

Chapter

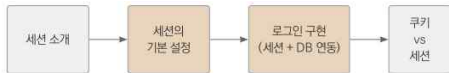
06

세션(Session)

## ■ 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.

### ■ 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

### ■ 세션의 주요 메서드

| 메서드명                                     | 설 명   |
|--|---|
| void setMaxInactiveInterval(int seconds) | 세션 유지시간을 초 단위로 설정<br>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>
```

### ■ 세션의 동작 방식

- 웹 브라우저가 처음 접속하면 서버(톰캣)는 세션을 새로 생성
- 응답 시 세션 아이디를 값으로 갖는 JSESSIONID 쿠키를 응답 헤더에 담아 웹 브라우저로 보냄
- 웹 브라우저는 재 요청 시마다 JSESSIONID를 요청 헤더에 추가하여 보냄
- 서버는 요청 헤더에 포함된 JSESSIONID로 해당 요청이 기존 세션에서 이어진 것임을 확인

#### 쿠키값 확인하기(쿠키가 생성된 이후의 페이지)

쿠키명 : JSESSIONID - 쿠키값 : A47F68902D1037537C4AD26F49378AA0

쿠키명 : myCookie - 쿠키값 : 쿠키맛나요

## ■ 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));

%>
<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 : C8DB3DFD8E43D9372BBF066085F44732
- 최초 요청 시각 : 18:58:23
- 마지막 요청 시각 : 18:58:23

#### Session 설정 확인

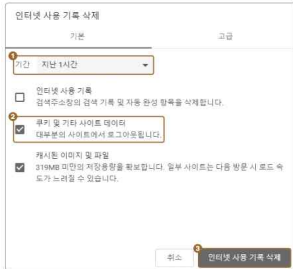
- 세션 유지 시간 : 1200
- 세션 아이디 : C8DB3DFD8E43D9372BBF066085F44732
- 최초 요청 시각 : 18:58:23
- 마지막 요청 시각 : 18:59:40

### ■ 세션 삭제

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



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



## 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);"> ❸
        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
- ❸ 따라서 로그인 폼을 출력
- ❹ 반대의 경우는 로그인 상태이므로 환영 메시지와 로그아웃 버튼 출력

#### ▶ 첫 실행시 화면

##### 로그인 페이지

아이디 :

패스워드 :



## ■ DB연동을 통한 로그인 처리

### . DTO

- . Data Transfer Object
- . 계층 사이에서 데이터를 교환하기 위해 생성하는 객체
- . 별 다른 로직 없이 속성(멤버 변수)과 그 속성에 접근하기 위한 게터/세터 메서드로만 구성
- . 값만 가진 객체라 하여 VO(Value Object)라고 하기도 함

### . DAO

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

### ■ DB 연동 - DTO생성

```
package membership;  
  
public class MemberDTO {  
    private String id;  
    private String pass;  
    private String name;  
    private String regdate; ← 멤버변수 선언  
  
    public String getId() { ← getter() 선언  
        return id;  
    }  
    public void setId(String id) { ← setter() 선언  
        this.id = id;  
    }  
    ..... 동일한 패턴이므로 하위 생략
```

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

## ■ DB 연동 - DAO생성

```
public class MemberDAO extends JDBCConnect { 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 JDBCConnect 클래스를 상속
- ❷ 부모 생성자 호출을 통해 DB 연결(접속)

## ■ DB 연동 - DAO생성(계속)

// 명시한 아이디/패스워드와 일치하는 회원 정보를 반환합니다.

```
public MemberDTO getMemberDTO(String uid, String upass) { ③
```

```
    MemberDTO dto = new MemberDTO(); // 회원 정보 DTO 객체 생성
```

```
    String query = "SELECT * FROM member WHERE id=? AND pass=?";
```

```
        // 쿼리문 템플릿
```

```
try { ⑤
```

```
    // 쿼리 실행
```

```
    pstmt = con.prepareStatement(query); // 동적 쿼리문 준비 ⑥
```

```
    pstmt.setString(1, uid); // 쿼리문의 첫 번째 인파라미터에 값 설정
```

```
    pstmt.setString(2, upass); // 쿼리문의 두 번째 인파라미터에 값 설정
```

```
    rs = pstmt.executeQuery(); // 쿼리문 실행 ⑦
```

```
// 결과 처리
```

```
if (rs.next()) { ⑧
```

```
    // 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));
```

```
}
```

```
}
```

③ 아이디와 패스워드를 매개변수로 정의

④ 인파라미터가 있는 select 쿼리문 작성

⑥~⑧ 인파라미터 값 설정 및 쿼리 실행

⑨ 쿼리 결과를 DTO에 저장

### ■ 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
```

## ■ 로그인 처리 JSP 구현(계속)

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

### ■ 로그아웃 처리

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

```
<%  
session.removeAttribute("UserId");  
session.removeAttribute("UserName"); ← 방법: 세션 영역의 속성 삭제  
  
session.invalidate(); ← 방법2: 세션 영역의 모든 속성 삭제  
  
response.sendRedirect("LoginForm.jsp"); ← 로그인 페이지로 이동  
%>
```

### ■ 쿠키 vs. 세션

- 로그인은 쿠키보다는 세션을 이용해 구현하는 것이 좋음

|          | 쿠키                                  | 세션  |
|----------|-------------------------------------|---|
| 저장 위치/형식 | 클라이언트 PC에 text로 저장.                 | 웹 서버에 Object 타입으로 저장.                             |
| 보안       | 클라이언트에 저장되므로 보안에 취약                 | 서버에 저장되므로 보안에 안전함.                                |
| 자원/속도    | 서버 자원을 사용하지 않으므로 세션보다 빠름.           | 서버 자원을 사용하므로 쿠키보다 느림                              |
| 용량       | 용량의 제한이 있음.                         | 서버가 허용하는 한 제한이 없음.                                |
| 유지 시간    | 쿠키 생성 시 설정. 단, 설정된 시간이 경과되면 무조건 삭제. | 서버의 web.xml에서 설정. 설정된 시간 내라도 동작이 있다면 삭제되지 않고 유지됨. |



## ■ 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

