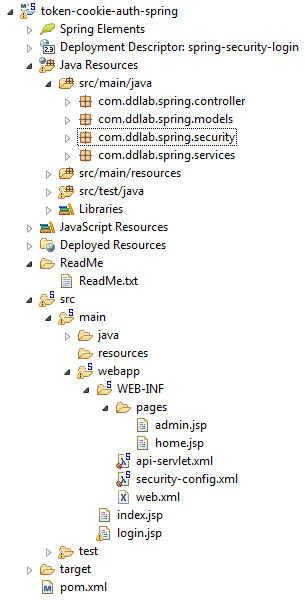
Cookie with Token based authentication in Spring with Restful web service (Apollo Type)

# Project Structure



# Maven(pom.xml)

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>token-cookie-auth-spring</groupId>

<artifactId>token-cookie-auth-spring</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>cookiebasedspringrestauth2 Maven Webapp</name>

<url>http://maven.apache.org</url>

<properties>

<!-- Spring -->

<org.springframework.version>4.0.5.RELEASE</org.springframework.version>

<org.springframework.security.version>3.2.4.RELEASE</org.springframework.security.version>

<!-- persistence -->

<hibernate.version>4.3.5.Final</hibernate.version>

<mysql-connector-java.version>5.1.30</mysql-connector-java.version>

<!-- logging -->

<org.slf4j.version>1.7.6</org.slf4j.version>

<logback.version>1.1.1</logback.version>

<!-- various -->

<hibernate-validator.version>5.1.1.Final</hibernate-validator.version>

<!-- util -->

<guava.version>17.0</guava.version>

<commons-lang3.version>3.3.2</commons-lang3.version>

<!-- testing -->

<org.hamcrest.version>1.3</org.hamcrest.version>

<junit.version>4.11</junit.version>

<mockito.version>1.9.5</mockito.version>

<httpcore.version>4.3.2</httpcore.version>

<httpclient.version>4.3.3</httpclient.version>

<rest-assured.version>2.3.1</rest-assured.version>

<!-- Maven plugins -->

<maven-compiler-plugin.version>3.1</maven-compiler-plugin.version>

<maven-war-plugin.version>2.4</maven-war-plugin.version>

<maven-surefire-plugin.version>2.17</maven-surefire-plugin.version>

<cargo-maven2-plugin.version>1.4.8</cargo-maven2-plugin.version>

</properties>

<dependencies>

<!-- Spring Security -->

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-web</artifactId>

<version>${org.springframework.security.version}</version>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-config</artifactId>

<version>${org.springframework.security.version}</version>

</dependency>

<!-- Spring -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${org.springframework.version}</version>

<exclusions>

<exclusion>

<artifactId>commons-logging</artifactId>

<groupId>commons-logging</groupId>

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-expression</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!-- web -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.0.1</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

<scope>runtime</scope>

</dependency>

<!-- marshalling -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.2.2</version>

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-core</artifactId>

<version>2.2.3</version>

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.2.3</version>

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-annotations</artifactId>

<version>2.2.3</version>

</dependency>

<!-- util -->

<dependency>

<groupId>com.google.guava</groupId>

<artifactId>guava</artifactId>

<version>${guava.version}</version>

</dependency>

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-lang3</artifactId>

<version>3.1</version>

</dependency>

<!-- logging -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>${org.slf4j.version}</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>${logback.version}</version>

<!-- <scope>runtime</scope> -->

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>jcl-over-slf4j</artifactId>

<version>${org.slf4j.version}</version>

</dependency>

<dependency> <!-- needed to bridge to slf4j for projects that use the log4j APIs directly -->

<groupId>org.slf4j</groupId>

<artifactId>log4j-over-slf4j</artifactId>

<version>${org.slf4j.version}</version>

</dependency>

<!-- test scoped -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit-dep</artifactId>

<version>${junit.version}</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.hamcrest</groupId>

<artifactId>hamcrest-core</artifactId>

<version>${org.hamcrest.version}</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.hamcrest</groupId>

<artifactId>hamcrest-library</artifactId>

<version>${org.hamcrest.version}</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>${mockito.version}</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.apache.httpcomponents</groupId>

