**Assisted Practice: 1.2 Default and Custom Event Handling**

This section will guide you to:

* Set up Eclipse to work with Spring
* Create a pom.xml configuration file to add all the required components
* Create a bean class to listen to Application events
* Create a bean class to stop listening to Application events
* Create a bean class to generate a Custom Event
* Create a bean class to publish the Custom Event
* Create a bean class to listen to the Custom Event
* Create a JSP file to call a Controller
* Create a Controller to test events
* Create a JSP view which the Controller calls after it finishes processing

**Development Environment**

* Eclipse IDE for Enterprise Java Developers v2019-03 (4.11.0)
* Apache Tomcat Server v9.0
* JRE: OpenJDK Runtime Environment 11.0.2
* All other software components are defined in the Project Object Model pom.xml

This lab has sixteen subsections, namely:

* + 1. Creating a Maven project which is web enabled
    2. Creating pom.xml for including the required components
    3. Creating a bean class StartEventHandler
    4. Creating a bean class StopEventHandler
    5. Creating a bean class CustomEvent
    6. Creating a bean class CustomEventPublisher
    7. Creating a bean class CustomEventListener
    8. Creating a Controller class MainController
    9. Creating the dispatcher servlet main-servlet.xml
    10. Creating a view index.jsp
    11. Creating a view customEvent.jsp
    12. Configuring web.xml
    13. Building the project
    14. Publishing and starting the project
    15. Running the project
    16. Pushing the code to your GitHub repositories

**Step 1.2.1:** Creating a Maven project which is web enabled

* Open Eclipse
* Go to the **File** menu. Choose **New->Maven Project**
* Uncheck **Create a Simple Project** and check **Use Default Workspace Location** and click on **Next**
* From the **archetype** list, choose the row that has **Artifact Id** as **maven-archetype-webapp** and click on **Next**
* Enter **Group Id** as **com** and **Artifact Id** as **SpringSetup** and click on **Finish**
* This will create the project files in the Project Explorer
* Before building the project, we need to confirm that the **servlet.jar** has been added to the project
* Servlet is already installed in your practice labs (Refer FSD: Lab Guide - Phase 2)
* To add it to the project, follow the below mentioned steps:
  + In the Project Explorer, right click and choose **Properties**
  + Select **Java Build Path** from the options on the left
  + Click on **Libraries** tab on the right
  + If there is an existing entry for the **servlet.jar,** then click on **Cancel** and exit the window
  + If it is not there, then click on **Classpath** entry and click on **Add External JARs** button on the right
  + From the **file** list, select **servlet.jar** file and click on **Ok**
  + Click on **Apply and Close**

**Step 1.2.2:** Creating pom.xml for including the required components

* In the Project Explorer, expand **SpringSetup** and double click on **pom.xml**
* Add the following entries:

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>SpringSetup</groupId>

<artifactId>SpringSetup</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>SpringSetup Maven Webapp</name>

<url>http://maven.apache.org</url>

<properties>

<org.springframework.version>5.1.6.RELEASE</org.springframework.version>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-expression</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>javassist</groupId>

<artifactId>javassist</artifactId>

<version>3.12.1.GA</version>

</dependency>

<dependency>

<groupId>taglibs</groupId>

<artifactId>standard</artifactId>

<version>1.1.2</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>commons-dbcp</groupId>

<artifactId>commons-dbcp</artifactId>

<version>1.4</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<dependency>

<groupId>javax.persistence</groupId>

<artifactId>persistence-api</artifactId>

<version>1.0.2</version>

<scope>provided</scope>

</dependency>

</dependencies>

<build>

<finalName>SpringSetup</finalName>

</build>

</project>

**Step 1.2.3:** Creating a bean class StartEventHandler

* In the Project Explorer, expand **SpringSetup->src->main**
* Right click **main** and choose **New->Other**
* In the **Wizard** list, choose **Java->Class** and click on **Next**
* In **Package,** enter com.ecommerce. and in **Name** enter StartEventHandlerand click on **Finish**
* Enter the following code:

