Spring Security Concurrent Session Implementation With Custom Form Login Filter

In Spring 4, we can use following configuration to adjust concurrent session count:

To use concurrent session support, you'll need to add the following to web.xml:

```
<listener>
    <listener-class>
        org.springframework.security.web.session.HttpSessionEventPublisher
    </listener-class>
</listener>
```

In addition, you will need to add the ConcurrentSessionFilter to your FilterChainProxy. The ConcurrentSessionFilter requires two properties, **sessionRegistry**, which generally points to an instance of SessionRegistryImpl, and **expiredUrl**, which points to the page to display when a session has expired. A configuration using the namespace to create the FilterChainProxy and other default beans might look like this (spring-security.xml file):

```
<!-- enable use-expressions -->
<http auto-config="false" use-expressions="true" pattern="/user/**"
     entry-point-ref="loginUrlAuthenticationEntryPoint">
    <!--If you use hasAnyAuthority method, you can ignore ROLE_ prefix -->
    <intercept-url pattern="/user/home/yoneticiler"
access="hasAnyAuthority('FULL_ADMIN','ADMIN')"/>
    <intercept-url pattern="/user/home/addUser" access="hasAnyAuthority('FULL_ADMIN','ADMIN')"/>
    <intercept-url pattern="/user/home/addUserGroup" access="hasAuthority('FULL_ADMIN')"/>
    <intercept-url pattern="/user/home/deleteUserGroup" access="hasAuthority('FULL_ADMIN')"/>
    <intercept-url pattern="/user/home/**"
                  access="hasAnyAuthority('FULL_ADMIN','ADMIN','EDITOR','SADECE_GORME')"/>
    <access-denied-handler error-page="/403"/>
    <custom-filter position="CONCURRENT_SESSION_FILTER" ref="concurrencyFilter" />
    <custom-filter position="FORM_LOGIN_FILTER" ref="myAuthFilter"/>
    <session-management session-authentication-strategy-ref="sas"/>
    <logout logout-url="/user/logout"
           invalidate-session="true"
           logout-success-url="/user/index?logout"/>
    <!-- enable csrf protection -->

    <csrf/>
</http>
<beans:bean id="concurrencyFilter"
           class="org.springframework.security.web.session.ConcurrentSessionFilter">
    <beans:constructor-arg index="0" ref="sessionRegistry"/>
    <beans:constructor-arg index="1" value="/user/index?logout"/>
</beans:bean>
<beans:bean id="myAuthFilter" class=
            "com.codesenior.telif.security.CustomUsernamePasswordAuthenticationFilter">
    <beans:property name="sessionAuthenticationStrategy" ref="sas" />
    <beans:property name="authenticationManager" ref="authenticationManager" />
    <beans:property name="filterProcessesUrl" value="/user/login"/>
    <beans:property name="authenticationFailureHandler" ref="failureHandler"/>
    <beans:property name="authenticationSuccessHandler" ref="authenticationSuccessHandler"/>
</beans:bean>
<beans:bean id="sas"
           class="org.springframework.security.web.authentication.session.CompositeSessionAuthe
nticationStrategy">
    <beans:constructor-arg>
        <beans:list>
            <beans:bean
                        class="org.springframework.security.web.authentication.session.Concurren
tSessionControlAuthenticationStrategy">
                <beans:constructor-arg ref="sessionRegistry"/>
```

```
                <beans:property name="maximumSessions" value="1" />
                <beans:property name="exceptionIfMaximumExceeded" value="true" />
            </beans:bean>
            <beans:bean
                        class="org.springframework.security.web.authentication.session.SessionFi
xationProtectionStrategy">
            </beans:bean>
            <beans:bean
                        class="org.springframework.security.web.authentication.session.RegisterS
essionAuthenticationStrategy">
                <beans:constructor-arg ref="sessionRegistry"/>
            </beans:bean>
        </beans:list>
    </beans:constructor-arg>
</beans:bean>
<!-- Select users and user_roles from database -->
<authentication-manager id="authenticationManager">
    <authentication-provider ref="daoAuthenticationProvider"/>
</authentication-manager>
<!-- Your DaoAuthenticationProvider will then use it like with any -->
<!-- other implementation of the PasswordEncoder interface. -->
<beans:bean id="daoAuthenticationProvider"
            class="org.springframework.security.authentication.dao.DaoAuthenticationProvider">
    <beans:property name="userDetailsService" ref="customUserDetailsService"/>
    <beans:property name="passwordEncoder" ref="passwordEncoder"/>
</beans:bean>
<beans:bean id="customUserDetailsService"
                class="com.codesenior.telif.security.CustomUserDetailsService">
        <beans:property name="userServiceImpl" ref="userServiceImpl"/>
    </beans:bean>
<!-- This Spring Security-friendly PasswordEncoder implementation will -->
<!-- wrap the PasswordEncryptor instance so that it can be used from -->
<!-- the security framework. -->
<beans:bean id="passwordEncoder"
class="org.jasypt.springsecurity3.authentication.encoding.PasswordEncoder">
    <beans:property name="passwordEncryptor" ref="jasyptPasswordEncryptor"/>
</beans:bean>
  <!-- Your application may use the PasswordEncryptor in several places, -->
    <!-- like for example at new user sign-up. -->
    <beans:bean id="jasyptPasswordEncryptor"
class="org.jasypt.util.password.StrongPasswordEncryptor"/>
<beans:bean id="successHandler"
            class="org.springframework.security.web.authentication.SavedRequestAwareAuthenticati
onSuccessHandler">
    <beans:property name="defaultTargetUrl" value="/user/home"/>
</beans:bean>
<beans:bean id="failureHandler"
            class="org.springframework.security.web.authentication.SimpleUrlAuthenticationFailur
eHandler">
    <beans:property name="defaultFailureUrl" value="/user/index?error=true"/>
</beans:bean>
```