<artifactId>httpclient</artifactId>

<version>4.3.6</version>

</dependency>

</dependencies>

<build>

<finalName>${pom.artifactId}</finalName>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.3</version>

<configuration>

<source>1.7</source>

<target>1.7</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

# Web Configuration (WEB-INF/web.xml)

<web-app id=*"WebApp\_ID"* version=*"2.4"*

xmlns=*"http://java.sun.com/xml/ns/j2ee"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://java.sun.com/xml/ns/j2ee*

*http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd"*>

<display-name>spring-security-login</display-name>

<servlet>

<servlet-name>api</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>api</servlet-name>

<url-pattern>/api/\*</url-pattern>

</servlet-mapping>

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>

/WEB-INF/security-config.xml

</param-value>

</context-param>

<!-- Spring Security -->

<filter>

<filter-name>springSecurityFilterChain</filter-name>

<filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>

</filter>

<filter-mapping>

<filter-name>springSecurityFilterChain</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<welcome-file-list>

<welcome-file>login.jsp</welcome-file>

</welcome-file-list>

</web-app>

# Spring MVC Configuration(WEB-INF/api-servlet.xml)

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xmlns:aop=*"http://www.springframework.org/schema/aop"*

xmlns:tx=*"http://www.springframework.org/schema/tx"*

xmlns:mvc=*"http://www.springframework.org/schema/mvc"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd*

*http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd*

*http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd*

*http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd"*>

<context:component-scan base-package=*"com.ddlab.spring.\*"* />

<context:annotation-config />

<mvc:annotation-driven />

<bean

class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<property name=*"prefix"*>

<value>/WEB-INF/pages/</value>

</property>

<property name=*"suffix"*>

<value>.jsp</value>

</property>

</bean>

</beans>

# Spring Security Configuration(WEB-INF/spring-security.xml)

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans:beans xmlns=*"http://www.springframework.org/schema/security"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:beans=*"http://www.springframework.org/schema/beans"*

xmlns:sec=*"http://www.springframework.org/schema/security"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"http://www.springframework.org/schema/security http://www.springframework.org/schema/security/spring-security-3.2.xsd*

*http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd"*>

<global-method-security secured-annotations=*"enabled"* pre-post-annotations=*"enabled"* />

<context:component-scan base-package=*"com.ddlab.spring.\*"* />

<context:annotation-config />

<beans:bean id=*"bankApiImpl"* class=*"com.ddlab.spring.models.BankApiImpl"* />

<beans:bean id=*"restAuthenticationEntryPoint"*

class=*"com.ddlab.spring.security.RestAuthenticationEntryPoint"* />

<beans:bean id=*"authenticationTokenProcessingFilter"*

class=*"com.ddlab.spring.security.AuthenticationTokenProcessingFilter"*>

</beans:bean>

<beans:bean id=*"logoutHandler"* class=*"com.ddlab.spring.security.LogoutSuccessHandler"* />

<http use-expressions=*"true"* entry-point-ref=*"restAuthenticationEntryPoint"*

disable-url-rewriting=*"true"*>

<intercept-url pattern=*"/api/\*\*"* access=*"isAuthenticated()"* />

<sec:custom-filter ref=*"authenticationTokenProcessingFilter"*

position=*"BASIC\_AUTH\_FILTER"* />

<sec:form-login authentication-success-handler-ref=*"mySuccessHandler"*

authentication-failure-handler-ref=*"myFailureHandler"* />

<logout success-handler-ref=*"logoutHandler"* />

</http>

<beans:bean id=*"mySuccessHandler"*

class=*"com.ddlab.spring.security.MySavedRequestAwareAuthenticationSuccessHandler"* />

<beans:bean id=*"myFailureHandler"*

class=*"org.springframework.security.web.authentication.SimpleUrlAuthenticationFailureHandler"* />

