1.Environmental Setup

Source code:

Pom.xml

<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>

2.Default & Custom Event Handling

Pom.xml

<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>

Package:com.ecommerce

Name :StartEventHandler

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");

}

}

Package:com.ecommmerce.beans

Name :StopEventHandler

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");

}

}

Package:com.ecommerce.beans

Name :CustomEvent

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";

}

}

Package:com.ecommerce.beans

Name :CustomEventPublisher

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);

}

}

Package:com.ecommerce.beans

Name :CustomEventListener

package com.ecommerce;

import org.springframework.context.ApplicationListener;

public class CustomEventListener implements ApplicationListener<CustomEvent> {

public void onApplicationEvent(CustomEvent event) {

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

}

}

Package:com.ecommerce.controller

Name :MainController

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";

}

}

main-servlet.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: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>

index.jsp

<html>

<body>

<h2>Spring Application</h2>

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

</body>

</html>

customEvent.jsp

<%@ 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>

Web.xml

<?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>

3.JDBC FrameWork

Pom.xml

<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>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.16</version>

<scope>runtime</scope>

</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>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.9</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>

CREATE DATABASE ecommerce

ecommerce

--

-- Table structure for table `eproduct`

--

DROP TABLE IF EXISTS `eproduct`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `eproduct` (

`ID` bigint(20) NOT NULL AUTO\_INCREMENT,

`name` varchar(100) DEFAULT NULL,

`price` decimal(10,2) DEFAULT NULL,

`date\_added` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (`ID`)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `eproduct`

--

LOCK TABLES `eproduct` WRITE;

/\*!40000 ALTER TABLE `eproduct` DISABLE KEYS \*/;

INSERT INTO `eproduct` VALUES (1,'HP Laptop ABC',21900.00,'2019-06-04 07:18:57'),(2,'Acer Laptop ABC',23300.00,'2019-06-04 07:19:07'),(3,'Lenovo Laptop ABC',33322.00,'2019-06-04 07:19:19');

/\*!40000 ALTER TABLE `eproduct` ENABLE KEYS \*/;

UNLOCK TABLES;

Package : com.ecommeerce.entity

Name : Eproduct

package com.ecommerce.entity;

import java.math.BigDecimal;

import java.util.Date;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.Table;

public class EProductEntity {

private long ID;

private String name;

private BigDecimal price;

private Date dateAdded;

public long getID() {return this.ID; }

public String getName() { return this.name;}

public BigDecimal getPrice() { return this.price;}

public Date getDateAdded() { return this.dateAdded;}

public void setID(long id) { this.ID = id;}

public void setName(String name) { this.name = name;}

public void setPrice(BigDecimal price) { this.price = price;}

public void setDateAdded(Date date) { this.dateAdded = date;}

}

package com.ecommerce.entity;

import java.math.BigDecimal;

import java.util.Date;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.Table;

public class EProductEntity {

private long ID;

private String name;

private BigDecimal price;

private Date dateAdded;

public long getID() {return this.ID; }

public String getName() { return this.name;}

public BigDecimal getPrice() { return this.price;}

public Date getDateAdded() { return this.dateAdded;}

public void setID(long id) { this.ID = id;}

public void setName(String name) { this.name = name;}

public void setPrice(BigDecimal price) { this.price = price;}

public void setDateAdded(Date date) { this.dateAdded = date;}

}

Package : com.ecommerce.dao

Name : EProductDAO

package com.ecommerce.dao;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.List;

import org.springframework.jdbc.core.BeanPropertyRowMapper;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.jdbc.core.RowMapper;

import com.ecommerce.entity.EProduct;

public class EProductDAO {

JdbcTemplate template;

public void setTemplate(JdbcTemplate template) {

this.template = template;

}

public List<EProduct> getProducts(){

return template.query("select \* from eproduct",new RowMapper<EProduct>(){

public EProduct mapRow(ResultSet rs, int row) throws SQLException {

EProduct e=new EProduct();

e.setId(rs.getInt(1));

e.setName(rs.getString(2));

e.setPrice(rs.getBigDecimal(3));

e.setDateAdded(rs.getDate(4));

return e;

