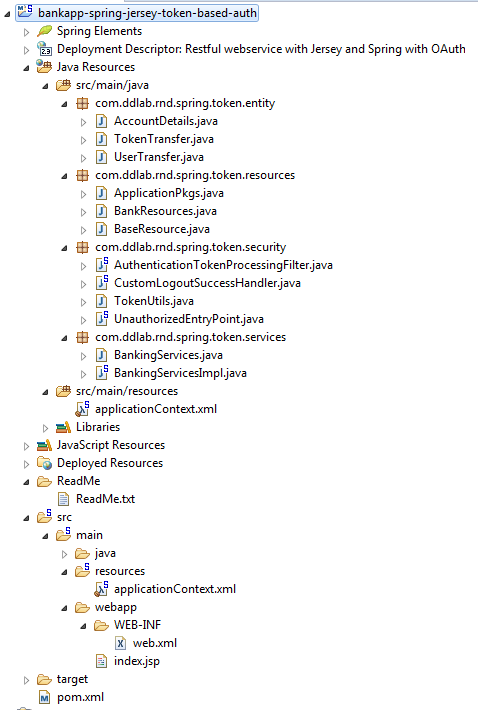
Token based Authentication in Spring Security with Jersey

# **Project Structure**



# **Maven Build (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>bankapp-spring-jersey-token-based-auth</groupId>

<artifactId>bankapp-spring-jersey-token-based-auth</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>bankapp-spring-jersey-token-based-auth Maven Webapp</name>

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

<properties>

<org.springframework.version>4.0.7.RELEASE</org.springframework.version>

<log4j.version>1.2.17</log4j.version>

<jdk.version>1.7</jdk.version>

<spring.security.version>3.2.5.RELEASE</spring.security.version>

</properties>

<dependencies>

<dependency>

<groupId>javax.ws.rs</groupId>

<artifactId>javax.ws.rs-api</artifactId>

<version>2.0</version>

</dependency>

<dependency>

<groupId>org.glassfish.jersey.containers</groupId>

<artifactId>jersey-container-servlet</artifactId>

<version>2.6</version>

</dependency>

<dependency>

<groupId>org.glassfish.jersey.core</groupId>

<artifactId>jersey-client</artifactId>

<version>2.6</version>

</dependency>

<dependency>

<groupId>org.codehaus.jackson</groupId>

<artifactId>jackson-core-asl</artifactId>

<version>1.9.13</version>

</dependency>

<dependency>

<groupId>org.codehaus.jackson</groupId>

<artifactId>jackson-mapper-asl</artifactId>

<version>1.9.13</version>

</dependency>

<dependency>

<groupId>org.glassfish.jersey.media</groupId>

<artifactId>jersey-media-multipart</artifactId>

<version>2.5.1</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

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

<version>3.1.0</version>

</dependency>

<dependency>

<groupId>org.glassfish.jersey.media</groupId>

<artifactId>jersey-media-json-jackson</artifactId>

<version>2.5.1</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

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

</dependency>

<!-- Expression Language (depends on spring-core) Define this if you use

Spring Expression APIs (org.springframework.expression.\*) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-expression</artifactId>

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

</dependency>

<!-- Bean Factory and JavaBeans utilities (depends on spring-core) Define

this if you use Spring Bean APIs (org.springframework.beans.\*) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

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

</dependency>

<!-- Aspect Oriented Programming (AOP) Framework (depends on spring-core,

spring-beans) Define this if you use Spring AOP APIs (org.springframework.aop.\*) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

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

</dependency>

<!-- Application Context (depends on spring-core, spring-expression, spring-aop,

spring-beans) This is the central artifact for Spring's Dependency Injection

Container and is generally always defined -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

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

</dependency>

<!-- Various Application Context utilities, including EhCache, JavaMail,

Quartz, and Freemarker integration Define this if you need any of these integrations -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

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

</dependency>

<!-- Transaction Management Abstraction (depends on spring-core, spring-beans,

spring-aop, spring-context) Define this if you use Spring Transactions or

DAO Exception Hierarchy (org.springframework.transaction.\*/org.springframework.dao.\*) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