<authentication-manager alias=*"authenticationManager"*

erase-credentials=*"false"*>

<authentication-provider>

<user-service>

<user name=*"admin"* password=*"admin"* authorities=*"ROLE\_ADMIN"* />

<user name=*"d"* password=*"d"* authorities=*"ROLE\_USER"* />

<user name=*"s"* password=*"s"* authorities=*"ROLE\_STAFF"* />

</user-service>

</authentication-provider>

</authentication-manager>

</beans:beans>

# JSP Pages(pages/admin.jsp)

<%@ taglib prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"* %>

<%@ page contentType=*"text/html;charset=UTF-8"* language=*"java"* %>

<html>

<head>

<title>Admin</title>

</head>

<body>

<a href=<c:url value=*"/j\_spring\_security\_logout"*/>>Logout</a><br/>

<h1>Only Admin allowed here</h1>

</body>

</html>

# JSP Pages(pages/home.jsp)

<%@ taglib prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"* %>

<html>

<head>

<title>Home</title>

</head>

<body>

<a href=<c:url value=*"/j\_spring\_security\_logout"*/>>Logout</a><br/>

<!-- <sec:authorize ifAnyGranted="ROLE\_ADMIN">

<h1>Only admin can see this</h1><br/>

<a href="admin"> Admin Home </a>

</sec:authorize> -->

<h1>Welcome, This page is for normal user</h1>

</body>

</html>

# JSP Pages(webapp/login.jsp)

<%@ taglib prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"* %>

<html>

<head>

<title>Login</title>

</head>

<body>

<c:if test=*"*${**not empty** param.error}*"*>

<font color=*"red"*>

Login error. <br />

Reason : ${sessionScope["SPRING\_SECURITY\_LAST\_EXCEPTION"].message}

</font>

</c:if>

<form method=*"POST"* action=*"*<c:url value=*"/j\_spring\_security\_check"* />*"*>

<table>

<tr>

<td align=*"right"*>Username</td>

<td><input type=*"text"* name=*"j\_username"* /></td>

</tr>

<tr>

<td align=*"right"*>Password</td>

<td><input type=*"password"* name=*"j\_password"* /></td>

</tr>

<tr>

<td colspan=*"2"* align=*"right"*>

<input type=*"submit"* value=*"Login"* />

</td>

</tr>

</table>

</form>

</body>

</html>

# Java Source Code

## Entity Layer

**BankApi.java**

**package** com.ddlab.spring.models;

**import** org.springframework.security.access.annotation.Secured;

**import** org.springframework.security.access.prepost.PreAuthorize;

**public** **interface** BankApi {

@PreAuthorize ("hasRole('ROLE\_USER')")

**public** WithdrawAmountModel withdrawAmount(String accountNo,**int** amount);

@Secured({"ROLE\_ADMIN"})

**public** UserNamePasswordModel resetPasswod(BankCustomerModel customer);

@PreAuthorize ("hasRole('ROLE\_STAFF')")

**public** String approveLoan(BankCustomerModel customer);

}

**BankApiImpl.java**

**package** com.ddlab.spring.models;

**public** **class** BankApiImpl **implements** BankApi {

@Override

**public** WithdrawAmountModel withdrawAmount(String accountNo, **int** amount) {

WithdrawAmountModel amountModel = **new** WithdrawAmountModel();

amountModel.setAmountWithdrawn(amount);

amountModel.setAvailableBalance(50000);

**return** amountModel;

}

@Override

**public** UserNamePasswordModel resetPasswod(BankCustomerModel customer) {

UserNamePasswordModel unamePwdModel = **new** UserNamePasswordModel();

unamePwdModel.setPassword("abcd$pass00rd");

unamePwdModel.setUserName(customer.getUserName());

**return** unamePwdModel;

}

@Override

**public** String approveLoan(BankCustomerModel customer) {

**return** "The loan account for "

+ customer.getUserName()

+ " has been approved and \n+ loan account has been created in the system";

}

}

**BankCustomerModel.java**

**package** com.ddlab.spring.models;

**import** com.fasterxml.jackson.annotation.JsonProperty;

**public** **class** BankCustomerModel {

@JsonProperty("accountNo")

**private** String accountNo;

@JsonProperty("userName")

**private** String userName;

@JsonProperty("userId")

