(가이드)

장표 종류

오리엔테이션 장표
전체 강의 중 1번 / 강사마다 1번 등장

챕터 시작 장표
현재 15챕터터까지 제작완료

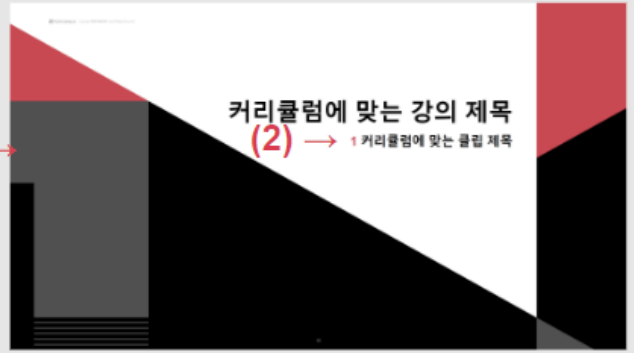
화면전환 장표
클립 속의 클립, 부제등 사용 가능

내지 장표

기본 장표, 내지, 본문, PPT내용 부분

예시 :

(1) 배경 속 숫자 = 챕터 넘버
(2) 빨간색 텍스트 숫자 = 클립 넘버



초격차 패키지 Online.

안녕하세요.
시그니처 백엔드 강의
백엔드 웹 개발 입문/실전을
진행하는 예상국입니다.

백엔드 웹 개발 입문/실전

PART1 | 웹 개발 입문과 데이터베이스

PART2 | 웹 서비스 개발 실전

PART3 | 최신/심화 웹 개발 실전

Web Service 의 인증

1 인증이란?

Web Service 인증

Web 인증이란?

웹 인증(Web Authentication)이란 웹 애플리케이션에서 사용자의 정체성을 확인하고 적절한 권한을 부여하는 과정입니다. 웹 인증은 사용자가 누구인지 확인하고, 해당 사용자가 액세스하려는 웹 리소스나 서비스에 대한 권한이 있는지 확인하는 데 사용됩니다.

웹 인증은 웹 서비스의 보안성을 높이며, 사용자 데이터의 무단 접근을 방지하기 위한 중요한 보안 요소입니다.

Web 인증이란?

사용자 등록

사용자는 서버로
아이디, 비밀번호 등의 인증 정보를 전송

서버는정보를 안전하게 저장

사용자 인증

ID, PW를 통하여
사용자는 웹에 인증

서버는 사용자를 확인하여
적절한 인증 및 인가

세션 관리

인증된 사용자는
웹 서버의 리소스에
접근할 수 있는
권한을 받는다.

서버는 쿠키, 세션, 토큰 등 적절한 기술을 통하여 사용자의 로그인 상태와 권한 관리

로그인 인증

로그인 인증

로그인 인증을 위한 다양한 인증 방식 / 로그인 방식이 존재 합니다.

1. ID/PW 기반 로그인
2. 소셜 로그인 (Oauth2)
3. 이메일 인증
4. 휴대폰 인증
5. 다중 인증요소 (MFA)

HTTP Session 인증

HTTP SESSION

HTTP SESSION 은 웹 어플리케이션에서 사용자 정보를 저장하는 기술 입니다.
사용자의 세션은 웹 어플리케이션에 접속 한 후, 일정 시간 동안 유지되는 정보

특징

1. HTTP 프로토콜은 Stateless 한 특성을 가지기 때문에 사용자가 다시 요청을 보낼 때마다 사용자 정보를 매번 다시 전송 해야 한다. HTTP Session 은 이러한 문제를 해결하기 위해 사용자 정보를 서버측에서 저장하고 관리하는 세션 ID를 발급 한다.

2. HTTP Session은 쿠키(Cookie)를 사용하여 구현된다.

3. HTTP Session은 사용자 로그인 정보를 관리 할때 사용하며, 사용자가 다시 접속 하여도 유지 된다.

4. HTTP Session은 서버에서 관리되기 때문에 사용자가 임의로 세션 정보를 조작 할 수 없다.
해당 값은 랜덤한 값으로 생성되며, HTTPS를 통해서 암호화 된다.

HTTP SESSION 인증

HTTP Session 인증은 웹 어플리케이션에서 사용하는 인증 방법으로

사용자 인증 정보를 **서버측에서 유지하고 관리**하기 위한 방법중 1가지 입니다.

인증 과정

1. 사용자가 로그인을 시도
2. 서버는 사용자의 인증 정보를 검증 하여 session Id를 생성
3. 세션은 서버측에서 관리되며, 서버에서 갱신 및 정보를 변경 할 수 있습니다.
4. 세션 ID는 쿠키(cookie) 방식으로 사용자에게 전달 되며, 웹 어플리케이션에서 사용한다.