CustomUsernamePasswordAuthenticationFilter Class

```
import org.jasypt.exceptions.EncryptionOperationNotPossibleException;
import org.springframework.security.authentication.AuthenticationServiceException;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;

public class CustomUsernamePasswordAuthenticationFilter
         extends UsernamePasswordAuthenticationFilter {

     @Override
     protected void successfulAuthentication(HttpServletRequest request,
                                             HttpServletResponse response,
                                             FilterChain chain,
                                             Authentication authResult)
             throws IOException, ServletException {
         super.successfulAuthentication(request, response, chain, authResult);
     }

     @Override
     protected void unsuccessfulAuthentication(HttpServletRequest request,
                                               HttpServletResponse response,
                                               AuthenticationException failed)
             throws IOException, ServletException {
         super.unsuccessfulAuthentication(request, response, failed);
     }

     @Override
     public Authentication attemptAuthentication(HttpServletRequest request, HttpServletResponse
response)
             throws AuthenticationException {
         HttpSession session = request.getSession();
         if (session.getAttribute("rand1") == null || session.getAttribute("rand2") == null)
             throw new IllegalArgumentException("Please type captcha result");
         int rand1 = Integer.parseInt((String) session.getAttribute("rand1"));
         int rand2 = Integer.parseInt((String) session.getAttribute("rand2"));
         if (Integer.valueOf(request.getParameter("captcha")) != rand1 + rand2) {
             throw new AuthenticationServiceException("What is " + rand1 + "+" + rand2 + "?");
         }
         try {
             return super.attemptAuthentication(request, response);
         } catch (EncryptionOperationNotPossibleException e) {
             throw new AuthenticationServiceException("Wrong username or password");
         }

     }
}
```

CustomUserDetailsService

```
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;

import java.lang.reflect.Field;
import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.List;


public class CustomUserDetailsService implements UserDetailsService {
     private Object userServiceImpl;
```

```
    public CustomUserDetailsService() {
    }

    public UserDetails loadUserByUsername(String username) {
        try {
            Method userMethod = userServiceImpl.getClass().getMethod("getUser", String.class);
            Object userFromDB = userMethod.invoke(userServiceImpl, username);
            Field field = userFromDB.getClass().getDeclaredField("password");
            field.setAccessible(true);
            String password = (String) field.get(userFromDB);
            return getUserDetails(username, password, userFromDB);
        } catch (Exception e) {
            throw new RuntimeException("Kullanıcı adınız veya şifreniz yanlış");
        }
    }

    private UserDetails getUserDetails(String username, String password, Object userFromDB) throws
Exception {
        return new User(
                username,
                password,
                true,
                true,
                true,
                true,
                getGrantedAuthorities(getRoleListAsStringArray(userFromDB))
        );
    }

    public static List<GrantedAuthority> getGrantedAuthorities(List<String> roles) {
        List<GrantedAuthority> authorities = new ArrayList<GrantedAuthority>();
        for (String role : roles) {
            authorities.add(new SimpleGrantedAuthority(role));
        }
        return authorities;
    }

    private List<String> getRoleListAsStringArray(Object userFromDB) throws Exception {
        List<String> roleList = new ArrayList<String>();
        Method userRoleListMethod = userFromDB.getClass().getMethod("getUserRoleList");
        for (Object role : (List) userRoleListMethod.invoke(userFromDB)) {
            Field roleField = role.getClass().getDeclaredField("role");
            roleField.setAccessible(true);
            String roleValue = (String) roleField.get(role);
            roleList.add(roleValue);
        }
        return roleList;
    }

    public void setUserServiceImpl(Object userServiceImpl) {
        this.userServiceImpl= userServiceImpl;
    }

}
```