# Lo *scope* di un bean

## Cosa imparerai

- Quali sono gli ambiti (o *scope*) in cui può essere utilizzato un bean

# Lo *scope* di un bean

Attraverso i metadati di configurazione, andiamo a specificare una serie di proprietà che caratterizzano un bean (proprietà e valori predefiniti, dipendenze da altri bean, ecc…)

Con i metadati di configurazione **è possibile indicare l'ambito (in inglese *scope*) di un'istanza di un bean**. Spring Framework supporta **6 ambiti**, quattro dei quali sono disponibili solo se si utilizza un ApplicationContext per le web application:

- **singleton**
- **prototype**
- **request**
- **session**
- **application**
- **websocket**

Spring consente anche di definire *scope* personalizzati qualora quelli predefiniti non siano sufficienti.

# Lo *scope* di un bean

Spring consente anche di definire *scope* personalizzati qualora quelli predefiniti non siano sufficienti.

Se per i metadati di configurazione si usa la notazione XML, per definire lo *scope* di un bean è necessario utilizzare l'attributo **scope="..."** nell'elemento **<bean />**.

```
<bean id="fatturaService" class="it.test.service.FatturaService" scope="prototype">
    <constructor-arg ref="clienteService" />
    <constructor-arg ref="prodottoService" />
    <constructor-arg type="String" value="ABCD" />
</bean>
```

Se non si specifica lo *scope*, il container considera quello predefinito: **singleton**.

# Lo *scope* di un bean

**Lo *scope* Singleton**

Questo è lo *scope* predefinito.

Con questo *scope*, il container crea solo un'istanza di un bean (singleton appunto…) e viene restitutita sempre questa, ad ogni richiesta di utilizzo.

L' istanza viene archiviata dal container in una cache di bean singleton e tutte le richieste e i riferimenti successivi per quel bean restituiranno l'oggetto memorizzato nella cache.

```
<bean id="clienteService" class="it.test.service.ClienteService">
    <constructor-arg type="String" value="PRIVATO" />
    <constructor-arg type="int" value="1" />
</bean>
```

```
<bean id="clienteService" class="it.test.service.ClienteService" scope="singleton">
    <constructor-arg type="String" value="PRIVATO" />
    <constructor-arg type="int" value="1" />
</bean>
```

# Lo *scope* di un bean

## Lo *scope* Singleton

*Il container, in fase di creazione del bean, crea **una sola istanza** della classe e la iniettata ai bean che la utilizzano.*

*FatturaService e OrdineService utilizzano la stessa istanza del bean ClienteService*

```
<bean id="clienteService" class="it.test.service.ClienteService">
    <constructor-arg type="String" value="PRIVATO" />
    <constructor-arg type="int" value="1" />
</bean>
```

*Istanza 1*

```
<bean id="fatturaService" class="it.test.service.FatturaService">
    <constructor-arg ref="clienteService" />
```

*Istanza 1*

```
<bean id="ordineService" class="it.test.service.OrdineService">
    <property name="cliente">
        <ref bean="clienteService"/>
    </property>
```

```
public class Main {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("services-xml.xml");

        FatturaService fattura = context.getBean("fatturaService", FatturaService.class);
        fattura.getCliente().setTipoPredefinito("PUBBLICA AMMINISTRAZIONE");

        System.out.println(fattura.getInfoFattura(33l));

        OrdineService o = context.getBean("ordineService", OrdineService.class);

        System.out.println(o.sayHello());
    }
}
```

```
Fattura n. 123/21 - Cliente: Mario Rossi - ELIMINATO PUBBLICA AMMINISTRAZIONE - TV 32'' TEI
CIAO sono un ordine del cliente Mario Rossi - ELIMINATO PUBBLICA AMMINISTRAZIONE - STATO OI
```

# Lo *scope* di un bean

**Lo *scope* Prototype**

Con questo scope, il container crea una nuova istanza del bean ogni volta che viene effettuata una richiesta per quel bean.

Il bean con scope «*prototype*» verrà istanziato ogni volta che ne viene richiesto l'utilizzo.

Di norma, è necessario utilizzare lo scope «**prototype**» se è necessario avere un'istanza dedicata di un determinato bean, altrimenti si può usare sempre singleton.

# Lo *scope* di un bean

## Lo *scope* Prototype

*Il container, inject del ClienteService crea una nuova istanza.*

*FatturaService e OrdineService utilizzano due istanze differenti del bean ClienteService.*

```xml
<bean id="clienteService" class="it.test.service.ClienteService" scope="prototype">
    <constructor-arg type="String" value="PRIVATO" />
    <constructor-arg type="int" value="1" />
</bean>
```

Istanza 1

```xml
<bean id="fatturaService" class="it.test.service.FatturaService">
    <constructor-arg ref="clienteService" />
    <constructor-arg ref="prodottoService" />
```

Istanza 2

```xml
<bean id="ordineService" class="it.test.service.OrdineService">
    <property name="cliente">
        <ref bean="clienteService"/>
```

```java
public class Main {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("services-xml.xml");

        FatturaService fattura = context.getBean("fatturaService", FatturaService.class);
        fattura.getCliente().setTipoPredefinito("PUBBLICA AMMINISTRAZIONE");

        System.out.println(fattura.getInfoFattura(33l));

        OrdineService o = context.getBean("ordineService", OrdineService.class);

        System.out.println(o.sayHello());
    }
}
```

```
Fattura n. 123/21 - Cliente: Mario Rossi - ELIMINATO PUBBLICA AMMINISTRAZIONE - TV 32'' TE
CIAO sono un ordine del cliente Mario Rossi - ELIMINATO PRIVATO - STATO ORDINE COMPLETATO
```

# Lo *scope* di un bean

**Gli *scope* Request, Session, Application, and WebSocket Scopes**

Questi *scope* sono utilizzabili sono in container web ad es. *XmlWebApplicationContext*.

**Request scope**

Con questo *scope*, viene create un'istanza di un bean per ogni HTTP request (ad es. la login action…).

**Session Scope**

Con questo *scope*, viene create un'istanza di un bean per ogni HTTP session (ad es. il bean con i dati utente).

**Application Scope**

Con questo scope, viene create un'istanza di un bean e sarà accessibile finché a web application sarà attiva (finché non verrà fatto l'undeploy o verrà spento il server).

NOTA: questi ambiti verranno affrontati meglio nella parte relativa a Spring web.

# Di cosa abbiamo parlato in questa lezione

- Quali sono gli ambiti (o *scope*) in cui può essere utilizzato un bean