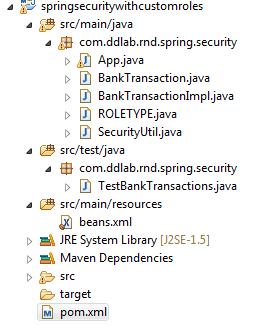
Spring Security with Custom Roles in Spring Standalone

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/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>springsecuritywithcustomroles</groupId>

<artifactId>springsecuritywithcustomroles</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>springsecuritywithcustomroles</name>

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

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<spring.version>3.2.8.RELEASE</spring.version>

<spring.security.version>3.2.3.RELEASE</spring.security.version>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

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

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

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

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

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

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.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</groupId>

<artifactId>spring-security-taglibs</artifactId>

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

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.30</version>

</dependency>

</dependencies>

</project>

# Spring Configuration

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

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

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

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

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.1.xsd*

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

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-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/util http://www.springframework.org/schema/util/spring-util-3.1.xsd"*>

**<bean id=*"roleHierarchy"* class=*"org.springframework.security.access.hierarchicalroles.RoleHierarchyImpl"*>**

**<property name=*"hierarchy"*>**

**<value>**

**ROLE\_ADMIN > ROLE\_MANAGER**

**ROLE\_MANAGER > ROLE\_TELLER**

**ROLE\_TELLER > ROLE\_USER**

**</value>**

**</property>**

**</bean>**

**<bean id=*"roleVoter"* class=*"org.springframework.security.access.vote.RoleHierarchyVoter"*>**

**<constructor-arg>**

**<ref bean=*"roleHierarchy"* />**

**</constructor-arg>**

**</bean>**

**<bean id=*"expressionHandler"* class=*"org.springframework.security.web.access.expression.DefaultWebSecurityExpressionHandler"*>**

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

**</bean>**

**<bean id=*"webExpressionHandler"* class=*"org.springframework.security.web.access.expression.WebExpressionVoter"*>**

**<property name=*"expressionHandler"*>**

**<ref bean=*"expressionHandler"* />**

**</property>**

**</bean>**

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

**<constructor-arg>**

**<list>**

**<ref bean=*"roleVoter"* />**

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

**<ref bean=*"webExpressionHandler"* />**

**</list>**

**</constructor-arg>**

**</bean>**

**<bean id=*"defaultMethodSecurityExpressionHandler"***

**class=*"org.springframework.security.access.expression.method.DefaultMethodSecurityExpressionHandler"*>**

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

**</bean>**

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

**<security:expression-handler ref=*"defaultMethodSecurityExpressionHandler"* />**

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

<bean id=*"bankTxn"* class=*"com.ddlab.rnd.spring.security.BankTransactionImpl"* />

</beans>

# Java Files

## BankTransaction.java

package com.ddlab.rnd.spring.security;

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

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

public interface BankTransaction {

/\*\*

\* Can be done by Manager and Admin

\*/

**@PreAuthorize ("hasRole('ROLE\_MANAGER')")**

public void approveLoan(String actNo);

/\*\*

\* Can be done only by USER

\*/

**@Secured("ROLE\_USER")**

public void openAccount(String customerName);

/\*\*

\* Can be done by Manager and Admin

\*/

**@PreAuthorize ("hasRole('ROLE\_MANAGER')")**

public void resetNetBankingPassword(String customerName);

/\*\*

\* Can be done by Teller,Manager and Admin

\*/

**@PreAuthorize ("hasRole('ROLE\_TELLER')")**

public void checkAccountStatus(String actNo);

/\*\*

\* Can be done by anybody in bank other than Normal user or customer

\*/

**@Secured({"ROLE\_ADMIN", "ROLE\_MANAGER" , "ROLE\_TELLER"})**

public void suggestCreditCard(String customerName);

/\*\*

\* Can be approved only by Manager and not by any other persons

\*/

**@Secured({"ROLE\_MANAGER"})**

public void approveOverDraftFacility(String customerName);

}

# BankTransactionImpl.java

package com.ddlab.rnd.spring.security;

public class BankTransactionImpl implements BankTransaction {

public void approveLoan(String actNo) {

System.out.println("The loan has been approved for the account no : "+actNo);

}

public void openAccount(String customerName) {

System.out.println("Hi "+customerName+" , your account will be activate within 24 hours");

