemente tendremos que actualizar nuestro código de Java para leer los datos de este nuevo tipo de fichero . En esta situación todo es sencillo ya que simplemente basta con cambiar un método.

```
package com.arquitecturajava;

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

public class Principal2 {

        public static void main(String[] args) {

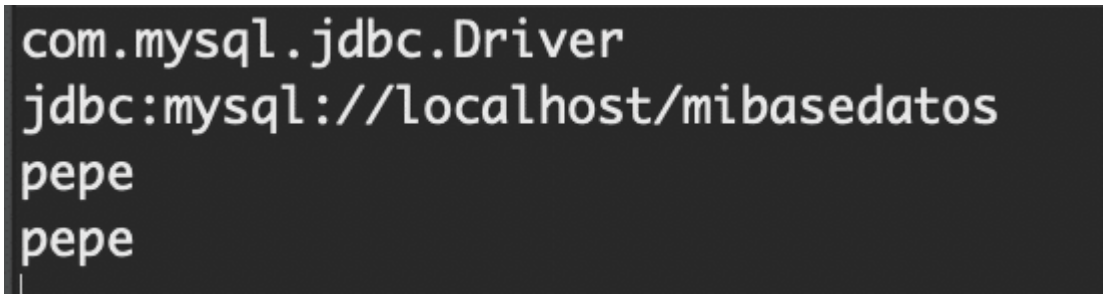
                Properties properties= new Properties();
                try {
                        properties.load(new FileInputStream(new
File("configuracion.properties")));
                        System.out.println(properties.get("DRIVER"));
                        System.out.println(properties.get("URL"));
                        System.out.println(properties.get("USUARIO"));
```

```
                System.out.println(properties.get("CLAVE"));
            } catch (FileNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

    }

}
```

El resultado es el mismo pero hay gente que prefiere esa sintaxis sobre la clásica ya que ayuda a validarlo con un DTD. El resultado es idéntico.



Otros artículos relacionados

- Spring Properties
- Properties y Encriptacion
- Java Files Walk
- Properties