Session & Cookie

# 수업차시	3

01. Session & Cookie

01-01. Session & Cookie 개요

01-01-01. HTTP 통신

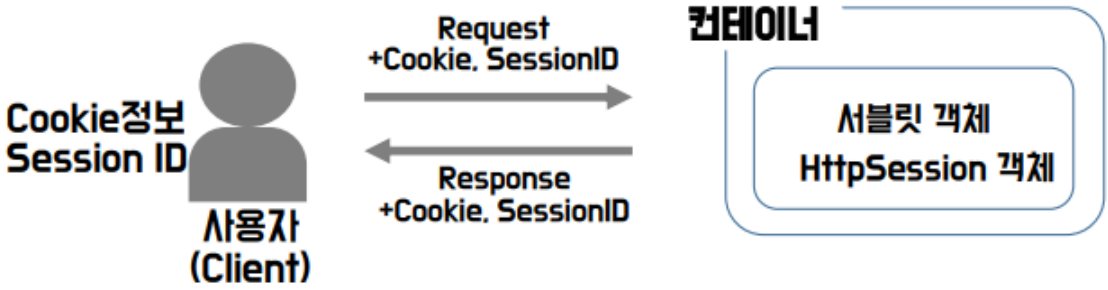
- HTTP 통신 방법
  - HTTP는 서버와 클라이언트 간의 요청과 응답으로 데이터를 주고 받는 형식이다.
  - 서버는 클라이언트의 요청에 응답을 완료하면 연결을 끊는다. (= stateless)
  - 따라서 클라이언트는 서버에 또 다른 요청을 하려면 새로 연결하여 응답을 받아야 한다.
- HTTP 통신의 특징
  1. 무 연결
     a. 클라이언트가 서버로 요청할 때, 요청과 응답이 존재하는데 계속 연결된 상태가 아니다.
     b. HTTP는 요청할 때 잠깐, 응답할 때 잠깐만 연결하는 **무 연결**의 특징을 가진다.
     c. 기본적으로 HTTP는 TCP 프로토콜(= 연결지향 프로토콜로 요청 시 서버가 수락해야 송수신 가능)에 특화 기능을 추가하여 사용하는 것이다.
  2. 무 상태
     a. 서블릿 컨테이너 내에는 여러 개의 서블릿이 있는데, 각각의 서블릿에서 상태값(속성,변수값)을 다른 서블릿에서 공유해 쓸 수 없음을 의미한다.
- HTTP 통신의 문제점
  - 연결이 끊어지면서 유지되어야 하는 정보가 사라지는 문제가 발생한다.

    (예시: 로그인 후의 로그인 계정 정보, 장바구니에 넣은 데이터 등)

01-01-02. Session과 Cookie

- HTTP 통신의 문제점을 보완하고자 연결이 끊어진 후에도 클라이언트에 대한 정보를 유지하기 위한 두 가지 방법이 있다.

⇒ 서버 측에 데이터를 보관하는 Session과 클라이언트 측에 데이터를 보관하는 Cookie이다.



01-02. Cookie

01-02-01. Cookie란

- 클라이언트 측, 즉 사용자 컴퓨터에 데이터를 텍스트 파일 형태로 저장하는 기술로 필요 시에 저장한 정보를 서버와 공유하여 정보를 유지하는 것이다.
  - 패키지 : javax.servlet.http.Cookie
  - 서블릿에서 쿠키 정보를 설정해 주면, 쿠키를 클라이언트가 가지고 있으면서 다른 서블릿에 요청할 때 담아 보낸다.
- 데이터는 Map형식으로 저장되고 데이터의 크기, 개수에 제한이 있다.
- Cookie 유지 시간, 유효 디렉토리, 유효 도메인 등을 설정할 수 있다.

  ⇒ 즉, 호출(조회)한 데이터를 클라이언트 PC에 가지고 있도록 하는 것으로 Cookie에 대한 보관 정보는 클라이언트가 주축이다.
- Cookie는 간단하게 이용할 수 있다는 장점이 있으나, 공용PC를 사용하거나 url에 일부 데이터를 포함하는 경우 보안에 취약하다.

