실전 JSP

1. 웹 프로그램 개요

1-1. 웹 프로그램이란?

- 웹 프로그램 : 인터넷 서비스를 이용해서 서로 다른 구성 요소들(PC 등)이 통신할 수 있는 프로그램이다.
- 사용자(브라우저) -> Request -> Web Server -> Response -> 사용자(브라우저)

1-2. 프로토콜(Protocol)과 IP

- 프로토콜 : 통신을 하기위한 규약으로, HTTP, FTP, SMTP, POP 등이 있다.
- 인터넷 객체 -> Request -> Web Server -> Response -> 인터넷 객체
- http://www.google.com:80/index.html 중, http는 protocol, www는 인터넷 서비스 구분, google.com은 도메인, 80은 port, index.html은 경로이다.

1-3. 웹 프로그램의 동작 원리

- User(browser) -> Request(html) -> Web Server -> query -> Database -> result -> Web Server -> Response(html) -> User(browser)
- Web Server : HTML(정적 데이터) , Container(동적 데이터)

2. 개발 환경 설정

2-1. JDK 설치

- http://www.oracle.com/technetwork/java/javase/downloads
- Development Tools -> Next

2-2. Path 설정

- javac.exe, java.exe(JVM 구동 명령 컴파일러)를 다른 디렉터리에서도 실행할 수 있도록 하기 위해 환경 변수(Path)에 bin 경로를 등록한다.
- JAVA_HOME 환경 변수 추가, Path 환경 변수에 JDK의 bin 디렉터리 추가

2-3. 이클립스 다운로드

- IDE(Integrated Development Environment : 통합 개발 환경) 설치 - eclipse

2-4. 웹 컨테이너(Apache Tomcat 8.5) 설치

- http://tomcat.apache.org/
- Eclipse - Window - show View - Other - Server - Servers - Tomcat v8.5 Server - Next - JRE 해당하는 것 선택 - Finish
- Sever Locations : Use Tomcat installation(takes control of Tomcat installation)
- Server Options : Publish module contexts to separate XML files
- Ports : HTTP/1.1 : 8090

3. JSP 맛보기

3-1. 웹 컨테이너 구조

- xxx.jsp -> request -> xxx_jsp.java -> xxx_jsp.class -> xxx_jsp.obj -> response -> HTML
- 웹 컨테이너(tomcat) : xxx_jsp.java, xxx_jsp.class, xxx_jsp.obj

3-2. JSP 파일 작성

- Dynamic Web Project -> Project name -> Finish
- Project 클릭 -> New -> JSP File -> WebContent -> File name -> Finish
- File 작성 -> Run As -> Run on Sever -> Tomcat v8.5 -> Finish

3-3. .java 파일 확인

- xxx.jsp -> xxx_jsp.java -> xxx_jsp.class

4. Servlet 맛보기

4-1. 웹 컨테이너 구조

- Browser -> request -> xxx.java -> xxx.class -> xxx.obj -> response -> Browser
- 웹 컨테이너(tomcat) : xxx.java, xxx.class, xxx.obj

4-2. Servlet 파일 작성

- New -> Dynamic Web Project -> Project name -> Next -> Next -> Content root, Content directory -> Finish
- src -> New -> Servlet -> Java package, Class name, Superclass -> Next -> URL mappings -> Servlet 지우기 -> Next -> Finish

4-3. Class 파일 확인

- browser -> xxx.java -> xxx.class

5. Servlet 맵핑

5-1. Servlet 맵핑이란?

- Browser -> request -> 웹 컨테이너(tomcat)=servlet
- requesr(서블릿 구분 필요) -> 서블릿 구분 방법 ->localhost:8090
- full path : http://localhost:8090/lec05Pjt001/servlet/com.servlet.ServletEx, 보안 취약, 복잡한 URL
- mapping path : http://localhost:8090/lec05Pjt001/SE, 간결한 URL

5-2. web.xml 파일을 이용한 맵핑

- web.xml은 배치 지시자(deployment description)

```
<servlet>
        <servlet-name>servletEx</servlet-name>
        <servlet-class>com.servlet.ServletEx</servlet-class>
</servlet>
<servlet>
        <servlet-name>servletEx</servlet-name>
        <url-pattern>/SE</url-pattern>
</servlet>
```

5-3. Java Annotation을 이용한 맵핑

- @WebServlet("/Hello") -> http://localhost:8090/lec05Pjt001/Hello

6. Servlet request, response

6-1. HttpServlet

- user -> Request -> 웹 컨테이너(tomcat) servlet -> Response -> user
- ServletEx(class) -> HttpServlet(abstract class) -> GenericServlet(avstract class) -> Servlet(interface), ServletConfig(interface), Serializable(interface)
- Servlet extends HttpServlet -> HttpServlet extends javax.servlet.GenericServlet

