Ex. No: 4_1	Study and Implement MVC using Spring Framework.
Date:	

Code:

Link.html:-

```
<html>
<body>
<a href="Link.spring">GO TO SPRING HOME PAGE</a>
</body>
</html>
```

Web.xml:-

```
<servlet>
<servlet-name>Dispatcher</servlet-name>
<servlet-class>
org.springframework.web.servlet.DispatcherServlet
</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>Dispatcher</servlet-name>
<url-pattern>*.spring</url-pattern>
</servlet-mapping>
```

Dispatcher-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:p="http://www.springframework.org/schema/p"
xsi:schemaLocation="http://www.springframework
.org/schema/beans http://www.springframework
.org/schema/beans/spring-beans-2.5.xsd">
<bean name="/Link.spring" class="LinkController"/>
</beans>
```

LinkController.java:-

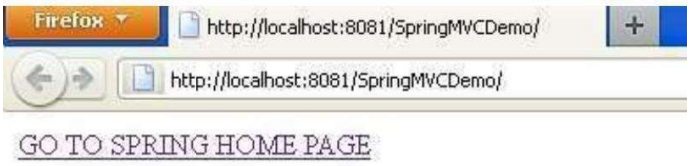
```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.Controller;
public class LinkController implements Controller {
public ModelAndView handleRequest(HttpServletRequest arg0,
```

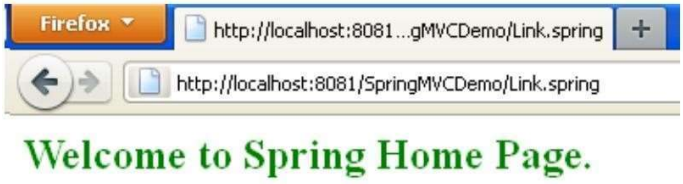
```
HttpServletResponse arg1) throws Exception {
return new ModelAndView("Welcome.jsp");
}
}
```

Welcome.jsp:-

```
<html>
<body>
<h2>
<font  color="Green">
Welcome to Spring Home Page.
</font>
</h2>
</body>
</html>
```

Output:-





Ex. No: 4_2	Develop a spring application which demonstrate bean life cycle.
Date:	

Code:

Beans.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"
xsi:schemaLocation = "http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

<bean id = "helloWorld" class = "com.ss.HelloWorld"
init-method = "init" destroy-method="destroy">
<property name = "msg" value = "Hello World!"/>
</bean>

</beans>
```

SpringBeanLife.java:-

```
package com.ss;

import org.springframework.beans.factory.BeanFactory;
import org.springframework.beans.factory.xml.XmlBeanFactory;
import org.springframework.core.io.ClassPathResource;
import org.springframework.core.io.Resource;

public class SpringBeanLife {

   public static void main(String[] args) {
      Resource r=new ClassPathResource("com/ss/Beans.xml");
      BeanFactory factory=new XmlBeanFactory(r);
      HelloWorld a = factory.getBean("helloWorld",HelloWorld.class);
      a.getMsg();
   }
}
```

HelloWorld.java:-

```
package com.ss;
public class HelloWorld {

   private String msg;

   public void getMsg() {
      System.out.println("Your Msg:"+msg);
```

```
    }

    public void setMsg(String msg) {
        this.msg= msg;
    }

    public void init(){

        System.out.println("initialized");

    }
    public void destroy(){

        System.out.println("destroyed");

    }
}
```

Output:-

```
run:
initialized
Your Msg:Hello World!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Ex. No: 4_3	Inject Service using Aspect Oriented Programming.
Date:	

Code:

applicationContext.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:p="http://www.springframework.org/schema/p"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

   <bean id="obj" class="A"></bean>
   <bean id="ba" class="BeforeAdvisor"></bean>

   <bean id="proxy" class="org.springframework.aop.framework.ProxyFactoryBean">
   <property name="target" ref="obj"></property>
   <property name="interceptorNames">
   <list>
   <value>ba</value>
   </list>
   </property>
   </bean>

   </beans>
```

A.java:-

```
public class A {
   public void m(){
      System.out.println("actual business logic");
   }
}
```

BeforeAdvisor.java:-

```
import java.lang.reflect.Method;
import org.springframework.aop.MethodBeforeAdvice;

public class BeforeAdvisor implements MethodBeforeAdvice{

   @Override
   public void before(Method method, Object[] args, Object target)throws Throwable {
      System.out.println("additional concern before actual logic");
   }
}
```

Test.java:-

```
import org.springframework.beans.factory.BeanFactory;
import org.springframework.beans.factory.xml.XmlBeanFactory;
import org.springframework.core.io.ClassPathResource;
import org.springframework.core.io.Resource;

public class Test {
   public static void main(String[] args) {
   Resource r=new ClassPathResource("applicationContext.xml");
   BeanFactory factory=new XmlBeanFactory(r);

   A a=(A)factory.getBean("proxy",A.class);
   a.m();
}

}
```

Output:

```
run:
additional concern before actual logic
actual business logic
BUILD SUCCESSFUL (total time: 0 seconds)
```

Ex. No: 4_4	Using Spring Template manages Database and Transaction.
Date:	

Code:

Beans.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:p="http://www.springframework.org/schema/p"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

<!-- Initialization for data source -->
  <bean id = "dataSource"
     class = "org.springframework.jdbc.datasource.DriverManagerDataSource">
     <property name = "driverClassName" value = "com.mysql.cj.jdbc.Driver"/>
     <property name = "url" value = "jdbc:mysql://localhost:3309/mu"/>
     <property name = "username" value = "root"/>
     <property name = "password" value = ""/>
  </bean>