01-02-02. Cookie 속성 설정

- name = value
  - ASCII 문자만 사용 가능하며, 한번 설정된 쿠키의 name은 수정할 수 없다.
  - 쿠키의 이름에는 공백문자와 일부 특수문자(**( ) = , " \ ?** @ **: ;**)를 사용할 수 없다.
- expire = ‘날짜’
  - Cookie의 유지 시간으로, 설정이 따로 없으면 브라우저 동작 동안 유지한다.
- path = ‘경로’
  - Cookie가 전달되는 서버의 유효 디렉토리를 지정하는 속성이다.
- domain = ‘서버정보’

    - Cookie가 전달되는 유효 서버를 설정한다.
- secure
    - https나 ssl보안 프로토콜로 요청할 때만 서버에 전송한다.

01-02-03. Cookie 설정 및 전송

1. Cookie 객체를 생성한다.
    - 이때 패키지를 import 해야 한다.

```
Cookie 쿠키명 = new Cookie('name','value');
```

2. 생성된 Cookie의 설정값을 지정한다.

메소드	내용
setMaxAge(int expiry)	유효 시간 설정
setPath(String uri)	경로 설정 (서버의 모든 요청이 아닌, 특정 경로를 통한 요청으로 Cookie를 사용하는 경우)
setDomain(String domain)	Cookie 도메인 설정 및 Cookie생성 (도메인 외의 도메인 설정 시 사용)

3. 응답 헤더에 쿠키를 담는다.

```
response.addCookie(Cookie cookie)
```

4. 응답을 보낸다.

01-02-04. 전송한 Cookie 활용

1. 전달받은 Cookie 목록을 HttpServletRequest객체를 이용해 배열 형태로 읽어온다.
    - request.getCookies() : Cookie 객체 배열 형태로 반환

```
Cookie[] list = HttpRequest.getCookies();
```

2. Cookie의 getName()와 getValue()를 이용해 담긴 값을 호출한다.

```
for(Cookie c : list) {
  System.out.println(c.getName());
  System.out.println(c.getValue());
}
```

- Cookie에 저장된 정보를 가지고 로그인 확인 및 팝업 설정 등을 할 수 있다.

01-02-05. Cookie 확인 방법

- IE (Internet Explorer)
  - 브라우저 확인
    - 개발자도구(F12) → network → 페이지 선택 → 우측 화면 Cookie
  - 저장 경로 : C:\Users*user*\AppData\Local\Microsoft\Windows\INetCache
- Chrome
  - 브라우저 확인
    - 개발자도구(F12) → Application → Cookies에서 확인
  - 저장 경로 : C:\Users*user*\AppData\Local\Google\Chrome\UserData\Default\Cache

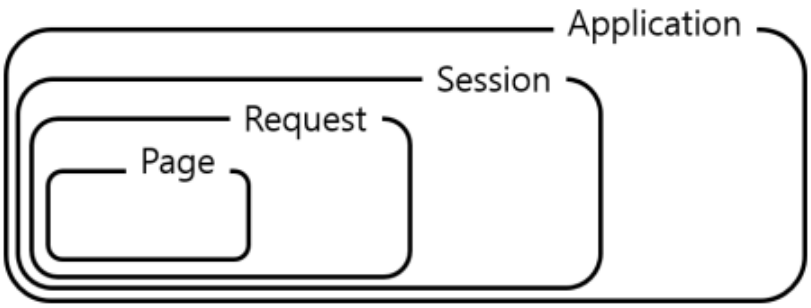
01-03. Session

01-03-01. Session이란

- 쿠키의 보안상 단점과 지원하지 않는 브라우저 문제 등으로 상태를 유지해야 하는 메커니즘에 적합하지 않은 경우(예: 회원 정보를 이용해서 해당 회원의 로그인 상태를 지속적으로 유지해야 하는 경우 등)가 다수 있다. 따라서 세션 인스턴스를 이용해서 상태를 유지하는 메커니즘을 제공한다.
- 서버에 데이터를 저장하는 기술로, 클라이언트에는 Session ID를 부여한다. 클라이언트가 요청 시 Session ID를 보내면 일치하는 Session정보를 컨테이너가 생성하여 그 객체의 데이터를 사용할 수 있다.

  즉, 브라우저마다 제공하는 HashMap으로 서버에서 클라이언트에게 제공한다. Session에 값을 넣으면 모든 서블릿에서 해당 Session ID로 고유 공간을 찾아 값을 공유해 사용할 수 있다.
  - 패키지 : javax.servlet.http.HttpSession
- 데이터를 서버에서 관리하므로 보안상 안전하고, 브라우저가 종료되면 세션도 함께 소멸한다.
- 만일 클라이언트가 보낸 Session ID가 없으면 새로 세션 객체를 생성한다.

