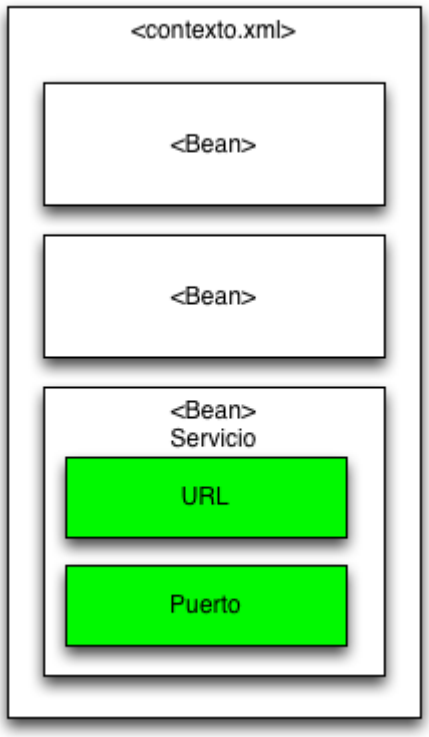
En muchas ocasiones cuando trabajamos con Spring Framework acabamos teniendo una serie de parametros configurados a nivel del fichero de contexto.xml (o applicationContext.xml) que están ligados a alguno de nuestros beans .Casos habituales son usuario y password de la base de datos , url de acceso a un servicio en concreto, dirección de correo a la que la aplicación envia mensajes etc .Imaginemonos que en nuestro caso disponemos de un bean de servicio que accede a una URL en un puerto determinado como muestra el siguiente fichero de Spring.

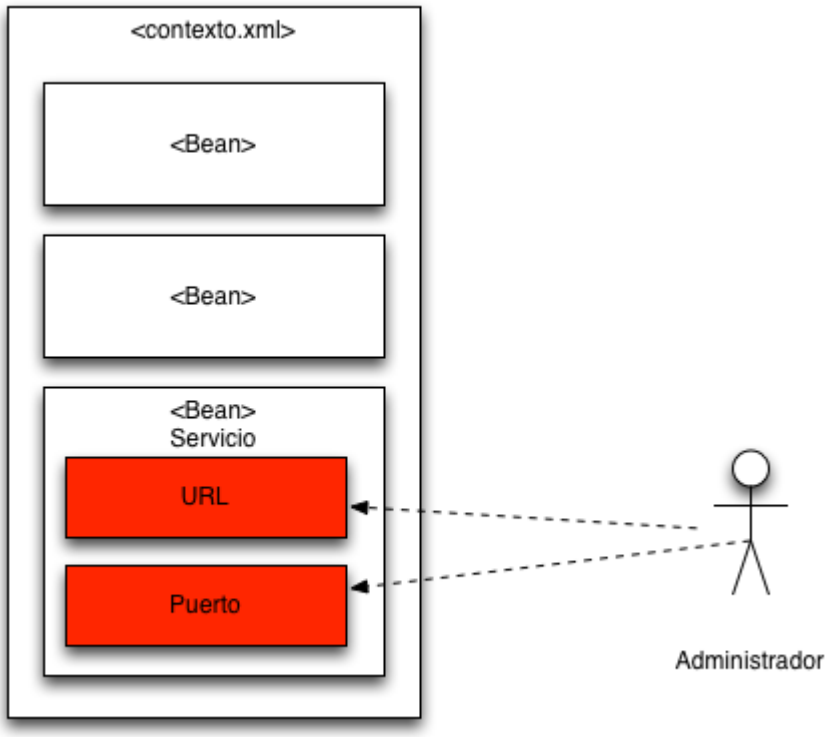
```
<?xml version="1.0" encoding="UTF-8"?></pre>
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.com/schema/context"
 xsi:schemaLocation="http://www.springframework.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
 http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd">

<bean id="servicio" class="com.arquitecturajava.properties.Servicio">
 <property name="url" value="http://tomcatRemoto"></property>
 <property name="puerto" value="8080"></property>
 </bean>
</beans>
```

