*Advance Java*

*Topics*

|  |  |  |
| --- | --- | --- |
| *S.no* | *Topic* | *Page.no* |
| *1.* | *Jdbc* |  |
| *2.* | *servlets* |  |
| *3.* | *jsp* |  |
| *4.* | *spring* |  |
| *5.* | *hibernate* |  |
| *6.* | *spring boot* |  |

*JDBC*

* *java database connectivity.*
* *java environment -> db. environment*
* *to store the data or to perform the operations on the data (select, insert, delete and update)*
* *it is a technology given by sunmicrosystem technology/specification/interfaces*

* *implementatin classes are provided by different different database vendors like oracle,mysql,sybase,ingress,redhat,postgres in the form of jar files*
* ***goal:*** *from java environment, we need to perform db. operations*

*On top of the existing data, we performed db. operations-*

*-insert the data*

*-update the data*

*-delete the data*

*-read the data*

***Steps to develop/write the jdbc program:***

*java<-------->db*

*each and every db provides 4 types of driver classes those are-*

*1.type1*

*2.type2*

*3.type3*

*4.type4(java language)*

*oracle database*

*type4 driver class name of oracle database?*

*->oracle.jdbc.driver.OracleDriver*

1. ***load the driver class***

*Class.forName("oracle.jdbc.driver.OracleDriver");*

1. ***obtain the connection from database***

*DriverManager.getConnection();*

*jdbc:oracle:thin:@localhost:1521:xe*

*jdbc -main protocol*

*oracle - sub protocol*

*thin- nick name of type 4 driver of oracle*

*localhost - ip address of db machine*

*1521 - default port number of database oracle*

*xe - default global database name of oracle express edition*

1. ***Create the Statement object***

***purpose***

*Creates a Statement object for sending SQL statements to the database.*

1. ***process the query***

*int x = statement.executeUpdate("insert/update/delete");*

1. ***close the connection***

*connection.close();*

***Summary:***

*->load the drier class*

*->obtain the connection*

*->Statement object creation*

*->Process the query*

*->Close the connection*

***example:***

*package com.sopra.jdbc;*

*import java.sql.Connection;*

*import java.sql.DriverManager;*

*import java.sql.Statement;*

*public class Test1 {*

*public static void main(String[] args) {*

*try {*

*//load the driver class*

*Class.forName("oracle.jdbc.driver.OracleDriver");*

*System.out.println("driver class is loaded");*

*Connection connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "sopra", "sopra");*

*System.out.println("i got the connection");*

*//Statement object creation for sending sql statemnts to db*

*Statement statement = connection.createStatement();*

*System.out.println("statement object got created for sending sql queries to the db");*

*//processing the queries*

*int x = statement.executeUpdate("insert into student values(800,'umesh')");*

*System.out.println(x + "row(s) inserted");*

*//close the conenction*

*connection.close();*

*}*

*catch(Exception e)*

*{*

*System.out.println(e);*

*}*

*}*

*}*

*-update and delete*

*dml operations -insert/update/delete*

*statement.executeUpdate();*

*select \* from student;*

*statement.executeQuery()*

*package com.sopra.jdbc;*

*import java.sql.Connection;*

*import java.sql.DriverManager;*

*import java.sql.ResultSet;*

*import java.sql.ResultSetMetaData;*

*import java.sql.Statement;*

*public class Test3 {*

*public static void main(String[] args) throws Exception {*

*Class.forName("oracle.jdbc.driver.OracleDriver");*

*Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","sopra","sopra");*

*Statement st = con.createStatement();*

*ResultSet rs = st.executeQuery("select \* from student");*

*ResultSetMetaData rsmd = rs.getMetaData();*

*System.out.println("No.of columns in my table is : " + rsmd.getColumnCount());*

*System.out.println("1st column data type and it's size is : " + rsmd.getColumnTypeName(1) + " " + rsmd.getPrecision(1));*

*System.out.println("2nd column data type and it's size is : " + rsmd.getColumnTypeName(2) + " " + rsmd.getPrecision(2));*

*for(int i=1;i<=rsmd.getColumnCount();i++)*

*System.out.print(rsmd.getColumnName(i) + "\s");*

*System.out.println();*

*while(rs.next())*

*{*

*System.out.println(rs.getInt(1) + " " + rs.getString(2));*

*}*

*}}*

*Servlet*

*java - james gosling - 1995*

*->jse -> standalone /desktop applications*

*->jee -> distributed/ web applications.*

*using servlet -> web applications*

*->static web applications -> html*

*->dynamic web applications -> facebook,flipkart,gmail...*

*we are using server side technologies to develop dynamic web*

*applications.*

*servlet -> server side technology*

*purpose: to develop dynamic web applications*

*given by: sun microsystem*

*->www.oracle.com*

*sunmicrosystem -> oracle*

*to develop web applications, usually we have to follow client server*

*architecture.*

*The client server computing works with a system of request and response.*

*The client sends a request to the server and the server responds with the*

*desired information*

*url: www.facebook.com -> fron the browser address bar*

*browser -> client(request)*

*from the server -> response -> facebook home page*

*click on signin -> request*

*-> response -> signin page ->*

*....*

*client ->*

*server -> server is available remotely.*

*client is used to send the request and server is used to process the*

*request and gives response back to the client.*

*using servlet technology -> web applications.*

*server -> tomcat*

*so many servers are availabe in the real world*

*tomcat,glassfish,weblogic,websphere,redhat....*

*tomcat -> apache software foundation -> web server*

*->https://www.flipkart.com/*

*Servlet is an interface -> javax.servlet*

*-> GenericServlet (javax.servlet)*

*-> HttpServlet(javax.servlet.http)*

*note: all the advanced java packages starts with javax.*

*note: all the core java packages starts with java.*

*java.sql*

*java.lang*

*java.io*

*java.util.*

*how we can define our own servlet:*

*->by implemeting Servlet interface*

*->by extending GenericServlet*

*->by extending HttpServlet*

*class MyServlet implements Servlet*

*{*

*//5 methods (init,destroy,service,getServletConfig,getServletInfo)*

*}*

*class MyServlet extends GenericServlet*

*{*

*//only service*

*}*

*class MyServlet extends HttpServlet*

*{*

*//only service*

*}*

*protocol independent applications-> any protocol support (http,smtp,tcp...)*

*protocol dependent applications ->using http protocol*

*note:*

*maintaing the protocols is very expensive.*

***why web.xml file is required?***

*-> deployment descriptor*

*client --> to whom we are sending the request -> server*

*src/main/webapp/WEB-INF/web.xml:*

*``````````````````````````````````*

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

*<web-app>*

*<servlet>*

*<servlet-name>sopra</servlet-name>*

*<servlet-class>com.sopra.servlets.MyServlet</servlet-class>*

*</servlet>*

*<servlet-mapping>*

*<servlet-name>sopra</servlet-name>*

*<url-pattern>/display</url-pattern>*

*</servlet-mapping>*

*</web-app>*

*src/main/java*

*`````````````*

*com.sopra.servlet*

***MyServlet.java:***

*package com.sopra.servlets;*

*import java.io.IOException;*

*import java.io.PrintWriter;*

*import javax.servlet.http.HttpServlet;*

*import javax.servlet.http.HttpServletRequest;*

*import javax.servlet.http.HttpServletResponse;*

*//servlet class*

*public class MyServlet extends HttpServlet {*

*//*

*public void doGet(HttpServletRequest request,HttpServletResponse response) throws IOException*

*{*

*response.setContentType("text/html");*

*PrintWriter pw = response.getWriter();*

*pw.println("<h1>welcome to servlets</h1>");*

*}*

*}*

*->select dynamic web project*

*->add runtime -> add the server*

*->Finish*

*->under src/main/java -> create one package and define one servlet class*

*by extending HttpServlet*

*->under src/webapp/WEB-INF/web.xml*

*Form Handling in Servlet*

*html form*

*-> submit*

*-> control goes to respective servlet*

*-> write the business logic*

*-> return response back to the client*

*under webapps folder:*

*``````````````````````*

*index.html:*

*````````````*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<form action="add">*

*<pre>*

*Enter first number: &nbsp;<input type="text" name="t1" /><br/>*

*Enter second number: <input type="text" name="t2" /><br />*

*<input type="submit" value="add" />*

*</pre>*

*</form>*

*</body>*

*</html>*

*AddServlet.java:*

*````````````````*

*package com.sopra.servlets;*

*import java.io.IOException;*

*import java.io.PrintWriter;*

*import javax.servlet.ServletException;*

*import javax.servlet.annotation.WebServlet;*

*import javax.servlet.http.HttpServlet;*

*import javax.servlet.http.HttpServletRequest;*

*import javax.servlet.http.HttpServletResponse;*

*public class AddServlet extends HttpServlet {*

*protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {*

*//business logic*

*int x = Integer.parseInt(request.getParameter("t1"));*

*int y = Integer.parseInt(request.getParameter("t2"));*

*int z = x+y;*

*PrintWriter pw = response.getWriter();*

*pw.println("<h2>");*

*pw.println("the sum is : " + z);*

*pw.println("</h2>");*

*}*

*}*

***web.xml:***

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

*<web-app>*

*<servlet>*

*<servlet-name>Ram</servlet-name>*

*<servlet-class>com.sopra.servlets.AddServlet</servlet-class>*

*</servlet>*

*<servlet-mapping>*

*<servlet-name>Ram</servlet-name>*

*<url-pattern>/add</url-pattern>*

*</servlet-mapping>*

*</web-app>*

***url:***

*http://localhost:2021/form-handling/index.html*

*http://localhost:2021/form-handling/add?t1=200&t2=678*

*http://localhost:2021/form-handling/add*

*doGet() and doPost():*

*if we want to handle very sensitive data it is highly recommended to override*

*doPost() in servlet class make sure html form method type must be post*

*form method by default value is get*

*->if we want to handle insensitive(normal data) we can happily go for*

*override doGet()*

*index.html:*

*`````````````*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<form action="add" method="post">*

*<pre>*

*Enter first number: &nbsp;<input type="text" name="t1" /><br/>*

*Enter second number: <input type="text" name="t2" /><br />*

*<input type="submit" value="add" />*

*</pre>*

*</form>*

*</body>*

*</html>*

*AddServlet.java:*

*`````````````````*

*package com.sopra.servlets;*

*import java.io.IOException;*

*import java.io.PrintWriter;*

*import javax.servlet.ServletException;*

*import javax.servlet.annotation.WebServlet;*

*import javax.servlet.http.HttpServlet;*

*import javax.servlet.http.HttpServletRequest;*

*import javax.servlet.http.HttpServletResponse;*

*public class AddServlet extends HttpServlet {*

*protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {*

*//business logic*

*int x = Integer.parseInt(request.getParameter("t1"));*

*int y = Integer.parseInt(request.getParameter("t2"));*

*int z = x+y;*

*PrintWriter pw = response.getWriter();*

*pw.println("<h2>");*

*pw.println("the sum is : " + z);*

*pw.println("</h2>");*

*}*

*}*

*ServletConfig and ServletContext*

*ServletConfig*

* *Servlet Config is for configuring specific information to a specific*

*servlet.*

* *in web.xml under servlet tag using <init-param> tag we generally configured this servlet information.*
* *Servlet Config object is per each and every servlet of entire web*

*application.*

*ServletContext*

* *Servlet Context, it is used for specifying common/global information*

*of a web application, which can be used by all the servlets.*

* *in web.xml using <context-param> tag we can configured*
* *Servlet Context object is per one web application.*

***FirstServlet.java:***

*package com.sopra.servlets;*

*import java.io.IOException;*

*import javax.servlet.ServletConfig;*

*import javax.servlet.ServletContext;*

*import javax.servlet.ServletException;*

*import javax.servlet.http.HttpServlet;*

*import javax.servlet.http.HttpServletRequest;*

*import javax.servlet.http.HttpServletResponse;*

*public class FirstServlet extends HttpServlet {*

*protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {*

*ServletConfig config = this.getServletConfig();*

*String value = config.getInitParameter("driver");*

*response.getWriter().println("<h1>" + value + "</h1>");*

*ServletContext context = this.getServletContext();*

*String value2 = context.getInitParameter("firm");*

*response.getWriter().println("<h1>" + value2 + "</h1>");*

*}*

*}*

***SecondServlet.java***

*package com.sopra.servlets;*

*import java.io.IOException;*

*import javax.servlet.ServletConfig;*

*import javax.servlet.ServletContext;*

*import javax.servlet.ServletException;*

*import javax.servlet.annotation.WebServlet;*

*import javax.servlet.http.HttpServlet;*

*import javax.servlet.http.HttpServletRequest;*

*import javax.servlet.http.HttpServletResponse;*

*public class SecondServlet extends HttpServlet {*

*protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {*

*ServletConfig config = this.getServletConfig();*

*String value = config.getInitParameter("url");*

*response.getWriter().println("<h1>" + value + "</h1>");*

*ServletContext context = this.getServletContext();*

*String value2 = context.getInitParameter("firm");*

*response.getWriter().println("<h1>" + value2 + "</h1>");*

*}*

}

**web.xml:**

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

*<web-app>*

*<!-- first servlet configuration -->*

*<servlet>*

*<servlet-name>Ram</servlet-name>*

*<servlet-class>com.sopra.servlets.FirstServlet</servlet-class>*

*<init-param>*

*<param-name>driver</param-name>*

*<param-value>oracle.jdbc.driver.OracleDriver</param-value>*

*</init-param>*

*</servlet>*

*<servlet-mapping>*

*<servlet-name>Ram</servlet-name>*

*<url-pattern>/first</url-pattern>*

*</servlet-mapping>*

*<!-- second servlet configuration -->*

*<servlet>*

*<servlet-name>Ram1</servlet-name>*

*<servlet-class>com.sopra.servlets.SecondServlet</servlet-class>*

*<init-param>*

*<param-name>url</param-name>*

*<param-value>jdbc:oracle:thin:@localhost:1521:xe</param-value>*

*</init-param>*

*</servlet>*

*<servlet-mapping>*

*<servlet-name>Ram1</servlet-name>*

*<url-pattern>/second</url-pattern>*

*</servlet-mapping>*

*<!-- configuring global information -->*

*<context-param>*

*<param-name>firm</param-name>*

*<param-value>sopra</param-value>*

*</context-param>*

*</web-app>*

***Summary***

*-formhandling*

*-doGet vs doPost*

*-ServletConfig vs ServletContext*

*-assigment: ....*

*->servlet collaboration / servlet chaining*

*->servlet redirecting mechanism*

*->@WebServlet annotation(alternative for web.xml)*

*->session tracking techniques(cookies,urlrewriting,HttpSession)*

*servlet redirecting mechanism:*

*if we make a request to one servlet*

*requirement: need to redirect to some other web address*

*response.sendRedirect("address of desired location");*

*ex: response.sendRedirect("http://www.google.com");*

*old websites --> new website addresses*

***@WebServlet:***

*Use the @WebServlet annotation to define a servlet component in a web*

*application. This annotation is specified on a class and contains metadata*

*about the servlet being declared. The annotated servlet must specify at*

*least one URL pattern. This is done by using the urlPatterns or value*

*attribute on the annotation.*

***MyServlet.java:***

*package com.sopra.servlets;*

*import java.io.IOException;*

*import javax.servlet.ServletException;*

*import javax.servlet.annotation.WebServlet;*

*import javax.servlet.http.HttpServlet;*

*import javax.servlet.http.HttpServletRequest;*

*import javax.servlet.http.HttpServletResponse;*

*@WebServlet("/redirect")*

*public class MyServlet extends HttpServlet {*

*protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {*

*response.getWriter().println("welcome to servlet redirecting mechanism");*

*response.sendRedirect("http://www.google.com");*

*}*

*}*

*ServletCollaboration*

*if we want to send some data from one servlet to any other resource*

*(html,jsp,servlet...)*

*->establishing communication in between servlet to any other resource*

*RequestDispatcher is one of the pre-defined interface*

*FirstServlet.java:*

*```````````````````*

*package com.sopra.servlets;*

*import java.io.IOException;*

*import javax.servlet.RequestDispatcher;*

*import javax.servlet.ServletException;*

*import javax.servlet.annotation.WebServlet;*

*import javax.servlet.http.HttpServlet;*

*import javax.servlet.http.HttpServletRequest;*

*import javax.servlet.http.HttpServletResponse;*

*@WebServlet("/FirstServlet")*

*public class FirstServlet extends HttpServlet {*

*protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {*

*response.getWriter().println("iam from first servlet");*

*RequestDispatcher requestDispatcher = request.getRequestDispatcher("SecondServlet");*

*//requestDispatcher.forward(request, response);*

*response.getWriter().println("<br/>");*

*requestDispatcher.include(request, response);*

*}*

*}*

***SecondServlet.java:***

*package com.sopra.servlets;*

*import java.io.IOException;*

*import javax.servlet.ServletException;*

*import javax.servlet.annotation.WebServlet;*

*import javax.servlet.http.HttpServlet;*

*import javax.servlet.http.HttpServletRequest;*

*import javax.servlet.http.HttpServletResponse;*

*@WebServlet("/SecondServlet")*

*public class SecondServlet extends HttpServlet {*

*protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {*

*response.getWriter().println("iam from second servlet");*

*}*

*}*

*Session Tracking Techniques*

*session:*

*````````*

*->9am to 6pm*

*time interval in between login and logout is the session*

*gmail(url) -> request*

*-> username:*

*password:*

*clicking on the login --> request*

*gmail inbox*

*compose -> request*

*each and every action which we are performing on the web page*

*absolutely those are reequests.*

*these requests can be processed by server(our servlet programs)*

*how the request will be moved to server?*

*by using one protocol (http)*

*http: stateless protocol*

*stateless -> it is not rembering any data*

*since http is a stateless protocol, each and every request can be*

*considering as a new request.*

*we have to maintain the session*

*by using session tracking techniques in servlets.*

*-cookies(server side)*

*-HttpSession(server side)*

*-UrlRewriting(server side)*

*-HidenForm Fields(client side and server side)*

*goal: to maintain the state*

*Cookie*

*A cookie is a small piece of information that is persisted between the*

*multiple client requests.*

*client1 -request --> server*

*resposne+cookie*

*client1 --request + cookie --> server1*

*response*

*index.html:*

*```````````*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<form action="FirstServlet">*

