

# **Autenticação e Autorização**

## **Introdução**

A segurança em aplicações corporativas está relacionada a diversos aspectos, tais como: autenticação, autorização e auditoria. A autenticação identifica quem acessa o sistema (se o usuário é quem ele diz que é – isto é, se ele é autêntico), a autorização determina o que um usuário autenticado pode fazer, e a auditoria diz o que o usuário fez.

## **Autenticação**

A autenticação determina se um usuário, que possui uma determinada identificação, é quem ele realmente diz que é. Durante a autenticação a identidade é verificada através de uma credencial (uma senha, por exemplo) fornecida pelo usuário.

## **Autorização**

A autorização define quais direitos e permissões tem o usuário do sistema. Após o usuário ser autenticado, o processo de autorização determina o que ele pode fazer no sistema.

## **Auditoria**

A auditoria está relacionada à coleta de informações relacionadas à utilização dos recursos de um sistema pelos seus usuários. Estas informações podem ser utilizadas para gerenciamento, planejamento, cobrança etc.

## **Segurança na plataforma Java EE**

A especificação Java EE define duas formas de implementação de recursos de segurança: declarativa e programática.

A segurança declarativa é aquela em que especificamos a configuração dos nossos serviços de segurança e o servidor de aplicações gerencia a segurança de acordo com o que foi definido em nossas especificações (baseada em arquivos de configuração – como o web.xml). Na segurança programática, a segurança é implementada através de codificação.

## Segurança programática com filtros de interceptação

O filtro de interceptação é um recurso da especificação Servlet 2.3 (ou posterior) que permite que as requisições que chegam a um servidor (como o Tomcat) sejam interceptadas por uma classe Java comum que pode realizar qualquer tipo de processamento e autorizar a requisição ou cancelá-la (por exemplo: redirecionando o usuário para algum recurso).

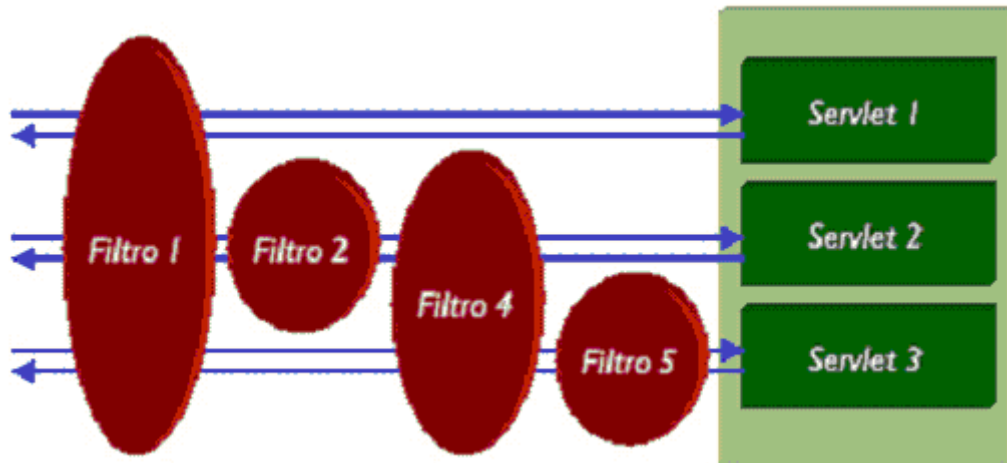


Figura 1. Filtros interceptando requisições (requests).

Sem o uso de filtro de interceptação, o acesso aos recursos de uma aplicação web pode ser ilustrado através do diagrama de seqüências seguinte.

