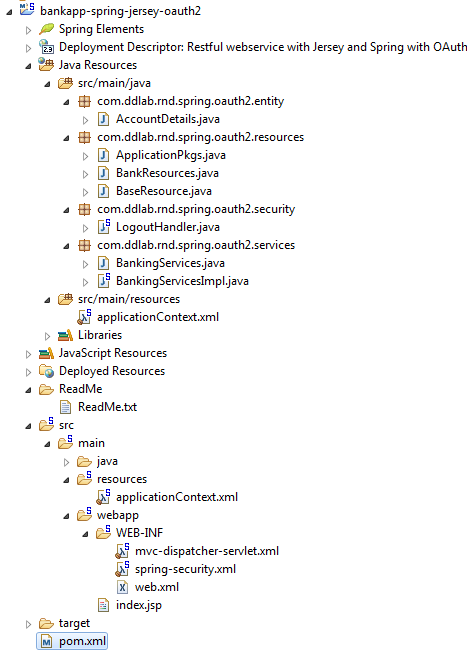
OAuth2 in Spring Security with Jersey RESTful web service

# 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-oauth2</groupId>

<artifactId>bankapp-spring-jersey-oauth2</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>bankapp-spring-jersey-oauth2 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>**

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

**<configuration>**

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

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

**</configuration>**

**</plugin>**

</plugins>

</build>

</project>

# Web Configuration( WEB-INF/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,/WEB-INF/mvc-dispatcher-servlet.xml,/WEB-INF/spring-security.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>

<!—For Jersey RESTful web services -->

**<servlet>**

**<servlet-name>com.ddlab.rnd.spring.oauth2.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.oauth2.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.oauth2.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>

# Spring Configuration

## Spring MVC (WEB-INF/mvc-dispatcher-servlet.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:context=*"http://www.springframework.org/schema/context"*

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

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

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

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

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

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

**<mvc:annotation-driven />**

</beans>

## Spring Security(spring-security.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:oauth=*"http://www.springframework.org/schema/security/oauth2"*

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

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

xsi:schemaLocation=*"http://www.springframework.org/schema/security/oauth2 http://www.springframework.org/schema/security/spring-security-oauth2-2.0.xsd*

*http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.2.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-4.1.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.1.xsd "*>