*Enter username: <input type="text" name="uname"/><br/>*

*<input type="submit" value="go"/>*

*</form>*

*</body>*

*</html>*

*FirstServlet.java:*

*``````````````````*

*package com.sopra.servlets;*

*import java.io.IOException;*

*import java.io.PrintWriter;*

*import javax.servlet.ServletException;*

*import javax.servlet.annotation.WebServlet;*

*import javax.servlet.http.Cookie;*

*import javax.servlet.http.HttpServlet;*

*import javax.servlet.http.HttpServletRequest;*

*import javax.servlet.http.HttpServletResponse;*

*@WebServlet("/FirstServlet")*

*public class FirstServlet extends HttpServlet {*

*protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {*

*String uname=request.getParameter("uname");*

*PrintWriter pw = response.getWriter();*

*pw.println("welcome to " + uname);*

*pw.println("<br/>");*

*//Cookie object creation*

*Cookie cookie = new Cookie("c1",uname);*

*//add the cookie object to response*

*response.addCookie(cookie);*

*//preparing form in the server(firstservlet)*

*pw.println("<form action='SecondServlet'>");*

*pw.println("<input type='submit' value='send'/>");*

*pw.println("</form>");*

*}*

*}*

***SecondServlet.java***

*package com.sopra.servlets;*

*import java.io.IOException;*

*import java.io.PrintWriter;*

*import javax.servlet.ServletException;*

*import javax.servlet.annotation.WebServlet;*

*import javax.servlet.http.Cookie;*

*import javax.servlet.http.HttpServlet;*

*import javax.servlet.http.HttpServletRequest;*

*import javax.servlet.http.HttpServletResponse;*

*@WebServlet("/FirstServlet")*

*public class FirstServlet extends HttpServlet {*

*protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {*

*javax.servlet.http.Cookie[] cookie=request.getCookies();*

*response.getWriter().println((String)cookie[0].getValue());*

*}*

*}*

***assignment:***

*perform login,logout,profile using cookies in servlet.*

*HttpSession*

*server/container creates a session id for each user.*

*sopra --> session id*

*ram --> session id*

*.....*

*session id is unique*

*The server uses this session id to identify the particular user.*

*on top of HttpServletRequest object*

*HttpSession session = request.getSession()*

***frequently used methods of HttpSession:***

* *getId()*
* *getLastAccessedTime()*
* *invalidate()*

***index.html:***

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<form action="FirstServlet">*

*Enter Username: <input type="text" name="uname"><br/>*

*<input type="submit" value="go"/>*

*</form>*

*</body>*

*</html>*

***FirstServlet.java:***

*package com.sopra.servlets;*

*import java.io.IOException;*

*import java.io.PrintWriter;*

*import javax.servlet.ServletException;*

*import javax.servlet.annotation.WebServlet;*

*import javax.servlet.http.HttpServlet;*

*import javax.servlet.http.HttpServletRequest;*

*import javax.servlet.http.HttpServletResponse;*

*import javax.servlet.http.HttpSession;*

*@WebServlet("/FirstServlet")*

*public class FirstServlet extends HttpServlet {*

*protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {*

*PrintWriter pw = response.getWriter();*

*String u1 = request.getParameter("uname");*

*pw.println("welcome to : " + u1);*

*//create the HttpSession object*

*HttpSession session = request.getSession();*

*session.setAttribute("uname", u1);*

*pw.println("<br/>");*

*pw.println("<a href='SecondServlet'>clickme</a>");*

*}*

*}*

***SecondServlet.java:***

*package com.sopra.servlets;*

*import java.io.IOException;*

*import java.io.PrintWriter;*

*import javax.servlet.ServletException;*

*import javax.servlet.annotation.WebServlet;*

*import javax.servlet.http.HttpServlet;*

*import javax.servlet.http.HttpServletRequest;*

*import javax.servlet.http.HttpServletResponse;*

*import javax.servlet.http.HttpSession;*

*@WebServlet("/SecondServlet")*

*public class SecondServlet extends HttpServlet {*

*protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {*

*HttpSession session = request.getSession(false);*

*String n = (String)session.getAttribute("uname");*

*PrintWriter pw = response.getWriter();*

*pw.println("Hello : " + n);*

*pw.println("<br/>");*

*pw.println("session creation time: " + session.getCreationTime());*

*pw.println("<br/>");*

*pw.println("session id: " + session.getId());*

*}*

*}*

***ASSIGNMENTS:***

* *HiddenFormFields*
* *UrlRewriting*
* *login,logout & profiles using cookies*

***EXAMPLE: gmail is implemented using cookies***

*-servlet redirecting mechanism*

*-servlet collobaration (forward and include)*

*-@WebServlet*

*-session tracking techniques*

*-cookies*

*-HttpSession*

*-HiddenFormFields*

*-UrlRewriting*

***How to invalidate the session:***

*package com.sopra.servlets;*

*import java.io.IOException;*

*import java.io.PrintWriter;*

*import java.util.Date;*

*import javax.servlet.ServletException;*

*import javax.servlet.annotation.WebServlet;*

*import javax.servlet.http.HttpServlet;*

*import javax.servlet.http.HttpServletRequest;*

*import javax.servlet.http.HttpServletResponse;*

*import javax.servlet.http.HttpSession;*

*@WebServlet("/FirstServlet")*

*public class FirstServlet extends HttpServlet {*

*int count;*

*protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {*

*PrintWriter pw = response.getWriter();*

*//create the HttpSession object*

*HttpSession session = request.getSession(true);*

*if(session.isNew())*

*{*

*pw.println("welcome to new user");*

*pw.println("<br/>");*

*count=0;*

*}*

*else*

*{*

*pw.println("welcome back");*

*count++;*

*pw.println("<br/>");*

*}*

*pw.println("session creation time: " + new Date(session.getCreationTime()));*

*pw.println("<br/>");*

*pw.println("session last accessed time: " + new Date(session.getLastAccessedTime()));*

*pw.println("<br/>");*

*pw.println("session id: " + session.getId());*

*pw.println("<br/>");*

*pw.println("visited count: " + count);*

*if(count>=5)*

*{*

*//destroying the session*

*session.invalidate();*

*}*

*}*

*}*

*JSP( Java Server Pages)*

* *jsp is also a technology given by sun microsystem*
* *implementations are provided by server vendors*
* *tomcat server -> server container*
* *jsp is a tag based programming language*
* *inside the jsp we also can write html tags + jsp tags*
* *non it programmers can easily understand this jsp technology*
* *we no need to restart the server just by resending the request*
* *automatically changes gets affected.*

*servlet container->it is responsible to process all the servlet classes*

*jsp container -> it is responsible to process all the jsp pages*

*ejb container*

*jsf container*

*->interfaces*

*implementations are provided by server vendors*

*how these implementations are available in the server?*

*in the form of jar files*

*for the servlet -> servlet-api.jar*

*for the jsp -> jsp-api.jar*

*purpose of servlet --> to develop dynamic web pages*

*purpose of jsp --> to develop dynamic web pages*

***Drawbacks of servlet:***

* *not recommended to have both business logic + presentation logic in single servlet class*
* *java knowledge is required to write the servlet class for non java programmers understanding the servlet is very much typical who has no knowledge on core java.*
* *everytime restart the server to make changes affected.*

*Types of JSP Tags*

*In jsp technology, we do have three types of tags-*

*1.scritpting tags/elements*

*2.directive tags/elements*

*3.action tags/elements*

***Scripting Tags***

***purpose:*** *to insert the java code in jsp page.*

*scripting tags can be again segarated into three types.*

*1.scriptlet element*

*2.expression element*

*3.declarative element*

*scriptlet element:*

***syntax:***

*<%*

*//java code*

*%>*

*where did you keep our jsp files?*

*under webapp folder*

*what is the extension for our jsp files?*

*<filename>.jsp*

*when we run the jsp file, jsp page is completely translated into*

*equivalent servlet class.*

*out -> JspWriter is the class which is extends from PrintWriter*

*out -> impliticit object of JspWriter*

*9 implicit objects*

*out,request,response,application,page,exception,...*

*example of scriptlet tag:*

*``````````````````````````*

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"%>*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<h1>welcome to jsp technology from sopra using html tag</h1>*

*<br/>*

*<%--scriptlet tag --%>*

*<%*

*out.println("<h1>");*

*out.println("welcome to jsp technology from sopra using jsp scriptlet tag");*

*out.println("</h1>");*

*%>*

*</body>*

*</html>*

*expression tag:*

***syntax:***

*<%=*

*//java code*

*%>*

***example:***

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"%>*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<%--expression tag --%>*

*<h1>*

*<%=*

*"welcome to jsp using expression tag"*

*%>*

*</h1>*

*</body>*

*</html>*

*declarative tag:*

*if we want to declare any variables or to define methods*

*we should use declarative tag.*

*syntax:*

*<%!*

*//declare the variable or*

*// define the method*

*%>*

*example:*

*<%!*

*int a= 100;*

*void m1()*

*{*

*}*

*%>*

*class \_ScriptLet*

*{*

*int a= 100;*

*void m1()*

*{*

*}*

*\_jspService()*

*{*

*}*

*}*

***note:***

* *using scirptlet or expression we can declare the variable but we cannot*

*define the method.*

* *using declarative tag, we can declare the variables and we can define the methods too.*

***example:***

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"%>*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<%!*

*int a=100;*

*int square(int x) {*

*return x\*x;*

*}*

*%>*

*<%*

*out.println("a value is : " +a);*

*int x = square(5);*

*out.println("<br/>");*

*out.println("square value is: " + x);*

*%>*

*</body>*

*</html>*

*scripting tags,directive tags, action tags*

*-scriptlet tags*

*-expression tag*

*-declarative tag*

***Directive Tags***

***purpose:*** *directive tags are telling to the jsp container how jsp page*

*is getting translated into equivalent servlet class.*

*jsp page --> equivalent servlet class (generated by jsp container.*

1. *include directive*
2. *page directive*
3. *taglib directive (custom tags)*

*include directive tag:*

***Purpose:*** *to include the content of any resource(jsp/html..)*

***syntax:***

*<%@ include file="resourcename" %>*

***example:***

*<%@ include file="index.html" %>*

*header.html:*

*```````````*

*footer.html:*

*`````````````*

***advantage of including directive:***

*->reusability*

***index.html***

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<h2>iam header tag</h2>*

*</body>*

*</html>*

***include.jsp:***

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"%>*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<%-- include directive -->*

*<%@include file="header.html" %>*

*</body>*

*</html>*

*page directive:*

***syntax:***

*<%@ page attribute="value" %>*

***List of attributes for page directive:***

* *import.*
* *contentType.*
* *extends.*
* *info.*
* *buffer.*
* *language.*
* *isELIgnored.*
* *isThreadSafe.*
* *isErrorPage*
* *errorPage*

***example*** *on import, contentType,language attributes of page directive:*

*<%@page*

*language="java"*

*import="java.util.Date"*

*contentType="application/msword; charset=ISO-8859-1"*

*%>*

*<body>*

*<%=*

*"Today's date is: "*

*+ new Date()*

*%>*

*</body>*

*</html>*

*exception handling in jsp*

*using (isErrorPage and error Page) of Page directive:*

***index.jsp:***

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"%>*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<form action="process.jsp">*

*<pre>*

*Enter Firstno: <input type="text" name="fno"/><br/>*

*Enter Secondno: <input type="text" name="sno"/><br/>*

*</pre>*

*<input type="submit" value="divide"/>*

*</form>*

*</body>*

*</html>*

***process.jsp:***

*<%@page errorPage="error.jsp" %>*

*<%*

*int fno = Integer.parseInt(request.getParameter("fno"));*

*int sno = Integer.parseInt(request.getParameter("sno"));*

*int result = fno/sno;*

*out.println("division is : " + result);*

*%>*

***error.jsp:***

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"*

*isErrorPage="true"%>*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<h2>sorry an exception occured!</h2>*

*Exception is:*

*<%= exception %>*

*</body>*

*</html>*

*taglib directive(custom tags):*

*``````````````````````````````*

*we can also define our own tags and provide our own functionality to the*

*custom tags which we are going to define.*

*-> taglib directives are used to define our own tags*

***syntax:***

*<%@ taglib prefix="prefixOfTag" uri="uriOfTagLibrary" %>*

*->action tags*

*->expression language in jsp*

*->registration use case*

*->login , logout, profile using cookies*

*Action Tags in Jsp:*

*purpose: each jsp action tag is used to perform some specific task.*

*these are used to control the flow between pages and to use java bean.*

*Types of Jsp action Tag*

*there are many jsp action tags / elements .*

* *jsp:forward*
* *jsp:include*
* *jsp:useBean*
* *jsp:setProperty*
* *jsp:getProperty*
* *jsp:param*

*<jsp:forward page="xyz.jsp"/>*

***jsp:param:***

*jsp:param tag is not independent tag it is depending on jsp:forward*

*while we are forwarding the request from one jsp to any other jsp with parameter*

*then we have to go for jsp:param tag under jsp:forward.*

***Example on jsp:forward and jsp:param:***

***index.jsp:***

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"%>*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<h2>this is index page</h2>*

*<jsp:forward page="displaydate.jsp">*

*<jsp:param name="company" value="sopra"/>*

*</jsp:forward>*

*</body>*

*</html>*

***displaydate.jsp:***

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"*

*import="java.util.Date" %>*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<%=*

*"Today's date is:" + new Date()*

*%>*

*<br/>*

*<%=*

*"parameter value is: " + request.getParameter("company")*

*%>*

*</body>*

*</html>*

*jsp:include:*

*jsp:include action tag is used to include the content of another resource*

*it may be jsp, html or servlet.*

*->the jsp include action tag includes the resource at request time so it is*

*better for dynamic pages because there might be changes in future.*

*jsp:include action tag can be used to include static as well as*

*dynamic pages.*

***index.jsp:***

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"%>*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<h2>this is index page</h2>*

*<jsp:include page="displaydate.jsp"/>*

*<h2>this is ending section of index page</h2>*

*</body>*

*</html>*

***displaydate.jsp:***

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"*

*import="java.util.Date" %>*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<%=*

*"Today's date is:" + new Date()*

*%>*

*</body>*

*</html>*

***Java Bean:***

* *a java bean is a normal java class that should follow some rules.*
* *it should have no-arg constructor*
* *It should be serializable*
* *it should provide methods to set and get the values of the properties*

*known as setters and getters methods.*

*Why use JavaBean?*

*According to Java white paper, it is a reusable software component. A bean encapsulates many objects into one object*

*so that we can access this object from multiple places.*

*Moreover, it provides easy maintenance.*

***example:***

*import java.io.Serializable;*

*class Student implements Serializable*

*{*

*private Address address;*

*private int stid;*

*public Student()*

*{*

*}*

*public void setStid(int stid)*

*{*

*this.stid=stid;*

*}*

*public int getStid()*

*{*

*return stid;*

*}*

*}*

*why should we use JavaBean?*

* *it is a reusable software component.*
* *A bean encapsulates many objects into one object so that we can*

*access this object from multiple palaces.*

***Student.java (java bean class):***

*package com.sopra.beans;*

*import java.io.Serializable;*

*public class Student implements Serializable{*

*private int stid;*

*private String stname;*

*public Student()*

*{*

*}*

*public int getStid() {*

*return stid;*

*}*

*public void setStid(int stid) {*

*this.stid = stid;*

*}*

*public String getStname() {*

*return stname;*

*}*

*public void setStname(String stname) {*

*this.stname = stname;*

*}*

*}*

***without jsp:useBean action element***

***useBean.jsp:***

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"*

*import="com.sopra.beans.Student"%>*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<%*

*Student obj = new Student();*

*obj.setStid(200);*

*obj.setStname("sopra");*

*out.println("Student id: " + obj.getStid());*

*out.println("Student name: " + obj.getStname());*

*%>*

*</body>*

*</html>*

***example using jsp:useBean,jsp:setProperty & jsp:getProperty:***

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"*

*import="com.sopra.beans.Student"%>*

*<body>*

*<jsp:useBean id="obj" class="com.sopra.beans.Student"></jsp:useBean>*

*<jsp:setProperty property="stid" name="obj" value="300"/>*

*<jsp:setProperty property="stname" name="obj" value="ram"/>*

*Records are: <br/>*

*Student id is : <jsp:getProperty property="stid" name="obj"/>*

*<br/>*

*Student name is: <jsp:getProperty property="stname" name="obj"/>*

*</body>*

*</html>*

***how to set dynamic data and get dynamic data using jsp:useBean :***

***index1.jsp:***

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"%>*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<form action="usebean.jsp">*

*Enter Student id: <input type="text" name="stid"/><br/>*

*Enter Student Name: <input type="text" name="stname"/><br/>*

*<input type="submit" value="go"/>*

*</form>*

*</body>*

*</html>*

***usebean.jsp:***

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"*

*import="com.sopra.beans.Student"%>*

*<body>*

*<jsp:useBean id="obj" class="com.sopra.beans.Student"></jsp:useBean>*

*<%-- <jsp:setProperty property="stid" name="obj" />*

*<jsp:setProperty property="stname" name="obj"/>*

*--%>*

*<jsp:setProperty property="\*" name="obj"/>*

*Records are: <br/>*

*Student id is : <jsp:getProperty property="stid" name="obj"/>*

*<br/>*

*Student name is: <jsp:getProperty property="stname" name="obj"/>*

*</body>*

*</html>*

*Registration use case:*

***index.jsp:***

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"%>*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<form action="process.jsp">*