}

public void resetNetBankingPassword(String customerName) {

System.out.println("Hi "+customerName+", Administrator has rest your internet banking password "

+ "and an email has been sent you");

}

public void checkAccountStatus(String actNo) {

System.out.println("You available balance is 2000 INR");

}

public void suggestCreditCard(String customerName) {

System.out.println("Hi "+customerName+" , a life time free Titanium credit card is available for you");

}

public void approveOverDraftFacility(String customerName) {

System.out.println("Hi "+customerName+" , a overdraft facility is approved for you");

}

}

## ROLETYPE.java

package com.ddlab.rnd.spring.security;

public enum ROLETYPE {

USER("ROLE\_USER") , ADMIN("ROLE\_ADMIN") , MANAGER("ROLE\_MANAGER") , TELLER("ROLE\_TELLER");

private String role;

private ROLETYPE(String role) {

this.role = role;

}

/\* (non-Javadoc)

\* @see java.lang.Enum#toString()

\*/

@Override

public String toString() {

return role;

}

}

## SecurityUtil.java

package com.ddlab.rnd.spring.security;

import java.util.List;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.authority.SimpleGrantedAuthority;

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

import org.springframework.security.core.userdetails.User;

public class SecurityUtil {

**public static void loginAs(String userName, String password, ROLETYPE roleType) {**

**setSecurity(userName, password, roleType.toString());**

**}**

**public static void logout() {**

**//SecurityContextHolder.getContext().setAuthentication(null);**

**//SecurityContextHolder.getContext().getAuthentication().setAuthenticated(false);**

**SecurityContextHolder.clearContext();**

**}**

**private static void setSecurity(String userName, String password,String roleType) {**

**List<SimpleGrantedAuthority> auths = new java.util.ArrayList<SimpleGrantedAuthority>();**

**auths.add(new SimpleGrantedAuthority(roleType));**

**User user = new User(userName, password, auths);**

**Authentication authToken = new UsernamePasswordAuthenticationToken (user.getUsername(), user.getPassword(), user.getAuthorities());**

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

**}**

}

## App.java

package com.ddlab.rnd.spring.security;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App

{

public static void main( String[] args )

{

//ApplicationContext applicationContext = new ClassPathXmlApplicationContext(new String[]{"beans1.xml","application-security.xml"});

ApplicationContext applicationContext = new ClassPathXmlApplicationContext("beans.xml");

// SecurityUtil.loginAs("Deb", "pqrs", "ROLE\_USER");

// SecurityUtil.loginAs("Deb", "pqrs", "ROLE\_MANAGER");

// SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.TELLER);

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.MANAGER);

BankTransaction bankTxn = (BankTransaction) applicationContext.getBean("bankTxn");

bankTxn.approveLoan("12345");

// bankTxn.openAccount("Deb");

// bankTxn.checkAccountStatus("12345");

}

}

## JUnit Test

TestBankTransactions.java

/\*\*

\* Junit Test Cases

\*/

package com.ddlab.rnd.spring.security;

import org.junit.BeforeClass;

import org.junit.Test;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.security.access.AccessDeniedException;

import org.springframework.security.authentication.AuthenticationCredentialsNotFoundException;

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

public class TestBankTransactions {

private static BankTransaction bankTxn = null;

@BeforeClass

public static void setup() {

@SuppressWarnings("resource")

ApplicationContext applicationContext = new ClassPathXmlApplicationContext("beans.xml");

bankTxn = (BankTransaction) applicationContext.getBean("bankTxn");

}

/\*\*

\* Loan can not be approved by Normal user

\*/

@Test(expected=AccessDeniedException.class)

public void testApproveLoanByNormalUser() {

SecurityUtil.logout();

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.USER);

bankTxn.approveLoan("12345");

}

/\*\*

\* Loan can be approved by Admin

\*/

@Test

public void testApproveLoanByAdmin() {

SecurityUtil.logout();

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.ADMIN);

bankTxn.approveLoan("12345");

}

/\*\*

\* Loan cannot be approved by Teller

\*/

@Test(expected=AccessDeniedException.class)

public void testApproveLoanByTeller() {

SecurityUtil.logout();

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.TELLER);

bankTxn.approveLoan("12345");

