권한에 따른 처리

한국기술교육대학교
온라인평생교육원

학습내용

- 로그인의 원리
- 세션(Session) 관리
- 권한 설정

학습목표

- 로그인의 원리에 대해 설명할 수 있다.
- 세션(Session) 관리에 대해 설명할 수 있다.
- 권한을 설정하는 방법에 대해 파악할 수 있다.

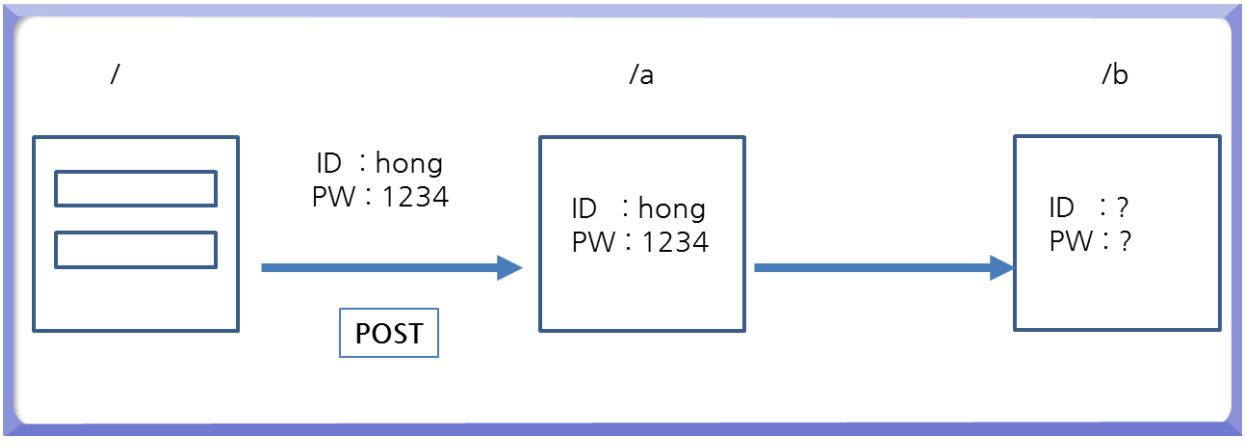
로그인의 원리

1 로그인의 기본

⟨1⟩ Model Attribute

1 http request에 저장한 Model변수는 하나의 요청 흐름에 대해서만 유효

2 변수가 지속적으로 존재하지 않으므로 로그인 처리 등에 활용하기 어려움



로그인의 원리

2 Session의 활용

⟨1⟩ Session의 활용

1. http session이란 서버가 자신에게 접속한 클라이언트를 식별하는 방법
2. 접근한 클라이언트 각각에게 session-id라는 식별자 부여
3. 이 식별자는 서버와 클라이언트의 메모리에 저장
4. 하나의 클라이언트에 대해 지속적으로 관리해야 하는 데이터 저장 장소로 세션 활용 가능

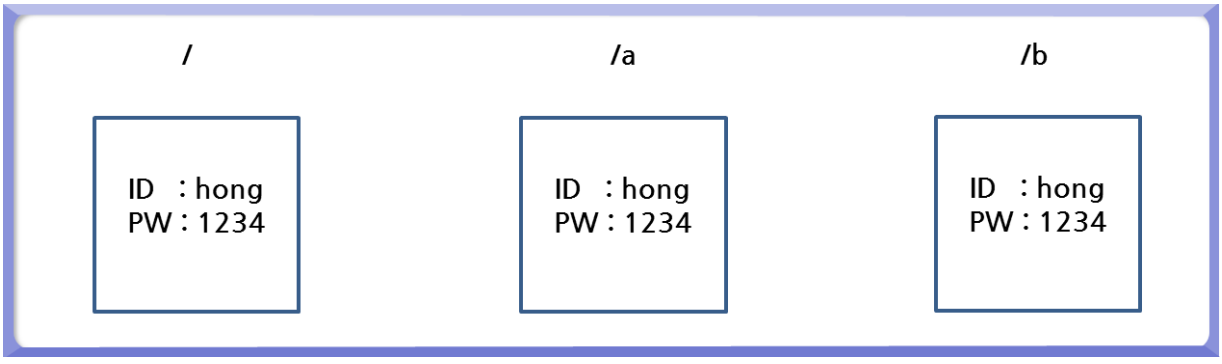
Session 생성 시기

- 임의의 웹 브라우저부터의 첫 번째 요청을 처리할 때 Session이 생성되고 관련 타이머가 동작함

Session 소멸 시기

- 브라우저 종료
- 세션 타이머가 만료
- 코드상에서 명시적으로 세션 소멸(예 : 로그아웃코드)

