JSF Authentication Login Logout Database Example

JULY 10, 2016 BY PANKAJ — 55 COMMENTS

Authentication mechanism allows users to have secure access to the application by validating the username and password. We will be using JSF view for login, DAO object ,HttpSession for session management, JSF managed bean and mysql database.

Lets now look in detail as how to create a JSF login logout authentication mechanism in JSF application.

Step 1: Create the table Users in mysql database as

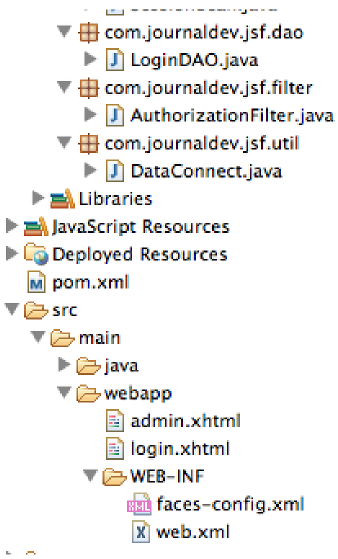
```
CREATE TABLE Users(
uid int(20) NOT NULL AUTO_INCREMENT,
uname VARCHAR(60) NOT NULL,
password VARCHAR(60) NOT NULL,
PRIMARY KEY(uid));
```

Here we create user table with uid as the primary key, username and password fields with not null constraints.

Step 2: Insert data into the table Users as;

```
INSERT INTO Users VALUES(1,'adam','adam');
```

Before we move on to our project related code, below image shows the project structure in Eclipse. Just create a dynamic web project and convert it to maven to get the project stub and then keep on adding different components.



```
▼ com.journaldev.jsf.dao
    ▶ LoginDAO.java
▼ com.journaldev.jsf.filter
    ▶ AuthorizationFilter.java
▼ com.journaldev.jsf.util
    ▶ DataConnect.java
▶ Libraries
▶ JavaScript Resources
▶ Deployed Resources
  pom.xml
▼ src
  ▼ main
    ▶ java
    ▼ webapp
        admin.xhtml
        login.xhtml
      ▼ WEB-INF
          faces-config.xml
          web.xml
```

Step 3: Create the JSF login page `login.xhtml` as;

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
        xmlns:h="http://java.sun.com/jsf/html">
<h:head>
        <title>login</title>
</h:head>
<h:body>
        <h:form>
                <h3>JSF Login Logout</h3>
                <h:outputText value="Username" />
                <h:inputText id="username" value="#{login.user}"></h:inputText>
                <h:message for="username"></h:message>
                <br></br><br></br>

                <h:outputText value="Password" />
                <h:inputSecret id="password" value="#{login.pwd}"></h:inputSecret>
                <h:message for="password"></h:message>
                <br></br><br></br>

                <h:commandButton action="#{login.validateUsernamePassword}"
                        value="Login"></h:commandButton>
        </h:form>
</h:body>
</html>
```

Step 4: Create the managed bean `Login.java` as;

```
package com.journaldev.jsf.beans;

import java.io.Serializable;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.context.FacesContext;
import javax.servlet.http.HttpSession;

import com.journaldev.jsf.dao.LoginDAO;
import com.journaldev.jsf.util.SessionUtils;

@ManagedBean
@SessionScoped
public class Login implements Serializable {

        private static final long serialVersionUID = 1094801825228386363L;

        private String pwd;
        private String msg;
        private String user;

        public String getPwd() {
                return pwd;
        }

        public void setPwd(String pwd) {
                this.pwd = pwd;
        }

        public String getMsg() {
                return msg;
        }
```

We declare three String variables user, pwd and msg for username, password and error message fields along with the getter and setter methods. We write a method `validateUsernamePassword()` for validating the username and password field by invoking the `LoginDAO` class to fetch the username and password from the database and compare it with the front end values passed. If the username and password does not match an error message is displayed as "Incorrect username and password" . Also a `logout()` method is written to perform logout by invalidating HTTPSession attached.