}

/\*\*

\* Loan can be approved by Manager

\*/

@Test

public void testApproveLoanByManager() {

SecurityUtil.logout();

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.MANAGER);

bankTxn.approveLoan("12345");

}

/\*\*

\* Loan can not be approved by unknown user

\*/

@Test(expected = AuthenticationCredentialsNotFoundException.class)

public void testApproveLoanAnonymously() {

SecurityUtil.logout();

System.out.println("----->"+SecurityContextHolder.getContext().getAuthentication());

bankTxn.approveLoan("12345");

}

//--------------------

//All bank employee suggest a customer for a free credit card

@Test

public void testSuggestForCreditCardLoanByManager() {

SecurityUtil.logout();

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.MANAGER);

bankTxn.suggestCreditCard("John Abraham");

}

@Test

public void testSuggestForCreditCardLoanByAdmin() {

SecurityUtil.logout();

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.ADMIN);

bankTxn.suggestCreditCard("John Abraham");

}

@Test

public void testSuggestForCreditCardLoanByTeller() {

SecurityUtil.logout();

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.TELLER);

bankTxn.suggestCreditCard("John Abraham");

}

@Test(expected=AccessDeniedException.class)

public void testSuggestForCreditCardLoanByUser() {

SecurityUtil.logout();

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.USER);

bankTxn.suggestCreditCard("John Abraham");

}

//Overdraft facility can only be approved by Bank Manager only and not by any other

@Test(expected=AccessDeniedException.class)

public void testOverDraftFacilityByUser() {

SecurityUtil.logout();

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.USER);

bankTxn.approveOverDraftFacility("John Abraham");

}

@Test(expected=AccessDeniedException.class)

public void testOverDraftFacilityByTeller() {

SecurityUtil.logout();

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.TELLER);

bankTxn.approveOverDraftFacility("John Abraham");

}

@Test(expected=AccessDeniedException.class)

public void testOverDraftFacilityByAdmin() {

SecurityUtil.logout();

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.ADMIN);

bankTxn.approveOverDraftFacility("John Abraham");

}

@Test

public void testOverDraftFacilityByManager() {

SecurityUtil.logout();

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.MANAGER);

bankTxn.approveOverDraftFacility("John Abraham");

}

//A user can open account but bank employees should not open in the same bank

@Test

public void testOpenAccountByUser() {

SecurityUtil.logout();

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.USER);

bankTxn.openAccount("John Abraham");

}

@Test(expected=AccessDeniedException.class)

public void testOpenAccountByAdmin() {

SecurityUtil.logout();

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.ADMIN);

bankTxn.openAccount("John Abraham");

}

@Test(expected=AccessDeniedException.class)

public void testOpenAccountByManager() {

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.MANAGER);

bankTxn.openAccount("John Abraham");

}

@Test(expected=AccessDeniedException.class)

public void testOpenAccountByTeller() {

SecurityUtil.logout();

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.TELLER);

bankTxn.openAccount("John Abraham");

}

//Rest user's Internet banking password

@Test(expected=AccessDeniedException.class)

public void testResetNetBankingPasswordByTeller() {

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.TELLER);

bankTxn.resetNetBankingPassword("John Abraham");

}

@Test(expected=AccessDeniedException.class)

public void testResetNetBankingPasswordByUser() {

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.USER);

bankTxn.resetNetBankingPassword("John Abraham");

}

@Test

public void testResetNetBankingPasswordByManager() {

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.MANAGER);

bankTxn.resetNetBankingPassword("John Abraham");

}

@Test

public void testResetNetBankingPasswordByAdmin() {

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.ADMIN);

bankTxn.resetNetBankingPassword("John Abraham");

}

//Account status can be checked by Bank Manager or Teller

@Test

public void testCheckAccountStatusByAdmin() {

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.ADMIN);

bankTxn.checkAccountStatus("12345");

}

@Test

public void testCheckAccountStatusByManager() {

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.MANAGER);

bankTxn.checkAccountStatus("12345");

}

@Test

public void testCheckAccountStatusByTeller() {

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.TELLER);

bankTxn.checkAccountStatus("12345");

}

@Test(expected=AccessDeniedException.class)

public void testCheckAccountStatusByUser() {

SecurityUtil.loginAs("Deb", "pqrs", ROLETYPE.USER);

bankTxn.checkAccountStatus("12345");

}

}