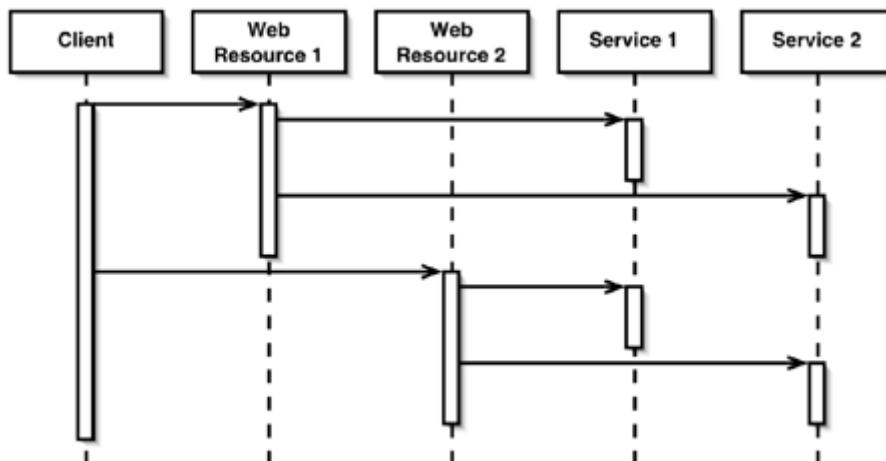


Figura 2. Diagrama de seqüências ilustrativos do acesso a recursos em uma aplicação que não usa filtros de interceptação.

Caso algum filtro de interceptação tivesse sido usado, o diagrama ficaria conforme a Figura 3.

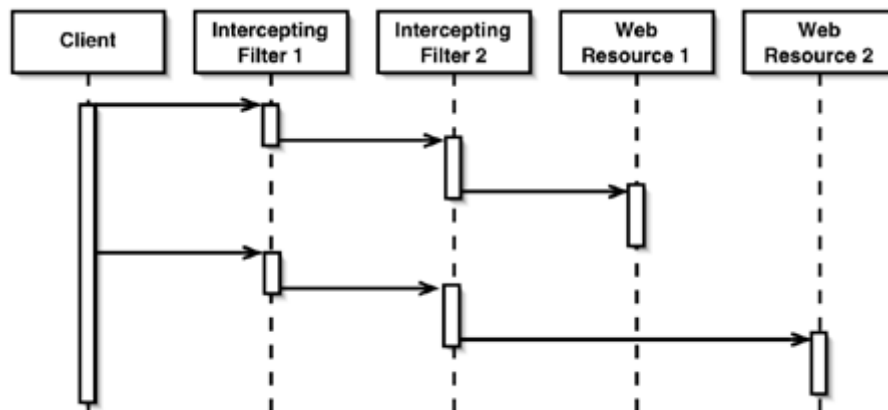


Figura 3. Diagrama de seqüências ilustrativos do acesso a recursos em uma aplicação que usa filtros de interceptação.

Ao escrever uma classe que implementa um filtro de interceptação, você lida com três interfaces no pacote `javax.servlet`:

- `Filter`
- `FilterConfig`
- `FilterChain`

Toda classe que implementa um filtro deve implementar a interface `Filter`. O ciclo de vida de um filtro é representado por três métodos desta interface: o **init**, **doFilter** e **destroy**.