*<pre>*

*Enter user name: <input type="text" name="name"/><br/>*

*Enter user email:<input type="text" name="email"/><br/>*

*Enter user password: <input type="password" name="password"/><br/>*

*<input type="submit" value="register"/>*

*</pre>*

*</form>*

*</body>*

*</html>*

***process.jsp:***

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"*

*import="com.sopra.beans.RegisterDao"%>*

*<body>*

*<jsp:useBean id="obj" class="com.sopra.beans.User"></jsp:useBean>*

*<jsp:setProperty property="\*" name="obj"/>*

*<%*

*int status = RegisterDao.register(obj);*

*if(status>0)*

*{*

*out.println("you are successfully registerd");*

*}*

*%>*

*</body>*

*</html>*

***User.java:***

*package com.sopra.beans;*

*import java.io.Serializable;*

*public class User implements Serializable{*

*private String name;*

*private String email;*

*private String password;*

*public User()*

*{*

*}*

*public String getName() {*

*return name;*

*}*

*public void setName(String name) {*

*this.name = name;*

*}*

*public String getEmail() {*

*return email;*

*}*

*public void setEmail(String email) {*

*this.email = email;*

*}*

*public String getPassword() {*

*return password;*

*}*

*public void setPassword(String password) {*

*this.password = password;*

*}*

*}*

***RegisterDao.java:***

*package com.sopra.beans;*

*import java.sql.Connection;*

*import java.sql.DriverManager;*

*import java.sql.PreparedStatement;*

*public class RegisterDao {*

*static int status=0;*

*public static int register(User u)*

*{*

*try {*

*Class.forName("oracle.jdbc.driver.OracleDriver");*

*Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","sopra","sopra");*

*PreparedStatement ps = con.prepareStatement("insert into user1 values(?,?,?)");*

*ps.setString(1, u.getName());*

*ps.setString(2, u.getEmail());*

*ps.setString(3, u.getPassword());*

*status = ps.executeUpdate();*

*}*

*catch(Exception e)*

*{*

*System.out.println(e);*

*}*

*return status;*

*}*

*}*

*jsp expression language:*

*it simplifies the accessibility of data stored in the java bean component.*

*and other objects like request,session,application etc..*

***syntax:***

*${expression}*

***index1.jsp:***

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"%>*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<form action="usebean.jsp">*

*Enter Student id: <input type="text" name="stid"/><br/>*

*Enter Student Name: <input type="text" name="stname"/><br/>*

*<input type="submit" value="go"/>*

*</form>*

*</body>*

*</html>*

***useBean.jsp:***

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"*

*import="com.sopra.beans.Student"*

*isELIgnored="false"%>*

*<body>*

*<jsp:useBean id="obj" class="com.sopra.beans.Student"></jsp:useBean>*

*<%-- <jsp:setProperty property="stid" name="obj" />*

*<jsp:setProperty property="stname" name="obj"/>*

*--%>*

*<jsp:setProperty property="\*" name="obj"/>*

*Records are: <br/>*

*<%-- Student id is : <jsp:getProperty property="stid" name="obj"/>*

*<br/>*

*Student name is: <jsp:getProperty property="stname" name="obj"/> --%>*

*<!--using jsp expression language --%>*

*Student id is: ${obj.stid} <br/>*

*Student name is: ${obj.stname}*

*</body>*

*</html>*

*Spring:*

* *it is a framework*
* *it means it contains both interfaces and implementation classes.*
* *2002 - Rod johnson - spring was released by him*
* *it is open source*
* ***purpose:*** *using this framework, what kind of applications we can develop?*

*->desktop applications*

*->web applications*

*What is the main advantage of spring f/w?*

* *we can achieve one important concept i.e. dependency injection*
* *using dependency injection, we can achieve loose coupling in between*

*classes.*

*tight coupling vs loose coupling:*

*class Car*

*{*

*Engine engine;*

*public Car(Engine engine)*

*{*

*//engine = new Engine("new engine");*

*}*

*}*

*class Engine{*

*public Engine("new new engine")*

*{*

*}*

*}*

* *maintainence of the program will be decreased*
* *From above program, both Car class and Engine classes are tightly coupled*

*with each other.*

* *To make it loosely coupled, we have to apply one design pattern*
* *In spring framework, we are applying inversion of control design pattern*

*to achieve that design pattern the concept name is 'dependency injection'.*

* *pivotal organization owned this spring in 2013 year.*

*spring f/w contains differnt modules*

* *spring core module*
* *spring jdbc module*
* *spring web mvc module*
* *spring orm module*
* *spring test module*
* *spring context module*

*steps to write a spring program using spring core module:*

*1.create a java project*

*2.create one package under src folder (com.sopra.bean)*

*3.create one bean class*

*4.under src we have to create spring configuration file to configure*

*beans.*

*5.add spring core jars to the buildpath of the application*

*6.create Test class and invoke business logic methods.*

***why should we configure java beans in spring configuration file?***

*because to create the object for java bean classes.*

***what is spring container in spring f/w?***

*spring container is responsible for creating java bean objects*

*managing , destrcution*

*spring container is the interface availabe in one package*

*-BeanFactory*

*-ApplicationContext*

***To work on spring core module what jar files we required?***

*commons logging*

*spring beans*

*spring context*

*spring expression*

*spring core*

***example:***

***Student.java:***

*package com.sopra.bean;*

*public class Student {*

*public void display()*

*{*

*System.out.println("iam from display");*

*}*

*}*

***spingcontext.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.xsd">*

*<bean id="stu" class="com.sopra.bean.Student"/>*

*</beans>*

***Test.java:***

*package com.sopra.bean;*

*import org.springframework.context.ApplicationContext;*

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

*public class Test {*

*public static void main(String[] args) {*

*//Student obj = new Student();*

*//obj.display();*

*ApplicationContext context = new ClassPathXmlApplicationContext("springcontext.xml");*

*Student stu = (Student)context.getBean("stu");*

*stu.display();*

*}*

*}*

***core java link:***

[*https://youtu.be/bsABDNNA9L0*](https://youtu.be/bsABDNNA9L0)

*Dependency injection can be achieved in spring in two ways*

*1.setter-based injection*

*2.constructor-based injection*

***setter injection:***

*Injecting primitive and string-based values by setter method*

***Student.java:***

*package com.sopra.beans;*

*import java.io.Serializable;*

*public class Student implements Serializable{*

*private int stid;*

*private String stname;*

*public int getStid() {*

*return stid;*

*}*

*public void setStid(int stid) {*

*this.stid = stid;*

*}*

*public String getStname() {*

*return stname;*

*}*

*public void setStname(String stname) {*

*this.stname = stname;*

*}*

*public Student()*

*{*

*}*

*@Override*

*public String toString() {*

*return "Student [stid=" + stid + ", stname=" + stname + "]";*

*}*

*}*

*springcontext.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.xsd">*

*<bean id="stu" class="com.sopra.beans.Student">*

*<property name="stid">*

*<value>300</value>*

*</property>*

*<property name="stname" value="Sopra"/>*

*</bean>*

*</beans>*

*Test.java:*

*```````````*

*package com.sopra.beans;*

*import org.springframework.context.ApplicationContext;*

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

*public class Test {*

*public static void main(String[] args) {*

*ApplicationContext context = new ClassPathXmlApplicationContext("springcontext.xml");*

*Student stu = (Student)context.getBean("stu");*

*/\**

*\* System.out.println("student id: " + stu.getStid());*

*\* System.out.println("student name: " + stu.getStname());*

*\*/*

*System.out.println(stu);*

*}*

*}*

*constructor based injection:*

*````````````````````````````*

*if we inject the values by calling the corresponding constructor of the*

*bean is called constructor injection.*

*springcontext.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.xsd">*

*<bean id="stu" class="com.sopra.beans.Student">*

*<constructor-arg name="stid" value="300"></constructor-arg>*

*<constructor-arg name="stname" value="ram"></constructor-arg>*

*</bean>*

*</beans>*

*Student.java*

*`````````````*

*package com.sopra.beans;*

*import java.io.Serializable;*

*public class Student implements Serializable{*

*private int stid;*

*private String stname;*

*public Student()*

*{*

*}*

*public Student(int stid, String stname) {*

*super();*

*this.stid = stid;*

*this.stname = stname;*

*}*

*@Override*

*public String toString() {*

*return "Student [stid=" + stid + ", stname=" + stname + "]";*

*}*

*}*

*note:*

*`````*

*when we apply both setter injection and construtor injection , spring container*

*will take setter injection values.*

*how to inject reference type values:*

*````````````````````````````````````*

*Student.java:*

*`````````````*

*package com.sopra.beans;*

*public class Student {*

*private Address address;*

*public Address getAddress() {*

*return address;*

*}*

*public void setAddress(Address address) {*

*this.address = address;*

*}*

*public Student()*

*{*

*}*

*@Override*

*public String toString() {*

*return "Student [address=" + address + "]";*

*}*

*}*

*Address.java:*

*`````````````*

*package com.sopra.beans;*

*public class Address {*

*private String cityName;*

*private String stateName;*

*private String countryName;*

*public String getCityName() {*

*return cityName;*

*}*

*public void setCityName(String cityName) {*

*this.cityName = cityName;*

*}*

*public String getStateName() {*

*return stateName;*

*}*

*public void setStateName(String stateName) {*

*this.stateName = stateName;*

*}*

*public String getCountryName() {*

*return countryName;*

*}*

*public void setCountryName(String countryName) {*

*this.countryName = countryName;*

*}*

*public Address()*

*{*

*}*

*@Override*

*public String toString() {*

*return "Address [cityName=" + cityName + ", stateName=" + stateName + ", countryName=" + countryName + "]";*

*}*

*}*

*springcontext.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.xsd">*

*<bean id="stu" class="com.sopra.beans.Student">*

*<property name="address">*

*<ref bean="add"/>*

*</property>*

*</bean>*

*<bean id="add" class="com.sopra.beans.Address">*

*<property name="cityName" value="Hyd"/>*

*<property name="stateName" value="Tel"/>*

*<property name="countryName" value="India"/>*

*</bean>*

*</beans>*

*Test.java:*

*```````````*

*package com.sopra.beans;*

*import org.springframework.context.ApplicationContext;*

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

*public class Test {*

*public static void main(String[] args) {*

*ApplicationContext context = new ClassPathXmlApplicationContext("springcontext.xml");*

*Student stu = (Student)context.getBean("stu");*

*System.out.println("city name is : " + stu.getAddress().getCityName());*

*System.out.println("state name is: " + stu.getAddress().getStateName());*

*System.out.println("country name is: " + stu.getAddress().getCountryName());*

*}*

*}*

*how to inject collection type values (list,set,map):*

*````````````````````````````````````````````````````*

*Student.java:*

*``````````````*

*package com.sopra.beans;*

*import java.util.List;*

*public class Student {*

*private int stid;*

*private String stname;*

*private List<String> subjects;*

*public int getStid() {*

*return stid;*

*}*

*public void setStid(int stid) {*

*this.stid = stid;*

*}*

*public String getStname() {*

*return stname;*

*}*

*public void setStname(String stname) {*

*this.stname = stname;*

*}*

*public List<String> getSubjects() {*

*return subjects;*

*}*

*public void setSubjects(List<String> subjects) {*

*this.subjects = subjects;*

*}*

*@Override*

*public String toString() {*

*return "Student [stid=" + stid + ", stname=" + stname + ", subjects=" + subjects + "]";*

*}*

*public Student()*

*{*

*}*

*}*

*springcontext.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.xsd">*

*<bean id="stu" class="com.sopra.beans.Student">*

*<property name="stid" value="100"/>*

*<property name="stname" value="Ram"/>*

*<property name="subjects">*

*<list>*

*<value>english</value>*

*<value>science</value>*

*<value>Hindhi</value>*

*</list>*

*</property>*

*</bean>*

*</beans>*

*Test.java:*

*````````````*

*package com.sopra.beans;*

*import org.springframework.context.ApplicationContext;*

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

*public class Test {*

*public static void main(String[] args) {*

*ApplicationContext context = new ClassPathXmlApplicationContext("springcontext.xml");*

*Student stu = (Student)context.getBean("stu");*

*System.out.println(stu);*

*}*

*}*

*assignments:*

*```````````````*

*inject collection type values (set & map)*

*-how to inject reference type values*

*-how to inject collection type values (list)*

*autowiring:*

*```````````*

*Autowiring feature of spring framework enables you to inject the object*

*dependency(reference) implicitly.*

*It internally uses setter or constructor injection*

*note:*

*"""""*

*if we want to achieve autowiring we have to use autowire attribute for a*

*bean element.*

*possible values:*

*-byName*

*-byType*

*-Constructor*

*if autowire=byName*

*springcontext.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.xsd">*

*<bean id="stu" class="com.sopra.beans.Student" autowire="byName"/>*

*<bean id="address" class="com.sopra.beans.Address">*

*<property name="cityName" value="Hyd"/>*

*<property name="stateName" value="Tel"/>*

*<property name="countryName" value="India"/>*

*</bean>*

*</beans>*

*Student.java:*

*``````````````*

*package com.sopra.beans;*

*public class Student {*

*private Address address;*

*public Address getAddress() {*

*return address;*

*}*

*public void setAddress(Address address) {*

*this.address = address;*

*}*

*public Student()*

*{*

*}*

*@Override*

*public String toString() {*

*return "Student [address=" + address + "]";*

*}*

*}*

*note:*

*`````*

*when we are using either autowire attribute values are byName or byType*

*internally setter injection will happen.*

*if we use autowire attribute value as constructor internally it uses byType*

*only but constructor will injection happen.*

*springcontext.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.xsd">*

*<bean id="stu" class="com.sopra.beans.Student" autowire="constructor"/>*

*<bean class="com.sopra.beans.Address">*

*<property name="cityName" value="Hyd"/>*

*<property name="stateName" value="Tel"/>*

*<property name="countryName" value="India"/>*

*</bean>*

*</beans>*

*annotation based wiring:*

*`````````````````````````*

*So instead of using XML to describe a bean wiring,*

*you can move the bean configuration into the component class*

*itself by using annotations on the relevant class, method, or field declaration.*

*in case of annotation based wiring autowire attribute is not required*

*for a bean element.*

*we have to add @autowired annotation in the java bean class.*

*we no need to maintain any setter,or constructor for a bean class.*

*note:*

*````*

*by default spring container could not recognize these annotations , to recognize*

*we have to add one xml element in spring configuration file*

*<context:annotation-config/>*

*note:*

*````*

*to make it execute this , we have to add one more jar spring-aop*

*Student.java:*

*``````````````*

*package com.sopra.beans;*

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

*public class Student {*

*@Autowired*

*private Address address;*

*public Student()*

*{*

*}*

*@Override*

*public String toString() {*

*return "Student [address=" + address + "]";*

*}*

*}*

*springcontext.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:context="http://www.springframework.org/schema/context"*

*xsi:schemaLocation="*

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

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd">*

*<context:annotation-config></context:annotation-config>*

*<bean id="stu" class="com.sopra.beans.Student"/> <!--Bean Tag used to create object of class-->*

*<bean class="com.sopra.beans.Address">*

*<property name="cityName" value="Hyd"/>*

*<property name="stateName" value="Tel"/>*

*<property name="countryName" value="India"/>*

*</bean>*

*</beans>*

*Test.java:*

*``````````*

*package com.sopra.beans;*

*import org.springframework.context.ApplicationContext;*

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

*public class Test {*

*public static void main(String[] args) {*

*ApplicationContext context = new ClassPathXmlApplicationContext("springcontext.xml");*

*Student stu = (Student)context.getBean("stu");*

*System.out.println(stu);*

*}*

*}*

*autodisovery:*

*``````````````*

*Autodiscovery feature frees us from writing all bean declarations*

*and bean autowirings.*

*Note:*

*`````*

*this feature can be enabled by adding one xml element i.e.*

*<context:component-scan base-package=""/>*

*@Component annotation:*

*``````````````````````*

*@Component is an annotation that allows Spring to automatically detect*

*our custom beans. In other words, without having to write any explicit code,*

*Spring will: Scan our application for classes annotated with @Component.*

*Instantiate them and inject any specified dependencies into them.*

*Inject them wherever needed.*

*Student.java:*

*`````````````*

*package com.sopra.beans;*

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

*import org.springframework.stereotype.Component;*

*@Component*

*public class Student {*

*@Autowired*

*private Address address;*

*public Student()*

*{*

*}*

*@Override*

*public String toString() {*

*return "Student [address=" + address + "]";*

*}*

*}*

*springcontext.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:context="http://www.springframework.org/schema/context"*

*xsi:schemaLocation="*

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

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd">*

*<context:component-scan base-package="com.sopra.beans"></context:component-scan>*

*<bean class="com.sopra.beans.Address">*

*<property name="cityName" value="Hyd"/>*

*<property name="stateName" value="Tel"/>*

*<property name="countryName" value="India"/>*

*</bean>*

*</beans>*

*Test.java:*

*```````````*

*package com.sopra.beans;*

*import org.springframework.context.ApplicationContext;*

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

*public class Test {*

*public static void main(String[] args) {*

*ApplicationContext context = new ClassPathXmlApplicationContext("springcontext.xml");*

*Student stu = (Student)context.getBean("student");*

*System.out.println(stu);*

*}*

*}*

*-autowiring (byName,byType,constructor)*

*-annotation based wiring(@autowired)*

*-autodiscovery*

*maven:*

*``````*

*maven is a automation build tool*

*build:*

*compile the source code*

*execute the test cases*

*package the entire project as a jar or war*

*fetching the dependencies*

*maven/gradle*

*pom file*

*-> project object model*

*@Qualifier annotation*

*````````````````````*

*The @Qualifier annotation is used to resolve the autowiring conflict,*

*when there are multiple beans of same type.*

*College.java:*

*`````````````*

*package com.sopra.beans;*

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

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

*public class College {*

*@Autowired*

*@Qualifier("stu2")*

*private Student student;*

*@Override*

*public String toString() {*

*return "College [student=" + student + "]";*

*}*

*public College()*

*{*

*}*

*}*

*Student.java:*

*`````````````*

*package com.sopra.beans;*

*public class Student {*

*private int stid;*

*private String stname;*

*public int getStid() {*

*return stid;*

*}*

*public void setStid(int stid) {*

*this.stid = stid;*

*}*

*public String getStname() {*

*return stname;*

*}*

*public void setStname(String stname) {*

*this.stname = stname;*

*}*

*public Student()*

*{*

*}*

*@Override*

*public String toString() {*

*return "Student [stid=" + stid + ", stname=" + stname + "]";*

*}*

*}*

*Test.java:*

*```````````*

*package com.sopra.beans;*

*import org.springframework.context.ApplicationContext;*

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

*public class Test {*

*public static void main(String[] args) {*

*ApplicationContext context = new ClassPathXmlApplicationContext("springcontext.xml");*

