# Spring Security Example Tutorial

APRIL 2, 2018 BY PANKAJ — 9 COMMENTS

Spring Security provides ways to perform authentication and authorization in a web application. We can use spring security in any servlet based web application.

**Table of Contents** [hide]

## Spring Security

Some of the benefits of using Spring Security are:

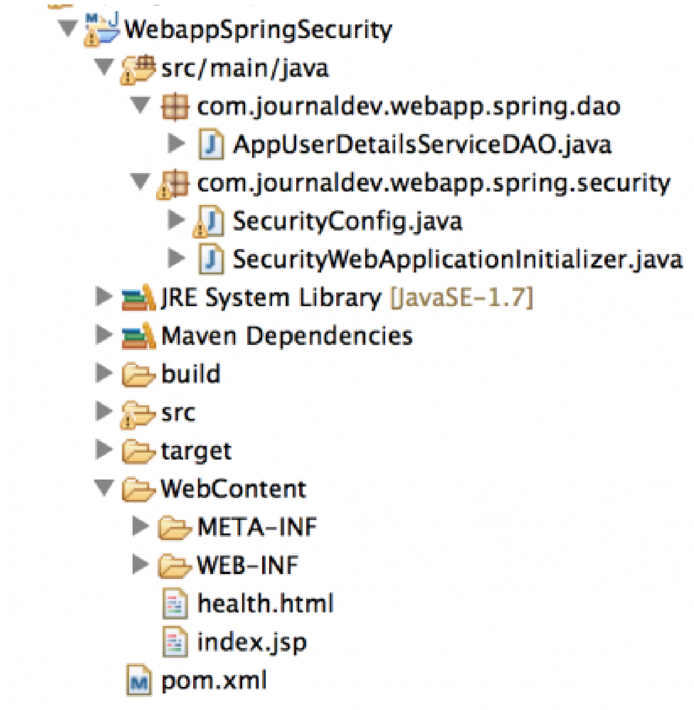
1. Proven technology, it's better to use this than reinvent the wheel. Security is something where we need to take extra care, otherwise our application will be vulnerable for attackers.
2. Prevents some of the common attacks such as CSRF, session fixation attacks.
3. Easy to integrate in any web application. We don't need to modify web application configurations, spring automatically injects security filters to the web application.
4. Provides support for authentication by different ways – in-memory, DAO, JDBC, LDAP and many more.
5. Provides option to ignore specific URL patterns, good for serving static HTML, image files.
6. Support for groups and roles.

## Spring Security Example

We will create a web application and integrate it with Spring Security.

Create a web application using "**Dynamic Web Project**" option in Eclipse, so that our skeleton web application is ready. Make sure to convert it to maven project because we are using Maven for build and deployment. If you are unfamiliar with these steps, please refer Java Web Application Tutorial.

Once we will have our application secured, final project structure will look like below image.


WebappSpringSecurity
src/main/java
com.journaldev.webapp.spring.dao
AppUserDetailsServiceDAO.java
com.journaldev.webapp.spring.security
SecurityConfig.java
SecurityWebApplicationInitializer.java
JRE System Library [JavaSE-1.7]
Maven Dependencies
build
src
target
WebContent
META-INF
WEB-INF
health.html
index.jsp
pom.xml

We will look into three spring security authentication methods.

1. **in-memory**
2. **DAO**
3. **JDBC**

For JDBC, I am using MySQL database and have following script executed to create the user details tables.

```
CREATE TABLE `Employees` (
  `username` varchar(20) NOT NULL DEFAULT '',
  `password` varchar(20) NOT NULL DEFAULT '',
  `enabled` tinyint(1) NOT NULL DEFAULT '1',
  PRIMARY KEY (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Roles` (
  `username` varchar(20) NOT NULL DEFAULT '',
  `role` varchar(20) NOT NULL DEFAULT '',
  PRIMARY KEY (`username`,`role`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `Employees` (`username`, `password`, `enabled`)
VALUES
        ('pankaj', 'pankaj123', 1);

INSERT INTO `Roles` (`username`, `role`)
```

```
VALUES
        ('pankaj', 'Admin'),
        ('pankaj', 'CEO');
```

We would also need to configure JDBC DataSource as JNDI in our servlet container, to learn about this please read Tomcat JNDI DataSource Example.

