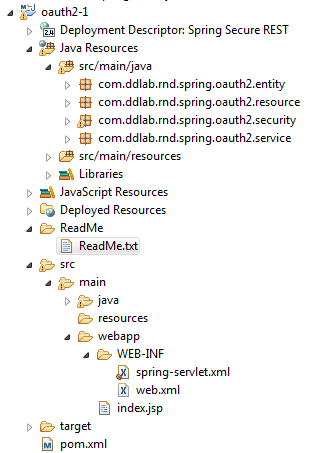
Tailor Made Custom OAuth2 Implementation for better customization

# 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>oauth2-1</groupId>

<artifactId>oauth2-1</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>oauth2-1 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>org.springframework.security.oauth</groupId>

<artifactId>spring-security-oauth2</artifactId>

<version>1.0.0.RELEASE</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

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

<version>3.0.1</version>

<scope>provided</scope>

</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)

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

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

xmlns=*"http://java.sun.com/xml/ns/javaee"* xmlns:web=*"http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*

xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*

id=*"WebApp\_ID"* version=*"2.5"*>

<display-name>Spring Secure REST</display-name>

<context-param>

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

<param-value>/WEB-INF/spring-servlet.xml</param-value>

</context-param>

<listener>

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

</listener>

<listener>

<listener-class>org.springframework.web.context.request.RequestContextListener</listener-class>

</listener>

<filter>

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

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

<init-param>

<param-name>contextAttribute</param-name>

<param-value>**org.springframework.web.servlet.FrameworkServlet.CONTEXT.spring**</param-value>

</init-param>

</filter>

<filter-mapping>

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

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

</filter-mapping>

<servlet>

<servlet-name>REST Service</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.oauth2.resource.ApplicationPkgs</param-value>

</init-param>

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

</servlet>

<servlet-mapping>

<servlet-name>REST Service</servlet-name>

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

</servlet-mapping>

<servlet>

<servlet-name>spring</servlet-name>

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

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

</servlet>

<servlet-mapping>

<servlet-name>spring</servlet-name>

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

</servlet-mapping>

</web-app>

# Spring Configuration (spring-servlet.xml)

# Spring Configuration (spring-servlet.cml)

<?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:oauth=*"http://www.springframework.org/schema/security/oauth2"*

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

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

xsi:schemaLocation=*"http://www.springframework.org/schema/security/oauth2*

*http://www.springframework.org/schema/security/spring-security-oauth2-1.0.xsd*

*http://www.springframework.org/schema/mvc*

*http://www.springframework.org/schema/mvc/spring-mvc-3.1.xsd*

*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-3.1.xsd*

*http://www.springframework.org/schema/context*

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

<context:component-scan base-package=*"com.ddlab.rnd.spring.oauth2"* />

<mvc:annotation-driven />

<mvc:default-servlet-handler />

<bean id=*"tokenStore"*

class=*"org.springframework.security.oauth2.provider.token.InMemoryTokenStore"* />

<bean id=*"tokenServices"*

class=*"org.springframework.security.oauth2.provider.token.DefaultTokenServices"*>

<property name=*"tokenStore"* ref=*"tokenStore"* />

<property name=*"supportRefreshToken"* value=*"true"* />

<property name=*"accessTokenValiditySeconds"* value=*"300000"*></property>

</bean>

<bean id=*"logoutSuccessHandler"* class=*"com.ddlab.rnd.spring.oauth2.security.LogoutImpl"*>

<property name=*"tokenstore"* ref=*"tokenStore"*></property>

</bean>

<bean id=*"oauthAccessDeniedHandler"*

class=*"org.springframework.security.oauth2.provider.error.OAuth2AccessDeniedHandler"*>

</bean>

<bean id=*"oauthAuthenticationEntryPoint"*

class=*"org.springframework.security.oauth2.provider.error.OAuth2AuthenticationEntryPoint"*>

</bean>

<bean id=*"clientDetails"*

class=*"com.ddlab.rnd.spring.oauth2.security.ClientDetailsServiceImpl"* />

<bean id=*"clientDetailsUserService"*

class=*"org.springframework.security.oauth2.provider.client.ClientDetailsUserDetailsService"*>