*College col = (College)context.getBean("college");*

*System.out.println(col);*

*}*

*}*

*springcontext.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:context="http://www.springframework.org/schema/context"*

*xsi:schemaLocation="*

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

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd">*

*<context:annotation-config></context:annotation-config>*

*<bean id="college" class="com.sopra.beans.College">*

*</bean>*

*<bean id="stu1" class="com.sopra.beans.Student">*

*<property name="stid" value="100"/>*

*<property name="stname" value="ram"/>*

*</bean>*

*<bean id="stu2" class="com.sopra.beans.Student">*

*<property name="stid" value="200"/>*

*<property name="stname" value="nandini"/>*

*</bean>*

*</beans>*

*refer spring aop:*

*`````````````````*

*spring aop:*

*```````````*

*aop --> aspect oriented programming*

*oop -> object oriented programming*

*to separate the main logic with secondary logic*

*->atm (withdrawl the money ) -> main logic*

*secondary logic -> sms ->secondary logic*

*authentication --> secondary logics*

*->maintainence of the application will get improved*

*authentication*

*before main logic*

*sms service*

*aspect,joinpoint,advice,weaving,proxy.....*

*spring aop module:*

*```````````````````*

*aop -> aspect oriented programming*

*oop -> object oriented programming*

*atm machine -> withdrawing/checking the balance --> primary logics*

*secondary logics -> pin/sms --> secondary logics/cross cutting concerns*

*AOP breaks the program logic into distinct parts(called concerns).*

*It is used to increase modularity by cross cutting concerns.*

*A cross cutting concern is a concern that can affect the whole application*

*and should be centralized in one location*

*ex:*

*```*

*transaction management,authentication,logging,security.etc..*

*class Test{*

*public void m1();{//logging}*

*public void m2();*

*public void m3();*

*public void m4();*

*public void m5();*

*public void n1();*

*public void n2();*

*public void p1();*

*public void p2();*

*public void p3();*

*}*

*aop -> aspect oriented programing*

*implementation*

*->aspectj*

*->spring aop*

*->jboss aop*

*//busines logic*

*public void withdraw()*

*{*

*}*

*aspect -> @Aspect declares the class as aspect*

*pointcut ->declares the pointcut expression*

*authentication serivce*

*@Before- before advice, it is applied before calling the actual method*

*@After*

*@AfterReturning*

*@Around*

*@AfterThrowing*

*advice*

*joinpoint-it is saying which advice is associated with which actual method.*

*weaving*

*weaver*

*proxy*

*...*

*pointcut:*

*`````````*

*pointcut is an expression language of spring AOP module.*

*@Pointcut("execution(\* Operation.\*(..)")*

*public void doSomething(){*

*//cross cutting conern logic/secondary logic*

*}*

*@Pointcut("execution(public \* \*(..)")*

*@Pointcut("execution(public Operation \*(..)")*

*@Pointcut("execution(public Employee.set\*(..)")*

*@Pointcut("execution(int Operation.\*(..)")*

*example on aop :*

*````````````````*

*Operation.java(interface):*

*``````````````````````````````*

*package com.sopra;*

*public interface Operation {*

*void m1();*

*int n1();*

*void p1();*

*}*

*OperationImpl.java:*

*````````````````````*

*package com.sopra;*

*public class OperationImpl implements Operation {*

*public void m1()*

*{*

*System.out.println("iam from m1 method ");*

*}*

*public int n1()*

*{*

*System.out.println("iam from n1 method");*

*return 3;*

*}*

*public void p1()*

*{*

*System.out.println("iam from p1 method");*

*}*

*}*

*TrackOperation.java:*

*````````````````````*

*package com.sopra;*

*import org.aspectj.lang.JoinPoint;*

*import org.aspectj.lang.annotation.After;*

*import org.aspectj.lang.annotation.Around;*

*import org.aspectj.lang.annotation.Aspect;*

*import org.aspectj.lang.annotation.Before;*

*@Aspect*

*public class TrackOperation {*

*@After("execution(\* com.sopra.\*.\*())")*

*public void myadvice(JoinPoint jp)//it is advice*

*{*

*System.out.println("additional concern");*

*}*

*}*

*App.java:*

*``````````*

*package com.sopra;*

*import org.springframework.context.ApplicationContext;*

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

*/\*\**

*\* Hello world!*

*\**

*\*/*

*public class App*

*{*

*public static void main( String[] args )*

*{*

*ApplicationContext context = new ClassPathXmlApplicationContext("config.xml");*

*Operation op = (Operation)context.getBean("op");*

*op.m1();*

*op.n1();*

*op.p1();*

*}*

*}*

*config.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:context="http://www.springframework.org/schema/context"*

*xmlns:aop="http://www.springframework.org/schema/aop"*

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

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

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

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

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

*http://www.springframework.org/schema/aop/spring-aop.xsd">*

*<aop:aspectj-autoproxy></aop:aspectj-autoproxy>*

*<bean id="op" class="com.sopra.OperationImpl" />*

*<bean id="to" class="com.sopra.TrackOperation" />*

*</beans>*

*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>com.cts</groupId>*

*<artifactId>spring-demo-8</artifactId>*

*<version>0.0.1-SNAPSHOT</version>*

*<packaging>jar</packaging>*

*<name>spring-demo-8</name>*

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

*<properties>*

*<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>*

*<maven.compiler.source>1.8</maven.compiler.source>*

*<maven.compiler.target>1.8</maven.compiler.target>*

*</properties>*

*<dependencies>*

*<dependency>*

*<groupId>junit</groupId>*

*<artifactId>junit</artifactId>*

*<version>3.8.1</version>*

*<scope>test</scope>*

*</dependency>*

*<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->*

*<dependency>*

*<groupId>org.springframework</groupId>*

*<artifactId>spring-context</artifactId>*

*<version>5.0.5.RELEASE</version>*

*</dependency>*

*<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjrt -->*

*<dependency>*

*<groupId>org.aspectj</groupId>*

*<artifactId>aspectjrt</artifactId>*

*<version>1.9.1</version>*

*</dependency>*

*<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjweaver -->*

*<dependency>*

*<groupId>org.aspectj</groupId>*

*<artifactId>aspectjweaver</artifactId>*

*<version>1.9.1</version>*

*</dependency>*

*<!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->*

*<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml -->*

*<dependency>*

*<groupId>org.apache.poi</groupId>*

*<artifactId>poi-ooxml</artifactId>*

*<version>3.9</version>*

*</dependency>*

*</dependencies>*

*</project>*

*java based configuration in spring*

*Java-based configuration option enables you to write most of your Spring*

*configuration without XML but with the help of few Java-based annotations*

*@Configuration*

*the class which is annotated with @Configuration it represents it contains*

*configurable information.*

*Annotating a class with the @Configuration indicates that the class can be*

*used by the Spring IoC container as a source of bean definitions.*

*The @Bean annotation tells Spring that a method annotated with @Bean will*

*return an object that should be registered as a bean in the Spring*

*application context.*

*Student.java:*

*`````````````*

*package com.sopra;*

*public class Student {*

*private int stid;*

*private String stname;*

*public int getStid() {*

*return stid;*

*}*

*public void setStid(int stid) {*

*this.stid = stid;*

*}*

*public String getStname() {*

*return stname;*

*}*

*public void setStname(String stname) {*

*this.stname = stname;*

*}*

*public Student()*

*{*

*}*

*}*

*MyConfig.java:*

*``````````````*

*package com.sopra;*

*import org.springframework.context.annotation.Bean;*

*import org.springframework.context.annotation.Configuration;*

*@Configuration*

*public class MyConfig {*

*@Bean("stu")*

*public Student getStudent()*

*{*

*return new Student();*

*}*

*}*

*App.java:*

*``````````*

*package com.sopra;*

*import org.springframework.context.ApplicationContext;*

*import org.springframework.context.annotation.AnnotationConfigApplicationContext;*

*/\*\**

*\* Hello world!*

*\**

*\*/*

*public class App*

*{*

*public static void main( String[] args )*

*{*

*ApplicationContext context = new AnnotationConfigApplicationContext(MyConfig.class);*

*Student stu = (Student)context.getBean("stu");*

*stu.setStid(100);*

*stu.setStname("ram");*

*System.out.println("student id: " + stu.getStid());*

*System.out.println("student name: " + stu.getStname());*

*}*

*}*

*spring jdbc module*

*purpose:*

*to implement persistence/database logic*

*jdbc:*

*load,connection,statement/prepareStatement,process the query,close the connection*

*JdbcTemplate*

*dao class*

*|*

*JdbcTemplate*

*|*

*DataSource(Interface)*

*|*

*DriverManagerDataSource*

*-driverClassName*

*-url*

*-username*

*-password*

*springcontext.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:context="http://www.springframework.org/schema/context"*

*xmlns:aop="http://www.springframework.org/schema/aop"*

*xsi:schemaLocation="*

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

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

*http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd">*

*<context:component-scan base-package="com.sopra"></context:component-scan>*

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

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

*</bean>*

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

*<property name="driverClassName" value="oracle.jdbc.driver.OracleDriver"/>*

*<property name="url" value="jdbc:oracle:thin:@localhost:1521:xe"/>*

*<property name="username" value="sopra"/>*

*<property name="password" value="sopra"/>*

*</bean>*

*</beans>*

*Employee.java:*

*````````````````*

*package com.sopra.beans;*

*public class Employee {*

*private int empid;*

*private String firstName;*

*private String lastName;*

*public int getEmpid() {*

*return empid;*

*}*

*public void setEmpid(int empid) {*

*this.empid = empid;*

*}*

*public String getFirstName() {*

*return firstName;*

*}*

*public void setFirstName(String firstName) {*

*this.firstName = firstName;*

*}*

*public String getLastName() {*

*return lastName;*

*}*

*public void setLastName(String lastName) {*

*this.lastName = lastName;*

*}*

*public Employee(int empid, String firstName, String lastName) {*

*super();*

*this.empid = empid;*

*this.firstName = firstName;*

*this.lastName = lastName;*

*}*

*public Employee() {*

*super();*

*}*

*@Override*

*public String toString() {*

*return "Employee [empid=" + empid + ", firstName=" + firstName + ", lastName=" + lastName + "]";*

*}*

*}*

*EmployeeDao.java:*

*`````````````````*

*package com.sopra.dao;*

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

*import org.springframework.jdbc.core.JdbcTemplate;*

*import org.springframework.stereotype.Component;*

*import com.sopra.beans.Employee;*

*@Component("dao")*

*public class EmployeeDao {*

*@Autowired*

*private JdbcTemplate jdbcTemplate;*

*public int saveEmployee(Employee e)*

*{*

*String sql = "insert into employee values(?,?,?)";*

*return jdbcTemplate.update(sql, e.getEmpid(),e.getFirstName(),e.getLastName());*

*}*

*//update,delete,select query,list of rows*

*}*

*App.java:*

*`````````*

*package com.sopra.beans;*

*import org.springframework.context.ApplicationContext;*

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

*import com.sopra.dao.EmployeeDao;*

*/\*\**

*\* Hello world!*

*\**

*\*/*

*public class App*

*{*

*public static void main( String[] args )*

*{*

*ApplicationContext context = new ClassPathXmlApplicationContext("springcontext.xml");*

*EmployeeDao dao = (EmployeeDao)context.getBean("dao");*

*Employee emp = new Employee();*

*emp.setEmpid(101);*

*emp.setFirstName("nandini");*

*emp.setLastName("jain");*

*int x = dao.saveEmployee(emp);*

*System.out.println(x + "row(s) inserted");*

*}*

*}*

*Assignment*

*-select single row*

*-select all row*

*springcontext.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:context="http://www.springframework.org/schema/context"*

*xmlns:aop="http://www.springframework.org/schema/aop"*

*xsi:schemaLocation="*

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

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

*http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd">*

*<context:component-scan base-package="com.sopra"></context:component-scan>*

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

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

*</bean>*

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

*<property name="driverClassName" value="oracle.jdbc.driver.OracleDriver"/>*

*<property name="url" value="jdbc:oracle:thin:@localhost:1521:xe"/>*

*<property name="username" value="sopra"/>*

*<property name="password" value="sopra"/>*

*</bean>*

*</beans>*

*Employee.java:*

*````````````````*

*package com.sopra.beans;*

*public class Employee {*

*private int empid;*

*private String firstName;*

*private String lastName;*

*public int getEmpid() {*

*return empid;*

*}*

*public void setEmpid(int empid) {*

*this.empid = empid;*

*}*

*public String getFirstName() {*

*return firstName;*

*}*

*public void setFirstName(String firstName) {*

*this.firstName = firstName;*

*}*

*public String getLastName() {*

*return lastName;*

*}*

*public void setLastName(String lastName) {*

*this.lastName = lastName;*

*}*

*public Employee(int empid, String firstName, String lastName) {*

*super();*

*this.empid = empid;*

*this.firstName = firstName;*

*this.lastName = lastName;*

*}*

*public Employee() {*

*super();*

*}*

*@Override*

*public String toString() {*

*return "Employee [empid=" + empid + ", firstName=" + firstName + ", lastName=" + lastName + "]";*

*}*

*}*

*EmployeeDao.java:*

*`````````````````*

*package com.sopra.dao;*

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

*import org.springframework.jdbc.core.JdbcTemplate;*

*import org.springframework.stereotype.Component;*

*import com.sopra.beans.Employee;*

*@Component("dao")*

*public class EmployeeDao {*

*@Autowired*

*private JdbcTemplate jdbcTemplate;*

*public int saveEmployee(Employee e)*

*{*

*String sql = "insert into employee values(?,?,?)";*

*return jdbcTemplate.update(sql, e.getEmpid(),e.getFirstName(),e.getLastName());*

*}*

*//update,delete,select query,list of rows*

*public Employee read(int id)*

*{*

*String sql = "select \* from employee where id=?";*

*EmployeeRowMapper rowmapper = new EmployeeRowMapper();*

*return jdbcTemplate.queryForObject(sql, rowmapper,id);*

*}*

*public List<Employee> findAll()*

*{*

*String sql = "select \* From employee";*

*EmployeeRowMapper rowmapper = new EmployeeRowMapper();*

*return jdbcTemplate.query(sql, rowmapper);*

*}*

*}*

*App.java:*

*`````````*

*package com.sopra.beans;*

*import java.util.List;*

*import org.springframework.context.ApplicationContext;*

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

*import com.sopra.dao.EmployeeDao;*

*/\*\**

*\* Hello world!*

*\**

*\*/*

*public class App*

*{*

*public static void main( String[] args )*

*{*

*ApplicationContext context = new ClassPathXmlApplicationContext("springcontext.xml");*

*EmployeeDao dao = (EmployeeDao)context.getBean("dao");*

*/\*Employee emp = new Employee();*

*emp.setEmpid(101);*

*emp.setFirstName("nandini");*

*emp.setLastName("jain");*

*int x = dao.saveEmployee(emp);*

*System.out.println(x + "row(s) inserted");\*/*

*/\* Employee emp = dao.read(101);*

*System.out.println("employee id is: " + emp.getEmpid());*

*System.out.println("employee first name is: " + emp.getFirstName());*

*System.out.println("employee last name is: " + emp.getLastName());*

*\*/*

*List<Employee> emps = dao.findAll();*

*for(Employee e: emps)*

*{*

*System.out.println(e.getEmpid() + " " + e.getFirstName() + " " + e.getLastName());*

*}*

*}*

*}*

*EmployeeRowMapper.java:*

*````````````````````````*

*package com.sopra.dao;*

*import java.sql.ResultSet;*

*import java.sql.SQLException;*

*import org.springframework.jdbc.core.RowMapper;*

*import com.sopra.beans.Employee;*

*public class EmployeeRowMapper implements RowMapper<Employee> {*

*public Employee mapRow(ResultSet rs, int rowNum) throws SQLException {*

*Employee emp = new Employee();*

*emp.setEmpid(rs.getInt(1));*

*emp.setFirstName(rs.getString(2));*

*emp.setLastName(rs.getString(3));*

*return emp;*

*}*

*}*

*spring web mvc module:*

*using this module, we are able to develop web applications.*

*flow:*

*`````*

*->explained daigram*

*1.create maven project by chooseing web-app archetype*

*2.right click on the project -> go to properties -> target runtime -> add server*

*3.open pom.xml and add spring web mvc dependecy*

*4.configure the DispatcherServlet class in web.xml file*

*5.create controller class under src/main/java under one package(com.sopra.controller)*

*make sure controller class must be annotated with @Controller annotation.*

*6.we configure view resolver class in spring configuration file*

*(InternalResourceViewResolver),*

*note: spring configuration file must be followed one naming conventions*

*<servlet-name>-servlet.xml*

*ex: dispatcher-servlet.xml*

*7.place the view page inside one location(WEB-INF/views/display.jsp)*

*8.run the application on top of server and pass the correct url so that*

*our view page will be rendered.*

*HelloController.java:*

*`````````````````````*

*package com.sopra.controller;*

*import org.springframework.stereotype.Controller;*

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

*import org.springframework.web.servlet.ModelAndView;*

*@Controller*

*public class HelloController {*

*@RequestMapping("/hello")*

*public ModelAndView hello()*

*{*

*ModelAndView mv = new ModelAndView();*

*//setting the view name*

*mv.setViewName("hello");*

*return mv;*

*}*

*}*

*web.xml:*

*````````*

*<!DOCTYPE web-app PUBLIC*

*"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"*

*"http://java.sun.com/dtd/web-app\_2\_3.dtd" >*

*<web-app>*

*<servlet>*

*<servlet-name>spring</servlet-name>*

*<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>*

*</servlet>*

*<servlet-mapping>*

*<servlet-name>spring</servlet-name>*

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

*</servlet-mapping>*

*</web-app>*

*spring-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:context="http://www.springframework.org/schema/context"*