01-03-02. 데이터 상태 저장 영역



1. Page scope : 하나의 Servlet, 하나의 Class에서만 공유 가능하다.
2. Request scope : forward에 한정해 공유할 수 있는 범위이다.
3. Session scope : redirect 방식에서도 활용할 수 있는 범위로 대표적으로 로그인 정보가 이 영역에 속한다. 사용자가 사용하는 모든 페이지에서 사용자의 정보를 가지고 있어야 하기 때문이다.
4. Application scope : 브라우저별 정보보다 넓은 범위이다.

01-03-03. Session 생성

- HttpSession은 직접 생성할 수 없고, request 객체의 getSession() 메소드를 이용해 반환 받을 수 있다.

```
HttpSession SessionID = HttpServletRequest.getSession();
```

- SessionID가 일치하는 Session 정보가 있으면 관련 객체를 호출하고, 없으면 boolean값에 따라 객체를 생성하거나 null 값을 반환한다. (true : 객체 생성 / false : null값 반환)

```
HttpRequest.getSession(boolean);
```

01-03-04. Session 설정 및 호출

- Session의 속성값을 설정한다.

```
SessionID.setAttribute('이름','값(Obj)'); //Session 데이터 설정
SessionID.setMaxInactiveInterval(숫자);   //Session 유지시간 설정
```

- Session의 속성값을 꺼내온다.

```
HttpSession SessionID = HttpServletRequest.getSession();
SessionID.getAttribute('이름'); // 데이터 호출
```

01-03-05. Session 데이터 삭제 방법

- 설정한 만료 시간이 지나면 세션이 자동으로 만료된다.

  ▼ (참고) 세션 timeout 적용 우선 순위

  1. setMaxInactiveInterval(int interval) 메소드를 통해 세션에 개별적으로 적용한 만료 시간

     ```
     session.setMaxInactiveInterval(60 * 10);  // 만료시간 10분 설정
     ```

  2. 각 웹 애플리케이션의 WEB-INF 디렉토리 하위의 web.xml (배포서술자) 파일을 통해 지정한 만료 시간

     ```
     <session-config>
       <session-timeout>60</session-timeout>
     </session-config>
     ```

  3. tomcat 설치 디렉터리 하위 conf 경로에 위치한 web.xml 파일을 통해 지정한 만료 시간

     ⇒ 톰캣의 conf 디렉터리에는 애플리케이션과 마찬가지로 web.xml 파일을 가지고 있고, 우선순위인 setMaxInactiveInterval() 메소드나 웹 애플리케이션의 web.xml에 설정하지 않으면 이곳에 설정된 값으로 적용된다.

     ```
     <session-config>
       <session-timeout>60</session-timeout>
     </session-config>
     ```

- removeAttribute() 메소드로 session의 attribute을 지운다.

  ```
  session.removeAttribute("속성명");
  ```

- invalidate() 메소드를 호출하면 세션의 모든 데이터를 제거한다.
  - 세션 자체를 무효화, 즉 강제 만료시키는 메소드이므로 실행 이후 세션을 이용하려고 하면 에러가 발생한다

```
session.invalidate();
```

01-03-06. Session Method

method명	**내용**
setAttribute(String, object)	request객체에 전달하고 싶은 값을 String 이름-Object 값으로 설정
getAttribute(String)	매개변수와 동일한 객체의 속성 값 가져옴
getAttributeNames()	객체에 등록되어 있는 모든 속성의 이름만 반환
removeAttribute(String)	request객체에 저장된 매개변수와 동일한 속성 값 삭제
getId()	SessionID값 가져옴
getCreationTime()	Session객체가 생성된 시간 반환 (msec)
getMaxInactiveInterval()	client 요청이 없을 때, 서버가 현재 Session을 언제까지 유지할지 초 단위로 반환(default = 30분)
getLastAccessedTime()	client 요청이 마지막으로 시도된 시간 반환 (msec)
isNew()	새로 생성된 Session이면 true, 아니면 false 반환
invalidate()	현재 Session 삭제
setMaxInactiveInterval(int)	객체 유지 시간을 설정하고, 지정 시간 지나면 객체 자동 삭제