**private** String userId;

**public** String getAccountNo() {

**return** accountNo;

}

**public** **void** setAccountNo(String accountNo) {

**this**.accountNo = accountNo;

}

**public** String getUserName() {

**return** userName;

}

**public** **void** setUserName(String userName) {

**this**.userName = userName;

}

**public** String getUserId() {

**return** userId;

}

**public** **void** setUserId(String userId) {

**this**.userId = userId;

}

}

**UserNamePasswordModel.java**

**package** com.ddlab.spring.models;

**public** **class** UserNamePasswordModel {

**private** String userName;

**private** String password;

**public** String getUserName() {

**return** userName;

}

**public** **void** setUserName(String userName) {

**this**.userName = userName;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

}

**WithdrawAmountModel.java**

**package** com.ddlab.spring.models;

**public** **class** WithdrawAmountModel {

**private** **int** amountWithdrawn;

**private** **int** availableBalance;

**public** **int** getAmountWithdrawn() {

**return** amountWithdrawn;

}

**public** **void** setAmountWithdrawn(**int** amountWithdrawn) {

**this**.amountWithdrawn = amountWithdrawn;

}

**public** **int** getAvailableBalance() {

**return** availableBalance;

}

**public** **void** setAvailableBalance(**int** availableBalance) {

**this**.availableBalance = availableBalance;

}

}

## Service Layer

**UserService.java**

**package** com.ddlab.spring.services;

**import** org.springframework.security.core.userdetails.UserDetails;

**public** **interface** UserService {

**public** UserDetails getUserByUserName(String userName);

}

**UserServiceImpl.java**

**package** com.ddlab.spring.services;

**import** java.util.ArrayList;

**import** java.util.List;

**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.stereotype.Service;

@Service("userService")

**public** **class** UserServiceImpl **implements** UserService {

@Override

**public** UserDetails getUserByUserName(String userName) {

UserDetails userDetails = **null**;

//A temporary solution, it should come from database

List<GrantedAuthority> gList = **new** ArrayList<GrantedAuthority>();

String password = **null**;

**if**(userName.equals("d")) {

GrantedAuthority ga = **new** SimpleGrantedAuthority("ROLE\_USER");

gList.add(ga);

password = "d";

}

**else** **if**(userName.equals("admin")) {

GrantedAuthority ga = **new** SimpleGrantedAuthority("ROLE\_ADMIN");

gList.add(ga);

password = "admin";

}

**else** **if**(userName.equals("s")) {

GrantedAuthority ga = **new** SimpleGrantedAuthority("ROLE\_STAFF");

gList.add(ga);

password = "s";

}

userDetails = **new** User(userName, password, gList);

**return** userDetails;

}

}

**Controller and REST Layer**

**package** com.ddlab.spring.controller;

**import** org.springframework.security.core.Authentication;

**import** org.springframework.security.core.context.SecurityContextHolder;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.servlet.ModelAndView;

@Controller

**public** **class** HomePageController {

@RequestMapping(value = "/home" , method = RequestMethod.***GET***)

**public** ModelAndView getHomePage() {

Authentication auth = SecurityContextHolder.*getContext*().getAuthentication();

String viewName = "home";

**if**( auth.getName().equals("admin"))

viewName = "admin";

ModelAndView model = **new** ModelAndView();

model.setViewName(viewName);

**return** model;

}

}

**BankApiController.java**

**package** com.ddlab.spring.controller;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.MediaType;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.ResponseBody;

**import** com.ddlab.spring.models.BankApi;

**import** com.ddlab.spring.models.BankCustomerModel;

**import** com.ddlab.spring.models.UserNamePasswordModel;

**import** com.ddlab.spring.models.WithdrawAmountModel;

@Controller

@RequestMapping(value = "/bankapi")

**public** **class** BankApiController {

@Autowired

**private** BankApi bankApi;

@RequestMapping(value = "/withdraw/{accountNo}/{amount}", method = RequestMethod.***GET***, produces = { MediaType.***APPLICATION\_JSON\_VALUE*** })

@ResponseBody()