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

</dependency>

<!-- JDBC Data Access Library (depends on spring-core, spring-beans, spring-context,

spring-tx) Define this if you use Spring's JdbcTemplate API (org.springframework.jdbc.\*) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

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

</dependency>

<!-- Object-to-Relation-Mapping (ORM) integration with Hibernate, JPA,

and iBatis. (depends on spring-core, spring-beans, spring-context, spring-tx)

Define this if you need ORM (org.springframework.orm.\*) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

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

</dependency>

<!-- Object-to-XML Mapping (OXM) abstraction and integration with JAXB,

JiBX, Castor, XStream, and XML Beans. (depends on spring-core, spring-beans,

spring-context) Define this if you need OXM (org.springframework.oxm.\*) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-oxm</artifactId>

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

</dependency>

<!-- Web application development utilities applicable to both Servlet and

Portlet Environments (depends on spring-core, spring-beans, spring-context)

Define this if you use Spring MVC, or wish to use Struts, JSF, or another

web framework with Spring (org.springframework.web.\*) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

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

</dependency>

<!-- Spring MVC for Servlet Environments (depends on spring-core, spring-beans,

spring-context, spring-web) Define this if you use Spring MVC with a Servlet

Container such as Apache Tomcat (org.springframework.web.servlet.\*) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

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

</dependency>

<!-- Spring MVC for Portlet Environments (depends on spring-core, spring-beans,

spring-context, spring-web) Define this if you use Spring MVC with a Portlet

Container (org.springframework.web.portlet.\*) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc-portlet</artifactId>

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

</dependency>

<!-- Support for testing Spring applications with tools such as JUnit and

TestNG This artifact is generally always defined with a 'test' scope for

the integration testing framework and unit testing stubs -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

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

<scope>test</scope>

</dependency>

<dependency>

<groupId>javax.annotation</groupId>

<artifactId>jsr250-api</artifactId>

<version>1.0</version>

</dependency>

<dependency>

<groupId>javax.inject</groupId>

<artifactId>javax.inject</artifactId>

<version>1</version>

</dependency>

<dependency>

<groupId>javax.enterprise</groupId>

<artifactId>cdi-api</artifactId>

<version>1.1-20130403</version>

</dependency>

<!-- Jersey Spring Integration -->

<dependency>

<groupId>org.glassfish.jersey.ext</groupId>

<artifactId>jersey-spring3</artifactId>

<version>2.10.1</version>

</dependency>

<!-- log4j -->

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>${log4j.version}</version>

</dependency>

<!-- Spring Security -->

<dependency>

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

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

<version>${spring.security.version}</version>

</dependency>

<dependency>

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

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

<version>${spring.security.version}</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

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

<version>3.0.1</version>

<scope>provided</scope>

</dependency>

</dependencies>

<build>

<finalName>bankapp-spring-jersey-token-based-auth</finalName>

<plugins>

<plugin>

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

<configuration>

<source>${jdk.version}</source>

<target>${jdk.version}</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

# **Spring configuration(src/main/resources/applicationContext.xml)**

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

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

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

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

xsi:schemaLocation=*"*

*http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.2.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.2.xsd*

*http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-3.2.xsd*

*http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-3.2.xsd*

*http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.2.xsd*

*http://www.springframework.org/schema/security http://www.springframework.org/schema/security/spring-security-3.2.xsd"*>

<context:annotation-config />

<tx:annotation-driven transaction-manager=*"transactionManager"* />

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

<bean id=*"logoutHandler"*

class=*"com.ddlab.rnd.spring.token.security.CustomLogoutSuccessHandler"* />

<!-- SPRING SECURITY SETUP -->

<bean id=*"passwordEncoder"*

class=*"org.springframework.security.crypto.password.StandardPasswordEncoder"*>

<constructor-arg value=*"ThisIsASecretSoChangeMe"* />

</bean>

<bean id=*"unauthorizedEntryPoint"* class=*"com.ddlab.rnd.spring.token.security.UnauthorizedEntryPoint"* />