}

});

}

}

Package : com.ecommerce.controller

Name : MainController

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;

import com.ecommerce.entity.EProduct;

import com.ecommerce.dao.EProductDAO;

@Controller

public class MainController {

@Autowired

EProductDAO eproductDAO;

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

public String listProducts(ModelMap map)

{

List<EProduct> list= eproductDAO.getProducts();

model.addAttribute("list",list);

return "listProducts";

}

}

main-servlet.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: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="dataSource"

class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close"

p:driverClassName="${jdbc.driverClassName}"

p:url="${jdbc.databaseurl}" p:username="${jdbc.username}"

p:password="${jdbc.password}"></bean>

<bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">

<property name="driverClassName" value="com.mysql.jdbc.Driver"></property>

<property name="url" value="jdbc:mysql://localhost:3306/ecommerce"></property>

<property name="username" value="root"></property>

<property name="password" value="master"></property>

</bean>

<bean id="jt" class="org.springframework.jdbc.core.JdbcTemplate">

<property name="dataSource" ref="ds"></property>

</bean>

<bean id="dao" class="com.ecommerce.dao.EProductDAO">

<property name="template" ref="jt"></property>

</bean>

<tx:annotation-driven />

<bean id="transactionManager"

class="org.springframework.orm.hibernate3.HibernateTransactionManager">

<property name="sessionFactory" ref="sessionFactory"></property>

</bean>

</beans>

Index.jsp

<html>

<body>

<h2>Spring Application</h2>

<a href="listProducts">List Products</a>

</body>

</html>

listProducts.jsp

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

pageEncoding="UTF-8"%>

<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>List Products</title>

</head>

<body>

<c:forEach var="product" items="${list}">

${product.name}, ${product.price}, ${product.dateAdded}<br>

</c:forEach>

</body>

</html>

web.xml

<?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>

4.Spring Boot Exception Handling

Package :com.ecommerce.entity

Name : Eproduct

package com.ecommerce.entity;

import java.math.BigDecimal;

import java.util.Collection;

import java.util.Date;

import java.util.List;

import java.util.Set;

import java.util.Map;

public class EProduct {

private long ID;

private String name;

private BigDecimal price;

private Date dateAdded;

public EProduct() {

}

public long getID() {return this.ID; }

public String getName() { return this.name;}

public BigDecimal getPrice() { return this.price;}

public Date getDateAdded() { return this.dateAdded;}

public void setID(long id) { this.ID = id;}

public void setName(String name) { this.name = name;}

public void setPrice(BigDecimal price) { this.price = price;}

public void setDateAdded(Date date) { this.dateAdded = date;}

}

Package : com.ecommerce.exceptions

Name : ProductNotFoundException

package com.ecommerce.exceptions;

public class ProductNotFoundException extends RuntimeException {

private static final long serialVersionUID = 1L;

}

Package : com.ecommerce.controller

Name : EproductExceptionController

package com.ecommerce.controllers;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.ControllerAdvice;

import org.springframework.web.bind.annotation.ExceptionHandler;

import com.ecommerce.exceptions.ProductNotFoundException;

@ControllerAdvice

public class EProductExceptionController {

@ExceptionHandler(value = ProductNotFoundException.class)

public ResponseEntity<Object> exception(ProductNotFoundException exception) {

return new ResponseEntity<>("Product not found", HttpStatus.NOT\_FOUND);

}

}

Name : MainController

package com.ecommerce.controllers;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.stereotype.Controller;

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

import org.springframework.web.bind.annotation.RequestBody;

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

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

import org.springframework.web.bind.annotation.ResponseBody;

import com.ecommerce.entity.EProduct;

import com.ecommerce.exceptions.ProductNotFoundException;

@Controller

public class MainController {

@RequestMapping(value = "/product/{id}", method = RequestMethod.GET)

@ResponseBody

public String getProduct(@PathVariable("id") String id) {

if (id.contentEquals("0"))

throw new ProductNotFoundException();

return "Product was found";

