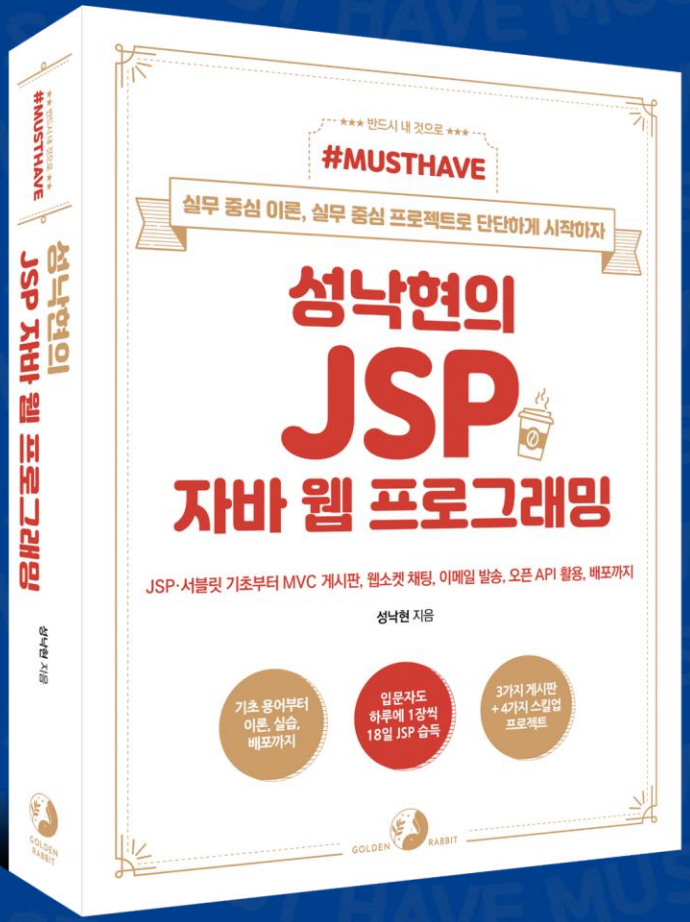


Chapter

06

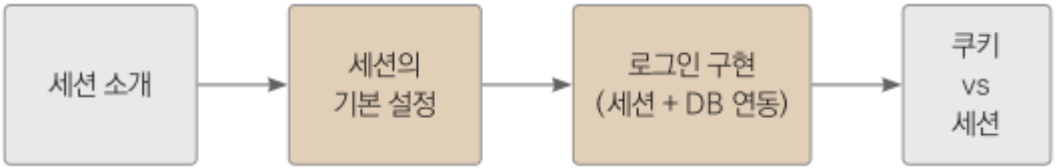
세션(Session)

학습 목표 및 순서

▣ 학습 목표

- 클라이언트의 상태 정보를 서버 측에 저장할 수 있는 세션에 대해 학습합니다
- 기본 사용법을 먼저 알아본 후 session 영역을 이용해 로그인 기능을 구현해봅니다.

▣ 학습 순서



▣ 활용 사례

- 상태 정보를 서버측에 저장하는 세션으로 로그인 정보 유지를 할 수 있습니다.

6.1 세션이란?

▣ 세션이란..??

- 클라이언트가 웹 브라우저를 통해 서버에 접속한 후
- 용무를 처리하고
- 웹 브라우저를 닫아 서버와의 접속을 종료하는 하나의 단위를 뜻함
- 세션은 클라이언트가 서버에 접속해 있는 동안 그 상태를 유지하는 것이 목적

6.2 세션 설정, 확인, 삭제

▣ 세션의 주요 메서드

메서드명	설 명
void setMaxInactiveInterval(int seconds)	세션 유지시간을 초 단위로 설정 web.xml에서 설정할 경우 <session-config>로 분 단위 설정
int getMaxInactiveInterval()	세션 유지시간을 초 단위로 출력
long getCreationTime()	세션의 최초 요청 시간
long getLastAccessedTime()	세션의 마지막 요청 시간
String getId()	웹 브라우저가 생성한 세션 ID 반환

