PHASE 2

1. JDBC – 4 hours (M)
2. Servlets – 12 hours (T W TH)
3. JSP – 12 hours (F M T)
4. Hibernate – 8 hours (W Th)

DATABASE:

1. LMS   
   sudo mysql -uroot -pSimplilearn
2. Show databases;
3. CREATE DATABASE voda2023;
4. Use voda2023;
5. create TABLE customer(email varchar(80) PRIMARY KEY, phone varchar(15),

password varchar(50), city varchar(50), ismember tinyint);

1. INSERT INTO `customer` VALUES ('sh@g.c','1111111111','sh','Mum',0);
2. INSERT INTO `customer` VALUES ('ab@g.c','2222222222','ab','Pune',1);
3. Select \* from customer;
4. Download driver i.e jar file [JAR -> Java Archive ] type 4 driver  
   <https://mvnrepository.com/artifact/mysql/mysql-connector-java/8.0.28>

Java and Database

1. Download the respective driver depending on the database
2. Add the driver in the project using build path if it is a core java application
3. Load the driver  
   Class.forName(<name of the driver>)
4. Connection to the database , specify the connection parameters  
   DriverManager.getConnection(<url>, <username>, <password>)  
   url : IP address + port number
5. Statement reference
6. Select query => stat.executeQuery()
7. Insert/ update/ delete => stat.executeUpdate()
8. Download tomcat zip from https://tomcat.apache.org/download-90.cgi

File -> New -> Project -> Dynamic Web Project

Add Project name => ServletDemo

Click on => New Runtime => Choose Apache => Tomcat 9.0 => Next

Choose Browse and point the path of tomcat unzipped folder then finish

Then Next innew Project Window => next => Selecyt web.xml and finish

Servlets

1. Servlets are the technology that allows request and response mechanism over HTTP
2. Tomcat server provides capabilities
   1. To handle HTTP protocol
   2. Able to take in the request , remember from where the request was until the response for that request is generated
   3. Stateless protocol
   4. Tomcat is able to take in the request sent by the client [browser], delegate the request to specific servlet based on the url mapping and send the response back to the same client
   5. Can also compile and execute servlets
3. In web applications you will not see main method. Developers do not create servlet objects. This is taken care by the server
4. To create a servlet => extends HttpServlet
   1. doGet -> <a> and form method=get  
      is not secured  
      data is exposed over the url  
      limitations on the amount of data that can be sent over the url
   2. doPost -> form method=post  
      secured and no limitations on the amount of data
   3. response.sendredirect => redirects to another page =>  
      this is a 2 req-response cycles
   4. inter-servlet communication  
      RequestDispatcher => dispatches request within the server

1 req-response cycle and is used for interservlet communication

* + 1. Include => If servlet 1 is calling servlet 2 then response of servlet 1 will be included in servlet 2
    2. forward => If servlet 1 is calling servlet 2 then response of servlet 1 will NOTN be included in servlet 2
    3. request and response objects are carried forward to next servlet. So servlet 2 can access data from servlet 1 request object
    4. if servlet 1 dispatches request from POST method then POST of servlet 2 will be called
  1. Session management
     1. HTTP is a stateless protocol. It remembers request only for 1 request – response cycle
     2. There is a need to maintain the state of the logged in user to process any functionalities related to the user
     3. Ways of doing session management
        1. url rewriting
        2. cookies
        3. hidden form fields
        4. HttpSession – usually http session is preferred as it is secured and data is on the server side

1. Filters
   1. Write the common logic to be processed against set of servlets at 1 location i.e filters/ interceptors/ middlewares
      1. Authenctication
      2. Security
      3. Compression
      4. Logging
      5. Caching
      6. encryption
   2. Filters sit between the request and response cycle
   3. Filters can be configured to listen for certain set of urls . They can intercept the request and based on the BL can either call the filter in chain or the servlet or can redirect the response
   4. They can also intercept the response and modify it if required
   5. Class extends HttpFilter or implements Filter interface
   6. doFilter method is called for every request and response => calling chain.doFilter() is IMP if needs to proceed with the request
2. Listeners
   1. They listen for any event that may happen and handle BL for that event
   2. 3 types of listeners
      1. Session listeners => created, destroyed, attribute added, attribute removed, attribute replaced, migration of servers  
         stay for the time invalide is not called
      2. Request listeners => created, destroyed, attribute added, attribute removed, attribute replaced, migration of servers  
         stay per request and response
      3. Context => created, destroyed, attribute added, attribute removed, attribute replaced, migration of servers  
         They are called as soon as the application is deployed on the server. Any DB connections, any other tech/ server startup/ any caching
   3. Create a class that implements XXXListener and override the different methods for implementation

JSP

1. Add mysql jar file in lib folder of WEB-INF

2. Add database + entity package from ServletDemo

3. Add LoginServlet.java under servlets folder

4. Create login.jsp, register.jsp, dashboard.jsp and update index.jsp

1. JSP comments <%-- --%>
2. JSP scriplets : used to write any valid java code. The code within the scriplets go inside the \_jspService() method  
   <% %>
3. JSP expressions : to send the output on the webpage  
   <%= %>
4. JSP declarations
5. JSP action tags
6. JSP page directives
   1. Page
   2. Taglib
   3. Include
7. JSTL -> core tag library

HIBERNATE

1. Created a core java project
2. Added hibernate and mysql jar files in the project build path
3. Created hibernate.cfg.xml file
   1. Database connection parameters
   2. Dialect
   3. DML =>
      1. Update => will create if table doesn’t exists
      2. Create => will always drop and recreate , you will loose the data
   4. Show\_sql
   5. Mapping of java classes =>
      1. XML => <mapping resource=”book.hbm.xml”></mapping>
      2. or annotation based approach  
         <mapping class=”com.bean.Author”></mapping>
4. Created our POJO’s => mapped with database tables
5. Configured or mapped java classes with database tables as follows:
   1. Book.hbm.xml file => mapping of class with table name and properties with column names
   2. Annotations
      1. @Entity => tells hibernate this class is database managed entity
      2. @Table => tells hibernate the table to map with
      3. @Id => for primary key
      4. @Column => to map with columns in database table