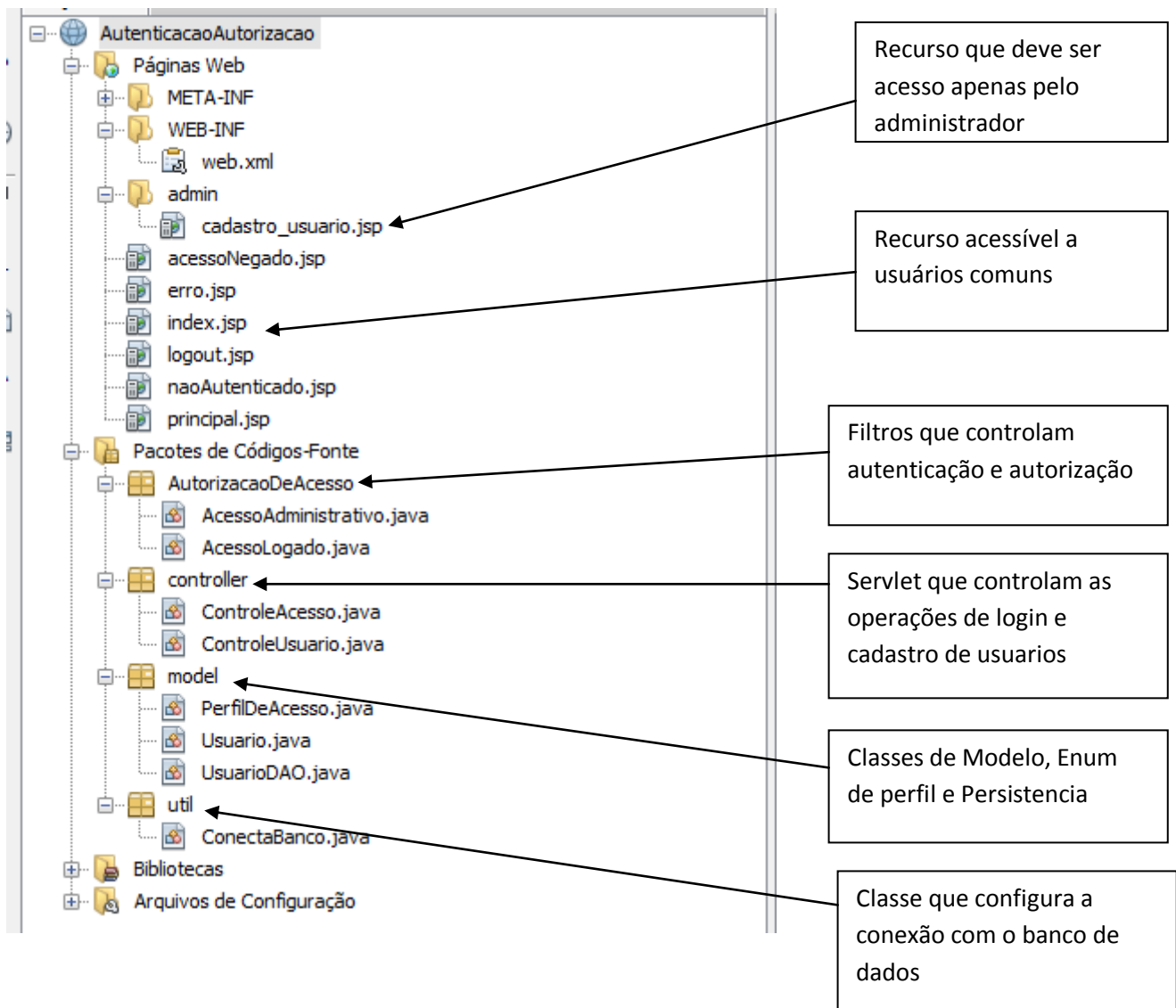
## Exercício – Implementando Autenticação e Autorização

Objetivo desta deste exercício é criar uma aplicação que faça uso de filtros para o controle de acesso a áreas restritas da aplicação.

### 1. Cria um banco de dados com o nome usuário e uma tabela também chamada usuário:

```
create table usuario(  
id serial PRIMARY KEY,  
login varchar NOT NULL,  
senha varchar NOT NULL,  
perfil varchar NOT NULL  
);
```

### 2. Cria uma aplicação web no NetBeans com a seguinte organização



### 3. No pacote Util crie a classe que configura a conexão com o banco de dados

```
package util;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ConectaBanco {
    public static Connection getConexao() {
        Connection conexao = null;
        try {
            //driver que será utilizado
            Class.forName("org.postgresql.Driver");
            //cria um objeto de conexao com um banco especificado no caminho...
            conexao = DriverManager.getConnection("jdbc:postgresql://127.0.0.1:5432/usuario",
"postgres", "admin");
        } catch (ClassNotFoundException erro1) {
            throw new RuntimeException(erro1);

        } catch (SQLException erro2) {
            throw new RuntimeException(erro2);

        }
        return conexao;
    }
}
```

### 4. No pacote modelo crie o Enum que define o perfil de cada usuário e a classe Usuario

```
1 //
2
3 package model;
4
5 public enum PerfilDeAcesso {
6     COMUM,
7     ADMINISTRADOR;
8 }
9
```

```

5  package model;
6
7  public class Usuario {
8      private String login;
9      private String senha;
10     private PerfilDeAcesso perfil;
11
12     //GETS E SETS
13
14
15
16 }