**public** WithdrawAmountModel withdraw(

@PathVariable(value = "accountNo") String accountNo,

@PathVariable(value = "amount") Integer amount) {

**return** bankApi.withdrawAmount(accountNo, amount);

}

@RequestMapping(value = "/reset", method = RequestMethod.***PUT***, produces = { MediaType.***APPLICATION\_JSON\_VALUE*** })

@ResponseBody()

**public** UserNamePasswordModel resetPasswod(@RequestBody BankCustomerModel customer) {

**return** bankApi.resetPasswod(customer);

}

}

## Security Layer

**TokenUtil.java**

**package** com.ddlab.spring.security;

**import** java.security.MessageDigest;

**import** java.security.NoSuchAlgorithmException;

**import** org.springframework.security.core.userdetails.UserDetails;

**import** org.springframework.security.crypto.codec.Hex;

**public** **class** TokenUtils {

**public** **static** **final** String ***MAGIC\_KEY*** = "obfuscate";

**public** **static** String createToken(UserDetails userDetails) {

/\* Expires in one hour \*/

**long** expires = System.*currentTimeMillis*() + 1000L \* 60 \* 60;

/\* Expires in 2 min \*/

// long expires = System.currentTimeMillis() + 1000L \* 60 \* 2;

StringBuilder tokenBuilder = **new** StringBuilder();

tokenBuilder.append(userDetails.getUsername());

tokenBuilder.append(":");

tokenBuilder.append(expires);

tokenBuilder.append(":");

tokenBuilder.append(TokenUtils.*computeSignature*(userDetails, expires));

**return** tokenBuilder.toString();

}

**public** **static** String computeSignature(UserDetails userDetails, **long** expires) {

StringBuilder signatureBuilder = **new** StringBuilder();

signatureBuilder.append(userDetails.getUsername());

signatureBuilder.append(":");

signatureBuilder.append(expires);

signatureBuilder.append(":");

// signatureBuilder.append(password);

signatureBuilder.append(userDetails.getPassword());

signatureBuilder.append(":");

signatureBuilder.append(TokenUtils.***MAGIC\_KEY***);

MessageDigest digest;

**try** {

digest = MessageDigest.*getInstance*("MD5");

} **catch** (NoSuchAlgorithmException e) {

**throw** **new** IllegalStateException("No MD5 algorithm available!");

}

**return** **new** String(Hex.*encode*(digest.digest(signatureBuilder.toString()

.getBytes())));

}

**public** **static** String getUserNameFromToken(String authToken) {

**if** (**null** == authToken) {

**return** **null**;

}

String[] parts = authToken.split(":");

**return** parts[0];

}

**public** **static** **boolean** validateToken(String authToken,

UserDetails userDetails) {

String[] parts = authToken.split(":");

**long** expires = Long.*parseLong*(parts[1]);

String signature = parts[2];

**if** (expires < System.*currentTimeMillis*()) {

**return** **false**;

}

**return** signature.equals(TokenUtils.*computeSignature*(userDetails,

expires));

}

}

**RestAuthenticationEntryPoint.java**

**package** com.ddlab.spring.security;

**import** java.io.IOException;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** org.springframework.security.core.AuthenticationException;

**import** org.springframework.security.web.AuthenticationEntryPoint;

**import** org.springframework.stereotype.Component;

/\*\*

\* The Entry Point will not redirect to any sort of Login - it will return the 401

\*/

@Component

**public** **final** **class** RestAuthenticationEntryPoint **implements** AuthenticationEntryPoint {

@Override

**public** **void** commence(**final** HttpServletRequest request, **final** HttpServletResponse response, **final** AuthenticationException authException) **throws** IOException {

response.sendError(HttpServletResponse.***SC\_UNAUTHORIZED***, "Unauthorized");

}

}

**MySavedRequestAwareAuthenticationSuccessHandler.java**

**package** com.ddlab.spring.security;

**import** java.io.IOException;

**import** javax.servlet.ServletException;

**import** javax.servlet.http.Cookie;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** org.apache.commons.codec.binary.Base64;