<bean id=*"authenticationTokenProcessingFilter"*

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

<!-- <constructor-arg ref="userDao" /> -->

</bean>

<security:authentication-manager id=*"authenticationManager"*>

<!-- <security:authentication-provider user-service-ref="userDao"> -->

<security:authentication-provider>

<security:user-service>

<security:user name=*"user"* password=*"user"*

authorities=*"ROLE\_USER"* />

<security:user name=*"piku"* password=*"piku"*

authorities=*"ROLE\_USER"* />

<security:user name=*"deba"* password=*"deba"*

authorities=*"ROLE\_ADMIN"* />

</security:user-service>

<!-- <security:password-encoder ref="passwordEncoder"></security:password-encoder> -->

</security:authentication-provider>

</security:authentication-manager>

<security:http realm=*"Protected API"* use-expressions=*"true"*

auto-config=*"false"* create-session=*"stateless"* entry-point-ref=*"unauthorizedEntryPoint"*

authentication-manager-ref=*"authenticationManager"*>

<security:logout logout-url=*"/api/1/bankingservices/logout"*

success-handler-ref=*"logoutHandler"* invalidate-session=*"true"*/>

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

position=*"FORM\_LOGIN\_FILTER"* />

<security:intercept-url pattern=*"/api/1/bankingservices/authenticate"*

access=*"permitAll"* />

<security:intercept-url pattern=*"/api/1/bankingservices/\*\*"*

access=*"isAuthenticated()"* />

</security:http>

</beans>

# **Web Configuration(web.xml)**

<!DOCTYPE web-app PUBLIC

"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"

"http://java.sun.com/dtd/web-app\_2\_3.dtd" >

<web-app>

<display-name>Restful webservice with Jersey and Spring with OAuth</display-name>

<context-param>

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

<param-value>classpath:applicationContext.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>

<listener>

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

</listener>

<servlet>

<servlet-name>com.ddlab.rnd.spring.token.resources.ApplicationPkgs</servlet-name>

<servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>

<init-param>

<param-name>javax.ws.rs.Application</param-name>

<param-value>com.ddlab.rnd.spring.token.resources.ApplicationPkgs</param-value>

</init-param>

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

</servlet>

<!-- <servlet>

<servlet-name>mvc-dispatcher</servlet-name>

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

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

</servlet> -->

<servlet-mapping>

<servlet-name>com.ddlab.rnd.spring.token.resources.ApplicationPkgs</servlet-name>

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

</servlet-mapping>

<!-- <servlet-mapping>

<servlet-name>mvc-dispatcher</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping> -->

<welcome-file-list>

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

</welcome-file-list>

</web-app>

# **Java Sources**

## **Java Resource Layer**

**ApplicationPkgs.java**

**package** com.ddlab.rnd.spring.token.resources;

**import** org.glassfish.jersey.jackson.JacksonFeature;

**import** org.glassfish.jersey.server.ResourceConfig;

**public** **class** ApplicationPkgs **extends** ResourceConfig {

**public** ApplicationPkgs() {

**super**(BankResources.**class**,JacksonFeature.**class**);

}

}

**BaseResource.java**

**package** com.ddlab.rnd.spring.token.resources;

**import** java.nio.file.AccessDeniedException;

**import** javax.ws.rs.NotFoundException;

**import** javax.ws.rs.WebApplicationException;

**import** javax.ws.rs.core.Response;

**import** javax.ws.rs.core.Response.Status;

**public** **class** BaseResource {

**protected** WebApplicationException createWebappException(Exception incomingException) {

Status status;

String string = **null**;

**if** (incomingException **instanceof** SecurityException || incomingException **instanceof** AccessDeniedException) {

status = Status.***FORBIDDEN***;

} **else** **if** (incomingException **instanceof** IllegalArgumentException) {

status = Status.***BAD\_REQUEST***;

string=incomingException.getMessage();

