Spring security 로그인 이후 로직!

앞서 Mybatis를 통해 유저정보를 확인하고 넘어온 username과 password를 통해 정상적인 사용자인지 구별했습니다.

이제는 로그인 이후 처리 로직에 대해 설명하겠습니다.
- 일반적으로 처리하는 로그인한 사용자의 IP정보 저장하기
- 로그인 이전 세션내에 요청한 URL이 있다면 그 URL로 Redirect하기

1. AuthenticationSuccessHandler 구현하기

```
    package com.example.demo.security.handlers;

    import java.io.IOException;

    import javax.servlet.ServletException;
    import javax.servlet.http.HttpServletRequest;
    import javax.servlet.http.HttpServletResponse;
    import javax.servlet.http.HttpSession;

    import com.example.demo.security.domain.SecurityMember;

    import org.springframework.security.core.Authentication;
    import
org.springframework.security.web.authentication.SavedRequestAwareAuthenticationS
uccessHandler;

    public class CustomLoginSuccessHandler extends
SavedRequestAwareAuthenticationSuccessHandler{

        public CustomLoginSuccessHandler(String defaultTargetUrl){
            setDefaultTargetUrl(defaultTargetUrl);
        }



    // 로그인 성공 시 동작을 정의해준다.
    // 로그인 성공 시 SecurityMember(세션정보 객체)에 IP정보를 추가해준다.
    // getClientIp 메소드를 통해 받아온다.

    // 또한 세션 정보를 받아와 앞서 접근한 URL에 대한 정보를 받아온다.
    // 로그인 시 앞서 접근한 세션내에서의 URL에 redirect해준다.
    @Override
    public void onAuthenticationSuccess(HttpServletRequest request,
HttpServletResponse response,Authentication authentication) throws
ServletException,IOException{

        // 로그인 성공시점에 IP 정보를 request를 통해 SecureMember객체에 추가한다.

((SecurityMember)authentication.getPrincipal()).setIp(getClientIp(request));

        HttpSession session = request.getSession();
```

```
        if(session != null){
            String redirectUrl = (String) session.getAttribute("prevPage");
            if(redirectUrl != null){
                session.removeAttribute("prevPage");
                getRedirectStrategy().sendRedirect(request, response,
redirectUrl);
            }
            else{
                super.onAuthenticationSuccess(request, response,
authentication);
            }
        }
        else{
            super.onAuthenticationSuccess(request, response, authentication);
        }

    }

    // 사용자의 IP를 가져오는 방법
    // 로그인 성공 동작 시 호출된다.
    public static String getClientIp(HttpServletRequest request) {
        String ip = request.getHeader("X-Forwarded-For");
         if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip)) {
             ip = request.getHeader("Proxy-Client-IP");
         }
         if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip)) {
             ip = request.getHeader("WL-Proxy-Client-IP");
         }
         if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip)) {
             ip = request.getHeader("HTTP_CLIENT_IP");
         }
         if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip)) {
             ip = request.getHeader("HTTP_X_FORWARDED_FOR");
         }
         if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip)) {
             ip = request.getRemoteAddr();
         }
         return ip;
    }
}
```

방법은 여러가지입니다. 참고만하세요

1. CustomLoginSuccessHandler 라는 java파일을 생성했습니다.
이 파일에서는 HttpServletRequest를 통해 사용자의 IP를 받아옵니다.

또한 이 IP를 SecurityMember라고 정의해놓은 세션정보 객체에 추가합니다.

2. HttpSession객체를 통해 Session에 접근합니다.
  이를 통해 세션 내에 앞서 접근한 URL을 받아옵니다.
  로그인 성공 이후 앞서 접근한 URL로 redirect해줍니다.

2. Adapter에 이에 대한 내용 추가하기

Adapter부분에 내용추가한 부분은 예시로 올린 소스를 참고하세요.

3. 설명

```
Adapter에 기본 정의되어 있는 로그인 로직에
successHandler를 추가로 실행시켜줌으로 로그인 이후 로직이 실행되는 것입니다.
기본적으로 사용자의 적합성을 판별해주는 부분은 그대로입니다.
```

Made by 장경석