5. Session에 ID : hong, PW : 1234 변수를 저장할 경우 브라우저를 닫기 전까지 모든 URL에서 사용 가능



Session 관리

1 Session Attribute 활용

Controller에서 Session객체 활용

1. Controller의 Parameter에 javax.servlet.http.HttpSession 객체를 추가함으로써 Session 객체 활용 가능
2. session.setAttribute("변수명", 값);과 같이 Session에 객체 저장 가능
3. session.removeAttribute("변수명");과 같이 Session에 저장된 객체 소멸 가능

```
@RequestMapping(value="/sessionTest")
  @ResponseBody
  public String sessionTest(HttpSession session) {
     session.setAttribute("a", 3);
     session.removeAttribute("a");
     return "success";
  }
```

4. 기본 request 객체에서도 Session 객체를 가져올 수 있음

```
@RequestMapping(value="/sessionTest")
  @ResponseBody
  public String sessionTest(HttpServletRequest request) {
     HttpSession session = request.getSession();
     session.setAttribute("a", 3);
     session.removeAttribute("a");
     return "success";
  }
```

Session 관리

2 Spring Security에서의 세션 활용

《1》 Spring Security에서의 로그인 유저

기존에는 Session을 통한 로그인 처리, 유저 관리를 직접 해줘야 했음

Spring Security에서는 SecurityContextHolder로부터 User Principal을 가져올 수 있음

이 Principal을 변수에 저장하여 활용 가능

《2》 Jsp에서의 활용

```
<c:set var="user"
value="${SPRING_SECURITY_CONTEXT.authentication.principal}"/>
```

《3》 Java에서의 활용

```
public static User current() {
    try {
        return (User) SecurityContextHolder.getContext()
                .getAuthentication().getPrincipal();
    } catch (Exception e) {
        return null;
    }
}
```

권한 설정

1 표현언어 기반 권한 설정

표현언어(Expression Language) 기반

1. jsp의 표현언어 기반으로 권한을 설정할 수 있음
2. security.xml에 <sec:http use-expressions="true"> 설정으로 인해 Expression Language Based 설정 가능
3. 이러한 설정을 xml 설정 파일과 java 파일 둘 다 가능

표현언어	설명
hasRole([role])	Principal이 [role]을 가지면 True
hasAnyRole([role1], [role2])	Principal이 여러 [role] 중 하나를 가지면 True
principal	현재 로그인 되어 있는 유저에 직접 접근
permitAll	모두 허용(항상 True)
denyAll	모두 불허용(항상 False)
isAnonymous()	현재 유저가 익명 유저라면 True
isAuthenticated()	현재 유저가 익명 유저가 아니라면 True

권한 설정

2 Java와 Xml에서의 권한 설정

〈1〉 Java에서의 권한 설정

@PreAuthorize annotation을
이용해 표현언어 방식으로
권한 설정 가능

함수 별로, 혹은 하나의 Controller
전체에 대한 권한 설정 가능

```
@PreAuthorize("hasRole('ROLE_USER')")
  @RequestMapping(value="/onlyUserByJava")
  @ResponseBody
  public String onlyUserByJava() {
      System.out.println("User.current() = " + User.current());
      return "user";
  }

  @PreAuthorize("hasRole('ROLE_ADMIN')")
  @RequestMapping(value="/onlyAdminByJava")
  @ResponseBody
  public String onlyAdminByJava() {
      System.out.println("User.current() = " + User.current());
      return "admin";
  }
```