<constructor-arg ref=*"clientDetails"* />

</bean>

<authentication-manager alias=*"authenticationManager"*

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

<authentication-provider user-service-ref=*"clientDetailsUserService"* />

</authentication-manager>

<bean id=*"clientCredentialsTokenEndpointFilter"*

class=*"org.springframework.security.oauth2.provider.client.ClientCredentialsTokenEndpointFilter"*>

<property name=*"authenticationManager"* ref=*"authenticationManager"* />

</bean>

<bean id=*"customUserAuthenticationProvider"*

class=*"com.ddlab.rnd.spring.oauth2.security.CustomUserAuthenticationProvider"*>

</bean>

<authentication-manager id=*"userAuthenticationManager"*

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

<authentication-provider ref=*"customUserAuthenticationProvider"* />

</authentication-manager>

<oauth:resource-server id=*"resourceServerFilter"*

resource-id=*"springsec"* token-services-ref=*"tokenServices"* />

<oauth:authorization-server

client-details-service-ref=*"clientDetails"* token-services-ref=*"tokenServices"*>

<oauth:authorization-code />

<oauth:implicit />

<oauth:refresh-token />

<oauth:client-credentials />

<oauth:password authentication-manager-ref=*"userAuthenticationManager"* />

</oauth:authorization-server>

<!-- End of OAUTH Configuration -->

<!-- All HTTP Pattern Configuration -->

<http pattern=*"/oauth/token"* create-session=*"stateless"*

authentication-manager-ref=*"userAuthenticationManager"*

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

<intercept-url pattern=*"/oauth/token"* access=*"IS\_AUTHENTICATED\_FULLY"* />

<anonymous enabled=*"false"* />

<http-basic entry-point-ref=*"oauthAuthenticationEntryPoint"* />

<custom-filter ref=*"clientCredentialsTokenEndpointFilter"*

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

<access-denied-handler ref=*"oauthAccessDeniedHandler"* />

</http>

<http pattern=*"/resources/\*\*"* create-session=*"never"*

entry-point-ref=*"oauthAuthenticationEntryPoint"* xmlns=*"http://www.springframework.org/schema/security"*>

<anonymous enabled=*"false"* />

**<intercept-url pattern=*"/resources/\*\*"* access=*"IS\_AUTHENTICATED\_FULLY"* />**

<custom-filter ref=*"resourceServerFilter"* before=*"PRE\_AUTH\_FILTER"* />

<access-denied-handler ref=*"oauthAccessDeniedHandler"* />

</http>

<http pattern=*"/logout"* create-session=*"never"*

entry-point-ref=*"oauthAuthenticationEntryPoint"* xmlns=*"http://www.springframework.org/schema/security"*>

<anonymous enabled=*"false"* />

<intercept-url pattern=*"/logout"* method=*"GET"* />

<sec:logout invalidate-session=*"true"* logout-url=*"/logout"*

success-handler-ref=*"logoutSuccessHandler"* />

<custom-filter ref=*"resourceServerFilter"* before=*"PRE\_AUTH\_FILTER"* />

<access-denied-handler ref=*"oauthAccessDeniedHandler"* />

</http>

</beans>

# Java Sources

## Entity layer

**Employee.java**

**package** com.ddlab.rnd.spring.oauth2.entity;

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

**public** **class** Employee {

@JsonProperty("ID")

**private** String empId ;

@JsonProperty("FIRST\_NAME")

**private** String firstName;

@JsonProperty("LAST\_NAME")

**private** String lastName;

**public** String getEmpId() {

**return** empId;

}

**public** **void** setEmpId(String empId) {

**this**.empId = empId;

}

**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;

}

}

## Service Layer

**EmpService.java**

**package** com.ddlab.rnd.spring.oauth2.service;

**import** com.ddlab.rnd.spring.oauth2.entity.Employee;

**public** **interface** EmpService {

**public** Employee getEmployeeById(String id);

}

**EmpServiceImpl.java**

**package** com.ddlab.rnd.spring.oauth2.service;

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

**import** com.ddlab.rnd.spring.oauth2.entity.Employee;

@Component("empServiceImpl")

**public** **class** EmpServiceImpl **implements** EmpService {

@Override

**public** Employee getEmployeeById(String id) {

Employee emp = **new** Employee();

emp.setEmpId(id);

emp.setFirstName("Debadatta");

emp.setLastName("Mishra");

