Utilizzare le configurazioni XML e Java insieme

Cosa imparerai

- Come utilizzare le due tipologie di configurazioni in un'unica applicazione



Combinare configurazioni XML e Java

Sebbene le annotation siano molto comode e consentano di snellire un'applicazione, spesso è importante mantenere dei parametri e delle configurazioni esterne al codice sorgente.

Utilizzare dei parametri esterni evita di ricompilare l'applicazione ad ogni modifica di un parametro.

Quando decidiamo di utilizzare la configurazione XML (insieme alle configurazioni tramite annotation) abbiamo due strade da seguire:

1. istanziare il container in modo «*XML-centric*» (ad es. usando la classe *ClassPathXmlApplicationContext*)
2. istanziare il container in modo «*Java-centric*» utilizzando la classe *AnnotationConfigApplicationContext*

Vediamo i due scenari…

Combinare configurazioni XML e Java

Approccio "XML-centric"

In questo scenario avremo:

1. La classe che conterrà le configurazioni (la classe annotata con *@Configuration*)
2. L'XML che conterrà le definizioni contenenti i parametri modificabili senza necessità di ricompilare il codice
3. Il file di properties con le proprietà che utilizzeremo nel bean

1

```java
@Configuration
public class Config {
    @Bean
    public OrdineService getOrdineService() {
        return new OrdineService(getProdottoService());
    }

    @Bean
    public ProdottoService getProdottoService() {
        return new ProdottoService();
    }
}
```

2

```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:p="http://www.springframework.org/schema/p"
xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    https://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    https://www.springframework.org/schema/context/spring-context.xsd">

    <context:annotation-config />
    <context:property-placeholder location="classpath:config.properties" />

    <bean class="it.test.Config" />

    <bean id="clienteService" class="it.test.service.ClienteService">
        <constructor-arg type="String" value="${cliente.tipo-predefinito}" />
        <constructor-arg type="int" value="${cliente.cerca-cancellati}" />
    </bean>
</beans>
```

3

```
1 cliente.tipo-predefinito=PRIVATO
2 cliente.cerca-cancellati=1
3
4
```

Combinare configurazioni XML e Java

Approccio "Java-centric"

In questo scenario avremo:

1. La classe annotate con *@Configuration* che conterrà le configurazioni, il puntamento al file XML definite con l'annotation *@ImportResource* e le property definite nel file di properties. Il valore di una property si legge tramite l'annotation *@Value*
2. L'XML che conterrà il puntamento al file di properties
3. Il file di properties con le proprietà che utilizzeremo nel bean

1

```java
@Configuration
@ImportResource("services.xml")
public class Config {
    @Value("${cliente.tipo-predefinito}")
    private String tipoPredefinito;

    @Value("${cliente.cerca-cancellati}")
    private int cercaCancellati;

    @Bean
    public ClienteService getClienteService() {
        return new ClienteService(tipoPredefinito, cercaCancellati);
    }

    @Bean
    public OrdineService getOrdineService() {
        return new OrdineService(getProdottoService());
    }
```

2

```xml
<beans
xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"
xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    https://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    https://www.springframework.org/schema/context/spring-context.xsd">

    <context:property-placeholder location="config.properties" />

    <bean class="it.test.Config" />
</beans>
```

3

```
cliente.tipo-predefinito=PRIVATO
cliente.cerca-cancellati=1
```

NOTA: il valore dell'annotation @ImportResource può anche essere: classpath:package/file.xml

Es. classpath:it/test/config/serv.xml

Di cosa abbiamo parlato in questa lezione

- Come utilizzare le due tipologie di configurazioni in un’unica applicazione