*xmlns:aop="http://www.springframework.org/schema/aop"*

*xsi:schemaLocation="*

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

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

*http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd">*

*<context:component-scan base-package="com.sopra"></context:component-scan>*

*<bean id="view" class="org.springframework.web.servlet.view.InternalResourceViewResolver">*

*<property name="prefix">*

*<value>/WEB-INF/jsp/</value>*

*</property>*

*<property name="suffix">*

*<value>.jsp</value>*

*</property>*

*</bean>*

*</beans>*

*hello.jsp:*

*`````````*

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"%>*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<h1>welcome to spring web mvc</h1>*

*</body>*

*</html>*

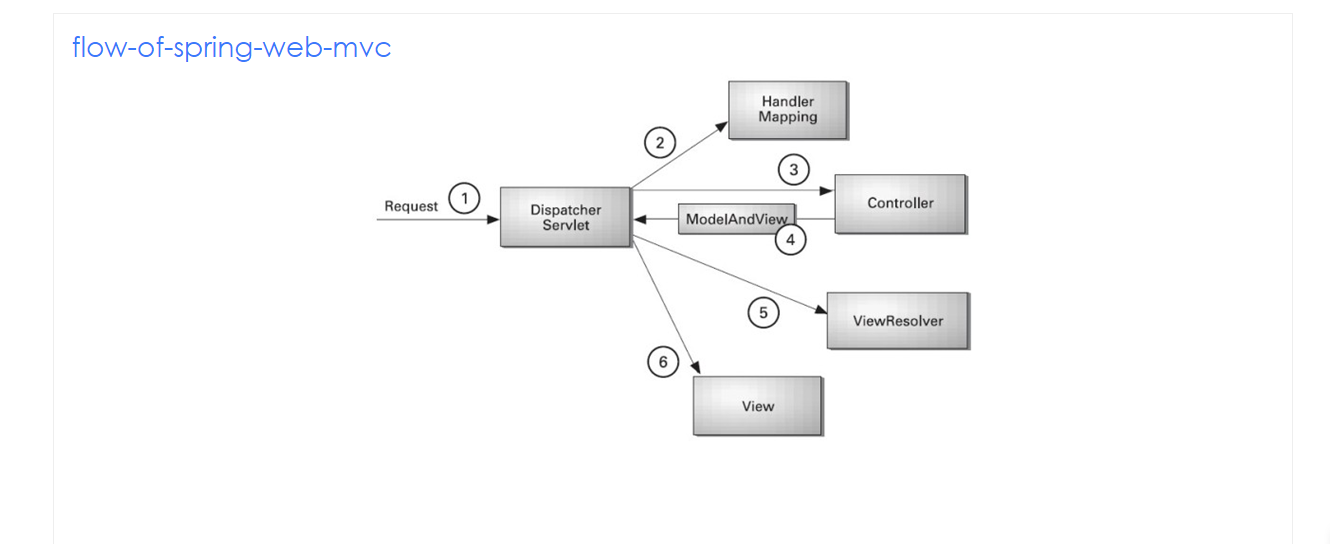
*note:*

*`````*

*don't forget to add the server to the spring web application.*

*->right click on project -> click on properties -> add target runtime*

*->add the server.*



*how to add the data to ModelAndView object:*

*```````````````````````````````````````````*

*HelloController.java:*

*````````````````````*

*package com.sopra.controller;*

*import org.springframework.stereotype.Controller;*

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

*import org.springframework.web.servlet.ModelAndView;*

*@Controller*

*public class HelloController {*

*@RequestMapping("/hello")*

*public ModelAndView hello()*

*{*

*ModelAndView mv = new ModelAndView();*

*mv.addObject("stid",100);*

*mv.addObject("stname","ram");*

*mv.addObject("staddress","hyd");*

*//setting the view name*

*mv.setViewName("hello");*

*return mv;*

*}*

*}*

*hello.jsp:*

*``````````*

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"*

*isELIgnored="false"%>*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<h1>welcome to spring web mvc</h1><br/>*

*Student id: ${stid}<br/>*

*Student name: ${stname} <br/>*

*Student address: ${staddress}*

*</body>*

*</html>*

*how to add the java object to ModelAndView and make it displayed in view page:*

***HelloController.java****:*

*package com.sopra.controller;*

*import org.springframework.stereotype.Controller;*

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

*import org.springframework.web.servlet.ModelAndView;*

*import com.sopra.beans.Student;*

*@Controller*

*public class HelloController {*

*@RequestMapping("/hello")*

*public ModelAndView hello()*

*{*

*ModelAndView mv = new ModelAndView();*

*Student stu = new Student();*

*stu.setStid(100);*

*stu.setStname("Ram");*

*mv.addObject("student", stu);*

*//setting the view name*

*mv.setViewName("hello");*

*return mv;*

*}*

*}*

*hello.jsp:*

*```````````*

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"*

*isELIgnored="false"%>*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<h1>welcome to spring web mvc</h1><br/>*

*Student Information: <br/>*

*${student}*

*</body>*

*</html>*

*how to add the List of java objects to ModelAndView and displayed in view page:*

***HelloController.java:***

*package com.sopra.controller;*

*import java.util.ArrayList;*

*import java.util.List;*

*import org.springframework.stereotype.Controller;*

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

*import org.springframework.web.servlet.ModelAndView;*

*import com.sopra.beans.Student;*

*@Controller*

*public class HelloController {*

*@RequestMapping("/hello")*

*public ModelAndView hello()*

*{*

*List<Student> students = new ArrayList<Student>();*

*ModelAndView mv = new ModelAndView();*

*Student stu1 = new Student();*

*stu1.setStid(100);*

*stu1.setStname("Ram");*

*Student stu2 = new Student();*

*stu2.setStid(200);*

*stu2.setStname("sopra");*

*students.add(stu1);*

*students.add(stu2);*

*mv.addObject("students",students);*

*//setting the view name*

*mv.setViewName("hello");*

*return mv;*

*}*

*}*

*hello.jsp:*

*```````````*

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"*

*isELIgnored="false"%>*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<h1>welcome to spring web mvc</h1><br/>*

*Student Information: <br/>*

*${students}*

*</body>*

*</html>*

*how to send the data from view to controller and then controller to view:*

***RegistrationController.java:***

*package com.sopra.controllers;*

*import javax.servlet.http.HttpServletRequest;*

*import javax.servlet.http.HttpServletResponse;*

*import org.springframework.stereotype.Controller;*

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

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

*import org.springframework.web.servlet.ModelAndView;*

*import com.sopra.beans.Student;*

*@Controller*

*public class RegistrationController {*

*@RequestMapping("/showreg")*

*public ModelAndView showRegistrationPage()*

*{*

*ModelAndView mv = new ModelAndView();*

*mv.setViewName("userReg");*

*return mv;*

*}*

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

*public ModelAndView registerStudent(HttpServletRequest request,HttpServletResponse response)*

*{*

*String stid = request.getParameter("stid");*

*String firstName = request.getParameter("firstName");*

*String lastName = request.getParameter("lastName");*

*Student stu = new Student();*

*stu.setStid(Integer.parseInt(stid));*

*stu.setFirstName(firstName);*

*stu.setLastName(lastName);*

*ModelAndView mv = new ModelAndView();*

*mv.addObject("stu", stu);*

*mv.setViewName("success");*

*return mv;*

*}*

*}*

***Student.java:***

*package com.sopra.beans;*

*public class Student {*

*private int stid;*

*private String firstName;*

*private String lastName;*

*public Student()*

*{*

*}*

*public int getStid() {*

*return stid;*

*}*

*public void setStid(int stid) {*

*this.stid = stid;*

*}*

*public String getFirstName() {*

*return firstName;*

*}*

*public void setFirstName(String firstName) {*

*this.firstName = firstName;*

*}*

*public String getLastName() {*

*return lastName;*

*}*

*public void setLastName(String lastName) {*

*this.lastName = lastName;*

*}*

*@Override*

*public String toString() {*

*return "Student [stid=" + stid + ", firstName=" + firstName + ", lastName=" + lastName + "]";*

*}*

*}*

***userReg.jsp:***

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"%>*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*<h2>Registration page</h2>*

*<form action="registerUser" method="post">*

*<pre>*

*Enter Student Id: <input type="text" name="stid"/><br/>*

*Enter Student FirstName: <input type="text" name="firstName"/><br/>*

*Enter Student LastName: <input type="text" name="lastName"/><br/>*

*<input type="submit" value="register"/>*

*</pre>*

*</form>*

*</body>*

*</html>*

***success.jsp:***

*<%@ page language="java" contentType="text/html; charset=ISO-8859-1"*

*pageEncoding="ISO-8859-1"*

*isELIgnored="false"%>*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="ISO-8859-1">*

*<title>Insert title here</title>*

*</head>*

*<body>*

*Students Records are: <br/>*

*${stu}*

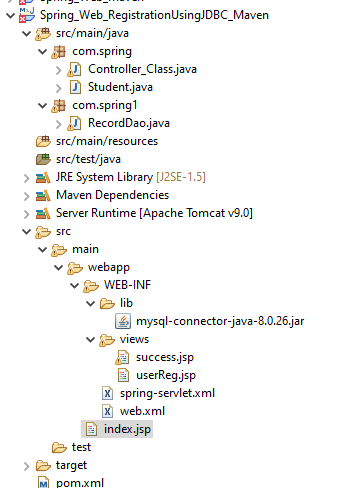
*</body>*

*</html>*

***assignment:****on top of this application apply spring jdbc module to store*

*student's data in the corresponding student table of db.*

*Registration with the help of JDBC in Spring Web Module*



***Web.xml***

<!-- Step1: configure the DispatcherServlet class in web.xml file -->

<!DOCTYPE web-app PUBLIC

"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"

"http://java.sun.com/dtd/web-app\_2\_3.dtd" >

<web-app>

<servlet>

<servlet-name>spring</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>spring</servlet-name>

<url-pattern>/</url-pattern></servlet-mapping></web-app>

***Controller\_class.java***

// Step2 :create controller class under src/main/java under one package(com.sopra.controller)

// make sure controller class must be annotated with @Controller annotation.

**package** com.spring;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

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

**import** org.springframework.jdbc.core.JdbcTemplate;

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

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

**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.servlet.ModelAndView;

**import** com.spring1.RecordDao;

@Controller

**public** **class** Controller\_Class {

@Autowired

**private** RecordDao dao;

@RequestMapping("/showreg")

**public** ModelAndView showRegistrationPage()

{

ModelAndView mv=**new** ModelAndView();

mv.setViewName("userReg");

**return** mv;

}

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

// First Approach

// public ModelAndView registerStudent(HttpServletRequest req,HttpServletResponse res)

// {

// String stid = req.getParameter("stid");

// String firstName = req.getParameter("firstName");

// String lastName = req.getParameter("lastName");

// Student stu=new Student();

// stu.setStid(stid);

// stu.setFirstName(firstName);

// stu.setLastName(lastName);

// dao.register(stu);

//

//

// ModelAndView mv=new ModelAndView();

// mv.addObject("stu",stu);

// mv.setViewName("success");

// return mv;

// Second Approach

// public ModelAndView registerStudent(@RequestParam("stid") int stid, @RequestParam("firstName") String firstName, @RequestParam("lastName") String lastName)

// {

//

// Student stu=new Student();

// stu.setStid(stid);

// stu.setFirstName(firstName);

// stu.setLastName(lastName);

// dao.register(stu);

//

// ModelAndView mv=new ModelAndView();

// mv.addObject("stu",stu);

// mv.setViewName("success");

// return mv;

// Third Approach

**public** ModelAndView registerStudent(@ModelAttribute("stu") Student stu)

{

dao.register(stu);

ModelAndView mv=**new** ModelAndView();

mv.addObject("stu",stu);

mv.setViewName("success");

**return** mv;

}

}

**spring\_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:context=*"http://www.springframework.org/schema/context"*

xmlns:aop=*"http://www.springframework.org/schema/aop"*

xsi:schemaLocation=*"*

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

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

*http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd"*>

<context:component-scan base-package=*"com.spring,com.spring1"*></context:component-scan>

<bean id=*"view"* class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<property name=*"prefix"*>

<value>/WEB-INF/views/</value>

</property>

<property name=*"suffix"*>

<value>.jsp</value>

</property>

</bean>

<!-- THE ABOVE CODE IS FOR JDBC -->

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

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

</bean>

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

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

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

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

<property name=*"password"* value=*"123N@ndini04"*/>

</bean>

</beans>

**Success.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"* isELIgnored=*"false"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Insert title here</title>

</head>

<body>

<!-- <h5>Successfully registered</h5><br/> -->

<h5>Successfully deleted</h5><br/>

<!-- <h5>Successfully updated</h5><br/> -->

</form>

</body>

</html>

**UserReg.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Insert title here</title>

</head>

<body>

<h2>Registration page</h2>

<form action=*"registerUser"* method=*"post"*>

<pre>

Enter Student Id: <input type=*"text"* name=*"stid"*/><br/>

Enter Student FirstName: <input type=*"text"* name=*"firstName"*/><br/>

Enter Student LastName: <input type=*"text"* name=*"lastName"*/><br/>

<input type=*"submit"* value=*"Register"*/>

</pre>

</form>

**Student.java**

**package** com.spring;

**public** **class** Student {

**private** **int** stid;

**private** String firstName;

**private** String lastName;

**public** Student()

{

}

**public** **int** getStid() {

**return** stid;

}

**public** **void** setStid(**int** stid) {

**this**.stid = stid;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

@Override

**public** String toString() {

**return** "Student [stid=" + stid + ", firstName=" + firstName + ", lastName=" + lastName + "]";

}

}

**RecordDao.java**

**package** com.spring1;

**import** java.sql.ResultSet;

**import** java.util.List;

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

**import** org.springframework.jdbc.core.JdbcTemplate;

**import** org.springframework.jdbc.core.ResultSetExtractor;

**import** org.springframework.stereotype.Component;

**import** org.springframework.stereotype.Repository;

**import** com.spring.Student;

@Repository

**public** **class** RecordDao {

@Autowired

**private** JdbcTemplate jdbcTemplate;

**public** **int** register(Student e)

{

// INSERT OPERATION

// String sql = "insert into employee values(?,?,?)";

// return jdbcTemplate.update(sql, e.getStid(),e.getFirstName(),e.getLastName());

// DELETE OPERATION

// String sql="Delete from Employee where empId=? ";

// return jdbcTemplate.update(sql,e.getStid());

// UPDATE OPERATION

String sql="update Employee set empfirstName=? where empId= ? ";

**return** jdbcTemplate.update(sql,e.getFirstName(),e.getStid());

}

}

**index.jsp**

<html>

<body>

<h2>Hello World!</h2>

</body>

</html>

**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>com.maven2</groupId>

<artifactId>Spring\_Web\_maven</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>Spring\_Web\_maven Maven Webapp</name>

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

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-web -->

<!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.10</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>5.3.10</version>

</dependency>

</dependencies>

<build><plugins><plugin><groupId>org.apache.maven.plugins</groupId><artifactId>maven-war-plugin</artifactId><version>3.3.1</version></plugin></plugins><finalName>check</finalName></build>

</project>

Hibernate:

hibernate is a orm framework

orm:

object relational mapping

dealing with java object --> database table

1 object --> 1 row of the table

instead of working on sql queries we can peform on top of our entity

class.

bean class

entity class

hibernate: to implment persistence logic

jdbc

spring jdbc

->

hibernate

hibernate architecutre includes many objects

-persistence object

-session factory

-transaction factory

-connection factory

-session

-transaction

steps to follow to develop hibernate application:

1.creat the maven project

2.add the hibernate core maven dependecy in pom.xml

3.under src/main/java right click -> new -> xml file(student.hbm.xml file)

4.under src/main/java right click -> new -> xml file(hibernate.cfg.xml)

5.under src/main/java create two packages (com.sopra.entity --> entity class,

com.sopra-> test class)

6.prepare entity class

7.prepare test class

8.run the application and check the db

**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>com.sopra</groupId>

<artifactId>hibernate-demo-1</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>hibernate-demo-1</name>

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

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.5.7.Final</version>

</dependency>

</dependencies>

</project>

**student.hbm.xml:**

````````````````

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

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD//EN"

"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="com.sopra.entity.Student" table="student">

<id name="stid">

<generator class="assigned"></generator>

</id>

<property name="firstName"></property>

<property name="lastName"></property>

</class>

</hibernate-mapping>

**hibernate.cfg.xml:**

`````````````````````

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

<!DOCTYPE hibernate-configuration SYSTEM

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<!-- data base properties -->

<property name="connection.driver\_class">oracle.jdbc.driver.OracleDriver</property>

<property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>

<property name="connection.username">sopra</property>

<property name="connection.password">sopra</property>

<!-- hibernate properties -->

<property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>

<property name="show\_sql">true</property>

<property name="format\_sql">true</property>

<!-- mapping file -->

<mapping resource="student.hbm.xml"/>

</session-factory>

</hibernate-configuration>

**Student.java:**

`````````````

package com.sopra.entity;

public class Student {

private int stid;

private String firstName;

private String lastName;

public int getStid() {

return stid;

}

public void setStid(int stid) {

this.stid = stid;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public Student()

{

}

}

**Test.java:**

``````````

package com.sopra;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.boot.Metadata;

import org.hibernate.boot.MetadataSources;

import org.hibernate.boot.registry.StandardServiceRegistry;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import com.sopra.entity.Student;

/\*\*

\* Hello world!