```
package com.journaldev.jsf.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import com.journaldev.jsf.util.DataConnect;

public class LoginDAO {

    public static boolean validate(String user, String password) {
        Connection con = null;
        PreparedStatement ps = null;

        try {
            con = DataConnect.getConnection();
            ps = con.prepareStatement("Select uname, password from Users where uname = ? and password = ?");
            ps.setString(1, user);
            ps.setString(2, password);

            ResultSet rs = ps.executeQuery();

            if (rs.next()) {
                //result found, means valid inputs
                return true;
            }
        } catch (SQLException ex) {
            System.out.println("Login error -->" + ex.getMessage());
            return false;
        } finally {
            DataConnect.close(con);
        }
```

In the `validate()` method we first establish connection to the database by invoking the `DataConnect` class `getConnection` method. We use `PreparedStatement` to build the query to fetch the data from the database with the user entered values. If we get any data in result set, it means input is valid and we return true, else false.

Step 6: Create the `DataConnect.java` class as;

```
package com.journaldev.jsf.util;
```

```
        public static Connection getConnection() {
                try {
                        Class.forName("com.mysql.jdbc.Driver");
                        Connection con = DriverManager.getConnection(
                                        "jdbc:mysql://localhost:3306/cardb", "pankaj",
"pankaj123");
                        return con;
                } catch (Exception ex) {
                        System.out.println("Database.getConnection() Error -->"
                                        + ex.getMessage());
                        return null;
                }
        }

        public static void close(Connection con) {
                try {
                        con.close();
                } catch (Exception ex) {
                }
        }
}
```

We load the JDBC driver using `Class.forName` method and use `DriverManager.getConnection` method passing the url, username and password to connect to the database.

Step 7: Create SessionUtils.java to obtain and manage session related user information.

```
package com.journaldev.jsf.beans;

import javax.faces.context.FacesContext;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

public class SessionUtils {

        public static HttpSession getSession() {
                return (HttpSession) FacesContext.getCurrentInstance()
                                .getExternalContext().getSession(false);
        }

        public static HttpServletRequest getRequest() {
                return (HttpServletRequest) FacesContext.getCurrentInstance()
                                .getExternalContext().getRequest();
                                .getExternalContext().getSession(false);
                return session.getAttribute("username").toString();
        }

        public static String getUserId() {
                HttpSession session = getSession();
                if (session != null)
                        return (String) session.getAttribute("userid");
                else
                        return null;
        }
}
```

Here we obtain a session for each user logged through the getUserId method thereby associating a session id to a particular user id.

Step 8: Create the authorization filter class as;

```
package com.journaldev.jsf.filter;

import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebFilter(filterName = "AuthFilter", urlPatterns = { "*.xhtml" })
public class AuthorizationFilter implements Filter {

        public AuthorizationFilter() {
        }

        @Override
        public void init(FilterConfig filterConfig) throws ServletException {

        }

        @Override
```

```
                        HttpServletRequest reqt = (HttpServletRequest) request;
                        HttpServletResponse resp = (HttpServletResponse) response;
                        HttpSession ses = reqt.getSession(false);
```

We implement the standard filter class by overriding the destroy and doFilter methods. In the doFilter method we will redirect user to login page if he tries to access other page without logging in.

Step 9: Create `admin.xhtml` as;

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
        xmlns:h="http://java.sun.com/jsf/html">
<h:head>
        <title>Facelet Title</title>
</h:head>
<h:body>
        <h:form>
                <p>Welcome #{login.user}</p>
                <h:commandLink action="#{login.logout}" value="Logout">
</h:commandLink>
        </h:form>
</h:body>
</html>
```

This page is rendered when the user logs in successfully. Logout functionality is implemented by calling the logout method of the `Login.java` class.

Step 10: Create `faces-config.xml` file as;

```
<?xml version='1.0' encoding='UTF-8'?>
<faces-config version="2.2" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_2.xsd">

        <navigation-rule>
                <from-view-id>/login.xhtml</from-view-id>
                <navigation-case>
                        <from-outcome>admin</from-outcome>
                        <to-view-id>/admin.xhtml</to-view-id>
                </navigation-case>
        </navigation-rule>
```

browser.