▣ 유지 시간 설정

예제 6-1] WEB-INF/web.xml

```
<session-config>
    <session-timeout>20</session-timeout>
</session-config>
```

6.2 세션 설정, 확인, 삭제

▣ 설정값 확인

예제 6-2] 06Session/SessionMain.jsp

```
<%
SimpleDateFormat dateFormat = new SimpleDateFormat("HH:mm:ss"); // 날짜 표시 형식 ❶

long creationTime = session.getCreationTime();  // 최초 요청(세션 생성) 시각
String creationTimeStr = dateFormat.format(new Date(creationTime));

long lastTime = session.getLastAccessedTime();  // 마지막 요청 시각
String lastTimeStr = dateFormat.format(new Date(lastTime));
%>                                                                    ❷
    <ul>
        <li>세션 유지 시간 : <%= session.getMaxInactiveInterval() %></li>  ❸
        <li>세션 아이디 : <%= session.getId() %></li>  ❹
        <li>최초 요청 시각 : <%= creationTimeStr %></li>
        <li>마지막 요청 시각 : <%= lastTimeStr %></li>                  ❺
    </ul>
```

Session 설정 확인

- 세션 유지 시간 : 1200
- 세션 아이디 : C8DB3DFDBE43D9372BBF066085F44732
- 최초 요청 시각 : 18:58:23
- 마지막 요청 시각 : 18:58:23

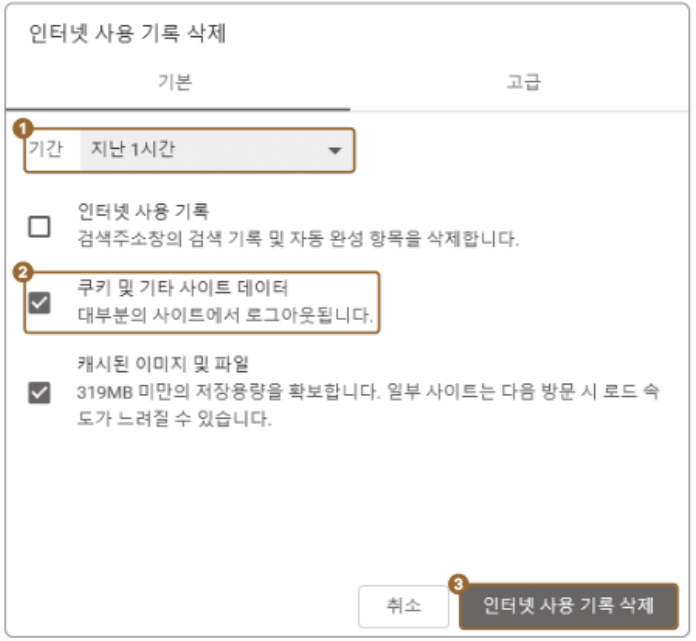
Session 설정 확인

- 세션 유지 시간 : 1200
- 세션 아이디 : C8DB3DFDBE43D9372BBF066085F44732
- 최초 요청 시각 : 18:58:23
- 마지막 요청 시각 : 18:59:40

6.2 세션 설정, 확인, 삭제

▣ 세션 삭제

- 웹 브라우저를 닫으면 세션은 삭제됨
- 웹 브라우저 ⇒ 설정 ⇒ [개인정보 및 보안] 영역에서 삭제해도 됨



- 톰켓은 세션아이디를 값으로 갖는 JSESSIONID 라는 쿠키를 통해 세션 영역에 상태를 유지해야 하는 값들을 저장
- 따라서 쿠키 삭제를 통해 세션을 지울 수 있음

6.3 세션과 DB를 이용한 로그인 구현

▣ 로그인 페이지 작성

예제 6-3] 06Session/LoginForm.jsp

```
<%
if (session.getAttribute("UserId") == null) {  // 로그인 상태 확인 ❷
    // 로그아웃 상태
%>
<form action="LoginProcess.jsp" method="post" name="loginFrm"
    onsubmit="return validateForm(this);">  ❺
    아이디 : <input type="text" name="user_id" /><br />
    패스워드 : <input type="password" name="user_pw" /><br />
    <input type="submit" value="로그인하기" />
</form>
<%
} else {  // 로그인된 상태
%>
    <%= session.getAttribute("UserName") %> 회원님, 로그인하셨습니다.<br />
    <a href="Logout.jsp">[로그아웃]</a>
<%
}
%>
```