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>

<style>
    /* 스마트폰 크기에서는 로그인 폼을 세로로 배치 */
    @media (max-width: 767px) {
        #login-form {
            display: flex;
            flex-direction: column;
        }
    }

    /* 로그인 폼 스타일 */
    #login-form {
        max-width: 400px;
        margin: 0 auto;
        padding: 20px;
        background-color: #f2f2f2;
        border-radius: 10px;
```

HTTP Cookie 인증

HTTP Cookie

HTTP Cookie(쿠키)는 웹 브라우저와 웹 서버 간에 상태 정보를 유지하기 위한 기술입니다. 쿠키는 서버가 브라우저는 이를 로컬에 저장하고 필요할 때마다 서버에 전송하여 사용자 상태 정보를 유지합니다.

쿠키는 HTTP 헤더에 Set-Cookie와 같은 헤더를 통해 서버에서 클라이언트에 전송됩니다.
쿠키는 키-값 쌍으로 이루어져 있으며, 이름, 값, 유효 기간, 도메인, 경로 등의 정보를 포함합니다.

특징

1. 쿠키는 클라이언트 측에 저장됩니다. 따라서, 서버가 클라이언트의 상태 정보를 확인하려면, 쿠키를 클라이언트에서 전송받아야 합니다.
2. 쿠키는 유효 기간을 가지고 있습니다. 유효 기간이 지나면, 쿠키는 삭제됩니다.
3. 쿠키는 보안 문제가 있을 수 있습니다. 쿠키에 민감한 정보를 저장하는 경우, HTTPS와 같은 보안 프로토콜을 사용하여 암호화해야 합니다.
4. 쿠키는 브라우저에서 관리되기 때문에, 브라우저에서 쿠키를 삭제하거나, 다른 브라우저에서 접속하는 경우에는 쿠키를 공유할 수 없습니다.

HTTP Cookie 를 통한 인증

Cookie를 통한 인증은 많은 곳에서 사용하고 있는 인증 방법 입니다.
앞서 배운 Cookie의 특성을 이용하여 사용자를 인증 합니다.

1. 사용자가 로그인 페이지에 접속하여 로그인 정보를 입력합니다
.
2. 서버는 사용자 정보를 검증하고, 인증이 성공하면 사용자의 고유 ID와 함께 인증 토큰(쿠키)을 생성합니다.
3. 서버는 생성된 인증 토큰(쿠키)을 HTTP 응답 헤더에 포함하여 클라이언트에게 전송합니다.
4. 클라이언트는 전송받은 인증 토큰(쿠키)을 로컬에 저장합니다.
5. 클라이언트는 이후 서버에 요청을 보낼 때마다 인증 토큰(쿠키)을 HTTP 요청 헤더에 포함하여 전송합니다.
6. 서버는 전송받은 인증 토큰(쿠키)을 검증하여, 인증이 성공하면 요청에 대한 응답을 생성합니다.

Header를 통한 인증

HTTP Header

HTTP Header를 통한 인증

서버와 클라이언트 간의 인증을 HTTP 헤더를 통해서 수행하는 방식

Http Basic, Http Digest, Oauth 와 같은 프로토콜을 통해서 구현 되는게 일반적이다.

그 외에는 특정 header 에 token을 넣어서 사용자를 인식 하고 인증을 한다.

JWT Token

JWT Token

JWT (JSON Web Token)는 웹 표준으로써, 데이터의 JSON 객체를 사용하여 가볍고 자가 수용적인 방식으로 정보를 안전하게 전달할 수 있도록 설계된 토큰 기반의 인증 방식입니다.

JWT는 URL, HTTP Header, HTML Form과 같은 다양한 방식으로 전달할 수 있으며, 서버와 클라이언트 간의 인증 정보를 포함합니다.

JWT는 Header, Payload, Signature 세 부분으로 구성됩니다.
Header는 JWT의 타입과 암호화 알고리즘 등을 포함하며, JSON 형식으로 인코딩됩니다.

Payload는 클레임 정보를 포함하며, JSON 형식으로 인코딩됩니다. 클레임 정보는 사용자 ID, 권한 등의 정보를 포함할 수 있습니다.

Signature는 Header와 Payload를 조합한 후, 비밀 키를 사용하여 생성된 서명 값입니다. 서명 값은 토큰의 무결성을 보장하며, JWT를 조작하지 않았다는 것을 검증합니다.

JWT Token

JWT Token을 이용한 인증 방식

1. 클라이언트가 서버에 로그인 요청을 보냅니다.
2. 서버는 로그인 요청을 검증하고, 유효한 사용자라면 JWT를 생성하여 클라이언트에게 반환합니다.
3. 클라이언트는 이후 요청에 JWT를 포함시켜 전송합니다.
4. 서버는 JWT를 검증하여, 클라이언트의 인증 여부를 판단합니다.

JWT Token

장점

1. 토큰 기반의 인증 방식이므로, 서버 측에서 별도의 세션 저장소를 유지할 필요가 없습니다.
2. JSON 형식으로 인코딩되므로, 다양한 플랫폼 간에 쉽게 전송 및 구현할 수 있습니다.
3. Signature를 사용하여 무결성을 보장하므로, 토큰이 변조되었는지 여부를 쉽게 검증할 수 있습니다.

단점

1. JWT의 크기가 커질 경우, 네트워크 대역폭이 증가하게 됩니다.
2. JWT는 한 번 발급된 후에는 내부 정보를 수정할 수 없으므로, 만료 시간을 짧게 설정해야 합니다.
3. JWT를 탈취당하면, 해당 토큰을 사용한 모든 요청이 인증 되므로, 보안 위협이 될 수 있습니다. 따라서, HTTPS와 같은 보안 프로토콜을 사용하여 JWT를 전송해야 합니다.