} **else** **if** (incomingException **instanceof** NotFoundException) {

status = Status.***NOT\_FOUND***;

} **else** **if** (incomingException **instanceof** Exception) {

status = Status.***INTERNAL\_SERVER\_ERROR***;

} **else** {

status = Status.***INTERNAL\_SERVER\_ERROR***;

}

**return** **new** WebApplicationException(Response.*status*(status).entity(string).type("text/plain").build());

}

}

**BankResources.java**

**package** com.ddlab.rnd.spring.token.resources;

**import** javax.servlet.ServletConfig;

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

**import** javax.ws.rs.Consumes;

**import** javax.ws.rs.FormParam;

**import** javax.ws.rs.GET;

**import** javax.ws.rs.POST;

**import** javax.ws.rs.Path;

**import** javax.ws.rs.Produces;

**import** javax.ws.rs.QueryParam;

**import** javax.ws.rs.core.Context;

**import** javax.ws.rs.core.MediaType;

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

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

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

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

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

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

**import** com.ddlab.rnd.spring.token.entity.AccountDetails;

**import** com.ddlab.rnd.spring.token.entity.TokenTransfer;

**import** com.ddlab.rnd.spring.token.security.TokenUtils;

**import** com.ddlab.rnd.spring.token.services.BankingServices;

@Path("1/bankingservices")

**public** **class** BankResources **extends** BaseResource {

@Context

HttpServletRequest request;

@Context

ServletConfig servletConfig;

@Autowired

@Qualifier(value="bankingService")

**private** BankingServices services;

@Autowired

@Qualifier("authenticationManager")

**private** AuthenticationManager authManager;

@Path("authenticate")

@POST

@Produces(MediaType.***APPLICATION\_JSON***)

**public** TokenTransfer authenticate(@FormParam("username") String username, @FormParam("password") String password)

{

UsernamePasswordAuthenticationToken authenticationToken =

**new** UsernamePasswordAuthenticationToken(username, password);

System.***out***.println("Auth Manager---------->"+authManager);

Authentication authentication = **this**.authManager.authenticate(authenticationToken);

SecurityContextHolder.*getContext*().setAuthentication(authentication);

**return** **new** TokenTransfer(TokenUtils.*createToken*(services.loadUserByUsername(username)));

}

@Path("/userid")

@GET

@Consumes({ MediaType.***APPLICATION\_XML***, MediaType.***APPLICATION\_JSON*** })

@Produces({ MediaType.***APPLICATION\_XML***, MediaType.***APPLICATION\_JSON*** })

**public** AccountDetails getUser(@QueryParam("id") **int** id) {

**if**(id == 0 )

**throw** createWebappException(**new** IllegalArgumentException("Incorrect ID"));

**return** services.getAccountDetails(String.*valueOf*(id));

}

}

## **Entity Layer**

**AccountDetails.java**

**package** com.ddlab.rnd.spring.token.entity;

**import** java.io.IOException;

**import** javax.xml.bind.annotation.XmlAccessType;

**import** javax.xml.bind.annotation.XmlAccessorType;

**import** javax.xml.bind.annotation.XmlElement;

**import** javax.xml.bind.annotation.XmlRootElement;

**import** org.codehaus.jackson.JsonGenerationException;

**import** org.codehaus.jackson.annotate.JsonProperty;

**import** org.codehaus.jackson.map.JsonMappingException;

**import** org.codehaus.jackson.map.ObjectMapper;

@XmlRootElement(name = "AccountDetails")

@XmlAccessorType(XmlAccessType.***FIELD***)

**public** **class** AccountDetails {

@XmlElement(name = "firstName")

@JsonProperty("firstName")

**private** String firstName;

@XmlElement(name = "lastName")

@JsonProperty("lastName")

**private** String lastName;

@XmlElement(name = "userId")

@JsonProperty("userId")

**private** String userId;

@XmlElement(name = "availableBalance")

@JsonProperty("availableBalance")