Login Page

Authentication Error Page

Login Success Page

Accessing admin.xhtml while logged in

Just click on the Logout link and the session will be invalidated, after that try to access admin.xhtml page and you will be redirected to the login page, go ahead and download the project from below link and try it out.

Download JSF Authentication Login Logout Project
9026 downloads

FILED UNDER: JAVA EE, JSF

About Pankaj

« 5 Advanced Java Books for Experienced Programmers

JSF Interview Questions And Answers »

Comments

shekar says

APRIL 17, 2017 AT 4:43 AM

Great examples

Reply

doppioB says

MARCH 21, 2017 AT 7:12 AM

Hi, i took some classes of your code and when I tried to login, it send me that error:
Etat HTTP 500 – java.lang.NullPointerException
The error pointed to "session.setAttribute ("key", value), I thought it came from my DBB but after long researchs I just TRY to change the return value in the method HttpSession getSession() and I put. "true" instead of "false" and I don't know why I could access to the next page after login. I would like to know why 😀

Reply

ryan says

FEBRUARY 16, 2017 AT 3:57 AM

works great but when i log out i can still go back using the back navigation button .How can i disable it

Reply

Rod Wilson says

DECEMBER 19, 2016 AT 6:46 AM

The warning messages will not render. An error message of "INFO: WARNING: FacesMessage(s) have been enqueued, but may not have been displayed." Any ideas as to why this is occurring?

Juan Lamata Feliz says

DECEMBER 13, 2016 AT 10:12 AM

gracias por el aporte <3

Reply

Willian says

DECEMBER 13, 2016 AT 6:41 AM

This is working for multiple users?

Reply

mhashimi says

NOVEMBER 30, 2016 AT 7:13 PM

this session can be used for multiple users or one, per machine/ip?
help please, a feedback would be nice 🙂

Reply

Aisha says

NOVEMBER 25, 2016 AT 7:40 AM

excuse me can you help me to do online nursery system by jsp it's important to finish it in this weekend.. please if you can send to me

Reply

Hans Newton says

NOVEMBER 13, 2016 AT 11:19 AM

It works perfect.

Thanks!

FEBRUARY 1, 2017 AT 9:05 AM

Hi , can you send me the Code !!

Reply

raed says

OCTOBER 29, 2016 AT 2:45 PM

Hello Pankaj,

could you please tell us or better show us with Code how can i check if the Session Timeout is happen and
the User hit a button to send a request how can we manage this situation to prevent an Exception because
Timeout of the Session ?

thanks Raed

Reply

yosser says

SEPTEMBER 30, 2016 AT 3:54 PM

thank u 🙂

Reply

Hrvoje says

AUGUST 9, 2016 AT 7:36 AM

I have a problem with this code, everything works great bu if I try to log in multiple users and then log out only one every users session is killed ? Quite a problem or just me ?

Reply

Fabio says

Tavakkaljon Dehqonov says

JUNE 23, 2016 AT 9:15 PM

HELO Pankaj.
I am using this proekt.
How can download package com.journaldev

Reply

Grzesiek says

JUNE 12, 2016 AT 1:25 AM

On
HttpSession session = SessionBean.getSession();
i've error: "error: cannot find symbol"

Can you help me?

Reply

Pankaj says

JUNE 12, 2016 AT 7:10 AM

Actually I changed the class name of SpringBean to SpringUtils and forgot to update the code in Login.java class. I have updated the code in the post as well as project zip file. You can download the project now, it will work fine.

Reply

Christian says

JUNE 8, 2016 AT 4:38 AM

java.lang.NullPointerException -> You have to add the mysql connector library.
It was perfect! That's work fine, thank you.

edward says

MAY 23, 2016 AT 3:36 AM

the app seems great though its throwing an exception "java.lang.NullPointerException"
why?