6-2. HttpServletRequest

- HttpServletRequest : 요청에 대한 정보를 가지고 있는 객체
- user -> Request -> 웹 컨테이너(tomcat) servlet

6-3. HttpServletResponse

- HttpServletResponse : 응답에 대한 정보를 가지고 있는 객체
- user <- Response <- 웹 컨테이너(tomcat) servlet

7. Servlet Life-Cycle

7-1. Servlet 생명주기

- @PostConstruct -> init() -> service -> destory() -> @PreDestory
- Servlet 생성 및 종료 : init(), service, destory()

7-2. 생명주기 관련 메서드

8. form 데이터 처리

8-1. form 태그

- Browser -> Request(DATA) -> 웹 컨테이너(tomcat) servlet
- form 관련 태그(user data) -> request 객체(user data)

8-2. doGet

- form 관련 태그(user data) -> 데이터가 웹 브라우저 URL에 노출되어 웹 서버로 전송(보안에 약하다) -> request 객체(user data)
- method="get" -> doGet(), 사용자 정보가 URL에 노출

8-3. doPost

- form 관련 태그(user data) -> 데이터가 HTTP Requesr에 포함되어 웹 서버로 전송(보안에 강하다) -> request 객체(user data)
- method="post" -> doPost(), 맵핑 정보만 노출

9. JSP 스크립트

9-1. Servlet vs JSP

- Servlet > JAVA 코드, xxx.java -> xxx.class
- JSP > HTML > JAVA 코드, xxx.jsp -> xxx_jsp.java -> xxx_jsp.class

9-2. JSP 파일 HTML5 포맷 설정

- Window -> Preferences -> Templates -> New -> html5 format, New JSP, Automatically insert

```
<%@ page language="java" contentType="text/html; charset=${encoding}"
pageEncoding="${encoding}"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="${encoding}">
<title>Insert title here</title>
</head>
<body>
</body>
</html>
```

9-3. JSP 주요 스크립트

- 선언 태그 : JSP 페이지에서 Java의 멤버변수 또는 메서드를 선언

```
<%!
   int num = 10;        // 변수 선언 및 초기화
   String str = "jsp";
   ArrayList<String> list = new ArrayList<String>();   // 객체 생성 및 인스턴스 초기화

   public void jspMethod() {     // 메서드 선언 및 정의
     System.out.println(" --jspMethod() --");
   }
%>
```

- 주석 태그 : jsp 주석은 jsp 파일이 서블릿 파일로 변환될 때 제외된다.

```
<!-- 주석 태그 --> : HTML 주석
<%-- Hello JSP World!! --%> : JSP 주석
```

- 스크립트릿 태그 : JSP 페이지에서 Java 코드를 넣기 위한 태그

```
<%
   if(num > 0) {         // Java 코드
%>
   <p> num > 0 </p>   // HTML 코드
<%
   } else {              // Java 코드
%>
   <p> num <= 0 </p>  // HTML 코드
<%
   }                     // Java 코드
%>
```

- 표현식 태그 : Java의 변수 및 메서드의 반환값을 출력하는 태그

num is <%=num%>, num is는 HTML 코드 <%=%> 안의 num은 Java 코드

- 지시어 : 서버에서 jsp 페이지를 처리하는 방법에 대한 정의

1) page : 페이지 기본 설정 -> <%page 속성="속성값">
2) include : include file 설정 -> <%include file="파일명">
3) taglib : 외부 라이브러리 태그 설정 -> <%taglib uri="uri" prefix="네임스페이스명">

10. JSP request, response

10-1. request 객체

- user -> Request -> 웹 서버(jsp)

```
<form action="mSignUp.jsp" method="get">
   user data
   <input type="submit" value="sign up">
</form>
```

-> Request

```
<%
   m_name = request.getParameter("m_name");
   m_pass = request.getParameter("m_pass");
   m_gender = request.getParameter("m_gender");
   m_hobby = request.getParameter("m_hobby");
```

```
    m_residence = request.getParameter("m_residence");
%>
```

10-2. response 객체

- user <- Response <- 웹 서버(jsp)

```
<body>
    First Page </br>
    <%
     response.sendRedirect("secondPage.jsp");
    %>
</body>
```

-> Response

```
<body>
    Second Page
</body>
```

11. JSP 내장 객체

11-1. config 객체

- web.xml(init param) -> getInitParameter() -> JSP(init param)

11-2. application 객체

- web.xml(context param) -> getInitParameter() -> JSP(context param)
- context -> setAttribute(), getAttribute() -> JSP