}

}

5.Consuming RESTful Web Services

Package :com.ecommerce.beans

Name :Quote

package com.ecommerce.beans;

import com.fasterxml.jackson.annotation.\*;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@JsonIgnoreProperties(ignoreUnknown = true)

public class Quote {

private String type;

private Value value;

public Quote() {

}

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

public Value getValue() {

return value;

}

public void setValue(Value value) {

this.value = value;

}

@Override

public String toString() {

return "Quote{" +

"type='" + type + '\'' +

", value=" + value +

'}';

}

}

Package :com.ecommerce.beans

Class :Quote

package com.ecommerce.beans;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@JsonIgnoreProperties(ignoreUnknown = true)

public class Value {

private Long id;

private String quote;

public Value() {

}

public Long getId() {

return this.id;

}

public String getQuote() {

return this.quote;

}

public void setId(Long id) {

this.id = id;

}

public void setQuote(String quote) {

this.quote = quote;

}

@Override

public String toString() {

return "Value{" +

"id=" + id +

", quote='" + quote + '\'' +

'}';

}

}

Package : com.ecommerce.controllers

Class :MainController

package com.ecommerce.controllers;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.stereotype.Controller;

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

import org.springframework.web.bind.annotation.RequestBody;

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

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

import org.springframework.web.bind.annotation.ResponseBody;

import org.springframework.web.client.RestTemplate;

import com.ecommerce.beans.Quote;

@Controller

public class MainController {

@RequestMapping("/")

@ResponseBody

public String index() {

RestTemplate restTemplate = new RestTemplate();

Quote quote = restTemplate.getForObject("https://gturnquist-quoters.cfapps.io/api/random", Quote.class);

return quote.toString();

}

}

6.Edge server and Routing

SpringEdgeApplication

Package :com.ecommerce

package com.ecommerce;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.netflix.zuul.EnableZuulProxy;

@SpringBootApplication

@EnableZuulProxy

public class SpringEdgeApplication {

public static void main(String[] args) {

SpringApplication.run(SpringEdgeApplication.class, args);

}

}

application.properties

spring.application.name = zuulserver

zuul.routes.products.path = /api/demo/\*\*

zuul.routes.products.url = http://localhost:8080/

server.port = 8111

7.File Handling

Index.html

<html>

<head><title>File Upload</title></head>

<body>

<form method="post" enctype="multipart/form-data" action="/upload">

Upload file&nbsp;

<input type="file" name="fileToUpload" id="fileToUpload"><br><br>

<input type="submit" value="Upload " name="submit">

</form>

</body>

</html>

Package :com.ecommerce.controllers

Class :MainController

package com.ecommerce.controllers;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import org.springframework.core.io.ClassPathResource;

import org.springframework.core.io.InputStreamResource;

import org.springframework.core.io.Resource;

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpStatus;

import org.springframework.http.MediaType;

import org.springframework.http.ResponseEntity;

import org.springframework.stereotype.Controller;

import org.springframework.util.ResourceUtils;

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

import org.springframework.web.bind.annotation.RequestBody;

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

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

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

import org.springframework.web.bind.annotation.ResponseBody;

import org.springframework.web.multipart.MultipartFile;

@Controller

public class MainController {

@RequestMapping(value = "/")

public String index() {

return "index.html";

}

@RequestMapping(value = "/upload", method = RequestMethod.POST, consumes = MediaType.MULTIPART\_FORM\_DATA\_VALUE)

public String fileUpload(@RequestParam("file") MultipartFile file) {

String result = "File was uploaded successfully";

try {

File convertFile = new File("/var/tmp/"+file.getOriginalFilename());

convertFile.createNewFile();

FileOutputStream fout = new FileOutputStream(convertFile);

fout.write(file.getBytes());

fout.close();

} catch (IOException iex) {

result = "Error " + iex.getMessage();

} finally {

return result;

}

}

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

public ResponseEntity<Object> downloadFile() throws IOException {

String fileName = "static/dump.txt";

ClassLoader classLoader = new MainController().getClass().getClassLoader();

File file = new File(classLoader.getResource(fileName).getFile());

InputStreamResource resource = new InputStreamResource(new FileInputStream(file));

