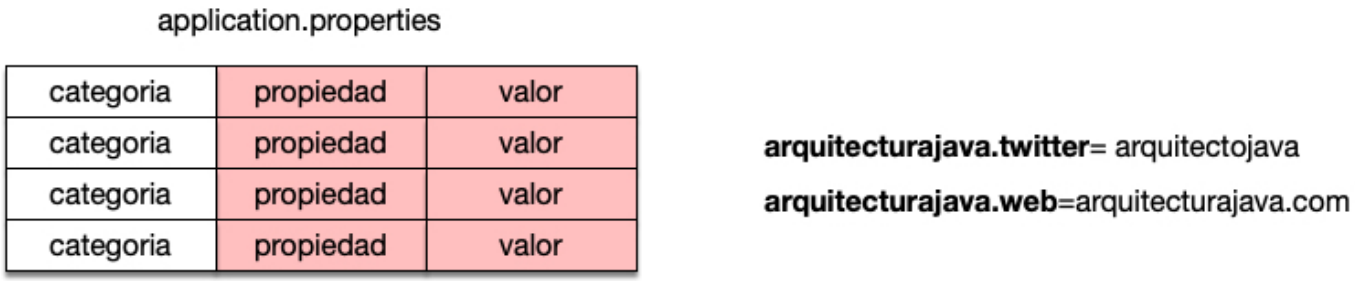
Spring Boot Yaml es una segunda opción que tenemos al uso clásico de ficheros de propiedades. En principio estos ficheros son prácticos y bastante sencillos de utilizar pero según la aplicación va creciendo en tamaño se hacen un poco más ilegibles ya que el conjunto de propiedades que almacenamos puede llegar a ser grande.

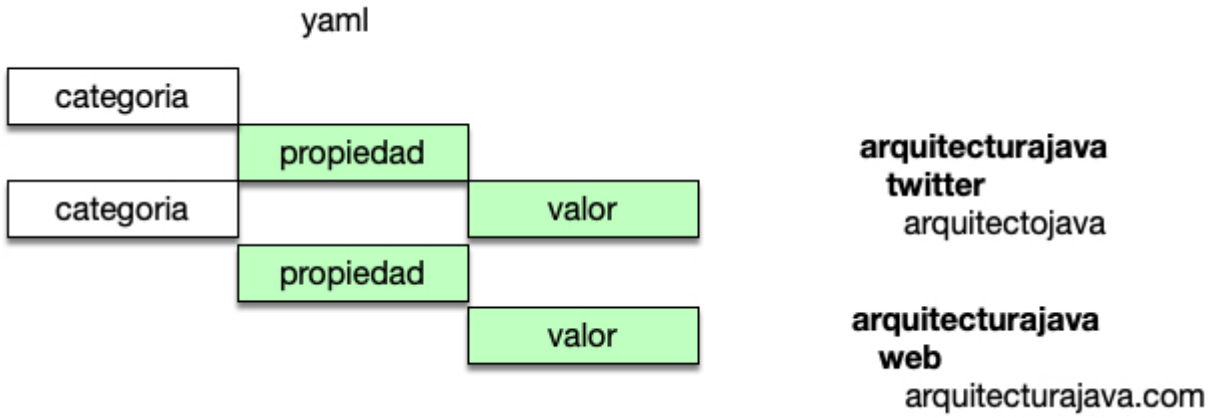


Vamos a construir un pequeño proyecto con Spring boot y probar estos nuevos ficheros YAML. Para ello como siempre lo primero será configurar las dependencias del proyecto de Spring Boot.

```
        <dependencies>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
thymeleaf</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
```

```
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>
```

Una vez tenemos configuradas las dependencias el siguiente paso es generar un fichero YAML que nos almacene las propiedades.Estos ficheros están tabulados utilizando 3 espacios de tal forma que se facilita sobremanera su lectura.



Eclipse soporta plugins para que su lectura sea más sencilla.

```
datos:
   twitter:
      arquitectojava
   web:
```

arquitecturajava.com

Una vez que tenemos definido el fichero YAML el siguiente paso es definir un fichero de configuración que cargue las propiedades para posteriormente inyectarlo en un controlador.

```
package com.arquitecturajava.yaml;

import
org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.stereotype.Component;

@Component
@ConfigurationProperties(prefix="datos")
public class ConfiguradorProperties {

	private String twitter;

	private String web;

	public String getTwitter() {
		return twitter;
	}

	public void setTwitter(String twitter) {
		this.twitter = twitter;
	}

	public String getWeb() {
		return web;
```

```
	}

	public void setWeb(String web) {
		this.web = web;
	}

}
```

Este fichero ese encarga de leer las propiedades que están ubicadas en el fichero de YAML.Una vez leemos las propiedades es momento de inyectar esta clase en un Controlador para que podamos imprimir los datos en una sencilla página html utilizando thymeleaf.

```
package com.arquitecturajava.yaml;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
@RequestMapping("/web")
public class WebController {

	@Autowired
	ConfiguradorProperties propiedades;

	@RequestMapping("/misdatos")
	public String mostrarDatos(Model modelo) {
modelo.addAttribute("twitter",propiedades.getTwitter());
```

```
                modelo.addAttribute("web",propiedades.getWeb());
                return "web/datos";
        }
}
```

Ya solo nos queda mostrar los datos en una vista HTML:

```
</pre>
<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">
<head>

</head>
MiTwitter:
<p th:text="${twitter}"></p>

MiWeb:
<p th:text="${web}"></p>

</html>
```

Si Cargamos la página veremos como las propiedades se imprimen:

MiTwitter:

arquitectojava

MiWeb:

arquitecturajava.com

Acabamos de utilizar Spring boot Yaml para configurar los típicos ficheros de propiedades de una forma mucho mas legible para el programador.

1. Spring Boot JSP y su configuración
2. Spring Boot JPA y su configuración
3. Spring Boot Properties utilizando @Value
4. spring boot