11-3. out 객체

- JSP > HTML

```
<%
    out.print("<h1>Hello JAVA World!!</h1>");
    out.print("<h2>Hello JSP World!!</h2>");
    out.print("<h3>Hello Servlet World!!</h3>");
%>
```

11-4. exception 객체

- JSP(error 발생) -> exception 객체

```
<!--exception 객체-->
<%
    out.print(str.toString());
%>
```

->

```
<%
    response.setStatus(200);
    String msg = exception.getMessage();
%>
<h1>error message : <%=msg %></h1>
```

12. Servlet 데이터 공유

12-1. servlet parameter

- web.xml(init param) -> getInitParameter() -> servlet(init param)

```
<init-param>
   <param-name>adminId</param-name>
   <param-value>admin</param-value>
</init-param>
<init-param>
   <param-name>adminPw</param-name>
   <param-value>1234</param-value>
</init-param>
 ->
String adminId= getServletConfig().getInitParameter("adminId");
String adminPw= getServletConfig().getInitParameter("adminPw");
```

12-2. context parameter

- web.xml(context param) -> getInitParameter() -> servlet(context param)

```
<context-param>
   <param-name>imgDir</param-name>
   <param-value>/upload/img</param-value>
</context-param>
<context-param>
   <param-name>testServerIP</param-name>
   <param-value>127.0.0.1</param-value>
</context-param>
 ->
String imgDir= getServletContext().getInitParameter("imgDir");
String testServerIP= getServletContext().getInitParameter("testServerIP");
```

12-3. context attribute

- context -> setAttribute(), getAttribute() -> servlet

13. Cookie

13-1. Cookie란?

- 브라우저(클라이언트) -> request -> 서버
- 서버 -> response, 연결 해제 -> 브라우저(클라이언트)
-> 브라우저(클라이언트) 기존 연결정보 저장(Cookie) -> request -> 서버
- 서버-> response, 연결 해제 -> 브라우저(클라이언트) 기존 연결정보 저장(Cookie)

13-2. Cookie 구현

- cookie == null -> false -> cookie 재활용 -> redirect : loginOk.jsp
- cookie == null -> true -> cookie 생성 -> redirect : loginOk.jsp

14. Session

14-1. Session이란?

- 브라우저(클라이언트) -> request -> 서버
- 서버 -> response, 연결 해제 -> 브라우저클라이언트)
-> 브라우저(클라이언트) -> request -> 서버 기존 연결정보 저장(Session)

- 서버 기존 연결정보 저장(Session) -> response, 연결 해제 -> 브라우저(클라이언트)

14-2. Session 구현

- session == null -> false -> 로그인 정보 출력 -> session = request.getSession(); out.print("memberId : "+ session.getAttribute("memberId") + "</br>"); 또는 redirect : loginOk.jsp -> 로그아웃 -> session.invalidate();
- session == null -> true -> 로그인 유도 -> HttpSession session= request.getSession(); session.setAttribute("memberId", mId); response.sendRedirect("loginOk.jsp"); 또는 redirect : loginOk.jsp -> 로그아웃 -> session.invalidate();

15. 한글 처리

15-1. 한글 처리

- 브라우저 -> post : 서블릿에 request.setCharacterEncoding("UTF-8"); / get : server.xml에 <Connector URIEncoding="UTF-8"/> 추가 -> 웹 서버

15-2. Filter

- 브라우저 -> request Filter -> 웹 서버
- 웹 서버 -> Filter response -> 브라우저
- UserFilter(class) -> Filter(interface)
- UserFilter(class) ->

```
publicclassTempFilter implementsFilter {

    @Override
    publicvoidinit(FilterConfig arg0) throwsServletException {}

    @Override
    publicvoiddoFilter(ServletRequest req, ServletResponse res, FilterChain chain)
 throwsIOException, ServletException {}

    @Override
    publicvoiddestroy() {}
}
```

16. 오라클 설치

16-1. 오라클 다운로드

- http://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html

16-2. 오라클 설치

- OracleWE_112_Win64 -> setup.exe -> Next -> I accept the in the license agreement -> Next

16-3. 계정 생성

- cmd 창에서 i>sqlplus -> Enter user-name : system, Enter password: 설정한 비밀번호 -> SQL> create user scott identified by tiger; -> User created -> SQL> grant connect, resource to scott; -> Grant succeeded -> SQL>exit

16-4. SQL developer

- http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html
- 처음 SQL developer를 실행하면 Java 경로를 설정해야 한다. 컴퓨터에 설치되어 있는 Java 경로를 설정한다. -> SQL developer가 실행된다. -> SQL developer가 실행되면 새로운 접속을 하기 위해서 왼쪽 접속 메뉴에서 오른쪽 마우스 버튼을 클릭해서 '새 접속' 메뉴를 클릭한다. -> 접속 이름, 사용자 이름, 비밀번호는 미리 생성한 계정을 넣는다.