**private** **long** avlBalance;

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

**public** String getUserId() {

**return** userId;

}

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

**this**.userId = userId;

}

**public** **long** getAvlBalance() {

**return** avlBalance;

}

**public** **void** setAvlBalance(**long** avlBalance) {

**this**.avlBalance = avlBalance;

}

**public** String toJSON() {

ObjectMapper mapper = **new** ObjectMapper();

**try** {

**return** mapper.writeValueAsString(**this**);

} **catch** (JsonGenerationException e) {

e.printStackTrace();

} **catch** (JsonMappingException e) {

e.printStackTrace();

} **catch** (IOException e) {

e.printStackTrace();

}

**return** **null**;

}

}

**UserTransfer.java**

**package** com.ddlab.rnd.spring.token.entity;

**import** java.util.Map;

**public** **class** UserTransfer {

**private** **final** String name;

**private** **final** Map<String, Boolean> roles;

**public** UserTransfer(String userName, Map<String, Boolean> roles) {

**this**.name = userName;

**this**.roles = roles;

}

**public** String getName() {

**return** **this**.name;

}

**public** Map<String, Boolean> getRoles() {

**return** **this**.roles;

}

}

**TokenTransfer.java**

**package** com.ddlab.rnd.spring.token.entity;

**public** **class** TokenTransfer

{

**private** **final** String token;

**public** TokenTransfer(String token)

{

**this**.token = token;

}

**public** String getToken()

{

**return** **this**.token;

}

}

## **Service Layer**

**BankingServices.java**

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

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

**import** com.ddlab.rnd.spring.token.entity.AccountDetails;

**public** **interface** BankingServices {

**public** AccountDetails getAccountDetails(String userId);

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

}

**BankingServicesImpl.java**

**package** com.ddlab.rnd.spring.token.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.Component;

**import** com.ddlab.rnd.spring.token.entity.AccountDetails;

@Component(value="bankingService")

**public** **class** BankingServicesImpl **implements** BankingServices {

@Override

**public** AccountDetails getAccountDetails(String userId) {

AccountDetails actDetails = **new** AccountDetails();

actDetails.setUserId(String.*valueOf*(userId));

actDetails.setAvlBalance(2000L);

actDetails.setFirstName("Debadatta");

actDetails.setLastName("Mishra");

**return** actDetails;

}

@Override

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

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

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

gList.add(ga);

UserDetails userDetails = **new** User(userName,"user",gList);

**return** userDetails;

}

}

## **Security Layer**

**AuthenticationTokenProcessingFilter.java**

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

**import** java.io.IOException;

**import** javax.servlet.FilterChain;

**import** javax.servlet.ServletException;

**import** javax.servlet.ServletRequest;

**import** javax.servlet.ServletResponse;

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

**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.context.SecurityContextHolder;

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

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

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

**import** com.ddlab.rnd.spring.token.services.BankingServicesImpl;

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

@Autowired

@Qualifier("bankingService")

**private** BankingServicesImpl services;

@Override

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

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

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

String authToken = **this**.extractAuthTokenFromRequest(httpRequest);

String userName = TokenUtils.*getUserNameFromToken*(authToken);

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

UserDetails userDetails = services.loadUserByUsername(userName);

**if** (TokenUtils.*validateToken*(authToken, userDetails)) {

UsernamePasswordAuthenticationToken authentication = **new** UsernamePasswordAuthenticationToken(

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

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

.buildDetails(httpRequest));

SecurityContextHolder.*getContext*().setAuthentication(

authentication);

}

}

chain.doFilter(request, response);

}

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

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

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

}

**return** (HttpServletRequest) request;

}

**private** String extractAuthTokenFromRequest(HttpServletRequest httpRequest) {

/\* Get token from header \*/

String authToken = httpRequest.getHeader("X-Auth-Token");

/\* If token not found get it from request parameter \*/

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

authToken = httpRequest.getParameter("token");

