Ficheros de propiedades con Spring

Spring te facilita la configuración de properties para poder cargarlos de diferentes formas, la más común a través de ficheros xml.

Imaginemos que tenemos el siguiente fichero de properties (pruebas.properties):

```
pruebas.nombre=Sergio
pruebas.apellidos=Gutiérrez
```

De esta forma lo primero que tendríamos que hacer sería registrar el fichero de propiedades en el XML de contexto de Spring o en cualquier otro fichero XML que se importe en este.

Esto lo podemos hacer de dos formas en el fichero XML del contexto de Spring:

1. A través de la definición de un Bean:

```
<bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
    <property name="locations" value="classpath:pruebas.properties" />
</bean>
```

2. A través del namespace context que te configura automáticamente un PropertyPlaceholderConfigurer en el contexto de Spring. Esta forma está soportada a partir de la versión 3.1 (3.1 incluida):

```
<context:property-placeholder location="classpath:pruebas.properties" />
```

Ya tenemos registrado el fichero de propiedades en el contexto de Spring. Ahora sólo nos queda utilizarlo. Para ello lo primero que vamos a hacer es crear una clase que contenga los dos atributos que queremos utilizar:

```
package pruebas.beans

public class Usuario {

    private String nombre;
    private String apellidos;

    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getApellidos() {
        return apellidos;
    }
    public void setApellidos(String apellidos) {
```

```
        this.apellidos = apellidos;
    }

}
```

Ahora vamos a configurar el bean en el fichero de contexto de Spring de la aplicación:

```
<bean id="usuario" class="pruebas.beans.Usuario">
    <property name="nombre" value="${pruebas.nombre}" />
    <property name="apellidos" value="${pruebas.apellidos}" />
</bean>
```

De esta forma cuando cogemos el bean usuario en la aplicación, bien con el método getBean o con un @Autowired, tendremos las propiedades con los valores que están establecidos en el fichero de propiedades.

Esta forma de hacerlo es todo a través de ficheros XML de configuración. Pero existe también otra posibilidad a partir de la versión 3.1 (3.1 incluida) y es a través de anotaciones.

Una opción sería tener el registro del properties en el fichero XML de configuración y a partir de ahí no meter las propiedades en la definición del bean usuario en el XML. Para ello cambiamos el bean en el fichero XML sería ahora:

```
<bean id="usuario" class="pruebas.beans.Usuario" />
```

Y la clase pruebas.beans.Usuario estaría configurada con anotaciones @Value para coger el valor de las propiedades de la siguiente forma:

```
package pruebas.beans

public class Usuario {

    @Value( "${pruebas.nombre}" )
    private String nombre;

    @Value( "${pruebas.apellidos}" )
    private String apellidos;

    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getApellidos() {
        return apellidos;
    }
    public void setApellidos(String apellidos) {
        this.apellidos = apellidos;
```

```
19      }
20
21}
22
23
```

Así, cuando cojamos el bean en la aplicación, también tendrá los valores de las propiedades establecidos como en el caso anterior cuando los pasábamos como propiedades en la definición del bean.

Spring 3.1 también introduce la posibilidad de configurar el fichero de properties a través de anotaciones utilizando *@PropertySource*, de la siguiente forma:

```
1@PropertySource("classpath:pruebas.properties")
```

aunque hay que utilizar esta anotación conjuntamente con *@Configuration* y tener activado las anotaciones en el fichero xml de contexto de Spring. Como inconveniente, comentar que el *@PropertySource* no registra automáticamente un *PropertySourcesPlaceholderConfigurer* por lo que el bean debe de estar explícitamente definido en la configuración.