17. SQL

17-1. 테이블 생성 및 삭제

- --table 생성

```
CREATE TABLEbook (
book_id NUMBER(4),
book_name VARCHAR2(20),
book_loc VARCHAR2(20)
);
--테이블검색
SELECT* FROMtab;
--테이블삭제
DROP TABLEbook;
--table 생성-제약조건
CREATE TABLEbook (
book_id NUMBER(4) CONSTRAINTbook_id_pk PRIMARY KEY,
book_name VARCHAR2(20),
book_loc
```

17-2. 테이블 추가, 수정, 삭제

- --시퀀스생성

```
CREATE SEQUENCEbook_seq;
--데이터추가
INSERT INTO
book(book_id, book_name, book_loc)
VALUES
(BOOK_SEQ.NEXTVAL, 'book5', '001-00005');
INSERT INTO
book
VALUES
(BOOK_SEQ.NEXTVAL, 'book6', '001-00006');
--수정
UPDATEbook SETbook_loc = '001-00006123'
WHEREbook_name = 'book6';
--삭제
DELETE FROM book
WHEREbook_id = 6;
```

17-3. 테이블 검색

- --데이터검색

```
SELECT* FROMbook;
SELECTbook_name, book_loc FROMbook;
SELECTbook_name AS책이름, book_loc AS책위치FROMbook;
--조건-WHERE
SELECT* FROMbook WHEREbook_id > 3;
SELECT* FROMbook WHEREbook_id > 3 ANDbook_id <= 5;
--조건-BETWEEN AND
SELECT* FROMbook WHEREbook_id BETWEEN2 AND4;
--조건-LIKE
SELECT* FROMbook WHEREbook_id LIKE3;
SELECT* FROMbook WHEREbook_loc LIKE'%3';
SELECT* FROMbook WHEREbook_name LIKE'book%';
SELECT* FROMbook WHEREbook_name LIKE'%ok%';
--정렬
SELECT* FROMbook ORDER BY book_id ASC;
SELECT* FROMbook ORDER BY book_id DESC;
SELECT* FROMbook ORDER BY book_name ASC;
SELECT* FROMbook ORDER BY book_name DESC;
```

18. JDBC

18-1. JDBC 설정

- Java와 DB가 통실할 수 있게 해주는 API
- eclipse에서도 Oracle의 API를 이용하기 위해서, Oracle LIB를 eclipse에 복사한다.
- Windows - Preference -> Java - ClasspathVariables - JRE_LIB
- C:\oraclexe\app\oracle\product\11.2.0\server\jdbc\lib -> copy : objdbc6_g.jar -> C:\Program Files\Java\jre1.8.0_151\lib\ext

18-2. JDBC를 이용한 데이터 관리

- JDBC 실행 순서 : OracleDriver 로딩(Driver loading) -> Java와 Oracle 연결(Connection) -> query 전송 객체(Statement) -> query 작성(query) -> query 전송(run)

18-3. PreparedStatement

- JDBC 실행 순서 : Driver loading -> Connection -> Statement -> query -> run
- Driver loading -> Connetcion -> query -> PreparedStatement -> run

19. DAO와 DTO

19-1. DAO, DTO란?

- 브라우저 -> request -> Servlet -> DAO -> DB -> DTO -> DAO -> Servlet -> response -> 브라우저
- 웹 서버 : Servlet, DAO
- DAO : Data Access Object, DTO : Data Transfer Object

19-2. DAO, DTO 구현

- 브라우저 <-> BookServlet <-> DB

-> 브라우저 <-> BookServlet <-> BookDAO <-> DTO -> DB

20. Connection Pool

20-1. 커넥션 풀이란?

- 브라우저 -> request -> 웹 서버 -> DB Access(DB Connection, Data handing, DB connection Close) -> DB

-> 브라우저 -> request -> 웹 서버 -> DB Access(<- rent <- Connection Pool) -> DB

20-2. 커넥션 풀 설정

- context.xml에서

```
<Resource
   auth="container"
   driverClassName="oracle.jdbc.driver.OracleDriver"
   url="jdbc:oracld:thin:@localhost:1521:xe"
   username="scott"
   password="tiger"
   name="jdbc/Oracle11g"
   type="javax.sql.DataSource"
   maxActive="4"
   masWait="10000" />
```

20-3. 커넥션 풀 구현

- DataSource dataSource;
- Context context = new InitialContext();

dataSource = (DataSource)context.lookup("java:comp/env/jdbc/Oracle11g");

- // con = DriverManager.getConnection(url, id, pw);

con = dataSource.getConnection();