HttpHeaders headers = new HttpHeaders();

headers.add("Content-Disposition", String.format("attachment; filename=\"%s\"", file.getName()));

headers.add("Cache-Control", "no-cache, no-store, must-revalidate");

headers.add("Pragma", "no-cache");

headers.add("Expires", "0");

ResponseEntity<Object>

responseEntity = ResponseEntity.ok().headers(headers).contentLength(file.length()).contentType(

MediaType.parseMediaType("application/txt")).body(resource);

return responseEntity;

}

}

8.Sending & Receiving Messages With Apache Kafka

Apache Kafka dependencies

Name :SpringKafka

Group :com.ecommerce

Package :com.ecommerce

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

<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/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.1.6.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<groupId>com.ecommerce</groupId>

<artifactId>SpringKafka</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>SpringKafka</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

</dependency>

<dependency>

<groupId>org.apache.kafka</groupId>

<artifactId>kafka-streams</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.kafka</groupId>

<artifactId>spring-kafka</artifactId>

<version>2.1.6.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>5.1.5.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework.kafka</groupId>

<artifactId>spring-kafka</artifactId>

<version>2.2.7.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

Package :com.ecommerce

Class :KafkaProducerConfig

package com.ecommerce;

import java.util.HashMap;

import java.util.Map;

import org.apache.kafka.clients.producer.ProducerConfig;

import org.apache.kafka.common.serialization.StringSerializer;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.kafka.core.DefaultKafkaProducerFactory;

import org.springframework.kafka.core.KafkaTemplate;

import org.springframework.kafka.core.ProducerFactory;

@Configuration

public class KafkaProducerConfig {

@Bean

public ProducerFactory<String, String> producerFactory() {

Map<String, Object> configProps = new HashMap<>();

configProps.put(ProducerConfig.BOOTSTRAP\_SERVERS\_CONFIG, "localhost:9092");

configProps.put(ProducerConfig.KEY\_SERIALIZER\_CLASS\_CONFIG, StringSerializer.class);

configProps.put(ProducerConfig.VALUE\_SERIALIZER\_CLASS\_CONFIG, StringSerializer.class);

return new DefaultKafkaProducerFactory<>(configProps);

}

@Bean

public KafkaTemplate<String, String> kafkaTemplate() {

return new KafkaTemplate<>(producerFactory());

}

}

Package :com.ecommerce

Name :KafkaConsumerConfig

package com.ecommerce;

import java.util.HashMap;

import java.util.Map;

import org.apache.kafka.clients.consumer.ConsumerConfig;

import org.apache.kafka.common.serialization.StringDeserializer;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.kafka.annotation.EnableKafka;

import org.springframework.kafka.config.ConcurrentKafkaListenerContainerFactory;

import org.springframework.kafka.core.ConsumerFactory;

import org.springframework.kafka.core.DefaultKafkaConsumerFactory;

@EnableKafka

@Configuration

public class KafkaConsumerConfig {

@Bean

public ConsumerFactory<String, String> consumerFactory() {

Map<String, Object> props = new HashMap<>();

props.put(ConsumerConfig.BOOTSTRAP\_SERVERS\_CONFIG, "localhost:2181");

props.put(ConsumerConfig.GROUP\_ID\_CONFIG, "group-id");

props.put(ConsumerConfig.KEY\_DESERIALIZER\_CLASS\_CONFIG, StringDeserializer.class);

props.put(ConsumerConfig.VALUE\_DESERIALIZER\_CLASS\_CONFIG, StringDeserializer.class);

return new DefaultKafkaConsumerFactory<>(props);

}

@Bean

public ConcurrentKafkaListenerContainerFactory<String, String> kafkaListenerContainerFactory() {

ConcurrentKafkaListenerContainerFactory<String, String>

factory = new ConcurrentKafkaListenerContainerFactory<>();

factory.setConsumerFactory(consumerFactory());

return factory;

}

}

Package : com.ecommerce.controllers

Name :MainControllers

package com.commerce.controllers;

import java.util.Calendar;

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

import org.springframework.kafka.core.DefaultKafkaProducerFactory;

import org.springframework.kafka.core.KafkaTemplate;

import org.springframework.kafka.core.ProducerFactory;

import org.springframework.stereotype.Controller;