## Spring Security Maven Dependencies

Here is our final pom.xml file.

```
<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>WebappSpringSecurity</groupId>
        <artifactId>WebappSpringSecurity</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>war</packaging>
        <dependencies>
                <!-- Spring Security Artifacts - START -->
                <dependency>
                        <groupId>org.springframework.security</groupId>
                        <artifactId>spring-security-web</artifactId>
                        <version>3.2.3.RELEASE</version>
                </dependency>
                <dependency>
                        <groupId>org.springframework.security</groupId>
                        <artifactId>spring-security-config</artifactId>
                        <version>3.2.3.RELEASE</version>
                </dependency>
                <dependency>
```

We have following dependencies related to Spring Framework.

1. **spring-jdbc**: This is used for JDBC operations by JDBC authentication method. It requires DataSource setup as JNDI. For complete example of it's usage, please refer Spring DataSource JNDI Example
2. **spring-security-taglibs**: Spring Security tag library, I have used it to display user roles in the JSP page. Most of the times, you won't need it though.

3. **spring-security-config**: It is used for configuring the authentication providers, whether to use JDBC, DAO, LDAP etc.
4. **spring-security-web**: This component integrates the Spring Security to the Servlet API. We need it to plugin our security configuration in web application.

Also note that we will be using Servlet API 3.0 feature to add listener and filters through programmatically, that's why servlet api version in dependencies should be 3.0 or higher.

## Spring Security Example View Pages

We have JSP and HTML pages in our application. We want to apply authentication in all the pages other than HTML pages.

`health.html`

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Health Check</title>
</head>
<body>
    <h3>Service is up and running!!</h3>
</body>
</html>
```

`index.jsp`

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://www.springframework.org/security/tags" prefix="sec" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Home Page</title>
</head>
<body>
<h3>Home Page</h3>
```

```
        <p>
      Hello <b><c:out value="${pageContext.request.remoteUser}"/></b><br>
      Roles: <b><sec:authentication property="principal.authorities" /></b>
    </p>

    <form action="logout" method="post">
      <input type="submit" value="Logout" />
```

I have included `index.jsp` as `welcome-file` in the application deployment descriptor.

Spring Security takes care of CSRF attack, so when we are submitting form for logout, we are sending the CSRF token back to server to delete it. The CSRF object set by Spring Security component is **_csrf** and we are using it's property name and token value to pass along in the logout request.

Let's look at the Spring Security configurations now.

## Spring Security Example UserDetailsService DAO Implementation

Since we will be using DAO based authentication also, we need to implement `UserDetailsService` interface and provide the implementation for `loadUserByUsername()` method.

Ideally we should be using some resource to validate the user, but for simplicity I am just doing basic validation.

`AppUserDetailsServiceDAO.java`

```
package com.journaldev.webapp.spring.dao;

import java.util.Collection;
import java.util.List;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;

public class AppUserDetailsServiceDAO implements UserDetailsService {

        protected final Log logger = LogFactory.getLog(getClass());

        @Override
        public UserDetails loadUserByUsername(final String username)
                        throws UsernameNotFoundException {

                logger.info("loadUserByUsername username="+username);
```

Notice that I am creating anonymous inner class of `UserDetails` and returning it. You can create an implementation class for it and then instantiate and return it. Usually that is the way to go in actual applications.

## Spring Security Example WebSecurityConfigurer implementation

We can implement `WebSecurityConfigurer` interface or we can extend the base implementation class `WebSecurityConfigurerAdapter` and override the methods.

`SecurityConfig.java`

```
package com.journaldev.webapp.spring.security;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.sql.DataSource;

import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationMa

import org.springframework.security.config.annotation.web.builders.WebSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerA


import com.journaldev.webapp.spring.dao.AppUserDetailsServiceDAO;

@Configuration
@EnableWebSecurity
```

Notice that we are ignoring all HTML files by overriding `configure(WebSecurity web)` method.

The code shows how to plugin JDBC authentication. We need to configure it by providing DataSource. Since we are using custom tables, we are also required to provide the select queries to get the user details and it's roles.

Configuring in-memory and DAO based authentication is easy, they are commented in above code. You can uncomment them to use them, make sure to have only one configuration at a time.

`@Configuration` and `@EnableWebSecurity` annotations are required, so that spring framework know that this class will be used for spring security configuration.