**<bean id=*"logoutHandler"* class=*"com.ddlab.rnd.spring.oauth2.security.LogoutHandler"*>**

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

**</bean>**

**<bean id=*"oauthAccessDeniedHandler"***

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

**<bean id=*"oauthAuthenticationEntryPoint"***

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

**<property name=*"realmName"* value=*"mytest"* />**

**</bean>**

**<bean id=*"clientAuthenticationEntryPoint"***

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

**<property name=*"realmName"* value=*"app/client"* />**

**<property name=*"typeName"* value=*"Basic"* />**

**</bean>**

**<bean id=*"clientCredentialsTokenEndpointFilter"***

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

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

**</bean>**

**<bean id=*"accessDecisionManager"* class=*"org.springframework.security.access.vote.UnanimousBased"***

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

**<constructor-arg>**

**<list>**

**<bean class=*"org.springframework.security.oauth2.provider.vote.ScopeVoter"* />**

**<bean class=*"org.springframework.security.access.vote.RoleVoter"* />**

**<bean class=*"org.springframework.security.access.vote.AuthenticatedVoter"* />**

**</list>**

**</constructor-arg>**

**</bean>**

<!-- This defined token store, we have used inmemory tokenstore for now

but this can be changed to a user defined one -->

**<bean id=*"tokenStore"***

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

<!-- This is where we defined token based configurations, token validity

and other things -->

**<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=*"120"* />**

**<property name=*"clientDetailsService"* ref=*"clientDetails"* />**

**</bean>**

**<bean id=*"userApprovalHandler"***

**class=*"org.springframework.security.oauth2.provider.approval.TokenServicesUserApprovalHandler"*>**

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

**</bean>**

**<bean id=*"clientDetailsUserService"***

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

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

**</bean>**

**<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=*"logoutHandler"* />**

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

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

**</http>**

<!-- This is default url to get a token from OAuth -->

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

**authentication-manager-ref=*"clientAuthenticationManager"***

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

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

**<anonymous enabled=*"false"* />**

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

<!-- include this only if you need to authenticate clients via request

parameters -->

**<custom-filter ref=*"clientCredentialsTokenEndpointFilter"***

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

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

**</http>**

<!-- This is where we tell spring security what URL should be protected

and what roles have access to them -->

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

**entry-point-ref=*"oauthAuthenticationEntryPoint"***

**access-decision-manager-ref=*"accessDecisionManager"***

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

**<anonymous enabled=*"false"* />**

**<!-- <intercept-url pattern="/api/\*\*" access="ROLE\_APP" /> -->**

**<intercept-url pattern=*"/api/\*\*"* access=*"ROLE\_USER"* />**

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

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

**</http>**

**<authentication-manager id=*"clientAuthenticationManager"***

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

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

**</authentication-manager>**

<!-- This is simple authentication manager, with a hardcoded user/password

combination. We can replace this with a user defined service to get few users

credentials from DB -->

**<authentication-manager alias=*"authenticationManager"***

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

**<authentication-provider>**

**<user-service>**

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

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

**</user-service>**

**</authentication-provider>**

**</authentication-manager>**

**<oauth:authorization-server**

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

**user-approval-handler-ref=*"userApprovalHandler"*>**

**<oauth:authorization-code />**

**<oauth:implicit />**

**<oauth:refresh-token />**

**<oauth:client-credentials />**

**<oauth:password />**

**</oauth:authorization-server>**

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

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

**<oauth:client-details-service id=*"clientDetails"*>**

**<!-- client -->**

**<oauth:client client-id=*"restapp"***

**authorized-grant-types=*"authorization\_code,client\_credentials"***

**authorities=*"ROLE\_USER"* scope=*"read,write,trust"* secret=*"secret"* />**

**<oauth:client client-id=*"restapp"***

**authorized-grant-types=*"password,authorization\_code,refresh\_token,implicit"***

**secret=*"restapp"* authorities=*"ROLE\_USER"* />**

**</oauth:client-details-service>**

**<sec:global-method-security**

**pre-post-annotations=*"enabled"* proxy-target-class=*"true"*>**

**<!--you could also wire in the expression handler up at the layer of the**

**http filters. See https://jira.springsource.org/browse/SEC-1452 -->**

**<sec:expression-handler ref=*"oauthExpressionHandler"* />**

**</sec:global-method-security>**

**<oauth:expression-handler id=*"oauthExpressionHandler"* />**

**<oauth:web-expression-handler id=*"oauthWebExpressionHandler"* />**

</beans>

## Spring Config(resources/application-context.xml)

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

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

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

</beans>

# Java Source Code

## Entity layer(AccountDetails.java)

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

}

}

## Service layer

**BankingServices.java**

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

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

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

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

}

**BankingServicesImpl.java**

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

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

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

}

}

# Resources Layer

**ApplicationPkgs.java**

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

**import** javax.servlet.ServletConfig;

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

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

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

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

**import** com.ddlab.rnd.spring.oauth2.services.BankingServices;

@Path("1/bankingservices")

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

@Context

HttpServletRequest request;

@Context

ServletConfig servletConfig;

@Autowired

@Qualifier(value="bankingService")

**private** BankingServices services;

@Path("/userid")

@GET

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

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

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

// GET http://localhost:8080/bankapp-spring-jersey-oauth2/api/1/bankingservices/userid?id=123

**if**(id == 0 )

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

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

}

}

## Security Layer

**LogoutHandler.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.common.DefaultOAuth2AccessToken;

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

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

**public** **class** LogoutHandler **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 in Rest Client

To get authentication token

**GET**

**http://localhost:8080/bankapp-spring-jersey-oauth2/oauth/token?grant\_type=password&client\_id=restapp&client\_secret=restapp&username=piku&password=piku**

**Response**

**{"access\_token":"16ce6fc2-f3e1-4842-a2d9-79b5bd26de11","token\_type":"bearer","refresh\_token":"eb1d2526-f54f-444b-8d3f-b515de5627c0","expires\_in":119}**

**GET**

**http://localhost:8080/bankapp-spring-jersey-oauth2/api/1/bankingservices/userid?id=123**

**Header**

**Authorization**

**bearer 16ce6fc2-f3e1-4842-a2d9-79b5bd26de11**

From the above request use the access\_token

**To Logout**

GET

http://localhost:8080/bankapp-spring-jersey-oauth2/logout

**Header**

**Authorization**

**bearer 16ce6fc2-f3e1-4842-a2d9-79b5bd26de11**

From the above request use the access\_token

**To get refresh token**

GET

**http://localhost:8080/bankapp-spring-jersey-oauth2/oauth/token?grant\_type=refresh\_token&client\_id=restapp&client\_secret=restapp&refresh\_token=eb1d2526-f54f-444b-8d3f-b515de5627c0**

From the above response, use the refresh\_token