}

**return** authToken;

}

}

**CustomSuccessLogoutHandler.java**

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

**import** java.io.IOException;

**import** javax.servlet.ServletException;

**import** javax.servlet.ServletRequest;

**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.AuthenticationManager;

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

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

**import** com.ddlab.rnd.spring.token.services.BankingServicesImpl;

**public** **class** CustomLogoutSuccessHandler **implements** LogoutSuccessHandler {

@Autowired

**private** AuthenticationManager authenticationManager;

@Autowired

@Qualifier("bankingService")

**private** BankingServicesImpl services;

@Override

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

HttpServletResponse response, Authentication authentication)

**throws** IOException, ServletException {

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

String authToken = **this**.extractAuthTokenFromRequest(httpRequest);

String userName = TokenUtils.*getUserNameFromToken*(authToken);

**try** {

TokenUtils.*removeUser*(authToken,services.loadUserByUsername(userName));

// SecurityContextHolder.clearContext();

} **catch** (Exception ex) {

ex.printStackTrace();

}

}

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

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

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

}

**return** (HttpServletRequest) request;

}

**private** String extractAuthTokenFromRequest(HttpServletRequest httpRequest) {

/\* Get token from header \*/

String authToken = httpRequest.getHeader("X-Auth-Token");

/\* If token not found get it from request parameter \*/

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

authToken = httpRequest.getParameter("token");

}

**return** authToken;

}

}

**TokenUtils.java**

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

**import** java.security.MessageDigest;

**import** java.security.NoSuchAlgorithmException;

**import** java.util.Map;

**import** java.util.concurrent.ConcurrentHashMap;

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

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

**public** **class** TokenUtils

{

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

**private** **static** Map<UserDetails,String> *tokenMap* = **new** ConcurrentHashMap<UserDetails,String>();

**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));

*tokenMap*.put(userDetails, tokenBuilder.toString());

**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(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)

{

**boolean** flag = **true**;

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

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

System.***out***.println("Expires token :::"+expires);

String signature = parts[2];

**if** (*tokenMap*.containsKey(userDetails)

&& *tokenMap*.get(userDetails).equals(authToken)) {

**if** (expires > System.*currentTimeMillis*()

&& signature.equals(TokenUtils.*computeSignature*(

userDetails, expires)))

flag = **true**;

**else**

{

flag = **false**;

}

}

**else** {

flag = **false**;

}

**return** flag;

}

**public** **static** **void** removeUser(String authToken,UserDetails userDetails) {

//Validate and then remove it.

**if**(*validateToken*(authToken, userDetails))

*tokenMap*.remove(userDetails);

}

**public** **static** **void** removeUser(UserDetails userDetails) {

//Validate and then remove it.

*tokenMap*.remove(userDetails);

}

}

**UnAuthorizedEntryPoint.java**

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

**import** java.io.IOException;

**import** javax.servlet.ServletException;

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

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

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

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

**public** **class** UnauthorizedEntryPoint **implements** AuthenticationEntryPoint {

@Override

**public** **void** commence(HttpServletRequest request,

HttpServletResponse response, AuthenticationException authException)

**throws** IOException, ServletException {

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

"Unauthorized: Authentication token was either missing or invalid.");

}

}

# **How to Use**

Use Firefox Rest client to test the application.

**Authenticate and get the token**

POST

<http://localhost:8080/bankapp-spring-jersey-token-based-auth/api/1/bankingservices/authenticate>

**Header**

Content-Type

application/x-www-form-urlencoded

**Body**

username=piku&password=piku

Response

{"token":"piku:1434892137808:c3de7857b6dbb3e260272b72351eccc6"}

Invoke Banking service to get the account details

**GET**

<http://localhost:8080/bankapp-spring-jersey-token-based-auth/api/1/bankingservices/userid?id=123>

**Header**

X-Auth-Token

piku:1434829290357:90399f46fbd41231f4d927e7a5ce9cff

and

Accept

application/json

**To Logout**

GET

<http://localhost:8080/bankapp-spring-jersey-token-based-auth/api/1/bankingservices/logout>

**Header**

X-Auth-Token

piku:1434829290357:90399f46fbd41231f4d927e7a5ce9cff