  <!-- Initialization for TransactionManager -->
  <bean id = "transactionManager"
     class  =  "org.springframework.jdbc.datasource.DataSourceTransactionManager">
     <property name = "dataSource"  ref = "dataSource" />
  </bean>

  <!-- Definition for studentJDBCTemplate bean -->
  <bean id = "studentJDBCTemplate"
     class = "StudentJDBCTemplate">
     <property name = "dataSource" ref = "dataSource" />
     <property name = "transactionManager" ref = "transactionManager" />
  </bean>


 </beans>
```

MainApp.java:-

```
import java.util.List;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
```

```
public class MainApp {
  public static void main(String[] args) {
    ApplicationContext context = new ClassPathXmlApplicationContext("Beans.xml");
    StudentJDBCTemplate studentJDBCTemplate =
      (StudentJDBCTemplate)context.getBean("studentJDBCTemplate");

    System.out.println("------Records creation --------" );
   studentJDBCTemplate.create("bhuna", 11, 99, 2010);
   studentJDBCTemplate.create("eshu", 20, 97, 2010);
  studentJDBCTemplate.create("neha", 25, 100, 2011);
    System.out.println("------Listing all the records --------" );
    List<StudentMarks> studentMarks = studentJDBCTemplate.listStudents();

    for (StudentMarks record : studentMarks) {
      System.out.print("ID : " + record.getId() );
      System.out.print(", Name : " + record.getName() );
      System.out.print(", Marks : " + record.getMarks());
      System.out.print(", Year : " + record.getYear());
      System.out.println(", Age : " + record.getAge());
    }
  }
}
```

StudentDAO.java:-

```
import java.util.List;
import javax.sql.DataSource;

public interface StudentDAO {
  /**
    * This is the method to be used to initialize
    * database resources ie. connection.
  */
  public void setDataSource(DataSource ds);

  /**
    * This is the method to be used to create
    * a record in the Student and Marks tables.
  */
  public void create(String name, Integer age, Integer marks, Integer year);

  /**
    * This is the method to be used to list down
    * all the records from the Student and Marks tables.
  */
  public List<StudentMarks> listStudents();
}
```

StudentJDBCTemplate.java:-

```
import java.util.List;
import javax.sql.DataSource;

import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.transaction.PlatformTransactionManager;
import org.springframework.transaction.TransactionDefinition;
import org.springframework.transaction.TransactionStatus;
import org.springframework.transaction.support.DefaultTransactionDefinition;

public class StudentJDBCTemplate implements StudentDAO {
  private DataSource dataSource;
  private JdbcTemplate jdbcTemplateObject;
  private PlatformTransactionManager transactionManager;

  public void setDataSource(DataSource dataSource) {
    this.dataSource = dataSource;
    this.jdbcTemplateObject = new JdbcTemplate(dataSource);
  }

  public void setTransactionManager(PlatformTransactionManager transactionManager) {
    this.transactionManager = transactionManager;
  }

  public void create(String name, Integer age, Integer marks, Integer year) {
    TransactionDefinition def = new DefaultTransactionDefinition();
    TransactionStatus status = transactionManager.getTransaction(def);

    try {
      String SQL1 = "INSERT INTO Student (sname, age) VALUES (?, ?)";
      jdbcTemplateObject.update(SQL1, name, age);

      // Get the latest student id to be used in the Marks table
      String SQL2 = "SELECT MAX(id) FROM Student";
      int sid = jdbcTemplateObject.queryForObject(SQL2, Integer.class);

      String SQL3 = "INSERT INTO Marks (sid, marks, yr) VALUES (?, ?, ?)";
      jdbcTemplateObject.update(SQL3, sid, marks, year);

      System.out.println("Created Name = " + name + ", Age = " + age);
      transactionManager.commit(status);
    } catch (DataAccessException e) {
      System.out.println("Error in creating record, rolling back");
      transactionManager.rollback(status);
      throw e;
    }
  }

  public List<StudentMarks> listStudents() {
```

```
    String SQL = "SELECT * FROM Student, Marks WHERE Student.id = Marks.sid";
    List<StudentMarks> studentMarks = jdbcTemplateObject.query(SQL, new StudentMarksMapper());
    return studentMarks;
  }
}
```

StudentMarks.java:-

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author admin
 */
public class StudentMarks {

  private Integer age;
  private String name;
  private Integer id;
  private Integer marks;
  private Integer year;
  private Integer sid;

  public void setAge(Integer age) {
    this.age = age;
  }
  public Integer getAge() {
    return age;
  }
  public void setName(String name) {
    this.name = name;
  }
  public String getName() {
    return name;
  }
  public void setId(Integer id) {
    this.id = id;
  }
  public Integer getId() {
    return id;
  }
  public void setMarks(Integer marks) {
    this.marks = marks;
  }
  public Integer getMarks() {
    return marks;
  }
  public void setYear(Integer year) {
```

```
    this.year = year;
  }
  public Integer getYear() {
    return year;
  }
  public void setSid(Integer sid) {
    this.sid = sid;
  }
  public Integer getSid() {
    return sid;
  }
}
```

StudentMarkMapper.java:-

```
import java.sql.ResultSet;
import java.sql.SQLException;
import org.springframework.jdbc.core.RowMapper;

public class StudentMarksMapper implements RowMapper<StudentMarks> {
  public StudentMarks mapRow(ResultSet rs, int rowNum) throws SQLException {
    StudentMarks studentMarks = new StudentMarks();
    studentMarks.setId(rs.getInt("id"));
    studentMarks.setName(rs.getString("sname"));
    studentMarks.setAge(rs.getInt("age"));
    studentMarks.setSid(rs.getInt("sid"));
    studentMarks.setMarks(rs.getInt("marks"));
    studentMarks.setYear(rs.getInt("yr"));

    return studentMarks;
  }
}
```

Output:-