Spring Security Configuration is using Builder Pattern and based on the authenticate method, some of the methods won't be available later on. For example, `auth.userDetailsService()` returns the instance of `UserDetailsService` and then we can't have any other options, such as we can't set DataSource after it.

## Integrating Spring Security Web with Servlet API

The last part is to integrate our Spring Security configuration class to the Servlet API. This can be done easily by extending `AbstractSecurityWebApplicationInitializer` class and passing the Security configuration class in the super class constructor.

`SecurityWebApplicationInitializer.java`

```
package com.journaldev.webapp.spring.security;

import
org.springframework.security.web.context.AbstractSecurityWebApplicationInitializer;

public class SecurityWebApplicationInitializer extends
                AbstractSecurityWebApplicationInitializer {

        public SecurityWebApplicationInitializer() {
        super(SecurityConfig.class);
    }
}
```

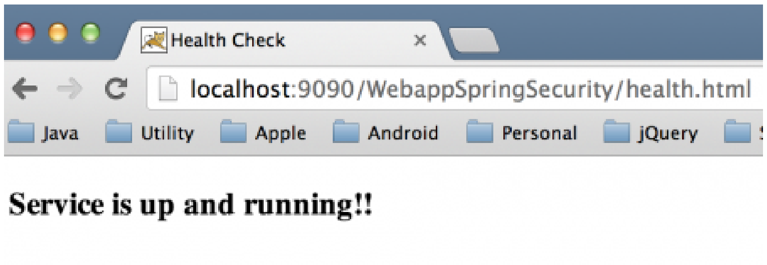
When our context startup, it uses ServletContext to add ContextLoaderListener listener and register our configuration class as Servlet Filter.

Note that this will work only in Servlet-3 complaint servlet containers. So if you are using Apache Tomcat, make sure it's version is 7.0 or higher.

Our project is ready, just deploy it in your favorite servlet container. I am using Apache Tomcat-7 for running this application.

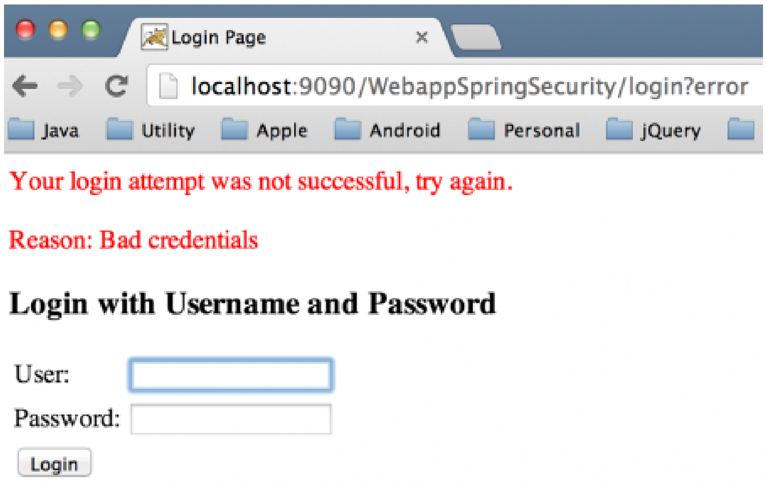
Below images show the response in various scenarios.

### Accessing HTML Page without Security



Service is up and running!!

### Authentication Failed for Bad Credentials



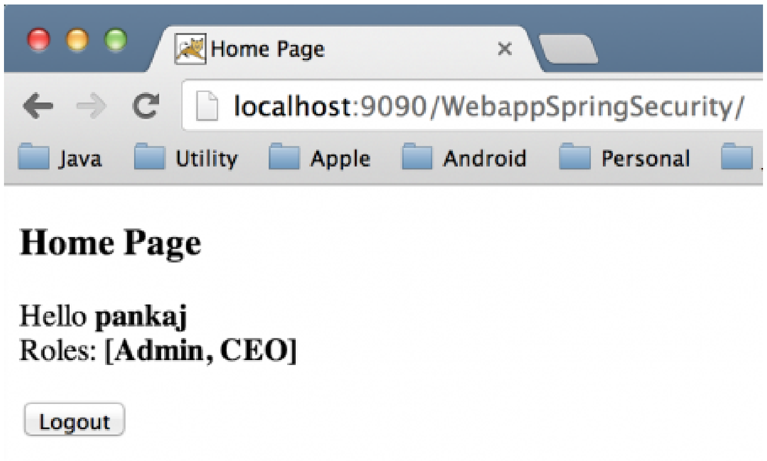
Your login attempt was not successful, try again.