```

- 5. Ainda no pacote Modelo, cria a classe UsuarioDAO com o método que cadastra um novo usuário e outro que autentica um usuário no banco de dados.**

```
package model;
```

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import util.ConectaBanco;
```

```
public class UsuarioDAO {
```

```
    private static final String CADASTRA_NOVO_USUARIO = "INSERT INTO usuario (login, senha, perfil) VALUES (?, ?, ?)";
```

```
    private static final String AUTENTICA_USUARIO = "SELECT * FROM usuario WHERE login=? AND senha=?";
```

```
    public void cadastraNovoUsuario(Usuario usuario) {
```

```
        Connection conexao = null;
```

```
        PreparedStatement pstmt = null;
```

```
        try {
```

```
            conexao = ConectaBanco.getConexao();
```

```
            pstmt = conexao.prepareStatement(CADASTRA_NOVO_USUARIO);
```

```
            pstmt.setString(1, usuario.getLogin());
```

```
            pstmt.setString(2, usuario.getSenha());
```

```
            pstmt.setString(3, usuario.getPerfil().toString());
```

```
            pstmt.execute();
```

```
        } catch (SQLException sqlErro) {
```

```
            throw new RuntimeException(sqlErro);
```

```
        } finally {
```

```
            if (conexao != null) {
```

```
                try {
```

```
                    conexao.close();
```

```

        } catch (SQLException ex) {
            throw new RuntimeException(ex);
        }
    }
}

}

public Usuario autenticaUsuario(Usuario usuario) {
    Usuario usuarioAutenticado = null;

    Connection conexao = null;
    PreparedStatement pstmt = null;
    ResultSet rsUsuario = null;
    try {

        conexao = ConectaBanco.getConexao();
        pstmt = conexao.prepareStatement(AUTENTICA_USUARIO);
        pstmt.setString(1, usuario.getLogin());
        pstmt.setString(2, usuario.getSenha());
        rsUsuario = pstmt.executeQuery();

        if (rsUsuario.next()) {
            usuarioAutenticado = new Usuario();
            usuarioAutenticado.setLogin(rsUsuario.getString("login"));
            usuarioAutenticado.setSenha(rsUsuario.getString("senha"));
            usuarioAutenticado.setPerfil(PerfilDeAcesso.valueOf(rsUsuario.getString("perfil")));
        }

    } catch (SQLException sqlErro) {
        throw new RuntimeException(sqlErro);
    } finally {
        if (conexao != null) {
            try {
                conexao.close();
            } catch (SQLException ex) {
                throw new RuntimeException(ex);
            }
        }
    }

    return usuarioAutenticado;
}
}

```

**6. Na pasta admin, crie um formulário para cadastro de um novo usuário do sistema.**

```
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>
8 <!DOCTYPE html>
9 <html>
10 <head>
11 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12 <title>Área Restrita</title>
13 </head>
14 <body>
15 <h1>Área de acesso restrito aos administradores!</h1>
16 <h2>Cadastro de novo usuário!</h2>
17
18 <%
19     String msg = (String) request.getAttribute("msg");
20     if(msg != null){
21
22         <font color="blue"><%=msg %></font>
23         <%=msg %>
24     <%>
25     <form action="ControleUsuario" method="POST">
26         Login: <input type="text" name="txtLogin"><br/>
27         Senha: <input type="password" name="txtSenha"><br/>
28         Perfil: <select name="optPerfil">
29             <option>COMUM</option>
30             <option>ADMINISTRADOR</option>
31         </select><br/>
32         <input type="submit" value="Cadastrar" name="acao">
33     </form>
34     <a href="../principal.jsp">Página Principal</a>
35 </body>
</html>
```

**7. Na pacote Controle, cria a Servlet ControleUsuario responsável pelo cadastro de um novo usuário.**

