Spring InitializingBean and DisposableBean example

By mkyong (http://www.mkyong.com/author/mkyong/) | March 27, 2010 | Updated : June 13, 2011

In Spring, **InitializingBean** and **DisposableBean** are two marker interfaces, a useful way for Spring to perform certain actions upon bean initialization and destruction.

1. For bean implemented InitializingBean, it will run `afterPropertiesSet()` after all bean properties have been set.
2. For bean implemented DisposableBean, it will run `destroy()` after Spring container is released the bean.

Example

Here's an example to show you how to use **InitializingBean**and **DisposableBean**. A CustomerService bean to implement both InitializingBean and DisposableBean interface, and has a message property.

```
package com.mkyong.customer.services;

import org.springframework.beans.factory.DisposableBean;
import org.springframework.beans.factory.InitializingBean;

public class CustomerService implements InitializingBean, DisposableBean
{
    String message;

    public String getMessage() {
      return message;
    }

    public void setMessage(String message) {
      this.message = message;
    }

    public void afterPropertiesSet() throws Exception {
      System.out.println("Init method after properties are set : " + message);
    }

    public void destroy() throws Exception {
      System.out.println("Spring Container is destroy! Customer clean up");
    }

}
```

File : Spring-Customer.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

       <bean id="customerService" class="com.mkyong.customer.services.CustomerService">
        <property name="message" value="i'm property message" />
       </bean>

</beans>
```

Run it

```
package com.mkyong.common;

import org.springframework.context.ConfigurableApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.mkyong.customer.services.CustomerService;

public class App
{
    public static void main( String[] args )
    {
        ConfigurableApplicationContext context =
            new ClassPathXmlApplicationContext(new String[] {"Spring-Customer.xml"});

        CustomerService cust = (CustomerService)context.getBean("customerService");

        System.out.println(cust);

        context.close();
    }
}
```

The **ConfigurableApplicationContext.close()** will close the application context, releasing all resources and destroying all cached singleton beans. It's use for `destroy()` method demo purpose only :)

Output

```
Init method after properties are set : im property message
com.mkyong.customer.services.CustomerService@47393f
...
INFO: Destroying singletons in org.springframework.beans.factory.
support.DefaultListableBeanFactory@77158a:
defining beans [customerService]; root of factory hierarchy
Spring Container is destroy! Customer clean up
```

The afterPropertiesSet() method is called, after the message property is set; while the destroy() method is call after the context.close();

Thoughts…
I would not recommend to use InitializingBean and DisposableBean interface, because it will tight coupled your code to Spring. A better approach should be specifying the init-method and destroy-method (http://www.mkyong.com/spring/spring-init-method-and-destroy-method-example/) attributes in your bean configuration file.

Download Source Code

Download It – Spring-InitializingBean-DisposableBean-Example.zip (http://www.mkyong.com/wp-content/uploads/2010/03/Spring-InitializingBean-DisposableBean-Example.zip)

References

1. InitializingBean Javadoc (http://static.springsource.org/spring/docs/2.5.x/api/org/springframework/beans/factory/InitializingBean.html)
2. DisposableBean Javadoc (http://static.springsource.org/spring/docs/2.5.x/api/org/springframework/beans/factory/DisposableBean.html)

Tags : spring (http://www.mkyong.com/tag/spring/)

Share this article on

Reader also read :

Spring – Mixing XML and JavaConfig (http://www.mkyong.com/spring/spring-mixing-xml-and-javaconfig/)

Spring embedded database examples (http://www.mkyong.com/spring/spring-embedded-database-examples/)

Spring MethodInvokingFactoryBean Example (http://www.mkyong.com/spring/spring-methodinvokingfactorybean-example/)

Spring @Value – Import a list from properties file (http://www.mkyong.com/spring/spring-value-import-a-list-from-properties-file/)