Dt : 29/3/2025

*imp

JSP Programming:

=>JSP stands for 'Java Server Page' and which is response from Web Application.

=>JSP is tag based programming language and which is more simple when compared to Servlet Programming.

=>JSP program is combination of both HTML code and Java Code,which means in JSP Programs we can write HTML code and Java Code.

=>JSP programs are saved with (.jsp) as an extention.

_Ex:_

View.jsp

=>we use the following JSP tags to write Java Code in JSP programs:

1.Scripting Tags

2.Directive Tags

3.Action Tags

1.Scripting Tags:

=>The tags which are used to write normal code are known as Scripting Tags.

=>Types:

(a)Scriptlet Tag

(b)Expression Tag

(c)Declarative Tag

(a)Scriptlet Tag:

=>Scriptlet Tag is used to write normal Java Code in JSP programs.

syntax:

```
<%
  ------JavaCode/ServletCode-----
%>
```

(b)Expression Tag:

=>Expression Tag is used to assign the value to variable or which is used to display the data directly to WebBrowser.

syntax:

```
<%=
  -----Value/Expression----
%>
```

(c)Declarative Tag:

=>Declarative tag is used to declare Variables and methods in JSP programs.

syntax:

```
<%!
  -----Variables/Methods-----
%>
```

--

2.Directive Tags:

=>Directive tags will specify the directions(specifications) in translation process.

=>Types:

(a)@page

(b)@include

(c)@taglib

(a)@page:

=>'@page' tag will provide specifications about current JSP page.

syntax:

```
<%@page language="java" setContentType="text/html" import="java.util.*" ...%>
```

(b)@include:

=>'@include' tag will specify the file which is included with the response.

syntax:

```
<%@include file="file-name" %>
```

(c)@taglib:

=>'@taglib' tag is used to link the external libraries.

syntax:

```
<%@taglib uri="lib...." prefix="nm"...%>
```

3.Action Tags:

=>Actions tags are used to peform some actions while execution process.

Types:

(a)<jsp:forward>

(b)<jsp:include>

(c)<jsp:param>

(d)<jsp:useBean>

(e)<jsp:setProperties>

(f)<jsp:getProperties>

===

****imp***

Bean Class in Servlet Programming:

=>The class which is declared with the following rules is known as 'Bean Class'

Rule-1 : The class must be implemented from 'java.io.Serializable' interface.

Rule-2 : The variables declared in the class must be 'private' variables.

Rule-3 : The class must be declared with 0-argument Constructor or 0-parameter Constructor

Rule-4 : The Class must be declared with 'Getter and Setter' methods.

Note:

=>These Bean classes will generate bean-objects and which are intermediate storages b/w

Servlet-prorams and Database product

===

Dt : 1/4/2025

****imp***

DAO in JDBC:

=>DAO stands for 'Data Access Object' and which is separate layer in MVC

(Model View Controller) to hold database related codes(Persistent Logics).

=>This DAO Layer includes DB-Connection,DB-Create,DB-Insert,DB-Retrieve,DB-Update and DB-Delete

=>In the process of establishing Communication b/w Servlet-Program and DB-Product, the DB-Jar file must be copied into "lib" folder of 'WEB-INF'.

Layout:

Ex-Application:

Servlet Application to perform the following two operations on Database Product

1.AddBookDetails

2.ViewBookDetailsByCode

DB Table : BookDetails72(code,name,author,price,qty)

primary key : code

create table BookDetails72(code varchar2(10),name varchar2(15),author varchar2(15),

price number(10,2),qty number(10),primary key(code));

BookBean.java

```
package test;
import java.io.*;
@SuppressWarnings("serial")
public class BookBean implements Serializable
{
   private String code,name,author;
   private float price;
   private int qty;
   public BookBean() {}
```

```java
public String getCode() {
      return code;
}
public void setCode(String code) {
      this.code = code;
}
public String getName() {
      return name;
}
public void setName(String name) {
      this.name = name;
}
public String getAuthor() {
      return author;
}
public void setAuthor(String author) {
      this.author = author;
}
public float getPrice() {
      return price;
}
public void setPrice(float price) {
      this.price = price;
}
public int getQty() {
      return qty;
}
public void setQty(int qty) {
      this.qty = qty;
}

}
```

DBInfo.java

```java
package test;
public interface DBInfo
{
   public static final String
driver="oracle.jdbc.driver.OracleDriver";
   public static final String
dbURL="jdbc:oracle:thin:@localhost:1521:xe";
   public static final String dbUName="system";
   public static final String dbPWord="tiger";
}
```

DBConnection.java

```
package test;
import java.sql.*;
public class DBConnection
{
    private static Connection con = null;
    private DBConnection() {}
    static
    {
      try {
            Class.forName(DBInfo.driver);
            con = DriverManager.getConnection
                        (DBInfo.dbURL,DBInfo.dbUName,DBInfo.dbPWord);
      }catch(Exception e) {
            e.printStackTrace();
      }
    }//end of block
    public static Connection getCon()
    {
      return con;
    }
}
```

home.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<a href="BookDetails.html">AddBookDetails</a>
<a href="view">ViewBookDetailsByCode</a>
</body>
</html>
```

BookDetails.html

```
<!DOCTYPE html>
```

```html
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="add" method="post">
BookCode:<input type="text" name="bcode"><br>
BookName:<input type="text" name="bname"><br>
BookAuthor:<input type="text" name="bauthor"><br>
BookPrice:<input type="text" name="bprice"><br>
BookQty:<input type="text" name="bqty"><br>
<input type="submit" value="AddBookDetails">
</form>
</body>
</html>
```

web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <welcome-file-list>
     <welcome-file>home.html</welcome-file>
  </welcome-file-list>
</web-app>
```