❷ session 영역에 저장된 UserId라는 속성이 null이면 로그아웃 상태

❹ 따라서 로그인 폼을 출력

❻ 반대의 경우는 로그인 상태이므로 환영 메세지와 로그아웃 버튼 출력

▶첫 실행시 화면

6.3 세션과 DB를 이용한 로그인 구현

▣ DB 연동 - DTO생성

예제 6-4] Java Resources/membership/MemberDTO.java

```
package membership;

public class MemberDTO {
    // 멤버 변수 선언
    private String id;
    private String pass;
    private String name;
    private String regidate;
```

member 테이블의 컬럼과 동일하게 멤버변수 선언
getter / setter 는 이클립스의 자동 생성 메뉴 사용

DTO 란..??

- DTO(Data Transfer Object)는 계층 사이에서 데이터를 교환하기 위해 생성하는 객체
- 속성(멤버변수)과 그 속성에 접근하기 위한 게터/세터 메서드만 갖춘 게 특징
- 다른 표현으로 값 객체(Value Object, VO)라고도 함

6.3 세션과 DB를 이용한 로그인 구현

▣ DB 연동 - DAO생성

DAO 란..??

- DAO(Data Access Object)는 데이터베이스의 데이터에 접근하기 위한 객체
- 보통 JDBC를 통해 구현하며, 하나의 테이블에서 수행할 수 있는 CRUD를 전담
- CRUD란 Create(생성), Read(읽기), Update(갱신), Delete(삭제) 작업을 말함

예제 6-5] Java Resources/membership/MemberDAO.java

```
public class MemberDAO extends JDBConnect {  ❶
    // 명시한 데이터베이스로의 연결이 완료된 MemberDAO 객체를 생성합니다.
    public MemberDAO(String drv, String url, String id, String pw) {
        super(drv, url, id, pw);  ❷
    }
```

❶ JDBConnect 클래스를 상속

❷ 부모 생성자 호출을 통해 DB 연결(접속)

6.3 세션과 DB를 이용한 로그인 구현

▣ DB 연동 - DAO생성(계속)

```
// 명시한 아이디/패스워드와 일치하는 회원 정보를 반환합니다.
public MemberDTO getMemberDTO(String uid, String upass) {  ❸
    MemberDTO dto = new MemberDTO();   // 회원 정보 DTO 객체 생성
    String query = "SELECT * FROM member WHERE id=? AND pass=?";  ❹
                                       // 쿼리문 템플릿

    try {  ❺
        // 쿼리 실행
        psmt = con.prepareStatement(query); // 동적 쿼리문 준비 ❻
        psmt.setString(1, uid);    // 쿼리문의 첫 번째 인파라미터에 값 설정
        psmt.setString(2, upass);  // 쿼리문의 두 번째 인파라미터에 값 설정  ❼
        rs = psmt.executeQuery();  // 쿼리문 실행 ❽

        // 결과 처리
        if (rs.next()) {  ❾
            // 쿼리 결과로 얻은 회원 정보를 DTO 객체에 저장
            dto.setId(rs.getString("id"));
            dto.setPass(rs.getString("pass"));
            dto.setName(rs.getString(3));
            dto.setRegidate(rs.getString(4));  ❿
        }
    }
```

❸ 아이디와 패스워드를 매개변수로 정의
❹ 인파라미터가 있는 select 쿼리문 작성
❻~❽ 인파라미터 값 설정 및 쿼리 실행
❾ 쿼리 결과를 DTO에 저장