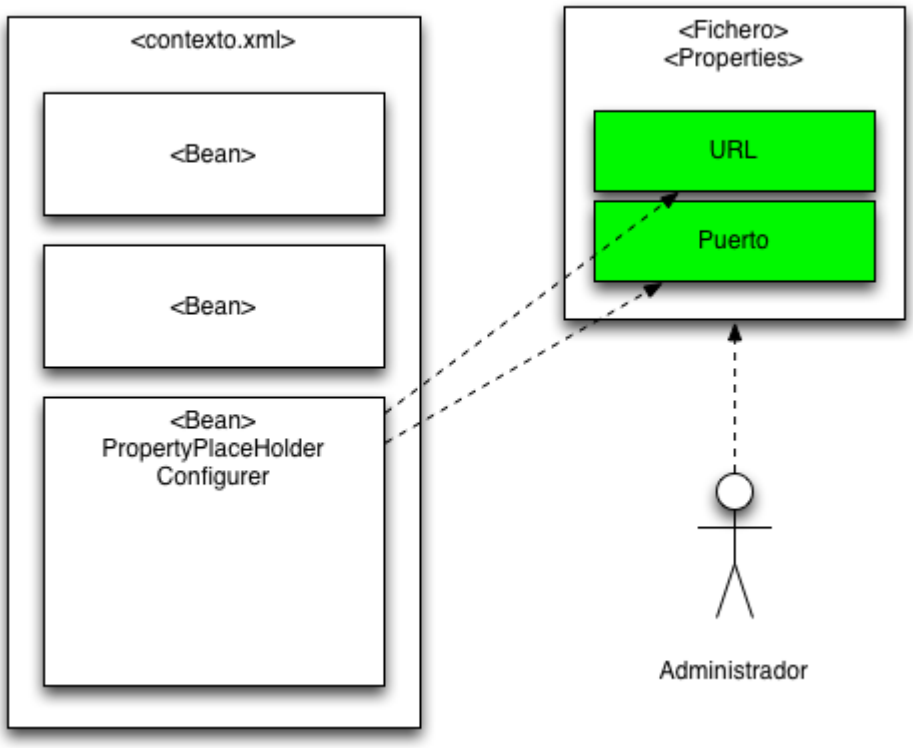
En principio nos puede parecer bastante cómodo configurarlo de esta manera ya que podremos cambiar los parametros a nivel fichero de configuración de Spring cuando lo necesitemos.




Sin embargo la realidad es que nosotros no seremos los que modifiquemos estos parámetros sino que habitualmente lo hará un perfil de administrador cuando la aplicación se despliegue o se mantenga en un entorno de producción . Esto complica un poco el tema ya que los administradores no suelen estar habituados a manejarse con este framework y no les resultara sencillo abrir un fichero XML que puede tener una estructura compleja y realizar modificaciones.




Para solventar este problema y organizar mejor todos los grupos de parámetros que nuestro fichero de configuración pueda tener Spring nos provee de una clase de utilidad denominada PropertyPlaceholderConfigurer . Esta clase nos permite extraer parametrizaciones del fichero de configuración de Spring  y ubicarlas en ficheros de propiedades.De tal forma que tanto para nosotros como para los administradores sea mas sencillo realizar modificaciones.




De tal forma que ahora dispondremos de al menos dos ficheros en la nueva solución .El primero almacena los beans de Spring y el segundo configura los parametros. De esta manera nos será mucho mas sencillo realizar modificaciones. Vamos a ver como quedan configurados estos ficheros.

contexto.xml

```
?xml version="1.0" encoding="UTF-8"?>
 <beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.com/schema/context"
 xsi:schemaLocation="http://www.springframework.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
```

```
 http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd">

<bean
class="org.springframework.beans.factory.config.PropertyPlaceholderCon
figurer">
<property name="location">
 <value>servicio.properties</value>
 </property>
 </bean>

<bean id="servicioPropiedades"
class="com.arquitecturajava.properties.Servicio">
 <property name="url" value="${url}"></property>
 <property name="puerto" value="${puerto}"></property>
 </bean>

</beans>
```

El bean PropertyPlaceholderConfigurer se encarga de extraer los parámetros de configuración a un fichero properties . Una vez realizada esta operación podemos ligar las propiedades a nuestro bien a través de la expresión ${url} o ${puerto} por último mostramos el contenido del fichero servicio.properties

```
url=http://tomcat
puerto=8081
```

Una vez realizada esta operación las responsabilidades de desarrollador y administrador quedarán mucho mas separadas.

Cecilio Álvarez Caules

Cecilio Álvarez Caules Oracle Java Certified Architech