Reply

Toshyjoe says

MAY 15, 2016 AT 1:23 AM

Perfect! That's work fine, thank you 🙂

Reply

Vitor Da Costa says

MAY 3, 2016 AT 11:49 PM

Thanks a lot… =)

Reply

Samy says

APRIL 20, 2016 AT 8:59 AM

An Error Occurred:

java.lang.NullPointerException

Reply

ravi says

APRIL 3, 2016 AT 10:19 AM

Dear Pankaj,
Thanks a lot. The code you provided helped a lot with my project. One question though, how would you exclude a page from authentication. For example, if you want the user to see the home page first, which

Mustafa Darcan says

MARCH 1, 2016 AT 1:23 PM

I get an error of java.lang.NullPointerException .How can I fix this ?

Reply

zongi says

OCTOBER 4, 2016 AT 3:03 AM

🙂
this question Sound like
The Project don´t want run, how to fix it 🙂

Reply

Martin Zwernemann says

FEBRUARY 10, 2016 AT 9:40 AM

Without any entries to web.xml the AuthorizationFilter is never used. Minimum is to include it in web.xml in follwing manner (replace xxxx with your package name):

AuthorizationFilter
xxxx.filter.AuthorizationFilter
This Filter authorizes user access to application.

error_page
/error/error.xhtml

Reply

Martin Zwernemann says

FEBRUARY 10, 2016 AT 9:44 AM

Sorry, the xml was eaten by your server. I replaced the XML-marks with asterisks:
filter
*filter-name*AuthorizationFilter*/filter-name*

param-value/ui/energy/error/error.xhtml*/param-value*
/init-param
/filter

Reply

Gholamali Irani says
JANUARY 11, 2016 AT 1:13 PM

Thanks a lot, so helpful
what is JSF managed bean behavior with static method?
Is it safe with multiple online user? (conflict sessions or not !!)

Reply

Alessandro Mattiuzzi says
DECEMBER 15, 2015 AT 5:17 AM

Really good my friend. Great example

Reply

Krzysiek says
DECEMBER 4, 2015 AT 10:04 AM

The class name “LoginDAO” is misleading as this it not a DAO object at all, it’s just a simple class which contain one (static) method.

Reply

Askat says
OCTOBER 27, 2015 AT 11:41 PM

let’s say in the Users table is a field department , how to map this field to the JSF page?

alfredo fernandes says

OCTOBER 10, 2015 AT 6:47 AM

very good example.
why after logout if you press back button in browser in not invalidated showing the admin page with the name of the logged user?
thank you for a reply.

Reply

BurakErk says

APRIL 18, 2016 AT 4:19 AM

I think this is a good question. We sould look for that.

Reply

BurakErk says

APRIL 18, 2016 AT 4:23 AM

```
public String logout() {
HttpSession session = SessionBean.getSession();
user = "";
pwd = "";
session.invalidate();
return "login";
}
```

This sould work.

Reply

BurakErk says

APRIL 20, 2016 AT 4:08 AM

Use this codefor redirection.
return "Login.xhtml?faces-redirect=true";
——————
return "login";
This code is just forwarding the page. And that situation user can click browser's "back button" and see the page which allready logouted.

Reply

Thien says

OCTOBER 3, 2015 AT 9:24 AM

Hi you. Thank you so much. But i have any question. In the file faces-config.xml, why not add a code:

controller.SercurityFilter

And. I can implements PhaseListener instead of implements Filter in the file AuthorizationFilter. Thank you.

Reply

Thien says

OCTOBER 3, 2015 AT 9:25 AM

controller.SercurityFilter

Reply

Ainsley says

SEPTEMBER 22, 2015 AT 5:34 PM

Hello Pankaj I was reading your tutorial and it really gave me some insights,I tried it myself but it does not work.It does not check username and password against the database but passes the values

Reply

Philip Grove says

SEPTEMBER 11, 2015 AT 9:36 AM

Upon further investigation of the example it appear to contain code that is never used and code that suggest it has been directly copied from another source.

Reveal this source immediately and stop taking credit for the work of others.

Reply

understanding the concept of authentication in JSF, if I will provide production level coding here, the length of post will be 3 times and it will loose the purpose of article.