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

@Controller

public class MainController {

@Autowired

private KafkaTemplate<String, String> kafkaTemplate;

@RequestMapping(value = "/")

public String index() {

this.sendMessage("This is a message sent at " + Calendar.getInstance().getTime());

return "Check Eclipse console for kafka output";

}

private void sendMessage(String msg) {

kafkaTemplate.send("ecommerce", msg);

}

}

SpringBootKafkaApplication

package com.ecommerce;

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

import org.springframework.boot.ApplicationArguments;

import org.springframework.boot.ApplicationRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.kafka.annotation.KafkaListener;

import org.springframework.kafka.core.KafkaTemplate;

@SpringBootApplication

public class SpringBootKafkaApplication {

@Autowired

private KafkaTemplate<String, String> kafkaTemplate;

public static void main(String[] args) {

SpringApplication.run(SpringBootKafkaApplication.class, args);

}

@KafkaListener(topics = "ecommerce", groupId = "group-id")

public void listen(String message) {

System.out.println("Received Message in group - group-id: " + message);

}

}

9.HTTPS for SpringBooot

Package :com.ecommerce.controllers

Name :MainController

package com.ecommerce.controllers;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.stereotype.Controller;

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

import org.springframework.web.bind.annotation.RequestBody;

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

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

import org.springframework.web.bind.annotation.ResponseBody;

@Controller

public class MainController {

@Autowired

private ProductRepository repository;

@RequestMapping("/")

@ResponseBody

public String index() {

return “This is running under SSL”;

}

}

application.properties

server.port=8443

server.ssl.key-alias=selfsigned\_localhost\_sslserver

server.ssl.key-password=changeit

server.ssl.key-store=classpath:ssl-server.jks

server.ssl.key-store-provider=SUN

server.ssl.key-store-type=JKS

10.Junit5:Standard Tests

Pom.xml

<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/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>UsingJUnit</groupId>

<artifactId>UsingJUnit</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<version>5.4.2</version>

</dependency>

<dependency>

<groupId>org.junit.platform</groupId>

<artifactId>junit-platform-launcher</artifactId>

<version>1.2.0</version>

</dependency>

</dependencies>

</project>

Package:com.ecommerce.tests

Name :StandardTestClass

package com.ecommerce.tests;

import static org.junit.jupiter.api.Assertions.fail;

import static org.junit.jupiter.api.Assumptions.assumeTrue;

import org.junit.jupiter.api.AfterAll;

import org.junit.jupiter.api.AfterEach;

import org.junit.jupiter.api.BeforeAll;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Disabled;

import org.junit.jupiter.api.DisplayName;

import org.junit.jupiter.api.Test;

import org.junit.platform.runner.JUnitPlatform;

import org.junit.runner.RunWith;

@DisplayName("JUnit 5 Standard Test Class Example")

@RunWith(JUnitPlatform.class)

public class StandardTestClass {

@BeforeAll

static void initAll() {

System.out.println("initAll()");

}

@BeforeEach

void init() {

System.out.println("init()");

}

@Test

void succeedingTest() {

System.out.println("succeedingTest()");

}

@Test

void failingTest() {

System.out.println("failingTest()");

fail("a failing test");

}

@Test

@Disabled("for demonstration purposes")

void skippedTest() {

// not executed

}

@Test

void abortedTest() {

System.out.println("abortedTest()");

assumeTrue("abc".contains("Z"));

fail("test should have been aborted");

}

@AfterEach

void tearDown() {

System.out.println("tearDown()");

}

@AfterAll

static void tearDownAll() {

System.out.println("tearDownAll()");

}

}

11.Assertions

Pom.xml

<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/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>UsingJUnit</groupId>

<artifactId>UsingJUnit</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<version>5.4.2</version>