\*

\*/

public class App {

public static void main(String[] args) {

StandardServiceRegistry ssr = new StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

Metadata meta = new MetadataSources(ssr).getMetadataBuilder().build();

SessionFactory factory = meta.getSessionFactoryBuilder().build();

Session session = factory.openSession();

Transaction transaction = session.beginTransaction();

Student stu = new Student();

stu.setStid(102);

stu.setFirstName("ram");

stu.setLastName("s");

session.save(stu);

transaction.commit();

System.out.println("successfully saved");

factory.close();

session.close();

}

}

**note:**

````

right click on the project, add ojdbc jar file for our database

Improvement of previous code using Annotation(@Entity, @Id) and without hibernate mapping file (”filename.hbm.xml”)

**Student.java:**

package com.sopra.entity;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

//@Table(name = "student")

public class Student {

@Id

//@Column(name = "stid")

private int stid;

// @Column(name="firstName")

private String firstName;

//@Column(name="lastName")

private String lastName;

public int getStid() {

return stid;

}

public void setStid(int stid) {

this.stid = stid;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public Student()

{

}

}

**note:**

jpa is the specification/interfaces for orm(object relational mapping)

-sunmicrosystem

-hibernate is the popular one

-ibatis

-toplink

**hibernate.cfg.xml**

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

<!-- HIBERNATE CONFIGURATION FILE -->

<!DOCTYPE hibernate-configuration SYSTEM

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<!-- data base properties -->

<property name=*"connection.driver\_class"*>com.mysql.jdbc.Driver</property>

<property name=*"connection.url"*>jdbc:mysql://localhost:3306/MyDatabase</property>

<property name=*"connection.username"*>root</property>

<property name=*"connection.password"*>123N@ndini04</property>

<!-- hibernate properties -->

<property name=*"dialect"*>org.hibernate.dialect.MySQL5Dialect</property>

<property name=*"show\_sql"*>true</property>

<property name=*"format\_sql"*>true</property>

<!-- NEWLY ADDED PROPERTY THIS <property name="hbm2ddl.auto">create/update</property> FOR ANNOTATION

create:when we give this property value as create, it will always drop the existing

table and it will create the new table. WHILE

update: when we give this property value as update, it will not drop the existing

table just it will perform the operations on top of the existing table.

-->

<!-- <property name="hbm2ddl.auto">create</property> -->

<property name=*"hbm2ddl.auto"*>update</property>

<!-- mapping file -->

<!-- INSTEAD OF THIS LINE <mapping resource="student.hbm.xml"/> USE BELOW ONE FOR ANNOTATION -->

<mapping class=*"com.hibernate.entity.Student"*/>

</session-factory>

</hibernate-configuration>

**Test.java**

**package** com.hibernate;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.hibernate.boot.Metadata;

**import** org.hibernate.boot.MetadataSources;

**import** org.hibernate.boot.registry.StandardServiceRegistry;

**import** org.hibernate.boot.registry.StandardServiceRegistryBuilder;

**import** com.hibernate.entity.Student;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

StandardServiceRegistry ssr = **new** StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

Metadata meta = **new** MetadataSources(ssr).getMetadataBuilder().build();

SessionFactory factory = meta.getSessionFactoryBuilder().build();

Session session = factory.openSession();

Transaction transaction = session.beginTransaction();

Student stu = **new** Student();

stu.setId(102);

stu.setFirstname("kajol");

stu.setLastname("verma");

session.save(stu);

transaction.commit();

System.***out***.println("Operation Successful");

factory.close();

session.close();

}

}

**create vs update of hbm2ddl.auto value:**

* create:

when we give this property value as create, it will always drop the existing

table and it will create the new table.

* update:

when we give this property value as update, it will not drop the existing

table just it will perform the operations on top of the existing table.

**Note:**

* @column and @ table is optional as if it is not mentioned then table created with name same as entity class name and column created with the name same as entity class instance variable name.
* @Entity(use above the entity class) and @Id(use above the primary key variable declaration) is mandatory.

custom generator:

we can apply the custom generator also for this we have

to define one java class that must be implements from

IdentifierGenerator and then we can override generate() with our own custom

generator logic and then we can apply this strategy in our model class.

**CustomGenerator.java:**

package com.sopra.entity;

import java.io.Serializable;

import java.util.Random;

import org.hibernate.HibernateException;

import org.hibernate.engine.spi.SharedSessionContractImplementor;

import org.hibernate.id.IdentifierGenerator;

public class CustomGenerator implements IdentifierGenerator {

public Serializable generate(SharedSessionContractImplementor session, Object object) throws HibernateException {

int id=0;

Random random = new Random();

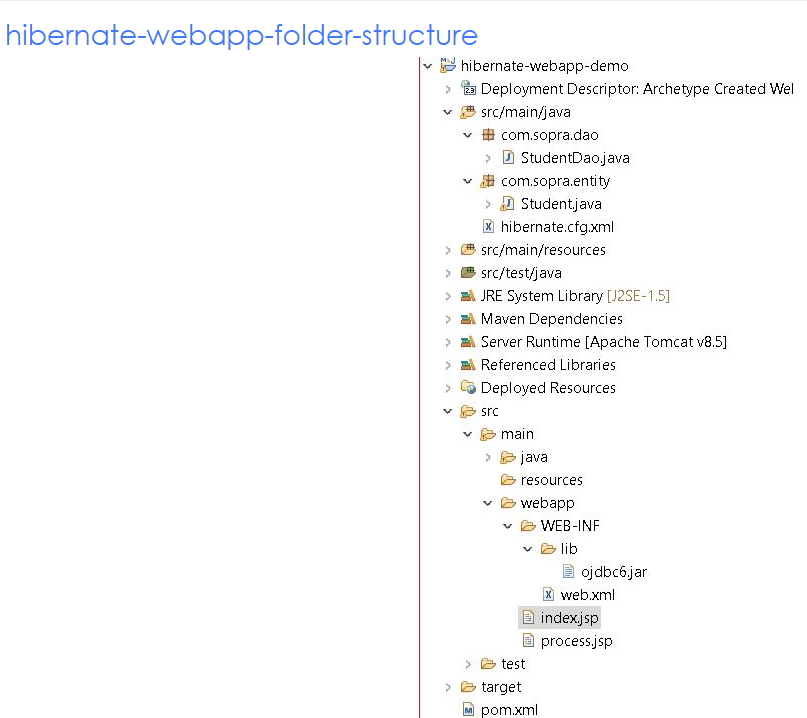
id=random.nextInt(1000);

return "stu-" + id;

}

}

hibernate web application (hibernate + jsp integration):



**index.jsp:**

<html>

<body>

<form action="process.jsp" method="post">

Student's First Name: <input type="text" name="firstName"/><br/>

Student's Last Name: <input type="text" name="lastName"/><br/>

<input type="submit" value="register"/>

</form>

</body>

</html>

**process.jsp:**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"

import="com.sopra.dao.StudentDao"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Insert title here</title>

</head>

<body>

<jsp:useBean id="stu" class="com.sopra.entity.Student"></jsp:useBean>

<jsp:setProperty property="\*" name="stu"/>

<%

int x = StudentDao.register(stu);

if(x>0)

{

out.println("successfully registered");

}

%>

</body>

</html>

**hibernate.cfg.xml:**

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

<!DOCTYPE hibernate-configuration SYSTEM

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<!-- data base properties -->

<property name="connection.driver\_class">oracle.jdbc.driver.OracleDriver</property>

<property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>

<property name="connection.username">sopra</property>

<property name="connection.password">sopra</property>

<!-- hibernate properties -->

<property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>

<property name="show\_sql">true</property>

<property name="format\_sql">true</property>

<property name="hbm2ddl.auto">create</property>

<!-- mapping file -->

<mapping class="com.sopra.entity.Student"/>

</session-factory>

</hibernate-configuration>

**Student.java:**

package com.sopra.entity;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import org.hibernate.annotations.GenericGenerator;

@Entity

public class Student {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private int stid;

private String firstName;

private String lastName;

public int getStid() {

return stid;

}

public void setStid(int stid) {

this.stid = stid;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public Student()

{

}

}

**StudentDao.java:**

package com.sopra.dao;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.boot.Metadata;

import org.hibernate.boot.MetadataSources;

import org.hibernate.boot.registry.StandardServiceRegistry;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import com.sopra.entity.Student;

public class StudentDao {

public static int register(Student st)

{

int i=0;

StandardServiceRegistry ssr = new StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

Metadata meta = new MetadataSources(ssr).getMetadataBuilder().build();

SessionFactory factory = meta.getSessionFactoryBuilder().build();

Session session = factory.openSession();

Transaction transaction = session.beginTransaction();

i = (Integer)session.save(st);

transaction.commit();

factory.close();

session.close();

return i;

}

}

**note:**

make sure add ojdbc jar to the buildpath and lib folder of project

Hibernate inheritance mapping:

->there are three inheritance mapping strategies are there

1.table per hierarchy

2.table per concrete class

3.table per subclass

**1.table per hierarchy:**

in table per hierarchy single table is required to map the whole hierarchy

an extra column is added to the the table to identify the class.

null values are stored in the table.

**2.table per concrete class:**

in case of table per concrete class, tables are created as per class.

But duplicate column added in subclass tables.

**3.table per subclass:**

in this strategy, tables are created as per class but related by

foregin key.

so there are no duplicate columns.

Employee emp = new Employee();

emp.setName("ram");

Regular\_Employee re = new Regular\_Employee();

re.setName("jain");

re.setSalary(70000);

re.setBonus(678);

Contract\_Employee ce = new Contract\_Employee();

ce.setName("rk");

ce.setPay\_per\_hour(8000);

ce.setContract\_duration("9 hours");

type

`````

employee

regularemployee

contractemployee

**Employee.java:**

`````````````

package com.sopra.entities;

import javax.persistence.DiscriminatorColumn;

import javax.persistence.DiscriminatorType;

import javax.persistence.DiscriminatorValue;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Inheritance;

import javax.persistence.InheritanceType;

@Entity

@Inheritance(strategy = InheritanceType.SINGLE\_TABLE)

@DiscriminatorColumn(name = "type",discriminatorType = DiscriminatorType.STRING)

@DiscriminatorValue("employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private int id;

private String name;

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;

}

}

**Regular\_Employee.java:**

``````````````````````

package com.sopra.entities;

import javax.persistence.DiscriminatorValue;

import javax.persistence.Entity;

@Entity

@DiscriminatorValue("regularemployee")

public class Regular\_Employee extends Employee {

private float salary;

private int bonus;

public float getSalary() {

return salary;

}

public void setSalary(float salary) {

this.salary = salary;

}

public int getBonus() {

return bonus;

}

public void setBonus(int bonus) {

this.bonus = bonus;

}

}

**Contract\_Employee.java:**

```````````````````````

package com.sopra.entities;

import javax.persistence.DiscriminatorValue;

import javax.persistence.Entity;

@Entity

@DiscriminatorValue("contractemployee")

public class Contract\_Employee extends Employee {

private float pay\_per\_hour;

private String contract\_duration;

public float getPay\_per\_hour() {

return pay\_per\_hour;

}

public void setPay\_per\_hour(float pay\_per\_hour) {

this.pay\_per\_hour = pay\_per\_hour;

}

public String getContract\_duration() {

return contract\_duration;

}

public void setContract\_duration(String contract\_duration) {

this.contract\_duration = contract\_duration;

}

}

**App.java:**

``````````

package com.sopra;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.boot.Metadata;

import org.hibernate.boot.MetadataSources;

import org.hibernate.boot.registry.StandardServiceRegistry;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import com.sopra.entities.Contract\_Employee;

import com.sopra.entities.Employee;

import com.sopra.entities.Regular\_Employee;

/\*\*

\* Hello world!

\*

\*/

public class App {

public static void main(String[] args) {

StandardServiceRegistry ssr = new StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

Metadata meta = new MetadataSources(ssr).getMetadataBuilder().build();

SessionFactory factory = meta.getSessionFactoryBuilder().build();

Session session = factory.openSession();

Transaction transaction = session.beginTransaction();

Employee emp = new Employee();

emp.setName("Ram");

Regular\_Employee re = new Regular\_Employee();

re.setName("jain");

re.setSalary(778787f);

re.setBonus(678);

Contract\_Employee ce = new Contract\_Employee();

ce.setName("xyz");

ce.setPay\_per\_hour(676);

ce.setContract\_duration("8hours");

session.save(emp);

session.save(re);

session.save(ce);

transaction.commit();

System.out.println("successfully saved");

factory.close();

session.close();

}

}

**hibernate.cfg.xml:**

``````````````````

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

<!DOCTYPE hibernate-configuration SYSTEM

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<!-- data base properties -->

<property name="connection.driver\_class">oracle.jdbc.driver.OracleDriver</property>

<property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>

<property name="connection.username">sopra</property>

<property name="connection.password">sopra</property>

<!-- hibernate properties -->

<property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>

<property name="show\_sql">true</property>

<property name="format\_sql">true</property>

<property name="hbm2ddl.auto">create</property>

<!-- mapping file -->

<mapping class="com.sopra.entities.Employee"/>

<mapping class="com.sopra.entities.Regular\_Employee"/>

<mapping class="com.sopra.entities.Contract\_Employee"/>

</session-factory>

</hibernate-configuration>

table per concrete class strategie:

```````````````````````````````````

**Employee.java:**

``````````````

package com.sopra.entities;

import javax.persistence.DiscriminatorColumn;

import javax.persistence.DiscriminatorType;

import javax.persistence.DiscriminatorValue;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Inheritance;

import javax.persistence.InheritanceType;

@Entity

@Inheritance(strategy = InheritanceType.TABLE\_PER\_CLASS)

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private int id;

private String name;

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;

}

}

**Regular\_Employee.java:**

```````````````````````

package com.sopra.entities;

import javax.persistence.DiscriminatorValue;

import javax.persistence.Entity;

@Entity

public class Regular\_Employee extends Employee {

private float salary;

private int bonus;

public float getSalary() {

return salary;

}

public void setSalary(float salary) {

this.salary = salary;

}

public int getBonus() {

return bonus;

}

public void setBonus(int bonus) {

this.bonus = bonus;

}

}

**Contract\_Employee.java:**

```````````````````````

package com.sopra.entities;

import javax.persistence.DiscriminatorValue;

import javax.persistence.Entity;

@Entity

public class Contract\_Employee extends Employee {

private float pay\_per\_hour;

private String contract\_duration;

public float getPay\_per\_hour() {

return pay\_per\_hour;

}

public void setPay\_per\_hour(float pay\_per\_hour) {

this.pay\_per\_hour = pay\_per\_hour;

}

public String getContract\_duration() {

return contract\_duration;

}

public void setContract\_duration(String contract\_duration) {

this.contract\_duration = contract\_duration;

}

}

**assignment:**

````````````

table per sub class strategy

hibernate query language:

hibernate query language -> sql

but it doesn't depends on table of the database

it is depending on entity class name and it's properties

so hql -> data base independent query lanugage

**Query interface:**

it is an object oriented representation of hibernate query.

the object of Query can be obtained -> createQuery() of Session interface

executeUpdate()- execute update /delete

list()

setFirstResult(int rowno)

**example:**

````````

package com.sopra;

import java.util.List;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.boot.Metadata;

import org.hibernate.boot.MetadataSources;

import org.hibernate.boot.registry.StandardServiceRegistry;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import org.hibernate.query.Query;

import com.sopra.entity.Student;

/\*\*

\* Hello world!

\*

\*/

public class App {

public static void main(String[] args) {

StandardServiceRegistry ssr = new StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

Metadata meta = new MetadataSources(ssr).getMetadataBuilder().build();

SessionFactory factory = meta.getSessionFactoryBuilder().build();

Session session = factory.openSession();

Transaction transaction = session.beginTransaction();

Query query = session.createQuery("from Student");

query.setFirstResult(0);

query.setMaxResults(3);

List list = query.list();

System.out.println(list);

transaction.commit();

System.out.println("successfully saved");

factory.close();

session.close();

}

}

**update query using hql:**

package com.sopra;

import java.util.List;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.boot.Metadata;

import org.hibernate.boot.MetadataSources;

import org.hibernate.boot.registry.StandardServiceRegistry;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import org.hibernate.query.Query;

import com.sopra.entity.Student;

/\*\*

\* Hello world!

\*

\*/

public class App {

public static void main(String[] args) {

StandardServiceRegistry ssr = new StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

Metadata meta = new MetadataSources(ssr).getMetadataBuilder().build();

SessionFactory factory = meta.getSessionFactoryBuilder().build();

Session session = factory.openSession();

Transaction transaction = session.beginTransaction();

Query query = session.createQuery("update Student set firstName=:fn where stid=:id");

query.setParameter("fn","verma");

query.setParameter("id", 1);

int status = query.executeUpdate();

transaction.commit();

System.out.println("successfully updated");

factory.close();

session.close();

}

}

spring and hibernate integration:

spring orm module:

spring orm module -> 1 line of code

->in hibernate f/w, we provide all the database information in

hibernate.cfg.xml

->But if we are going to integrate the hibernation with spring

we don't need to create the hibernate.cfg.xml

we can provide all this information in spring configuration file.

advantages of spring f/w with hibernate:

the spring f/w provides HibernateTemplate class, so you don't need to follow

so many steps like creating configuration , buildsessionfactory,

session,beginning and commiting transaction etc.

->so it saves a lot of code.

Student stu = new Student(100,"ram","sopra");

hibernateTemplate.save(stu);

**spring orm crud use case:**

1.create a table in the database

-it is optional

2.create a spring configuration file(springcontext.xml)

-it contains information of SessionFactory,DataSource...

3.Create Student.java file

-it is the persistent class/entity class/model class

4.create StudentDao.java

-it is the dao class that uses HibernateTemplate

5.create Test file

-it calls methods of EmployeeDao

**Student.java:**

package com.sopra.beans;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class Student {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private int id;

private String firstName;

private String lastName;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

@Override

public String toString() {

return "Student [id=" + id + ", firstName=" + firstName + ", lastName=" + lastName + "]";

}

}

**StudentDao.java:**

````````````````

package com.sopra.dao;

import com.sopra.beans.Student;

public interface StudentDao {

int saveStudent(Student student);

}

**StudentDaoImpl.java:**

`````````````````````

package com.sopra.dao;

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

import org.springframework.orm.hibernate5.HibernateTemplate;

import org.springframework.transaction.annotation.Transactional;

import com.sopra.beans.Student;

public class StudentDaoImpl implements StudentDao {

@Autowired

private HibernateTemplate template;

public HibernateTemplate getTemplate() {

return template;

}

public void setTemplate(HibernateTemplate template) {

this.template = template;

}

@Transactional

public int saveStudent(Student student) {

Integer result=(Integer)template.save(student);

return result;

}

}

**Test.java:**

```````````

package com.sopra;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.sopra.beans.Student;

import com.sopra.dao.StudentDao;

/\*\*

\* Hello world!