**import** org.springframework.security.core.Authentication;

**import** org.springframework.security.core.userdetails.UserDetails;

**import** org.springframework.security.web.authentication.SimpleUrlAuthenticationSuccessHandler;

**import** org.springframework.security.web.savedrequest.HttpSessionRequestCache;

**import** org.springframework.security.web.savedrequest.RequestCache;

**import** org.springframework.security.web.savedrequest.SavedRequest;

**import** org.springframework.util.StringUtils;

**public** **class** MySavedRequestAwareAuthenticationSuccessHandler **extends**

SimpleUrlAuthenticationSuccessHandler {

**private** RequestCache requestCache = **new** HttpSessionRequestCache();

@Override

**public** **void** onAuthenticationSuccess(**final** HttpServletRequest request,

**final** HttpServletResponse response,

**final** Authentication authentication) **throws** ServletException,

IOException {

**final** SavedRequest savedRequest = requestCache.getRequest(request,

response);

UserDetails userDetails = (UserDetails) authentication.getPrincipal();

**if** (savedRequest == **null**) {

String cookieToken = TokenUtils.*createToken*(userDetails);

System.***out***.println("Now cookie Token ::::"+cookieToken);

String ecodedCookieToken = **new** String(

Base64.*encodeBase64*(cookieToken.getBytes()));

System.***out***.println("Now ecodedCookieToken ::::"+ecodedCookieToken);

// Create the cookie

Cookie cook = **new** Cookie("APPASSERTION", ecodedCookieToken);

cook.setPath("/");

cook.setMaxAge(1 \* 2 \* 60);// 24 hours is 60 seconds x 60 minutes x

// 24 hours

cook.setVersion(1);

response.addCookie(cook);

clearAuthenticationAttributes(request);

setDefaultTargetUrl("/api/home");

getRedirectStrategy().sendRedirect(request, response, "/api/home");

**return**;

} **else** {

}

**final** String targetUrlParameter = getTargetUrlParameter();

**if** (isAlwaysUseDefaultTargetUrl()

|| (targetUrlParameter != **null** && StringUtils.*hasText*(request

.getParameter(targetUrlParameter)))) {

requestCache.removeRequest(request, response);

clearAuthenticationAttributes(request);

**return**;

}

clearAuthenticationAttributes(request);

**super**.onAuthenticationSuccess(request, response, authentication);

}

**public** **void** setRequestCache(**final** RequestCache requestCache) {

**this**.requestCache = requestCache;

}

}

**AuthenticationTokenProcessingFilter.java**

**package** com.ddlab.spring.security;

**import** java.io.IOException;

**import** java.util.HashMap;

**import** java.util.Map;

**import** javax.servlet.FilterChain;

**import** javax.servlet.ServletException;

**import** javax.servlet.ServletRequest;

**import** javax.servlet.ServletResponse;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.beans.factory.annotation.Qualifier;

**import** org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

**import** org.springframework.security.core.Authentication;

**import** org.springframework.security.core.context.SecurityContextHolder;

**import** org.springframework.security.core.userdetails.UserDetails;

**import** org.springframework.security.crypto.codec.Base64;

**import** org.springframework.security.web.authentication.WebAuthenticationDetailsSource;

**import** org.springframework.web.filter.GenericFilterBean;

**import** com.ddlab.spring.services.UserService;

**public** **class** AuthenticationTokenProcessingFilter **extends** GenericFilterBean {

@Autowired

@Qualifier("userService")

**private** UserService userService;

@Override

**public** **void** doFilter(ServletRequest request, ServletResponse response,

FilterChain chain) **throws** IOException, ServletException {

HttpServletResponse response1 = **null**;

**try** {

response1 = (HttpServletResponse) response;

HttpServletRequest httpRequest = **this**.getAsHttpRequest(request);

String cookieValue = getCookieValueByName(httpRequest,

"APPASSERTION");

System.***out***.println("Cookie Value :::"+cookieValue);

**if** (cookieValue != **null**) {

cookieValue = **new** String(Base64.*decode*(cookieValue.getBytes()));

}

Authentication auth = SecurityContextHolder.*getContext*()

.getAuthentication();

UserDetails userDetails = **null**;

**if** (auth == **null** && cookieValue == **null**) {

// User has not yet logged in

System.***out***.println("user has not yet logged in...........");