</dependency>

<dependency>

<groupId>org.junit.platform</groupId>

<artifactId>junit-platform-launcher</artifactId>

<version>1.2.0</version>

</dependency>

</dependencies>

</project>

Package :com.ecommerce.tests

Name :Assertions

package com.ecommerce.tests;

import static org.junit.jupiter.api.Assertions.assertArrayEquals;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.junit.jupiter.api.Assertions.assertFalse;

import static org.junit.jupiter.api.Assertions.assertNotEquals;

import static org.junit.jupiter.api.Assertions.assertNotNull;

import static org.junit.jupiter.api.Assertions.assertNotSame;

import static org.junit.jupiter.api.Assertions.assertNull;

import static org.junit.jupiter.api.Assertions.assertSame;

import static org.junit.jupiter.api.Assertions.assertThrows;

import static org.junit.jupiter.api.Assertions.assertTrue;

import org.junit.jupiter.api.\*;

import org.junit.jupiter.api.condition.DisabledIf;

import org.junit.jupiter.api.condition.EnabledOnOs;

import org.junit.jupiter.api.condition.OS;

import org.junit.platform.runner.JUnitPlatform;

import org.junit.runner.RunWith;

@DisplayName("JUnit 5 Assertions Example")

@RunWith(JUnitPlatform.class)

public class Assertions {

@Test

@DisplayName("assert Examples")

public void assertTests() {

String str = null;

String str2 = "some value";

String[] a1 = { "A", "B" };

String[] a2 = { "A", "B" };

assertTrue(4 > 0);

assertFalse(5 < 1);

assertNull(str);

assertNotNull(str2);

assertSame(str, str);

assertNotSame(str, str2);

assertEquals(5, 5);

assertNotEquals(5, 6);

assertArrayEquals(a1, a2);

assertThrows(RuntimeException.class, () -> {

throw new RuntimeException();

});

}

}

12.Conditional test Execution

Pom.xml

<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/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>UsingJUnit</groupId>

<artifactId>UsingJUnit</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<version>5.4.2</version>

</dependency>

<dependency>

<groupId>org.junit.platform</groupId>

<artifactId>junit-platform-launcher</artifactId>

<version>1.2.0</version>

</dependency>

</dependencies>

</project>

Package :com.ecommerce.tests

Name :ConditionalTets

package com.ecommerce.tests;

import org.junit.jupiter.api.\*;

import org.junit.jupiter.api.AfterAll;

import org.junit.jupiter.api.BeforeAll;

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.condition.DisabledIf;

import org.junit.jupiter.api.condition.EnabledOnOs;

import org.junit.jupiter.api.condition.OS;

import org.junit.platform.runner.JUnitPlatform;

import org.junit.runner.RunWith;

@DisplayName("JUnit 5 Conditional Tests Example")

@RunWith(JUnitPlatform.class)

public class ConditionalTests {

@Test

@EnabledOnOs({OS.WINDOWS})

public void runOnWindows() {

System.out.println("This runs on Windows");

}

@Test

@EnabledOnOs({OS.LINUX})

public void runOnLinux() {

System.out.println("This runs on Linux");

}

@Test

@DisabledIf("Math.random() < 0.314159")

void mightNotBeExecuted() {

System.out.println("This may or not execute ");

}

}

13.Nested & Repeated Tests

Pom.xml

<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/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>UsingJUnit</groupId>

<artifactId>UsingJUnit</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<version>5.4.2</version>

</dependency>

<dependency>

<groupId>org.junit.platform</groupId>

<artifactId>junit-platform-launcher</artifactId>

<version>1.2.0</version>

</dependency>

</dependencies>

</project>

Package :com.ecommerce.tests

Name :Caluclator

package com.ecommerce.tests;

public class Calculator

{

public int add(int a, int b) {

return a + b;

}

}

Package :com.ecommerce.tests

Name :NestedCases

package com.ecommerce.tests;

import org.junit.jupiter.api.\*;

import org.junit.jupiter.api.AfterAll;

import org.junit.jupiter.api.BeforeAll;

import org.junit.jupiter.api.Test;

import org.junit.platform.runner.JUnitPlatform;

import org.junit.runner.RunWith;

@DisplayName("JUnit 5 Nested Example")

@RunWith(JUnitPlatform.class)

public class NestedCases {

@BeforeAll

static void beforeAll() {

System.out.println("Before all test methods");