6.3 세션과 DB를 이용한 로그인 구현

▣ 로그인 처리 JSP 구현

예제 6-6] 06Session/LoginProcess.jsp

```
<%
// 로그인 폼으로부터 받은 아이디와 패스워드 ❶
String userId = request.getParameter("user_id");
String userPwd = request.getParameter("user_pw");

// web.xml에서 가져온 데이터베이스 연결 정보 ❷
String oracleDriver = application.getInitParameter("OracleDriver");
String oracleURL = application.getInitParameter("OracleURL");
String oracleId = application.getInitParameter("OracleId");
String oraclePwd = application.getInitParameter("OraclePwd");

// 회원 테이블 DAO를 통해 회원 정보 DTO 획득
MemberDAO dao = new MemberDAO(oracleDriver, oracleURL, oracleId, oraclePwd);  ❸
MemberDTO memberDTO = dao.getMemberDTO(userId, userPwd);  ❹
dao.close();  ❺
```

6.3 세션과 DB를 이용한 로그인 구현

▣ 로그인 처리 JSP 구현(계속)

```
// 로그인 성공 여부에 따른 처리
if (memberDTO.getId() != null) {  ❻
    // 로그인 성공
    session.setAttribute("UserId", memberDTO.getId());
    session.setAttribute("UserName", memberDTO.getName());  ❼
    response.sendRedirect("LoginForm.jsp");  ❽
}
else {
    // 로그인 실패
    request.setAttribute("LoginErrMsg", "로그인 오류입니다.");  ❾
    request.getRequestDispatcher("LoginForm.jsp").forward(request, response);  ❿
}
%>
```

❻~❼ 로그인 성공시 세션 영역에 아이디, 이름을 저장
❽ 로그인 페이지로 “이동”
❾ 로그인 실패시 리퀘스트 영역에 에러메세지 저장
❿ 로그인 페이지로 “포워드”

6.3 세션과 DB를 이용한 로그인 구현

▣ 로그아웃 처리

- 로그아웃은 session 영역에 저장된 로그인 관련 속성을 삭제하면 됨
- 방법1 : 속성명을 명시한 후 삭제
- 방법2 : 모든 속성을 한꺼번에 삭제

예제 6-6] 06Session/LoginProcess.jsp

```
<%
// 방법 1: 회원인증정보 속성 삭제 ❶
session.removeAttribute("UserId");
session.removeAttribute("UserName");

// 방법 2: 모든 속성 한꺼번에 삭제 ❷
session.invalidate();

// 속성 삭제 후 페이지 이동 ❸
response.sendRedirect("LoginForm.jsp");
%>
```

6.4 쿠키 vs. 세션

▣ 쿠키 vs. 세션

- 로그인은 쿠키보다는 세션을 이용해 구현하는 것이 좋음

	쿠키	세션
저장 위치/형식	클라이언트 PC에 text로 저장됩니다.	웹 서버에 Object 타입으로 저장됩니다.
보안	클라이언트에 저장되므로 보안에 취약합니다.	서버에 저장되므로 보안에 안전합니다.
자원/속도	서버 자원을 사용하지 않으므로 세션보다 빠릅니다.	서버 자원을 사용하므로 쿠키보다 느립니다.
용량	용량의 제한이 있습니다.	서버가 허용하는 한 제한이 없습니다.
유지 시간	쿠키 생성 시 설정합니다. 단, 설정된 시간이 경과되면 무조건 삭제됩니다.	서버의 web.xml에서 설정합니다. 설정된 시간 내라도 동작이 있다면 삭제되지 않고 유지됩니다.

학습 마무리

▣ 핵심요약

- 클라이언트의 상태 정보를 저장하는 방법에는 쿠키와 세션이 있음
- 쿠키
  - 상태 정보를 클라이언트 PC에 저장
  - 보안 측면에서 크게 중요하지 않은 정보 저장시 활용
- 세션
  - 상태 정보를 서버에 저장
  - 보안이 중요한 정보 저장시 활용
- 세션은 클라이언트가 웹 브라우저를 통해 접속 후 종료시점까지의 단위를 뜻함
- 유지 시간 설정은 web.xml을 이용
- 유지 시간동안 아무런 동작이 없다면 소멸되지만, 동작이 있다면 유지됨