Reason: Bad credentials

Login with Username and Password

User:
Password:

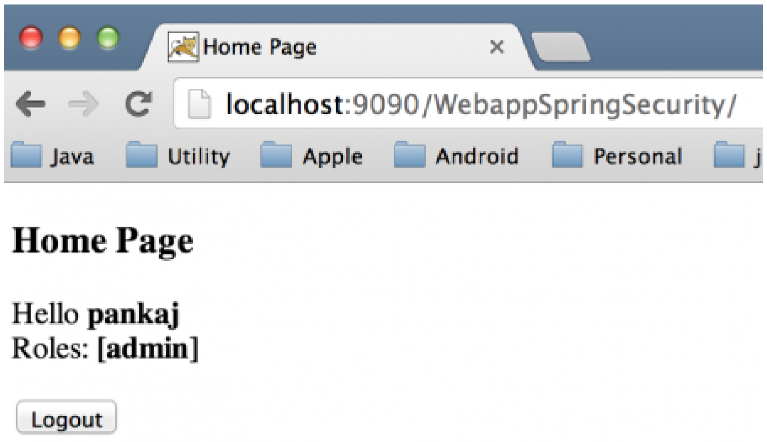
### Home Page with Spring Security JDBC Authentication



Home Page

Hello **pankaj**
Roles: **[Admin, CEO]**

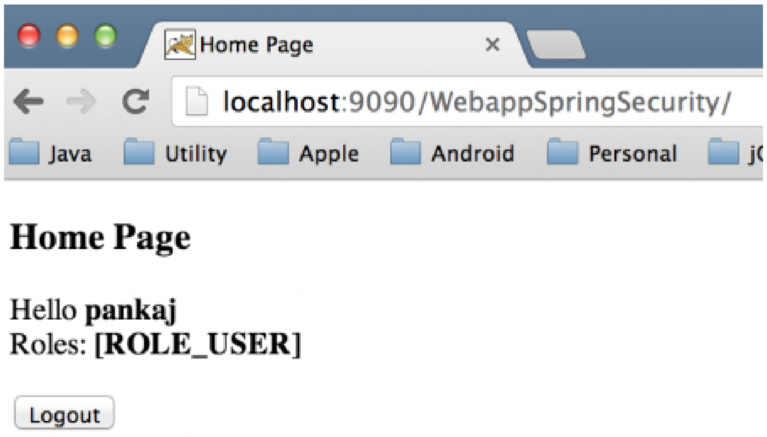
### Home Page with Spring Security UserDetailsService DAO Authentication



Home Page

Hello **pankaj**
Roles: **[admin]**

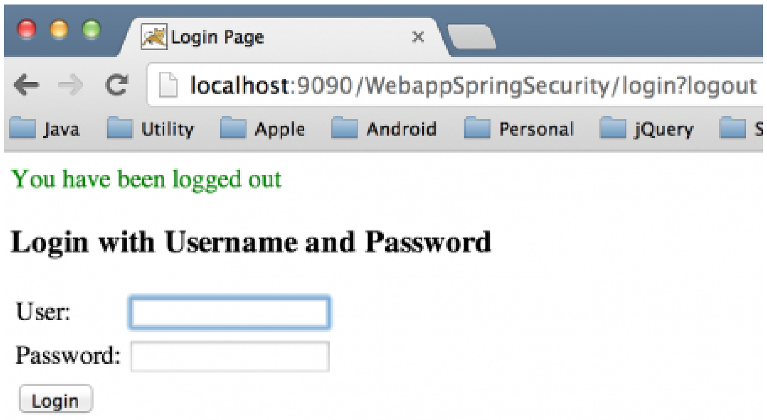
## Home Page with Spring Security In-Memory Authentication



Home Page

Hello **pankaj**
Roles: **[ROLE_USER]**

## Logout Page



You have been logged out

Login with Username and Password

If you want to use Servlet Container that doesn't support Servlet Specs 3, then you would need to register `DispatcherServlet` through deployment descriptor. See JavaDoc of `WebApplicationInitializer` for more details.