}

@BeforeEach

void beforeEach() {

System.out.println("Before each test method");

}

@AfterEach

void afterEach() {

System.out.println("After each test method");

}

@AfterAll

static void afterAll() {

System.out.println("After all test methods");

}

@Nested

@DisplayName("Tests for the method A")

class A {

@BeforeEach

void beforeEach() {

System.out.println("Before each test method of the A class");

}

@AfterEach

void afterEach() {

System.out.println("After each test method of the A class");

}

@Test

@DisplayName("Example test for method A")

void sampleTestForMethodA() {

System.out.println("Example test for method A");

}

@Nested

@DisplayName("When X is true")

class WhenX {

@BeforeEach

void beforeEach() {

System.out.println("Before each test method of the WhenX class");

}

@AfterEach

void afterEach() {

System.out.println("After each test method of the WhenX class");

}

@Test

@DisplayName("Example test for method A when X is true")

void sampleTestForMethodAWhenX() {

System.out.println("Example test for method A when X is true");

}

}

}

}

Package :com.ecommerce.tests

Name :RepeatedTests

package com.ecommerce.tests;

import org.junit.jupiter.api.\*;

import org.junit.jupiter.api.AfterAll;

import org.junit.jupiter.api.BeforeAll;

import org.junit.jupiter.api.Test;

import org.junit.platform.runner.JUnitPlatform;

import org.junit.runner.RunWith;

import static org.junit.jupiter.api.Assertions.assertEquals;

@DisplayName("JUnit 5 Repeated Tests Example")

@RunWith(JUnitPlatform.class)

public class RepeatedTests {

@BeforeAll

public static void init(){

System.out.println("Before All init() method called");

}

@BeforeEach

public void initEach(){

System.out.println("Before Each initEach() method called");

}

@Test

@DisplayName("Add operation test")

@RepeatedTest(5)

void addNumber(TestInfo testInfo) {

Calculator calculator = new Calculator();

Assertions.assertEquals(2, calculator.add(1, 1), "1 + 1 should equal 2");

System.out.println("===addNumber testcase executed===");

}

@AfterEach

public void cleanUpEach(){

System.out.println("After Each cleanUpEach() method called");

}

@AfterAll

public static void cleanUp(){

System.out.println("After All cleanUp() method called");

}

}

14.Dynamic Tests

Pom.xml

<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/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>UsingJUnit</groupId>

<artifactId>UsingJUnit</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<version>5.4.2</version>

</dependency>

<dependency>

<groupId>org.junit.platform</groupId>

<artifactId>junit-platform-launcher</artifactId>

<version>1.2.0</version>

</dependency>

</dependencies>

</project>

Package :com.ecommerce.tests

Name :DynamicTests

package com.ecommerce.tests;

import java.util.Arrays;

import java.util.Collection;

import java.util.List;

import org.junit.jupiter.api.\*;

import org.junit.jupiter.api.AfterAll;

import org.junit.jupiter.api.BeforeAll;

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.condition.DisabledIf;

import org.junit.jupiter.api.condition.EnabledOnOs;

import org.junit.jupiter.api.condition.OS;

import org.junit.jupiter.api.function.Executable;

import org.junit.platform.runner.JUnitPlatform;

import org.junit.runner.RunWith;

import org.junit.jupiter.api.DynamicTest.\*;

import org.junit.jupiter.api.TestFactory;

@DisplayName("JUnit 5 Dynamic Tests Example")

@RunWith(JUnitPlatform.class)

public class DynamicTests {

@TestFactory

Collection<DynamicTest> dynamicTests() {

return Arrays.asList(

dynamicTest("simple dynamic test", () -> assertTrue(true)),

dynamicTest("My Executable Class", new MyExecutable()),

dynamicTest("Exception Executable", () -> {throw new Exception("Exception Example");}),

dynamicTest("simple dynamic test-2", () -> assertTrue(true))

);

}

}

class MyExecutable implements Executable {

@Override

public void execute() throws Throwable {

System.out.println("Hello World!");

}

}

15.Dependency Injection

Pom.xml

<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/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>UsingJUnit</groupId>

