Event Handling

Like any other frameworks, Spring also supports event handling. Sometimes it is necessary to do some extra work based on Application State, Like if you want to load Roles and its permitted operations in a HashMap what will be the ideal way to do it? Obviously, we want to load them when the **ApplicationContext** started.

It is necessary to intercept/hook into **Spring** **ApplicationContext** , so we can aware of the different stages of Object Lifecycle and do some work based on the lifecycle phases.

To let Application developer to know about Object state Spring **ApplicationContext** published different **ApplicationEvent**s , If any Spring bean implements the **ApplicationInterface<T>** interface then this bean will be notified every time when an **ApplicationEvent** published by **ApplicationContext** so this bean work as a Subscriber. As you can guess Spring Event handling based on **Observer design pattern**.

**Spring Application Events**

|  |  |
| --- | --- |
| **Event Name** | **Event Description** |
| **ContextStartedEvent** | This event is published when context is started, to fire this event ApplicationContext has to implement *ConfigurableApplicationContext* interface and call its start() method. |
| **ContextStoppedEvent** | This event is published when context is stopped, to fire this event ApplicationContext has to implement *ConfigurableApplicationContext* interface and call its stop() method. |
| **ContextRefreshedEvent** | This event is published when context is refreshed or initialized, to fire this event ApplicationContext has to implement *ConfigurableApplicationContext* interface and call its refresh() method. |
| **ContextClosedEvent** | This event is published when context is closed to fire this event ApplicationContext has to implement *ConfigurableApplicationContext* interface and call its close() method.Please note that once a ApplicationContext is closed it can’t be started or refreshed. |
| **RequestHandledEvent** | This is a web specific event which is fired when a request is serviced. |

**How to listen an Application Event:**

Step 1. Create a Bean which implements **ApplicationListener<T>** Interface.

Step 2: Override public void **onApplicationEvent(T event)** method and provide the business logic.

Step3. Instantiate Application context ,which implement ***ConfigurableApplicationContext****.*

Step4. call **start(),refresh(),close()** to fires different Events.

**Example**:

**ContextStartEvenListener.java**

package com.example.event;

import org.springframework.context.ApplicationListener;

import org.springframework.context.event.ContextStartedEvent;

public class ContextStartEventListener implements ApplicationListener<ContextStartedEvent>{

@Override

public void onApplicationEvent(ContextStartedEvent event) {

System.out.println(event.getSource());

System.out.println("ContextStarted Event Captured");

}

}

**ContextStoppedEventListner.java**

package com.example.event;

import org.springframework.context.ApplicationListener;

import org.springframework.context.event.ContextStoppedEvent;

public class ContextStoppedEventListener implements ApplicationListener<ContextStoppedEvent>{

@Override

public void onApplicationEvent(ContextStoppedEvent event) {

System.out.println(event.getSource());

System.out.println("ContextStopped Event Captured");

}

}

**Main.java**

package com.example.event;

import org.springframework.context.ConfigurableApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Main {

public static void main(String[] args) {

ConfigurableApplicationContext context = new ClassPathXmlApplicationContext("configFiles/event.xml"); // Let us raise a start event.

context.start();//fire start event

context.stop();//fire stop event

}

}

**Event.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-3.0.xsd">

<bean id="contextStartEvent" class="com.example.event.ContextStartEventListener"/>

<bean id="contextStoppedEvent" class="com.example.event.ContextStoppedEventListener"/>

</beans>

**Output:**

org.springframework.context.support.ClassPathXmlApplicationContext@c596a7a: startup date [Mon Oct 31 18:03:57 IST 2016]; root of context hierarchy

ContextStarted Event Captured

org.springframework.context.support.ClassPathXmlApplicationContext@c596a7a: startup date [Mon Oct 31 18:03:57 IST 2016]; root of context hierarchy

ContextStopped Event Captured