**JFSD: A-Z of Back-end and Database Development**

**Day 12 : 18 Aug 24**

Session Tracking :

Session is a collection of http request and response within a particular period of time. By default http is known as state less protocol. Default session time for every server is 3 minute.

1. Cookies : Cookies is a small text file created by server when client send first request to server. Cookies file store in client machine. This file contains lot of information as sessionId, expiry, path etc. when 2nd request pass from client by default request + sessionid also pass.

1st req ----------------🡪

2nd req + cookies file information mainly session id

Client Server

🡨----------1st res + cookies

🡨-------2nd res + cookies

Limitation of cookies.

Client can disable cookies option in browser. When we disable cookies server not able to keep the track about the client.

Cookies is not secure.

1. URL-Rewrite : server side programming they use URL-Rewrite technique. In this technique they append session Id through URL

URL?sessionId=%%%%^^^^@@@@XXXX&&&

This technique support for only Get method not for post..

1. API : HttpSession : it is a interface which provides set of methods which help to keep the track about the session.

In Servlet we need to create the reference of HttpSession

HttpSession hs = request.getSession();

In JSP, they already provided implicit object ie session.

We can navigate from one page to another page with help of response.sendRedirect(“path”);

RequestDispatch rd1 = request.getReqeustDispatcher(“path”)

rd1.include(request,reponse); source + target page ie merge the page

rd1.forward(request,response); we can see the output of only target page.

In this request will not destroy. If we store any value is request scope. Those value we can access only when we are moving from one page to another but it maintain same request. Using forward we can navigate within a application.

response.sendRedirect(“path”); we can see the output of only target page.

It generate new request. Using sendRedirect we can navigate to another application.

If we want to share the data between one page to another page and it generating new request then we can take the help session object.

This object also provide same method as request ie setAttribute(),getAttribute(), removeAttribute();

Old request : forward http://localhost:9090/Product\_Management\_System\_App/ProductViewController

New request : sendRedirect

http://localhost:9090/Product\_Management\_System\_App/view\_product.jsp

JSTL : JSP Standard Tab library

JSTL provides lot of pre defined tags which hep to do core functionality, sql functionality and more using tag base.

JSTL use one small EL (Expression Language).

<%

%>

If we want to use jsp in jsp page

We need to take the help of taglib directive tags.

<%@ taglib uri=”” prefix=”” %>

**Course 4 end project**

**Funfit .**

**Course-end Project 1**

**Description**

**Objective**

* **To plan and develop the backend for a given problem statement using Java Enterprise**
* **To configure project using Maven for build and dependency management**
* **To connect Java Code with Database**
* **To use the SQL and NoSQL Databases**

**PreRequisites**

* **Java**
* **Servlets**
* **JSP**
* **Maven**
* **JDBC**
* **MySQL**

**Description**

**This assignment is designed to help gain understanding of how to plan and develop the back end for a given problem. Further, to gain hands-on experience in designing Server Side Controller Structure with Servlets. Implementing the Database connections to perform CRUD operations**

**Problem Statement**

**Yamin is a Gym Instructor. He takes multiple sessions of Zumba during the day for which he has divided his participants into batches running in morning hours and evening hours. In order to manage his participants he needs a software solution to record the details of his participants and batches. Luckily his friend Mike runs a software company. Yamin has requested the company to develop the software solution to manage his participants and batches.**

**Company has to develop a backend solution for the project.**

**Tasks**

**As a task you need to design and develop the front end for the problem statement along with database design:**

1. **Create a Dynamic Web Project in Eclipse and configure it to maven project**
2. **Create a Java Classes**
   * **Create a Participant Class**
   * **Create a Batch Class**
3. **Create Servlets in the Project**
   * **Create a Servlet for Participant**
   * **Create a Servlet for Batch**
4. **Create HTML Pages**
   * **Create a Welcome Page with Navigation Menu**
   * **Create a HTML Page to Add Batch**
   * **Create a HTML Page to Add Participant**
   * **Create a HTML Page to Update Batch**
   * **Create a HTML Page to Update Participant**

* **Perform CRUD Operations using JDBC**
  + **Create DataBase and Table in MySQL**
  + **Configure JDBC Dependencies for MySQL**
  + **Implement a DAO Design Pattern**
  + **Create a Repository Class which uses DAO to perform DB Interactions**
  + **In Participant Servlet implement CRUD Operations using doGet, doPost, doDelete and doPut Http Methods**
  + **In Batch Servlet implement CRUD Operations using doGet, doPost, doDelete and doPut Http Methods**

1. **Create JSP Pages in the Project**
   * **Create a JSP Page to view the list of Participants with delete option**
   * **Create a JSP Page to view the list of Batches with delete option**
   * **Create a JSP Page to view the list of Participants in a Batch using query parameter**

1. **Build and Run the Project on Apache Tomcat web server**
2. **Validate the working of project**
3. **Package the project as a Jar file using Maven Package Goal**

**To download the complete document click here - [](https://lms.simplilearn.com/user/project/download-attachment?file=1688109319_phase_end_project.pdf)**

Steps

1. create dynamic web project with version 4 with tomcat 9.x version. Select option as web.xml file.

2. after project created please convert this project to maven project. In release tag write version of java as 11 and then add two dependencies as mysql connector and jstl tags.

3. create database funfit db and create two tables.

Batch BI(PK) Auto increment, typeofbatch(morning/evening), time (8am,9am,8pm,9pm)

create table batch(bid int primary key auto\_increment, typeofbatch enum('morning','evening'), time varchar(20));

insert into batch(typeofbatch,time) values('morning','9am');

Student Sid(PK) auto increment, ,SName, Age,phone\_number, BI(FK)

create table student(sid int primary key auto\_increment, sname varchar(25), age int, bid int, foreign key(bid) references batch(bid));

create table student(sid int primary key auto\_increment, sname varchar(25), age int, bid int, foreign key(bid) references batch(bid) on delete cascade);

insert into student(sname,age,bid) values('Steven',23,1);

4. create package as com.funfit.bean

a. Batch : bid,typeofbatch,time with data type, setter/getter, constructor, tostring

b. Student: sid, sname,age,bid with data type, setter/getter, constructor and tostring

1. Create package as com.funfit.resource : this class contains getConnection method responsible to provide database connection.
2. Create package as com.funfit.dao

Which contains StudentDao and BatchDao method

Generally it contains method as insert, delete, update and retrieve. If you need any more than you can write it. Jdbc logic.

1. Create package as com.funfit.service :

Which contains StudentService and BatchService methods. If you need any business logic depending requirement that you can write before calling dao method after before passing result to controller.

1. Create package as com.funfit.controller

Inside that package you can create more than one controller to do Batch as well as Student operations.

1. Create more than one view pages using html or jsp

Index.jsp ---🡪

Batch.jsp ---🡪

Add Batch , Delete Batch, Update Batch, View Batch

Student.jsp -🡪

Add Student, Delete Student, Update Student, View Student