} **else** **if** (auth != **null** && cookieValue == **null**) {

userDetails = (UserDetails) auth.getPrincipal();

} **else** **if** (auth == **null** && cookieValue != **null**) {

//It is for the use case where use is passing APPASSERTION as cookie in Firefox rest client

//Get the user name from the cookie token

String cookiTokenUserName = TokenUtils.*getUserNameFromToken*(cookieValue);

userDetails = userService.getUserByUserName(cookiTokenUserName);

UsernamePasswordAuthenticationToken authentication = **new** UsernamePasswordAuthenticationToken(

userDetails, **null**, userDetails.getAuthorities());

authentication.setDetails(**new** WebAuthenticationDetailsSource()

.buildDetails(httpRequest));

SecurityContextHolder.*getContext*().setAuthentication(

authentication);

**boolean** flag = TokenUtils.*validateToken*(cookieValue,userDetails);

**if** (!flag) {

response1.sendError(HttpServletResponse.***SC\_UNAUTHORIZED***,

"Unauthorized");

}

} **else** **if** (auth != **null** && cookieValue != **null**) {

// do nothing

}

**try** {

chain.doFilter(request, response);

} **catch** (Exception e) {

response1.sendError(HttpServletResponse.***SC\_UNAUTHORIZED***,

"Unauthorized");

}

} **catch** (Exception e) {

e.printStackTrace();

response1.sendError(HttpServletResponse.***SC\_UNAUTHORIZED***,

"Unauthorized");

}

}

**private** HttpServletRequest getAsHttpRequest(ServletRequest request) {

**if** (!(request **instanceof** HttpServletRequest)) {

**throw** **new** RuntimeException("Expecting an HTTP request");

}

**return** (HttpServletRequest) request;

}

**private** String getCookieValueByName(HttpServletRequest httpRequest,

String cookieName) {

String completeCookies = httpRequest.getHeader("Cookie");

completeCookies = completeCookies.replaceAll("\"", "");

String cookieValue = **null**;

**if** (completeCookies != **null**) {

String[] cooks = completeCookies.split(";");

Map<String, String> cookieMap = **new** HashMap<String, String>();

**for** (String cookie : cooks) {

String[] keyVal = cookie.split("=",2);

cookieMap.put(keyVal[0].trim(), keyVal[1].trim());

}

cookieValue = cookieMap.get(cookieName);

}

**return** cookieValue;

}

}

**LogoutSuccessHandler.java**

**package** com.ddlab.spring.security;

**import** java.io.IOException;

**import** javax.servlet.ServletException;

**import** javax.servlet.http.Cookie;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** org.springframework.security.core.Authentication;

**import** org.springframework.security.web.authentication.logout.SimpleUrlLogoutSuccessHandler;

**public** **class** LogoutSuccessHandler **extends** SimpleUrlLogoutSuccessHandler {

@Override

**public** **void** onLogoutSuccess(HttpServletRequest request,

HttpServletResponse response, Authentication authentication)

**throws** IOException, ServletException {

Cookie cook = **new** Cookie("APPASSERTION", "someJunkValue");

cook.setPath("/");

cook.setMaxAge(0);// 24 hours is 60 seconds x 60 minutes x 24 hours

cook.setVersion(1);

response.addCookie(cook);

// do whatever you want

**super**.onLogoutSuccess(request, response, authentication);

}

}

**How to Use**

POST

<http://localhost:8080/token-cookie-auth-spring/j_spring_security_check?j_username=d&j_password=d>

GET

<http://localhost:8080/token-cookie-auth-spring/api/bankapi/withdraw/12345/500>

<http://localhost:8080/token-cookie-auth-spring/api/bankapi/reset>

PUT

Request Body

{

"accountNo": "12345",

"userName": "Deb",

"userId":"222"

}

GET

<http://localhost:8080/token-cookie-auth-spring/j_spring_security_logout>

Login from Browser

<http://localhost:8080/token-cookie-auth-spring>

If you want to make rest call from another browser, use the following

Header

Cookie

APPASSERTION=adascsa565sdgdgddfdd