\*

\*/

public class App

{

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("springcontext.xml");

StudentDao dao = (StudentDao)context.getBean("dao");

Student stu = new Student();

stu.setFirstName("ram");

stu.setLastName("sopra");

int x = dao.saveStudent(stu);

System.out.println(x + "row(s) inserted");

}

}

**springcontext.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:context="http://www.springframework.org/schema/context"

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

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

xsi:schemaLocation="

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

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

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

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

<tx:annotation-driven/>

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

<bean id="dao" class="com.sopra.dao.StudentDaoImpl">

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

</bean>

<bean id="template"

class="org.springframework.orm.hibernate5.HibernateTemplate">

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

</bean>

<bean id="mySessionFactory"

class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">

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

<property name="annotatedClasses">

<list>

<value>com.sopra.beans.Student</value>

</list>

</property>

<property name="hibernateProperties">

<props>

<prop key="hibernate.dialect">org.hibernate.dialect.Oracle10gDialect</prop>

<prop key="hibernate.hbm2ddl.auto">create</prop>

<prop key="hibernate.show\_sql">true</prop>

<prop key="hibernate.format\_sql">true</prop>

</props>

</property>

</bean>

<bean id="ds"

class="org.springframework.jdbc.datasource.DriverManagerDataSource">

<property name="driverClassName"

value="oracle.jdbc.driver.OracleDriver" />

<property name="url"

value="jdbc:oracle:thin:@localhost:1521:xe" />

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

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

</bean>

<bean class="org.springframework.orm.hibernate5.HibernateTransactionManager" name="transactionManager">

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

</bean>

</beans>

**pom.xml:**

````````

->spring context

->hibernate core

->spring orm

<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>com.sopra</groupId>

<artifactId>spring-orm-deom</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-orm-deom</name>

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

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.10</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.5.7.Final</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-orm -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>5.3.10</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

spring web mvc application w.r.to java based configuration:

index.jsp:

`````````

<html>

<body>

<form action="greet">

<input type="submit" value="press to greet">

</form>

</body>

</html>

GreetController.java:

````````````````````

package com.sopra.web;

import org.springframework.stereotype.Controller;

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

import org.springframework.web.servlet.ModelAndView;

@Controller

public class GreetController {

@RequestMapping("/greet")

public ModelAndView showView()

{

ModelAndView mv = new ModelAndView();

mv.setViewName("result");

mv.addObject("res","Sopra welcomes you to Spring");

return mv;

}

}

MyConfig.java:

`````````````

package com.sopra.web;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.ViewResolver;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

import org.springframework.web.servlet.view.InternalResourceViewResolver;

@Configuration

public class MyConfig implements WebMvcConfigurer {

@Bean

public ViewResolver view()

{

InternalResourceViewResolver ivr = new InternalResourceViewResolver();

ivr.setPrefix("/WEB-INF/jsps/");

ivr.setSuffix(".jsp");

return ivr;

}

}

WebInitializer.java:

````````````````````

package com.sopra.web;

import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;

public class WebInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {

@Override

protected Class<?>[] getRootConfigClasses() {

// TODO Auto-generated method stub

return null;

}

@Override

protected Class<?>[] getServletConfigClasses() {

// TODO Auto-generated method stub

return new Class[] {MyConfig.class};

}

@Override

protected String[] getServletMappings() {

// TODO Auto-generated method stub

return new String[] {"/"};

}

}

result.jsp:

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"

isELIgnored="false"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Insert title here</title>

</head>

<body>

${res}

</body>

</html>

**Assignment:**

develop the registration application using spring web mvc orm module w.r.to java based configuration

http://javainsimpleway.com/spring-mvc-with-hibernate-crud-example/

Development Steps

1.Create a Maven Web Application

2.Add Dependencies - pom.xml File

3.Project Structure

4.AppInitializer - Register a DispatcherServlet using Java-based Spring configuration

5.AppContext - Spring and Hibernate Integration using Java-based Spring configuration

6.WebMvcConfig - Spring MVC Bean Configuration using Java-based Spring configuration

7.JPA Entity - Customer.java

8.Spring MVC Controller Class - CustomerController.java

9.Service Layer - CustomerService.java and CustomerServiceImpl.java

10.DAO Layer - CustomerDAO.java and CustomerDAOImpl.java 11 JSP Views - customer-form.jsp and list-customers.jsp

11.Serve Static Resources - CSS and JS

12.Build and Run an application

spring orm mvc registration application:

**EmployeeController.java:**

package com.sopra.controller;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

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

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.servlet.ModelAndView;

import com.sopra.entity.Employee;

import com.sopra.service.EmployeeService;

@Controller

public class EmployeeController {

@Autowired

private EmployeeService service;

public EmployeeService getService() {

return service;

}

public void setService(EmployeeService service) {

this.service = service;

}

@RequestMapping("registrationPage")

public String showRegistrationPage() {

return "userReg";

}

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

public ModelAndView registerUser(@RequestParam("empid") int empid,@RequestParam("firstName") String firstName,@RequestParam("lastName") String lastName ) {

Employee emp = new Employee();

emp.setId(empid);

emp.setFirstName(firstName);

emp.setLastName(lastName);

int result = service.save(emp);

ModelAndView mv = new ModelAndView();

mv.addObject("result", "employee created with empid: " + result);

mv.setViewName("userReg");

return mv;

}

}

**EmployeeService.java:**

`````````````````````

package com.sopra.service;

import com.sopra.entity.Employee;

public interface EmployeeService {

int save(Employee emp);

}

**EmployeeServiceImpl.java:**

`````````````````````````

package com.sopra.service.impl;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.sopra.dao.EmployeeDao;

import com.sopra.entity.Employee;

import com.sopra.service.EmployeeService;

@Service

public class EmployeeServiceImpl implements EmployeeService {

@Autowired

private EmployeeDao dao;

public EmployeeDao getDao() {

return dao;

}

public void setDao(EmployeeDao dao) {

this.dao = dao;

}

@Override

@Transactional

public int save(Employee emp) {

//business logic

return dao.create(emp);

}

}

**EmployeeDao.java**:

`````````````````

package com.sopra.dao;

import com.sopra.entity.Employee;

public interface EmployeeDao {

int create(Employee emp);

}

**EmployeeDaoImpl.java:**

package com.sopra.dao.impl;

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

import org.springframework.orm.hibernate5.HibernateTemplate;

import org.springframework.stereotype.Repository;

import com.sopra.dao.EmployeeDao;

import com.sopra.entity.Employee;

@Repository

public class EmployeeDaoImpl implements EmployeeDao {

@Autowired

private HibernateTemplate template;

public HibernateTemplate getTemplate() {

return template;

}

public void setTemplate(HibernateTemplate template) {

this.template = template;

}

@Override

public int create(Employee emp) {

Integer result = (Integer)template.save(emp);

System.out.println(emp.getId());

return result;

}

}

**Employee.java:**

package com.sopra.entity;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name="employee")

public class Employee {

@Id

@Column(name="id")

private int id;

@Column(name="firstname")

private String firstName;

@Column(name="lastname")

private String lastName;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public Employee()

{

}

public Employee(int id, String firstName, String lastName) {

super();

this.id = id;

this.firstName = firstName;

this.lastName = lastName;

}

public String toString()

{

return id + " " + firstName + " " +lastName;

}

}

**note:** ojdbc6.jar must be added to the buildpath and lib folder of

application.

**userReg.java:**

<%@ page isELIgnored="false" language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Employee Registraion Page</title>

</head>

<body>

<h2> Employee Registration Page</h2>

<form action="registerUser" method="post">

<pre>

Enter Employee Id: <input type="text" name="empid"/><br/>

Enter Employee FirstName: <input type="text" name="firstName"/><br/>

Enter Employee LastName: <input type="text" name="lastName"/><br/>

<input type="submit" value="register"/>

</pre>

</form>

<br/>

${result}

</body>

</html>

**web.xml:**

<!DOCTYPE web-app PUBLIC

"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"

"http://java.sun.com/dtd/web-app\_2\_3.dtd" >

<web-app>

<servlet>

<servlet-name>sopra</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>sopra</servlet-name>

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

</servlet-mapping>

</web-app>

**sopra-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:context="http://www.springframework.org/schema/context"

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

xsi:schemaLocation="

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

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

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

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

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

<context:component-scan

base-package="com.sopra"/>

<tx:annotation-driven/>

<bean id="dataSource"

class="org.springframework.jdbc.datasource.DriverManagerDataSource">

<property name="driverClassName"

value="oracle.jdbc.driver.OracleDriver"></property>

<property name="url"

value="jdbc:oracle:thin:@localhost:1521:xe"></property>

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

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

</bean>

<bean id="mysessionFactory"

class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">

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

<property name="annotatedClasses">

<list>

<value>com.sopra.entity.Employee</value>

</list>

</property>

<property name="hibernateProperties">

<props>

<prop key="hibernate.dialect">org.hibernate.dialect.Oracle9Dialect</prop>

<prop key="hibernate.hbm2ddl.auto">create</prop>

<prop key="hibernate.show\_sql">true</prop>

<prop key="hibernate.format\_sql">true</prop>

</props>

</property>

</bean>

<bean id="template"

class="org.springframework.orm.hibernate5.HibernateTemplate">

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

</bean>

<bean

class="org.springframework.orm.hibernate5.HibernateTransactionManager"

name="transactionManager">

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

</bean>

<bean id="view" class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix">

<value>/WEB-INF/jsps/</value>

</property>

<property name="suffix">

<value>.jsp</value>

</property>

</bean>

</beans>

**pom.xml:**

required dependencies:

-spring orm

-spring web mvc

-hibernate core

spring boot:

spring - pivotal organization

Spring Boot is Spring's convention-over-configuration solution for

creating stand-alone, production-grade Spring-based Applications that you

can "just run".

It is preconfigured with the Spring team's "opinionated view" of the best

configuration and use of the Spring platform and third-party libraries so

you can get started with minimum fuss. Most Spring Boot applications need

very little Spring configuration.

**Key Features:**

Create stand-alone Spring applications

Embed Tomcat or Jetty directly (no need to deploy WAR files)

Provide opinionated 'starter' Project Object Models (POMs) to simplify

your Maven configuration

Automatically configure Spring whenever possible

Provide production-ready features such as metrics, health checks and

externalized configuration

Absolutely no code generation and no requirement for XML configuration.

Smooth Integration and supports all Enterprise Integration Patterns.

->spring boot is a projet that actually built on top of the spring f/w.

->it provides an easier and faster way to setup,configure and run both

simple and web applications.

->in spring boot, there is no requirement for xml configuration.

->It uses convention over configuration software design pattern that means

it decreases the developer's effort.

->we can use sts ide(spring tool suite) or spring intializer to develop

spring boot java applications.

along with spring boot f/w,some other spring projects are there to help

to build applications addresing moder business needs.

->spring data

. it simplifies data accessf from the relational and

nosql databases.

->spring security

->spring social

prerequisites to develop applications using spring boot?

* java 1.8
* Maven 3.0+
* Spring f/w 5.x
* an ide (sts) is recommended

**latest version of spring boot:**

The latest version of Spring Boot is 2.2.1

**spring** ->spring f/w is a widely used java ee f/w for building applications

**spring boot** -> it is also a f/w , widely used to devlop REST API's

**REST** -Representational State Transfer(web services)

sprinf f/w ->

doesnot support in memory databases ->

but where as spring boot supports in memory databses such as

derby ,h2 etc.

Developers manually define dependencies for the spring project

in pom.xml file.

spring boot comes with the concept of starter in pom.xml file

that internally takes care of downloading dependencies jars based on spring boot requirement.

spring boot vs spring mvc:

**spring boot:**

there is no requirement for a deployment descriptor(web.xml file)

it reduces development time and increase productivity

**spring:**

a deployment descriptor is required.

it takes more time to achieve the same.

**spring boot architecture:**

json => java script object notation

presentation layer

-handles the HTTP requests, convert the json to object

and after that it transfers this object to business layer

business layer

-it consists of service classes and uses services provided by

data access layers.

persistence layer

-translates business objects from and to database rows.

database layer

-crud operations performed.

**NOTE:**

creating spring boot application using spring initializer tool:

https://start.spring.io

**com.sopra(SpringBootApplication.java):**

package com.sopra;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import org.springframework.context.ConfigurableApplicationContext;

import org.springframework.context.annotation.ComponentScan;

import com.sopra.beans.Student;

@SpringBootApplication

public class SpringbootDemoApplication {

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(SpringbootDemoApplication.class, args);

Student student = context.getBean(Student.class);

//setting the values to Student object by calling setter methods - setter injection

student.setStid(100);

student.setStname("rama");

//we are displaying student object - toString()

System.out.println(student);

}

}

**com.sopra.beans(Student)**

package com.sopra.beans;

import org.springframework.stereotype.Component;

@Component

public class Student {

@Override

public String toString() {

return "Student [stid=" + stid + ", stname=" + stname + "]";

}

private int stid;

private String stname;

public int getStid() {

return stid;

}

public void setStid(int stid) {

this.stid = stid;

}

public String getStname() {

return stname;

}

public void setStname(String stname) {

this.stname = stname;

}

public Student() {

}

}

**note:**

**@SpringBootConfiguration vs @EnableAutoConfiguration:**

@SpringBootConfiguration annotation tells us that a class is a configuration class,

and @EnableAutoConfiguration automatically configures the Spring application based

on its included jar files.

spring boot web application:

**HelloController.java:**

package com.sopra.controller;

import org.springframework.stereotype.Controller;

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

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

import org.springframework.web.servlet.ModelAndView;

@Controller

public class HelloController {

@RequestMapping("/home")

public ModelAndView hello()

{

ModelAndView mv = new ModelAndView();

mv.setViewName("hello");

return mv;

}

}

**application.properties:**

server.port=2021

spring.mvc.view.prefix=/WEB-INF/jsps/

spring.mvc.view.suffix=.jsp

**pom.xml:**

->tomcat jasper 9.9.53

->spring boot web

->spring boot dev tools

official documentation for spring data jpa module:

https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#reference

**for mysql application.properties:**

spring.datasource.url=jdbc:mysql://localhost:3306/mydb?useSSL=false

spring.datasource.username=root

spring.datasource.password=root

spring.jpa.hibernate.ddl-auto=create

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

crud application using spring data jpa module:

**Student.java:**

package com.sopra.entity;

import javax.persistence.Entity;

import javax.persistence.Id;

@Entity

public class Student {

@Id

private int stid;

private String firstName;

private String lastName;

public int getStid() {

return stid;

}

public void setStid(int stid) {

this.stid = stid;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

@Override

public String toString() {

return "Student [stid=" + stid + ", firstName=" + firstName + ", lastName=" + lastName + "]";

}

public Student() {

super();

}

}

**StudentRepository:**

package com.sopra.repository;

import org.springframework.data.repository.CrudRepository;

import com.sopra.entity.Student;

public interface StudentRepository extends CrudRepository<Student, Integer> {

}

**Test application:**

package com.sopra;

import java.util.Optional;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import com.sopra.entity.Student;

import com.sopra.repository.StudentRepository;

@SpringBootApplication

public class SpringbootJpaDemoApplication {

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(SpringbootJpaDemoApplication.class, args);

StudentRepository repo = context.getBean(StudentRepository.class);

Student stu = new Student();

stu.setStid(105);

stu.setFirstName("tanisq");

stu.setLastName("sopra");

//to save we have to call save methods

// Student st= repo.save(stu);

//Iterable<Student> findAll = repo.findAll();

// System.out.println(findAll);

/\* Student st = repo.findById(100).get();

repo.delete(st);

System.out.println("row will be delted");

\*/

Student st = repo.findById(101).get();

st.setFirstName("rk");

Student s= repo.save(st);

System.out.println(s);

}

}

application.properties:

```````````````````````

spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe

spring.datasource.username=sopra

spring.datasource.password=sopra

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

**finder methods of Repository:**

package com.sopra.repository;

import org.springframework.data.repository.CrudRepository;

import com.sopra.entity.Student;

public interface StudentRepository extends CrudRepository<Student, Integer> {

public Student findByLastName(String lastName);

public Student findByFirstName(String firstName);

public Student findByFirstNameOrStid(String firstName,int stid);

}

Test:

``````

package com.sopra;

import java.util.Optional;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import com.sopra.entity.Student;

import com.sopra.repository.StudentRepository;

@SpringBootApplication

public class SpringbootJpaDemoApplication {

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(SpringbootJpaDemoApplication.class, args);

StudentRepository repo = context.getBean(StudentRepository.class);

Student st = repo.findByFirstName("jain");

System.out.println(st);

Student st1 = repo.findByFirstNameOrStid("umesh", 107);

System.out.println(st1);

}

}

lombok api:

https://projectlombok.org/download

->DOUBLE CLICK ON downloaded file, click on specify location and upto eclipse.exe

choose the location

->click on install/update

->close it

->restart the eclipse and then start to use lombok annotations on top of entity

classes

note: in pom.xml file add the lombok dependecy

**Student.java:**

package com.sopra.entity;

import javax.persistence.Entity;

import javax.persistence.Id;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

@Entity

@Data

@AllArgsConstructor

@NoArgsConstructor

public class Student {

@Id

private int stid;

private String firstName;

private String lastName;

}

**StudentRepository.java:**

package com.sopra.repository;

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

import org.springframework.stereotype.Repository;

import com.sopra.entity.Student;

public interface StudentRepository extends JpaRepository<Student,Integer> {

}

**Test**

package com.sopra;

import java.util.List;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import org.springframework.data.domain.Page;

import org.springframework.data.domain.PageRequest;

import org.springframework.data.domain.Sort;

import org.springframework.data.domain.Sort.Direction;

import com.sopra.entity.Student;

import com.sopra.repository.StudentRepository;

@SpringBootApplication

public class SpringbootJpaPaginationDemoApplication {

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(SpringbootJpaPaginationDemoApplication.class, args);

StudentRepository repo= context.getBean(StudentRepository.class);

//Page<Student> findAll = repo.findAll(PageRequest.of(1,1));

//findAll.forEach(p->System.out.println(p));

//List<Student> findAll = repo.findAll(Sort.by("firstName","lastName"));

//findAll.forEach(p->System.out.println(p.getFirstName() + " " + p.getLastName()));

//List<Student> findAll = repo.findAll(Sort.by(Sort.Direction.DESC, "firstName"));

//findAll.forEach(p->System.out.println(p.getFirstName()));

// repo.findAll(PageRequest.of(0, 2, Direction.DESC, "firstName")).forEach(p->System.out.println(p.getFirstName()));

}

}

**application.properties:**

`

spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe

spring.datasource.username=sopra

spring.datasource.password=sopra

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

JPQL

* java persistence query language
* JPQL is a powerful query language that allows you to define database

queries based on your entity model.