That's all for Spring Security example tutorial and it's integration in Servlet Based Web Application. Please download the sample project from below link and play around with it to learn more.

Download Spring Servlet Security Project

« PREVIOUS
Spring Interview Questions and Answers

NEXT »
Spring Security Example UserDetailsService

**About Pankaj**

If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on **Google Plus**, **Facebook** or **Twitter**. I would love to hear your thoughts and opinions on my articles directly.

Recently I started creating video tutorials too, so do check out my videos on **Youtube**.

FILED UNDER: SPRING

## Comments

**Ganesh says**
AUGUST 31, 2017 AT 1:15 AM
How to create REST web service Spring security in Spring 4.
Reply

**safdar says**
JULY 10, 2016 AT 9:28 AM
How to integrate spring security with the application which is combination of spring, JSF 2.0 and Hibernate.
I'm using spring for transaction management at service level with @Transactional annotation and @ Service annotation and @Repository annotation with the DAO layer.
Jsf for the view layer and using the using the @ManagedBean for the service layer as well. and Hibernate for persistence.
Actually, I referred to your example JSF + Spring +Hibernate and I'm trying to integrate Spring Security with the examp[le..The above is the detail of the same example.
Reply

**Praveen Kumar says**
MARCH 7, 2016 AT 11:55 PM
Hello, Pankaj, I am a great fan of your blogs.
I have a query. I have observed you are using Mac for development. I have been working on Mac since long time and thinking of switching to Mac, can you give some advice. Is it worth? Also, I will be developing on cloud, so that shouldn't make much difference whether it's windows or Mac, is it?
Thanks,
Praveen
Reply

**Pankaj says**
MARCH 8, 2016 AT 6:58 AM
It's personal choice, use any OS that you feel comfortable. For me, I work mostly on Mac OS X and Windows 10.
Reply

**cinematik says**
DECEMBER 5, 2015 AT 10:46 AM

How can we define custom login page with java based spring config in Servlet Web Application?

Reply

**Rimmy Arora says**

MAY 18, 2015 AT 12:09 AM

can u please provide me an example on Leave management system with the use of servlet with the jdbc connection..

Reply

**Rohitee Vilas Kawade says**

NOVEMBER 27, 2016 AT 5:08 AM

please can you give me login and account balance inquiry using spring 4

Reply

**Ivan says**

JANUARY 27, 2015 AT 11:41 PM

Shouldn't there be some kind of login page jsp?

Reply

**krishna kumar says**

APRIL 16, 2014 AT 6:38 AM

please provide the example based on xml and annotations configuration.

Reply

## Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

Search for tutorials...

DOWNLOAD ANDROID APP

GET IT ON Google Play

SPRING FRAMEWORK

## Spring Tutorial

### Spring Core

- Spring Framework
- Spring Dependency Injection
- Spring IoC and Bean
- Spring Bean Life Cycle
- Spring REST
- Spring REST XML
- Spring RestTemplate

- Spring ActiveMQ Example

## Spring Integrations

- Spring JDBC
- Spring DataSource JNDI
- Spring Hibernate
- Spring Primefaces JPA
- Spring Primefaces MongoDB
- Spring Primefaces Hibernate
- SpringRoo Primefaces Hibernate
- Spring JSF
- Spring JDF Hibernate

## Miscellaneous

- Spring Data MongoDB
- Spring Interview Questions

RECOMMENDED TUTORIALS

## Java Tutorials

- Java IO
- Java Regular Expressions
- Multithreading in Java
- Java Logging
- Java Annotations
- Java XML
- Collections in Java
- Java Generics
- Exception Handling in Java
- Java Reflection
- Java Design Patterns
- JDBC Tutorial

## Java EE Tutorials

- Servlet JSP Tutorial
- Struts2 Tutorial
- Spring Tutorial
- Hibernate Tutorial
- Primefaces Tutorial
- Apache Axis 2
- JAX-RS
- Memcached Tutorial

**Computer Security Certification**

**50% Off Internet Security**

**Web Design Tutorials**

**Java Tutorial for Beginners**

**Learn Programming Online**

**Oracle Database Tools**

**Java Programming Courses**

**Python Programming Tutorial**