<artifactId>UsingJUnit</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<version>5.4.2</version>

</dependency>

<dependency>

<groupId>org.junit.platform</groupId>

<artifactId>junit-platform-launcher</artifactId>

<version>1.2.0</version>

</dependency>

</dependencies>

</project>

Package :com.ecommerce.tests

Name :DependencyInjection

package com.ecommerce.tests;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.junit.jupiter.api.Assertions.assertTrue;

import org.junit.jupiter.api.\*;

import org.junit.jupiter.api.AfterAll;

import org.junit.jupiter.api.BeforeAll;

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.condition.DisabledIf;

import org.junit.jupiter.api.condition.EnabledOnOs;

import org.junit.jupiter.api.condition.OS;

import org.junit.platform.runner.JUnitPlatform;

import org.junit.runner.RunWith;

@DisplayName("JUnit 5 Dependency Injection Example")

@RunWith(JUnitPlatform.class)

public class DependencyInjection {

DependencyInjection(TestInfo testInfo) {

assertEquals("JUnit 5 Dependency Injection Example", testInfo.getDisplayName());

}

@BeforeEach

void init(TestInfo testInfo) {

String displayName = testInfo.getDisplayName();

assertTrue(displayName.equals("TEST 1") || displayName.equals("test2()"));

}

@Test

@DisplayName("TEST 1")

@Tag("my-tag")

void test1(TestInfo testInfo) {

assertEquals("TEST 1", testInfo.getDisplayName());

assertTrue(testInfo.getTags().contains("my-tag"));

}

@Test

void test2() {

}

}

16.Build RESTful with SpringBoot

Main Method

package com.test.example;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class RestApiTestApplication {

public static void main(String[] args) {

SpringApplication.run(RestApiTestApplication.class, args);

}

}

Entity class

package com.test.example;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class ProductEntity {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

@Column(name = "id", updatable = false, nullable = false)

private int id;

@Column

private String name;

@Column

private String description;

public ProductEntity() {

super();

}

public ProductEntity(int id, String name, String description) {

super();

this.id = id;

this.name = name;

this.description = description;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

}

Repository class

package com.test.example;

import org.springframework.data.jpa.repository.JpaRepository;

public interface ProductRepository extends JpaRepository<ProductEntity, Integer>{

}

Service class

package com.test.example;

import java.util.List;

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

import org.springframework.stereotype.Service;

@Service

public class ProductService {

@Autowired

ProductRepository productRepository;

public List<ProductEntity> getAllProduct(){

return productRepository.findAll();

}

public ProductEntity getProduct(int id){

return productRepository.findById(id).get();

}

public void addProduct(ProductEntity pe){

productRepository.save(pe);

}

public void updateProduct(int id, ProductEntity pe){

if(productRepository.findById(id).isPresent()) {

ProductEntity oldProductEntity=productRepository.findById(id).get();

oldProductEntity.setName(pe.getName());

oldProductEntity.setDescription(pe.getDescription());

productRepository.save(oldProductEntity);

}

}

public void deleteProduct(int id){

productRepository.deleteById(id);

}

}

Controller class

package com.test.example;

import java.util.List;

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

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

import org.springframework.web.bind.annotation.RequestBody;

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

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

import org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping(path = "/webapi")

public class ProductController {

@Autowired

ProductService productService;

@RequestMapping("/allproduct")

public List<ProductEntity> getAllProduct(){

return productService.getAllProduct();

}

@RequestMapping("/product/{id}")

public ProductEntity getProduct(@PathVariable int id){

return productService.getProduct(id);

}

@RequestMapping(method=RequestMethod.POST, value="/product")

public void addProduct(@RequestBody ProductEntity pe ) {

productService.addProduct(pe);

}

@RequestMapping(method=RequestMethod.PUT, value="/product/{id}")

public void updateProduct(@PathVariable int id, @RequestBody ProductEntity pe ) {

productService.updateProduct(id, pe);

}

@RequestMapping(method=RequestMethod.DELETE, value="/product/{id}")

public void deleteProduct(@PathVariable int id) {

productService.deleteProduct(id);

}

}