**return** emp;

}

}

## Resource Layer

**ApplicationPkgs.java**

**package** com.ddlab.rnd.spring.oauth2.resource;

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

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

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

**public** ApplicationPkgs() {

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

}

}

**EmployeeInfo.java**

**package** com.ddlab.rnd.spring.oauth2.resource;

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

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

**import** javax.ws.rs.PathParam;

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

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

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

**import** com.ddlab.rnd.spring.oauth2.entity.Employee;

**import** com.ddlab.rnd.spring.oauth2.service.EmpService;

@Path("/emp")

**public** **class** EmployeeInfo {

@Autowired

**private** EmpService empService;

@Path("/info/{id}")

@GET

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

**public** Employee getEmpInfo( @PathParam("id") String empId ) {

**return** empService.getEmployeeById(empId);

}

}

## Security Layer

**ClientDetailsServiceImpl.java**

**package** com.ddlab.rnd.spring.oauth2.security;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.springframework.security.oauth2.common.exceptions.OAuth2Exception;

**import** org.springframework.security.oauth2.provider.BaseClientDetails;

**import** org.springframework.security.oauth2.provider.ClientDetails;

**import** org.springframework.security.oauth2.provider.ClientDetailsService;

**import** org.springframework.security.oauth2.provider.NoSuchClientException;

**import** org.springframework.stereotype.Service;

@Service

**public** **class** ClientDetailsServiceImpl **implements** ClientDetailsService {

/\*

\* From this method, get the user details

\* This class is tailor made, manually doing everything for

\* better customization

\*/

@Override

**public** ClientDetails loadClientByClientId(String clientId)

**throws** OAuth2Exception {

**if** (clientId.equals("client1")) {

List<String> authorizedGrantTypes = **new** ArrayList<>();

authorizedGrantTypes.add("password");

authorizedGrantTypes.add("refresh\_token");

authorizedGrantTypes.add("client\_credentials");

BaseClientDetails clientDetails = **new** BaseClientDetails();

clientDetails.setClientId("client1");

clientDetails.setClientSecret("client1");

clientDetails.setAuthorizedGrantTypes(authorizedGrantTypes);

**return** clientDetails;

} **else** **if** (clientId.equals("client2")) {

List<String> authorizedGrantTypes = **new** ArrayList<>();

authorizedGrantTypes.add("password");

authorizedGrantTypes.add("refresh\_token");

authorizedGrantTypes.add("client\_credentials");

BaseClientDetails clientDetails = **new** BaseClientDetails();

clientDetails.setClientId("client2");

clientDetails.setClientSecret("client2");

clientDetails.setAuthorizedGrantTypes(authorizedGrantTypes);

**return** clientDetails;

}

**else** {

**throw** **new** NoSuchClientException("No client with requested id: "

+ clientId);

}

}

}

**CustomUserAuthenticationProvider.java**

**package** com.ddlab.rnd.spring.oauth2.security;

**import** java.util.ArrayList;

**import** java.util.List;

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

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

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

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

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

**public** **class** CustomUserAuthenticationProvider **implements** AuthenticationProvider {

@Override

**public** Authentication authenticate(Authentication authentication)

**throws** AuthenticationException {

/\*

\* From this method, get the user details

\* This class is tailor made, manually doing everything for

\* better customization

\*/

**if** (authentication.getPrincipal().equals("user")

&& authentication.getCredentials().equals("user")) {

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

CustomUserPasswordAuthenticationToken auth = **new** CustomUserPasswordAuthenticationToken( authentication.getPrincipal(),

authentication.getCredentials(), grantedAuthorities);

**return** auth;

} **else** **if** (authentication.getPrincipal().equals("admin")

&& authentication.getCredentials().equals("admin")) {

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

CustomUserPasswordAuthenticationToken auth = **new** CustomUserPasswordAuthenticationToken(

authentication.getPrincipal(),

authentication.getCredentials(), grantedAuthorities);

**return** auth;

} **else** **if** (authentication.getPrincipal().equals("user1")

&& authentication.getCredentials().equals("user1")) {

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

CustomUserPasswordAuthenticationToken auth = **new** CustomUserPasswordAuthenticationToken(

authentication.getPrincipal(),

authentication.getCredentials(), grantedAuthorities);

