4. 스프링프레임워크중요개념3

Session과 Cookie는 웹 페이지에서 로그인 정보를 유지하는 중요한 기능들입니다. 이 두 가지는 사용자의 로그인 상태를 관리하는데, 각각의 역할과 관리 방식이 다릅니다. 여기서 다루는 Session과 Cookie는 자바스크립트에서 배운 로컬 스토리지나 세션 스토리지와는 다른 개념입니다. 웹 프로그램에서 프론트엔드와 백엔드가 구분되듯이, Session과 Cookie도 서로 다른 영역에서 관리됩니다.

1. Session

- **Session**은 주로 **백엔드**에서 관리되며, 사용자의 로그인 상태를 유지하는 역할을 합니다. 예를 들어, 사용자가 네이버에서 로그인 후, 페이지를 이동할 때마다 로그인 상태를 유지하려면 Session이 필요합니다.
- 웹 페이지는 기본적으로 상태를 유지하지 않는 **Stateless** 구조를 가지고 있기 때문에, 로그인 상태를 유지하기 위해서는 Session이 필요합니다.
- Session은 사용자의 정보를 저장하고, 사용자가 페이지를 이동해도 로그인 상태를 유지할 수 있도록 돕습니다. 서버가 종료되거나, 사용자가 로그아웃하거나, 브라우저를 닫을 때까지 Session은 유지됩니다.
- **Session ID**는 중요한 정보를 담고 있지 않으며, 사용자의 로그인 상태를 식별하는 식별자 역할을 합니다. 프론트엔드로 Session ID만 전달되며, 사용자의 실제 로그인 정보는 서버에 저장됩니다.

2. Cookie

- **Cookie**는 클라이언트 측에서 관리되며, 브라우저에 저장되는 작은 데이터 조각입니다. Session ID와 같은 정보를 저장하여, 사용자의 로그인 상태를 서버와 프론트엔드가 서로 확인할 수 있도록 합니다.
- Cookie에는 보안 처리된 **Session ID**가 저장되며, 이 정보를 기반으로 사용자가 로그인한 상태임을 확인합니다. 이로 인해 사용자의 중요한 정보가 탈취되는 것을 방지할 수 있습니다.

3. 상태 관리 방식

- **Stateless**: 서버가 클라이언트의 상태를 전혀 저장하지 않는 방식입니다. 매번 로그인을 해야 하며, 주로 **JWT 토큰**과 같은 인증 방식을 사용할 때 적용됩니다.
- **Stateful**: 서버와 클라이언트가 상태를 유지하는 방식으로, 일반적인 로그인 상태 유지 기능을 사용할 때 적용됩니다. 웹 애플리케이션이나 쇼핑몰과 같은 로그인 상태가 필요한 서비스에서 많이 사용됩니다.
- **IfRequired**: 필요한 경우에만 Session을 생성하는 방식으로, 기본적으로 로그인 시에만 Session이 생성됩니다.
- **Never**: 새로운 Session을 생성하지 않지만, 이미 존재하는 Session은 사용할 수 있는 방식입니다.

결론

Session은 서버 측에서, Cookie는 클라이언트 측에서 각각 관리되며, 두 가지 모두 로그인 상태를 유지하는데 중요한 역할을 합니다. Spring Framework에서 주로 사용하는 **Spring Security**를 통해 보다 안전하게 세션을 관리할 수 있으며, 필요한 경우 세션 유지 시간을 설정할 수도 있습니다.

이제 세션과 쿠키의 기본 개념을 이해했으니, 앞으로는 실제 개발에 적용해보며 더 깊이 알아가면 좋을 것입니다.