권한 설정

2 Java와 Xml에서의 권한 설정

2 XML에서의 권한 설정

1. Security.xml에서 intercept-url 설정을 통해 주소에 대한 접근을 제어할 수 있음
2. 위에서부터 순서대로 접근을 제어할 주소에 대한 권한 설정을 부여하고, 그 외에는 모두 허용하는 방식
3. 다음 예제는 /user/onlyUserByXml과 /user/onlyAdminByXml 주소 각각에 대해 접근 가능한 권한을 부여하고, 그 외의 모든 주소에 대해서는 인증 없이 접근 가능하도록 설정한 예

```
<sec:http use-expressions="true">
    <sec:form-login login-page="/user/signin" default-target-
url="/user/signinSuccess"
      authentication-failure-url="/user/signinFailed"/>
    <sec:logout logout-url="/user/signout" logout-success-url="/user/signin"
/>
    <sec:intercept-url pattern="/user/onlyUserByXml"
access="hasRole('ROLE_USER')" />
    <sec:intercept-url pattern="/user/onlyAdminByXml"
access="hasRole('ROLE_ADMIN')" />
    <sec:intercept-url pattern="/**" access="permitAll" />
  </sec:http>
```

권한 설정

2 Java와 Xml에서의 권한 설정

3 Jsp에서의 권한 설정

Jsp에서는 전체 페이지 단위가 아닌 페이지의 부분 단위로 권한 설정 가능

Jsp에서도 동일한 표현식을 이용해서 권한에 따른 View처리를 할 수 있음
예) 관리자만 볼 수 있는 특별 문구 혹은 메뉴 지정 가능

Jsp 상단에 아래와 같이 한 줄을 추가해야 함

```
<<%@ taglib prefix="sec"
uri="http://www.springframework.org/security/tags" %>
```

```
<sec:authorize access="hasRole('ROLE_USER')">
이 문장은 ROLE_USER 권한을 가진 사람에게만 보입니다.<br/>
</sec:authorize>
<sec:authorize access="hasRole('ROLE_ADMIN')">
이 문장은 ROLE_ADMIN 권한을 가진 사람에게만 보입니다.<br/>
</sec:authorize>
<sec:authorize access="hasAnyRole('ROLE_USER', 'ROLE_ADMIN')">
이 문장은 ROLE_USER 혹은 ROLE_ADMIN 권한을 가진 사람에게만 보입니다.<br/>
</sec:authorize>
```

학습정리

1. 로그인의 원리

- Model Attribute
  - Http Request에 저장한 Model변수는 하나의 요청 흐름에 대해서만 유효
  - 변수가 지속적으로 존재하지 않으므로 로그인 처리 등에 활용하기 어려움

- Session의 활용
  - Http Session이란 서버가 자신에게 접속한 클라이언트를 식별하는 방법
  - 접근한 클라이언트 각각에게 Session-id라는 식별자를 부여함
  - 이 식별자는 서버와 클라이언트의 메모리에 저장됨
  - 하나의 클라이언트에 대해 지속적으로 관리해야 하는 데이터 저장 장소로서 세션 활용 가능
  - Session에 ID : hong, PW : 1234 변수를 저장할 경우 브라우저를 닫기 전까지 모든 URL에서 사용 가능

학습정리

2. Session 관리

- Session Attribute 활용
  - Controller의 Parameter에 javax.servlet.http.HttpSession 객체를 추가함으로써 Session객체 활용 가능
  - session.setAttribute(“변수명”, 값); 과 같이 Session에 객체 저장 가능
  - session.removeAttribute(“변수명”);과 같이 Session에 저장된 객체 소멸 가능
  - 기본 Request객체에서도 Session 객체를 가져올 수 있음

- Spring Security에서의 세션 활용
  - Spring Security에서는 SecurityContextHolder로부터 User Principal을 가져올 수 있고 이 principal을 변수에 저장하여 활용할 수 있음

학습정리

3. 권한 설정

- 표현언어(Expression Language) 기반 권한 설정
  - Jsp의 표현언어 기반으로 권한을 설정할 수 있음
  - Security.xml에 <sec:http use-expressions="true"> 설정으로 인해 Expression Language Based 설정 가능
  - 이러한 설정을 xml 설정 파일과 Java 파일 둘 다 가능

- Java에서의 권한 설정
  - @PreAuthorize Annotation을 이용해 표현언어 방식으로 권한 설정 가능
  - 함수별로, 혹은 하나의 Controller 전체에 대한 권한 설정 가능

- XML에서의 권한 설정
  - Security.xml에서 Intercept-URL 설정을 통해 주소에 대한 접근 제어 가능