Reply

Philip Grove says

SEPTEMBER 14, 2015 AT 5:41 PM

No way am I doing your work for you. There is redundant code in the example, and if you are the programmer you claim to be you should be able it find it yourself.

Not catching Exception but rather subclasses is not only the way to go in production code it also makes better examples as it reveals the potential problems. Nor would it increase your post length by a factor of 3, as it can be done simply by editing the catch part of the statement and doesn't added a single line. So your argumentation is invalid, and the example needs updating.

Reply

Pankaj says

JUNE 5, 2016 AT 1:11 AM

Imagine I accuse you to stole Monalisa painting and then asking you to gather proof for me, sounds familiar?

And for the common code, every java program will have try catch blocks with exception being caught and logged, you should know this.

Reply

Paul Kelly says

OCTOBER 25, 2016 AT 2:33 PM

Your a dick. He is just trying to help beginners. If you don't like it, fuck off and write your own brilliant tutorials for the world to see

Reply

Never ever catch “Exception” in production code, it has loads on unforeseen consequences. I had hoped that it was not done here to promote proper exception handling.

Catching “Exception” is sometimes done in the test phase before proper exception handling is done, because proper exception handling on something that might not even work is a waste of time.

Reply

Philip Grove says

SEPTEMBER 10, 2015 AT 3:54 PM

Rename the class “SessionBean” in the example immediately. It's not a bean so the name is confusing.

Reply

daniel says

MAY 10, 2016 AT 7:22 AM

YES. because the clase is named “SessionBean” i have lost time truing to understand what it means.

Reply

Boris says

AUGUST 14, 2015 AT 6:43 AM

Excellent but I have a question… what happend with AuthorizationFilter

Reply

jacklyn onye says

JULY 23, 2015 AT 1:58 AM

Thanks for your tutorial, it was very helpful. is there any way we can use entity class that connect to database? trying not to code the sql statement.

Thanks so much

Name says

JULY 2, 2015 AT 7:11 AM

is it specified which version of JSF is used here ?

and what is its jar ?

Reply

Maor says

JUNE 17, 2015 AT 2:39 AM

Thanks, really helped!

Reply

Faycal says

MAY 26, 2015 AT 7:28 AM

Thank's for this example,

I've an error in SessionBean, i'm using JSF 2.1, how to import it ?

Reply

akasozi says

MAY 12, 2015 AT 3:55 PM

nice tutorial, however you forgot to specify mappings on web.xml file i.e.

AuthorizationFilter
*.AuthirizationFilter

AuthorizationFilter
/secured/*

Reply

Can you please provide a complete web.xml example showing the filter mappings? Thanks!

Reply

Ivan says

MAY 10, 2015 AT 2:46 AM

Can I use this to show parts of a web site?

```
Content logged users
```

is secure? or rendered param can be inyectable

Reply

Ivan says

MAY 10, 2015 AT 2:42 AM

Can I use this to show parts of a web site?

Content logged users

is secure? or render param can be inyectable

Reply

Leave a Reply

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

Comment

Email

Website

POST COMMENT

Custom Search

STAY UPDATED!

Name

E-Mail Address

I AM IN!

RECOMMENDED TUTORIALS

Java Tutorials: Java IO Tutorial, Java Regular Expressions Tutorial, Multithreading in Java, Java Logging API Tutorial, Java Annotations,Java XML Tutorial, Collections in Java, Java Generics, Exception Handling in Java, Java Reflection, Java Design Patterns, JDBC Tutorial

Java EE: Servlet JSP Tutorial, Struts2 Tutorial, Spring Tutorial, Hibernate Tutorial, Primefaces Tutorial

Web Services: Apache Axis 2 Tutorial, JAX-

Hosting

IMPORTANT INTERVIEW QUESTIONS

Java String Interview Questions, Java Multithreading Interview Questions, Java Programming Interview Questions, Java Interview Questions, Java Collections Interview Questions, Java Exception Interview Questions

Servlet Interview Questions, JSP Interview Questions, Struts2 Interview Questions, JDBC Interview Questions, Spring Interview Questions, Hibernate Interview Questions