**package** com.ecommerce.beans;

**import** org.springframework.context.ApplicationListener;

**import** org.springframework.context.event.ContextStartedEvent;

**public** **class** StartEventHandler **implements** ApplicationListener<ContextStartedEvent>{

**public** void onApplicationEvent(ContextStartedEvent event) {

**System**.out.println("ContextStartedEvent Received");

}

}

**Step 1.2.4:** Creating a bean class StopEventHandler

* In the Project Explorer, expand **SpringSetup->src->main**
* Right click on **main** and choose **New->Other**
* In the **Wizard** list, choose **Java->Class** and click on **Next**
* In **Package,** enter com.ecommerce.beans and in **Name** enter StopEventHandler and click on **Finish**
* Enter the following code:

**package** com.ecommerce.beans;

**import** org.springframework.context.ApplicationListener;

**import** org.springframework.context.event.ContextStoppedEvent;

**public** **class** StopEventHandler **implements** ApplicationListener<ContextStoppedEvent>{

**public** void onApplicationEvent(ContextStoppedEvent event) {

**System**.out.println("ContextStoppedEvent Received");

}

}

**Step 1.2.5:** Creating a bean class CustomEvent

* In the Project Explorer, expand **SpringSetup->src->main**
* Right click on **main** and choose **New->Other**
* In the **Wizard** list, choose **Java->Class** and click on **Next**
* In **Package,** enter com.ecommerce.beans and in **Name** enter CustomEvent and click on **Finish**
* Enter the following code:

**package** com.ecommerce.beans;

**import** org.springframework.context.ApplicationEvent;

**public** **class** CustomEvent **extends** ApplicationEvent{

**public** CustomEvent(**Object** source) {

**super**(source);

}

**public** **String** toString(){

**return** "This is a custom event";

}

}

**Step 1.2.6:** Creating a bean class CustomEventPublisher

* In the Project Explorer, expand **SpringSetup->src->main**
* Right click on **main** and choose **New->Other**
* In the **Wizard** list, choose **Java->Class** and click **Next**
* In **Package,** enter com.ecommerce.beans and in **Name** enter CustomEventPublisher and click on **Finish**
* Enter the following code:

**package** com.ecommerce.beans;

**import** org.springframework.context.ApplicationEventPublisher;

**import** org.springframework.context.ApplicationEventPublisherAware;

**public** **class** CustomEventPublisher **implements** ApplicationEventPublisherAware {

**private** ApplicationEventPublisher publisher;

**public** void setApplicationEventPublisher (ApplicationEventPublisher publisher) {

**this**.publisher = publisher;

}

**public** void publish() {

CustomEvent ce = **new** CustomEvent(**this**);

publisher.publishEvent(ce);

}

}

**Step 1.2.7:** Creating a bean class CustomEventListener

* In the Project Explorer, expand **SpringSetup->src->main**
* Right click on **main** and choose **New->Other**
* In the **Wizard** list, choose **Java->Class** and click on **Next**
* In **Package,** enter com.ecommerce.beans and in **Name** enter CustomEventListener and click on **Finish**
* Enter the following code:

**package** com.ecommerce;

**import** org.springframework.context.ApplicationListener;

**public** **class** CustomEventListener **implements** ApplicationListener<CustomEvent> {

**public** void onApplicationEvent(CustomEvent event) {

**System**.out.println(event.toString());

}

}

**Step 1.2.8:** Creating a Controller class MainController

* In the Project Explorer, expand **SpringSetup->src->main**
* Right click on **main** and choose **New->Other**
* In the **Wizard** list, choose **Java->Class** and click on **Next**
* In **Package,** enter com.ecommerce.controller and in **Name** enter MainControllerand click on **Finish**
* Enter the following code:

**package** com.ecommerce.controller;

**import** org.apache.log4j.Logger;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.ModelMap;

**import** org.springframework.validation.BindingResult;

**import** org.springframework.web.bind.annotation.ModelAttribute;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** org.springframework.context.ApplicationContext;

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

**@Controller**

**public** **class** MainController {

**@RequestMapping(value = "/customevent", method = RequestMethod.GET)**

**public** **String** customEvent(ModelMap map)

{

**String** confFile = "main-servlet.xml";

ApplicationContext context = **new** ClassPathXmlApplicationContext(confFile);

CustomEventPublisher cvp =

(CustomEventPublisher) context.getBean("customEventPublisher");

cvp.publish();

cvp.publish();

**return** "customEvent";

}

}

**Step 1.2.9:** Creating the dispatcher servlet main-servlet.xml

* In the Project Explorer, expand **SpringSetup->src->main->webapp**
* Right click on **WEB-INF** and choose **New->File**
* In **filename,** enter **main-servlet.xml** and click on **Finish**
* Enter the following script:

<?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:aop="http://www.springframework.org/schema/aop"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:jee="http://www.springframework.org/schema/jee"

xmlns:lang="http://www.springframework.org/schema/lang"

xmlns:p="http://www.springframework.org/schema/p"

xmlns:tx="http://www.springframework.org/schema/tx"

xmlns:util="http://www.springframework.org/schema/util"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-2.5.xsd

http://www.springframework.org/schema/aop

http://www.springframework.org/schema/aop/spring-aop-2.5.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-2.5.xsd

http://www.springframework.org/schema/tx

http://www.springframework.org/schema/tx/spring-tx-2.5.xsd">

<context:annotation-config />

<context:component-scan base-package="com.ecommerce.controller" />

<bean id="jspViewResolver"

class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="viewClass"

value="org.springframework.web.servlet.view.JstlView"></property>

<property name="prefix" value="/WEB-INF/view/"></property>

<property name="suffix" value=".jsp"></property>

</bean>

<bean id="messageSource"

class="org.springframework.context.support.ReloadableResourceBundleMessageSource">

<property name="basename" value="classpath:messages"></property>

<property name="defaultEncoding" value="UTF-8"></property>

</bean>

<bean id = "startEventHandler" class = "com.ecommerce.beans.StartEventHandler"/>

<bean id = "stopEventHandler" class = "com.ecommerce.beans.StopEventHandler"/>

<bean id = "customEventListener" class = "com.ecommerce.beans.CustomEventListener"/>

<bean id = "customEventPublisher" class = "com.ecommerce.beans.CustomEventPublisher"/>

<tx:annotation-driven />

</beans>

**Step 1.2.10:** Creating a view index.jsp

* In the Project Explorer, expand **SpringSetup->src->main->webapp->WEB-INF**
* Right click on **view** and choose **New->File**
* In **filename,** enter **index.jsp** and click on **Finish**
* Enter the following code:

<**html**>

<**body**>

<**h2**>Spring Application</**h2**>

<**a** href="customevent">Custom Event Handling</**a**>

</**body**>

</**html**>

**Step 1.2.11:** Creating a view customEvent.jsp

* In the Project Explorer, expand **SpringSetup->src->main->webapp->WEB-INF->view**
* Right click on **view** and choose **New->File**
* In **filename,** enter **customEvent.jsp and** click on **Finish**
* Enter the following code:

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Custom Event</**title**>

</**head**>

<**body**>

Custom Event has been generated in the Java Console.

</**body**>

</**html**>

**Step 1.2.12:** Configuring web.xml

* In the Project Explorer expand **SpringSetup->src->main->webapp->WEB-INF**
* Double click on **web.xml** to open it in the editor
* Enter the following script:

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://java.sun.com/xml/ns/javaee"

version="2.5"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee

http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd">

<display-name>Archetype Created Web Application</display-name>

<welcome-file-list>

<welcome-file>/WEB-INF/view/index.jsp</welcome-file>

</welcome-file-list>

<servlet>

<servlet-name>main</servlet-name>

<servlet-class>

org.springframework.web.servlet.DispatcherServlet

</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>main</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/main-servlet.xml</param-value>

</context-param>

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

</web-app>

**Step 1.2.13:** Building the project

* From the **Project** menu at the top, click on **Build**
* If any compile errors are shown, fix them as required

**Step 1.2.14:** Publishing and starting the project

* If you do not see the **Servers** tab near the bottom of the IDE, go to **Window** menu and click on **Show View->Servers**
* Right click the **Server** entry and choose **Add and Remove**
* Click the **Add** button to move **SpringSetup** from the **Available** list to the **Configured** list
* Click on **Finish**
* Right click the **Server** entry and click on **Publish**
* Right click the **Server** entry and click on **Start**
* This will start the server

**Step 1.2.15:** Running the project

* To run the project, open a web browser and type: [**http://localhost:8080/**](http://localhost:8080/ServletConcept)**SpringSetup**

**Step 1.2.16:** Pushing the code to your GitHub repositories

* Open your command prompt and navigate to the folder where you have created your files.

**cd <folder path>**

* Initialize your repository using the following command:

**git init**

* Add all the files to your git repository using the following command:

**git add .**

* Commit the changes using the following command:

**git commit . -m “Changes have been committed.”**

* Push the files to the folder you initially created using the following command:

**git push -u origin master**