How to use Custom Captcha and Authentication Manager in Spring Security

Spring Configuration

```
<http auto-config="false" use-expressions="true"
      entry-point-ref="loginUrlAuthenticationEntryPoint">
    <intercept-url pattern="/index" access="permitAll"/>
    <intercept-url pattern="/kayit" access="permitAll"/>
    <intercept-url pattern="/sifre-hatirlatma" access="permitAll"/>
    <intercept-url pattern="/checkUser" access="permitAll"/>
    <intercept-url pattern="favicon.ico" access="permitAll"/>
    <intercept-url pattern="/kullanici/**"
access="hasAnyAuthority('ROLE_ADMIN','ROLE_OFFICER','ROLE_CLIENT')"/>

    <access-denied-handler error-page="/403"/>
    <custom-filter position="FORM_LOGIN_FILTER" ref="customUsernamePasswordAuthenticationFilter"/>

    <logout logout-url="/cikis"
            invalidate-session="true"
            logout-success-url="/index?logout=true"/>
    <csrf/>
</http>

<beans:bean id="loginUrlAuthenticationEntryPoint"
            class="org.springframework.security.web.authentication.LoginUrlAuthenticationEntryPoint">
    <beans:constructor-arg value="/index"/>
</beans:bean>

<beans:bean id="customUsernamePasswordAuthenticationFilter"
            class="com.codesenior.telif.security.CustomUsernamePasswordAuthenticationFilter">
    <beans:property name="filterProcessesUrl" value="/login" />
    <beans:property name="authenticationFailureHandler" ref="failureHandler"/>
    <beans:property name="authenticationSuccessHandler" ref="successHandler"/>
    <beans:property name="authenticationManager" ref="authenticationManager"/>
</beans:bean>

<beans:bean id="successHandler"
            class="org.springframework.security.web.authentication.SavedRequestAwareAuthenticationSuc
cessHandler">
    <beans:property name="defaultTargetUrl" value="/kullanici/index"/>
</beans:bean>

<beans:bean id="failureHandler"
            class="org.springframework.security.web.authentication.SimpleUrlAuthenticationFailureHand
ler">
    <beans:property name="defaultFailureUrl" value="/index?error=true"/>
</beans:bean>

<!-- Select users and user_roles from database -->
<authentication-manager id="authenticationManager">
    <authentication-provider user-service-ref="customUserDetailsService">
        <password-encoder hash="plaintext">
        </password-encoder>
    </authentication-provider>
</authentication-manager>
```

Login Page

```
<form method="post" action="<c:url value="/login"/>">
    <header class="in_head_1">
        <h2 id="pageTitle">ADMIN LOGIN</h2>
    </header>
    <article class="art_loop">
        <label>Username:</label>
        <div class="pure-control-group">
            <input name="username" type="text" placeholder="Username" maxlength="12"
required="true"/>
        </div>
        <label>Password:</label>
        <div class="pure-control-group">
            <input name="password" type="password" placeholder="Password" maxlength="20"
required="true"/>
        </div>
        <c:set var="rand1"><%= java.lang.Math.round(java.lang.Math.random() * 10) %>
        </c:set>
        <c:set var="rand2"><%= java.lang.Math.round(java.lang.Math.random() * 10) %>
        </c:set>

        <c:set var="rand1" value="${rand1}" scope="session"/>
        <c:set var="rand2" value="${rand2}" scope="session"/>
        <label>What is ${rand1}+${rand2} ?</label>

        <div class="pure-control-group">
            <input name="captcha" type="text" placeholder="What is ${rand1}+${rand2} ?"
required="true"/>
        </div>

        <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}"/>

        <div class="pure-controls">
            <button type="submit" class="buttonSubmit">Login</button>
        </div>
    </article>
</form>
```

CustomUsernamePasswordAuthenticationFilter

```
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 {
        String id = request.getParameter("captcha");
        HttpSession session = request.getSession();
        int rand1 = Integer.parseInt((String) session.getAttribute("rand1"));
        int rand2 = Integer.parseInt((String) session.getAttribute("rand2"));
        if (Integer.valueOf(id) != rand1 + rand2) {
            throw new AuthenticationServiceException("Please correctly answer: "+rand1+ "+"+rand2);
        }
        return super.attemptAuthentication(request, response);
    }
}
```

CustomUserDetailsService

```
@Service
public class CustomUserDetailsService implements UserDetailsService {

    @Autowired
    private UserService userService;

    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;
    }

    public UserDetails loadUserByUsername(String userName)
            throws UsernameNotFoundException {
        User domainUser = userService.getUser(userName);

        boolean enabled = true;
        boolean accountNonExpired = true;
        boolean credentialsNonExpired = true;
        boolean accountNonLocked = true;

        return new org.springframework.security.core.userdetails.User(
                domainUser.getUsername(),
                domainUser.getPassword(),
                enabled,
                accountNonExpired,
                credentialsNonExpired,
                accountNonLocked,
                getAuthorities(domainUser.getUserRoleList())
        );
    }

    public Collection<? extends GrantedAuthority> getAuthorities(List<UserRole> userRoleList) {
        return getGrantedAuthorities(getRoles(userRoleList));
    }

    public List<String> getRoles(List<UserRole> userRoleList) {

        List<String> roles = new ArrayList<String>();

        for (UserRole userRole : userRoleList) {
            roles.add(userRole.getRole());
        }
        return roles;
```

```
    }
}
```