**return** auth;

} **else** {

**throw** **new** BadCredentialsException("Bad User Credentials.");

}

}

@Override

**public** **boolean** supports(Class<? **extends** Object> authentication) {

**return** **true**;

}

}

**CustomUserPasswordAuthenticationToken.java**

**package** com.ddlab.rnd.spring.oauth2.security;

**import** java.util.Collection;

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

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

**public** **class** CustomUserPasswordAuthenticationToken **extends** AbstractAuthenticationToken {

**private** **static** **final** **long** ***serialVersionUID*** = 310L;

**private** **final** Object principal;

**private** Object credentials;

**private** **int** loginId;

**public** **int** getLoginId() {

**return** loginId;

}

**public** **void** setLoginId(**int** loginId) {

**this**.loginId = loginId;

}

**public** CustomUserPasswordAuthenticationToken(Object principal,

Object credentials) {

**super**(**null**);

**this**.principal = principal;

**this**.credentials = credentials;

setAuthenticated(**false**);

}

**public** CustomUserPasswordAuthenticationToken(Object principal,

Object credentials,

Collection<? **extends** GrantedAuthority> authorities) {

**super**(authorities);

**this**.principal = principal;

**this**.credentials = credentials;

**super**.setAuthenticated(**true**);

}

**public** Object getCredentials() {

**return** **this**.credentials;

}

**public** Object getPrincipal() {

**return** **this**.principal;

}

**public** **void** setAuthenticated(**boolean** isAuthenticated)

**throws** IllegalArgumentException {

**if** (isAuthenticated) {

**throw** **new** IllegalArgumentException(

"Cannot set this token to trusted - use constructor which takes a GrantedAuthority list instead");

}

**super**.setAuthenticated(**false**);

}

**public** **void** eraseCredentials() {

**super**.eraseCredentials();

**this**.credentials = **null**;

}

}

**LogoutImpl.java**

**package** com.ddlab.rnd.spring.oauth2.security;

**import** java.io.IOException;

**import** javax.servlet.ServletException;

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

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

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

**import** org.springframework.security.oauth2.provider.token.InMemoryTokenStore;

**import** org.springframework.security.oauth2.common.DefaultOAuth2AccessToken;

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

**public** **class** LogoutImpl **implements** LogoutSuccessHandler {

InMemoryTokenStore tokenstore;

**public** InMemoryTokenStore getTokenstore() {

**return** tokenstore;

}

**public** **void** setTokenstore(InMemoryTokenStore tokenstore) {

**this**.tokenstore = tokenstore;

}

@Override

**public** **void** onLogoutSuccess(HttpServletRequest paramHttpServletRequest,

HttpServletResponse paramHttpServletResponse,

Authentication paramAuthentication) **throws** IOException,

ServletException {

removeaccess(paramHttpServletRequest);

paramHttpServletResponse.getOutputStream().write("\n\tYou Have Logged Out successfully.".getBytes());

}

**public** **void** removeaccess(HttpServletRequest req) {

String tokens=req.getHeader("Authorization");

System.***out***.println(tokens);

String value=tokens.substring(tokens.indexOf(" ")).trim();

DefaultOAuth2AccessToken token= **new** DefaultOAuth2AccessToken(value);

System.***out***.println(token);

tokenstore.removeAccessToken(value);

System.***out***.println("\n\tAccess Token Removed Successfully!!!!!!!!");

}

}

**How to Use**

**Request**

POST

http://localhost:8080/oauth2-1/oauth/token?username=user1&password=user1&client\_id=client1&client\_secret=client1&grant\_type=password

**Response**

{

"access\_token": "c870aa37-3734-4b3b-93e4-4b8281d33b62",

"token\_type": "bearer",

"refresh\_token": "14f60a9a-f9a3-49d8-af19-60d85ea64c79",

"expires\_in": 299999

}

GET

http://localhost:8080/oauth2-1/resources/emp/info/123

**Header**

Authorization

bearer c870aa37-3734-4b3b-93e4-4b8281d33b62

<http://localhost:8080/oauth2-1/logout>

**Header**

Authorization

bearer c870aa37-3734-4b3b-93e4-4b8281d33b62