* Its structure and syntax are very

similar to SQL.

**StudentRepository.java:**

package com.sopra.repository;

import java.util.List;

import javax.transaction.Transactional;

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

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

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

import org.springframework.data.repository.query.Param;

import org.springframework.stereotype.Repository;

import com.sopra.entity.Student;

public interface StudentRepository extends JpaRepository<Student,Integer> {

@Query("from Student")

public List<Student> findAllStudents();

@Query("select st.firstName ,st.lastName from Student st")

public List<Object[]> findStudentsPartialInfo();

@Query("from Student where firstName=:fname")

public List<Student> findByFirstName(@Param("fname") String firstName);

@Modifying

@Transactional

@Query("delete from Student where stid=:id")

int deleteStudentById(@Param("id") int stid);

@Query(value = "select \* from student",nativeQuery = true)

public List<Student> getAllStudents();

}

**Student.java:**

package com.sopra.entity;

import javax.persistence.Entity;

import javax.persistence.Id;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

@Entity

@Data

@AllArgsConstructor

@NoArgsConstructor

public class Student {

@Id

private int stid;

private String firstName;

private String lastName;

}

**Test.java:**

package com.sopra;

import java.util.List;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import org.springframework.data.domain.Page;

import org.springframework.data.domain.PageRequest;

import org.springframework.data.domain.Sort;

import org.springframework.data.domain.Sort.Direction;

import com.sopra.entity.Student;

import com.sopra.repository.StudentRepository;

@SpringBootApplication

public class SpringbootJpaPaginationDemoApplication {

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(SpringbootJpaPaginationDemoApplication.class, args);

StudentRepository repo= context.getBean(StudentRepository.class);

//Page<Student> findAll = repo.findAll(PageRequest.of(1,1));

//findAll.forEach(p->System.out.println(p));

//List<Student> findAll = repo.findAll(Sort.by("firstName","lastName"));

//findAll.forEach(p->System.out.println(p.getFirstName() + " " + p.getLastName()));

//List<Student> findAll = repo.findAll(Sort.by(Sort.Direction.DESC, "firstName"));

//findAll.forEach(p->System.out.println(p.getFirstName()));

// repo.findAll(PageRequest.of(0, 2, Direction.DESC, "firstName")).forEach(p->System.out.println(p.getFirstName()));

/\*List<Student> students = repo.findAllStudents();

for(Student st: students)

{

System.out.println(st);

}\*/

/\*List<Object[]> objs = repo.findStudentsPartialInfo();

for(Object[] ob:objs)

{

System.out.println(ob[0] + " " + ob[1]);

}\*/

/\* List<Student> students = repo.findByFirstName("nishanth");

for(Student st:students)

{

System.out.println(st);

}\*/

/\* int x = repo.deleteStudentById(106);

System.out.println(x + "row(s) delted");\*/

List<Student> students= repo.getAllStudents();

for(Student st: students)

{

System.out.println(st);

}

}

}

aggregate functions:

```````````````````````

@Query("SELECT AVG(u.age) from User u")

int getAverageAge();

spring rest module:

WEBSERVICE:

icici card --> icici atm's (same type of applications)

icici card(java) -> sbi atm(.net)

Web services are XML-based information exchange/JSON systems that use the

Internet for direct application-to-application interaction. These systems

can include programs, objects, messages, or documents.

-soap(simple object acccess protocol)->protocol based -> supports only xml

-restful(representational state transfer)->architral style-> supports xml and json

rest developed by Roy Fielding who also developed http.

resource has some reprsentations like xml,json.

HTTP methods:

Get-retrieving the data

put -updating the data

post - inserting the data

delete -deleting the data

Http also defines standard stats codes.

404 - Resource not foundation

200 - success

201 - created

401 - unauthorized

500 - internal server error

...

official url to check all http status codes:

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>

**advantages:**

->these are platform independent

it can be writtin in any programming language and can be executed

on any platform.

->it supports different data formats like html,xml,json,text..

->these are reusable

->they are language neutral.

**@Controller** : is the annotation is used to mark the class as controller class

**@RestController** : it is the annotation which is the combination of two annotations those are @Controller + @ResponseBody

**note:**

@ResponseBody is a Spring annotation which binds a method return value to the

web response body. It is not interpreted as a view name. It uses HTTP Message

converters to convert the return value to HTTP response body, based on the

content-type in the request HTTP header

**@PathVariable:**

put, the @PathVariable annotation can be used to handle template variables

in the request URI mapping, and set them as method parameters.

package com.sopra.model;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@AllArgsConstructor

@NoArgsConstructor

public class HelloBean {

private String message;

}

**HelloController.java:**

package com.sopra.controller;

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

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

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

import com.sopra.model.HelloBean;

@RestController

public class HelloController {

@GetMapping("/hello")

public String hello()

{

return "Welcome to SpringBoot Rest Demo";

}

@GetMapping("/hello-bean")

public HelloBean helloBean()

{

return new HelloBean("welcome to spring boot bean class");

}

@GetMapping("/hello/{name}")

public String helloPath(@PathVariable("name") String x)

{

return "welcome to " + x;

}

}

**application.properties:**

server.port=2021

Spring Rest application crud usecase:

**Department.java:**

package com.sopra.entity;

import javax.persistence.Entity;

import javax.persistence.Id;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@AllArgsConstructor

@NoArgsConstructor

@Entity

public class Department {

@Id

private int departmentId;

private String departmentName;

private String departmentAddress;

private String departmentCode;

}

**DepartmentController.java:**

package com.sopra.controller;

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

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

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

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

import com.sopra.entity.Department;

import com.sopra.service.DepartmentService;

@RestController

public class DepartmentController {

@Autowired

private DepartmentService service;

@PostMapping("/department")

public Department saveDepartment(@RequestBody Department department)

{

return service.saveDepartment(department);

}

@GetMapping("/departments")

public List<Department> fetchAllDepartments()

{

return service.fetchAllDepartments();

}

@GetMapping("/department/{departmentId}")

public Department fetchById(@PathVariable("departmentId") int departmentId)

{

return service.fetchById(departmentId);

}

@DeleteMapping("/department/{id}")

public String deleteById(@PathVariable("id") int departmentId)

{

service.deleteById(departmentId);

return "department - " + departmentId + " deleted successfully";

}

@PutMapping("/department/{id}")

public Department updateDepartment(@PathVariable("id") int departmentId, @RequestBody Department department)

{

return service.updateDepartment(departmentId,department);

}

@GetMapping("department/{dname}/{daddress}")

public Department findDepartmentByDepartmentNameAndDepartmentAddress(@PathVariable("dname") String dname,@PathVariable("daddress") String daddress)

{

return service.findDepartmentByDepartmentNameAndDepartmentAddress(dname,daddress);

}

}

/\*localhost:2021/department->post

body raw json

{

"DepartmentId":100,

"DepartmentName":"csc",

"DepartmentAddress":"hyd",

"DepartmentCode":"csc-001"

}

}\*/

**DepartmentRepository.java:**

package com.sopra.repository;

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

import com.sopra.entity.Department;

public interface DepartmentRepository extends JpaRepository<Department, Integer> {

public Department findByDepartmentNameAndDepartmentAddress(String departmentName,String departmentAddress);

}

**DepartmentService.java:**

package com.sopra.service;

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

import org.springframework.stereotype.Service;

import com.sopra.entity.Department;

import com.sopra.repository.DepartmentRepository;

@Service

public class DepartmentService {

@Autowired

private DepartmentRepository repository;

public Department saveDepartment(Department department) {

// TODO Auto-generated method stub

return repository.save(department);

}

public List<Department> fetchAllDepartments()

{

return repository.findAll();

}

public Department fetchById(int departmentId)

{

return repository.findById(departmentId).get();

}

public void deleteById(int departmentId) {

repository.deleteById(departmentId);

}

public Department updateDepartment(int departmentId, Department department) {

// TODO Auto-generated method stub

Department deptDb = repository.findById(departmentId).get();

deptDb.setDepartmentName(department.getDepartmentName());

deptDb.setDepartmentAddress(department.getDepartmentAddress());

deptDb.setDepartmentCode(department.getDepartmentCode());

return repository.save(deptDb);

}

public Department findDepartmentByDepartmentNameAndDepartmentAddress(String dname,String daddress)

{

return repository.findByDepartmentNameAndDepartmentAddress(dname, daddress);

}

}

**application.properties:**

server.port=2021

spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe

spring.datasource.username=sopra

spring.datasource.password=sopra

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

spring.jpa.database-platform=org.hibernate.dialect.Oracle10gDialect

**assignment:**

fetchByDepartmentNameandAddress(String departmentName,String departmentAddress)

Swagger:

if it is soap web based service -> wsdl (web service description language)

swagger2 is an open source project used to generate the REST API documents

for RESTful web services.

It provides a user interface to access our RESTful web services via the

web browser.

it specifies the format (url,method,and representation) to describe REST webservices.

A swagger is open source tool

it is one of the popular api documenation format for restful webservices.

It provides both json and ui support.

->It allows us to describe our entire api's including

->available endpoints and operations on each endpoints

->operation parameters for each operation

->authentication methods.

->contact information,license,terms and user

add following two dependencies in pom.xml file:

````````````````````````````````````````````````

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.9.2</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger-ui -->

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger-ui</artifactId>

<version>2.9.2</version>

</dependency>

**define one configuration class for swagger like below: Swagger Configuration**

`

package com.learnersstop;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import springfox.documentation.spi.DocumentationType;

import springfox.documentation.spring.web.plugins.Docket;

import springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration

@EnableSwagger2

public class SwaggerConfig {

@Bean

public Docket api()

{

return new Docket(DocumentationType.SWAGGER\_2);

}

}

**NOTE:**

to check swagger documentation for our application, pass below url in the

browser: http://localhost:2021/swagger-ui.html

**Assignment:**

refer content negotiation

crud use case:

``````````````

----spring boot mvc data jpa-----

pom.xml:

spring data jpa

spring web

spring dev tools

oracle driver

lombok

entity class:

`````````````

package com.sopra.entity;

import java.util.Objects;

import javax.persistence.Entity;

import javax.persistence.Id;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@AllArgsConstructor

@NoArgsConstructor

@Entity

public class Student {

@Id

private int stid;

private String stname;

private double stmarks;

}

SpringBOOT DATA JPA CRUD APPLICATION-STUDENT MANAGEMENT APP - JSP:

```````````````````````````````````````````````````````````````````

package com.sopra;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringbootMvcDatajpaApplication {

public static void main(String[] args) {

SpringApplication.run(SpringbootMvcDatajpaApplication.class, args);

}

}

StudentController.java:

````````````````````````

package com.sopra.controller;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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 com.sopra.entity.Student;

import com.sopra.service.StudentService;

@Controller

public class StudentController {

@Autowired

StudentService studentService;

@RequestMapping("/read-student")

public String showReadStudentPage(Model model) {

model.addAttribute("students", studentService.findAll());

return "readstudent";

}

@RequestMapping("/create-student")

public String showCreateStudentPage(Model model) {

model.addAttribute("command", new Student());

return "createstudent";

}

@RequestMapping(value = "/create-student", method = RequestMethod.POST)

public String createStudent(@ModelAttribute("student") Student student) {

studentService.saveStudent(student);

return "redirect:/read-student";

}

@RequestMapping(value = "/update-student/{id}")

public String showUpdateStudentPage(@PathVariable int id, Model model) {

model.addAttribute("id", id);

model.addAttribute("command", studentService.findById(id).orElse(null));

return "updatestudent";

}

@RequestMapping(value = "/update-student/{id}", method = RequestMethod.POST)

public String updateStudent(@PathVariable int id, @ModelAttribute("student") Student student) {

studentService.updateStudent(id, student);

return "redirect:/read-student";

}

@RequestMapping(value = "/delete-student/{id}")

public String deleteStudent(@PathVariable int id) {

studentService.deleteById(id);

return "redirect:/read-student";

}

}

Student.java:

``````````````

package com.sopra.entity;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@AllArgsConstructor

@NoArgsConstructor

@Entity

public class Student {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int stid;

private String stname;

private double stmarks;

}

StudentRepository.java:

````````````````````````

package com.sopra.repository;

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

import org.springframework.stereotype.Repository;

import com.sopra.entity.Student;

@Repository

public interface StudentRepository extends JpaRepository<Student, Integer> {

}

StudentService.java:

````````````````````

package com.sopra.service;

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.sopra.entity.Student;

import com.sopra.repository.StudentRepository;

@Service

public class StudentService {

@Autowired

private StudentRepository studentRepository;

public List<Student> findAll() {

return studentRepository.findAll();

}

public Optional<Student> findById(int id) {

return studentRepository.findById(id);

}

public Student saveStudent(Student student) {

return studentRepository.save(student);

}

public Student updateStudent(int id, Student student) {

Student updatedStudent = studentRepository.findById(id).orElse(null);

updatedStudent.setStname(student.getStname());

updatedStudent.setStmarks(student.getStmarks());

return studentRepository.save(updatedStudent);

}

public void deleteById(int id) {

studentRepository.deleteById(id);

}

}

application.properties:

````````````````````````

spring.datasource.url=jdbc:mysql://localhost:3306/mydb?useSSL=false

spring.datasource.username=root

spring.datasource.password=root

spring.jpa.hibernate.ddl-auto=create

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

server.port=2021

spring.mvc.view.prefix:/WEB-INF/jsps/

spring.mvc.view.suffix:.jsp

index.html:

```````````

<!DOCTYPE html>

<html>

<head>

<title>My Students - Spring Boot Web MVC</title>

</head>

<body>

<h1>Student Management System- CRUD App</h1>

<a href="/read-student">Get Students</a>|

<a href="/create-student">Create Student</a>

</body>

</html>

createstudent.jsp:

````````````````````

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

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

<h1>Create Student</h1>

<form:form method="post" action="/create-student">

<table>

<tr>

<td>Enter Student name: </td>

<td><form:input path="stname"/></td>

</tr>

<tr>

<td>Enter Student marks: </td>

<td><form:input path="stmarks"/></td>

</tr>

<tr>

<td></td>

<td><input type="submit" value="Create"/></td>

</tr>

</table>

</form:form>

readstudent.jsp:

````````````````

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

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

<h1>Read Students</h1>

<table border="2" width="70%" cellpadding="2">

<tr>

<th>Id</th>

<th>Name</th>

<th>Marks</th>

<th>update</th>

<th>delete</th>

</tr>

<c:forEach var="student" items="${students}">

<tr>

<td>${student.stid}</td>

<td>${student.stname}</td>

<td>${student.stmarks}</td>

<td><a href="/update-student/${student.stid}">Update</a></td>

<td><a href="/delete-student/${student.stid}">Delete</a></td>

</tr>

</c:forEach>

</table>

<br/>

<a href="/create-student">Create Student</a>

updatestudent.jsp:

`````````````````````

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

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

<h1>Update Student</h1>

<form:form method="post" action="/update-student/${id}">

<table>

<tr>

<td>Student Name: </td>

<td><form:input path="stname"/></td>

</tr>

<tr>

<td>Student Marks: </td>

<td><form:input path="stmarks"/></td>

</tr>

<tr>

<td></td>

<td><input type="submit" value="Update" /></td>

</tr>

</table>

</form:form>

pom.xml:

````````

<?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 https://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>2.5.4</version>

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

</parent>

<groupId>com.sopra</groupId>

<artifactId>springboot-mvc-datajpa</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>springboot-mvc-datajpa</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-data-jpa</artifactId>

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>mysql</groupId>

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

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</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>

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

</plugin>

</plugins>

</build>

</project>

DEMOSTRATION

1.SpringBoot Actuators

2.YAML File

3.SpringBoot Hateous (link in the REST crud application)

@GetMapping("/department/{departmentId}")

public EntityModel<Department> fetchById(@PathVariable("departmentId") int departmentId)

{

Department department = service.fetchById(departmentId);

EntityModel<Department> model = EntityModel.of(department);

WebMvcLinkBuilder linkToDepts = linkTo(methodOn(this.getClass()).fetchAllDepartments());

model.add(linkToDepts.withRel("all-depts"));

return model;

}

Git Command

1.git config --global user.email "jainnandini550@gmail.com"

2. git config --global user.password "123N@ndini04"

3. clone the git projects

git clone "https://github.com/jainnandini550/Git-GitHub-Learning.git"

4. Go to the Directory or folder

cd c:\Git-GitHub-Learning

5. made some changes by opening the file

6. write command= git add .

(these files will moved to staging area)

7. git commit -m "initial commit"

8. files will be moved to unmodified area

9. git push -u origin master

**Note:**

''''''''''

Clear command used for clearing the git console.