```
package controller;
```

```
import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import model.PerfilDeAcesso;
import model.Usuario;
import model.UsuarioDAO;
```

```
public class ControleUsuario extends HttpServlet {
```



```

protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");

    try {

        String acao = request.getParameter("acao");
        if (acao.equals("Cadastrar")) {
            Usuario usuario = new Usuario();
            usuario.setLogin(request.getParameter("txtLogin"));
            usuario.setSenha(request.getParameter("txtSenha"));
            String perfil = request.getParameter("optPerfil");
            if (perfil.equalsIgnoreCase("administrador")) {
                usuario.setPerfil(PerfilDeAcesso.ADMINISTRADOR);
            } else {
                usuario.setPerfil(PerfilDeAcesso.COMUM);
            }

            UsuarioDAO usuarioDAO = new UsuarioDAO();
            usuarioDAO.cadastraNovoUsuario(usuario);
            request.setAttribute("msg", "cadastrado com sucesso");
            RequestDispatcher rd =
request.getRequestDispatcher("/admin/cadastro_usuario.jsp");
            rd.forward(request, response);
        }

        } catch (Exception erro) {
            RequestDispatcher rd = request.getRequestDispatcher("/erro.jsp");
            request.setAttribute("erro", erro);
            rd.forward(request, response);
        }

    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }
}

```

**8. Dentro do diretório Paginas Web crie o JSP erro.jsp**

```
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>
8 <!DOCTYPE html>
9 <html>
10 <head>
11 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12 <title>Error Page</title>
13 </head>
14 <body>
15 <h1>Erro!</h1>
16 <%= ((Exception) request.getAttribute("erro")).getMessage() %>
17 </body>
18 </html>
19
```

**9. Execute e o arquivo cadastro\_usuario.jsp e o funcionamento da aplicação até o momento.**

**10. No arquivo index.jsp, vamos criar um formulário para Autenticar um usuário que acessa a aplicação.**

```
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>
8 <!DOCTYPE html>
9 <html>
10 <head>
11 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12 <title>Autenticação</title>
13 </head>
14 <body>
15 <h1>Autenticação de Usuário!</h1>
16 <%
17     String msg = (String) request.getAttribute("msg");
18     if(msg!=null){
19
20
21         <font color="red"> <%=msg%></font>
22         <%=msg%>
23     <form action="ControleAcesso" method="POST">
24         Login: <input type="text" name="txtLogin"><br/>
25         Senha: <input type="password" name="txtSenha"><br/>
26         <input type="submit" value="Entrar" name="acao">
27     </form>
28 </body>
29 </html>
30
```

**11. No pacote Controle, cria a Servlet ControleAcesso responsável por autenticar o usuário.**

```
package controller;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import model.Usuario;
import model.UsuarioDAO;

public class ControleAcesso extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        try {

            String acao = request.getParameter("acao");
            if (acao.equals("Entrar")) {
                Usuario usuario = new Usuario();
                usuario.setLogin(request.getParameter("txtLogin"));
                usuario.setSenha(request.getParameter("txtSenha"));

                UsuarioDAO usuarioDAO = new UsuarioDAO();
                Usuario usuarioAutenticado = usuarioDAO.autenticaUsuario(usuario);
                //se o usuario existe no banco de dados
                if (usuarioAutenticado != null) {

                    //cria uma sessao para o usuario
                    HttpSession sessaoUsuario = request.getSession();
                    sessaoUsuario.setAttribute("usuarioAutenticado", usuarioAutenticado);
                    //redireciona para a pagina principal
                    response.sendRedirect("principal.jsp");
                } else {
                    RequestDispatcher rd = request.getRequestDispatcher("/index.jsp");
                    request.setAttribute("msg", "Login ou Senha Incorreto!");
                    rd.forward(request, response);
                }
            }

        } else
```

```

        if(acao.equals("Sair")){
            HttpSession sessaoUsuario = request.getSession();
            sessaoUsuario.removeAttribute("usuarioAutenticado");
            response.sendRedirect("logout.jsp");
        }

    } catch (Exception erro) {
        RequestDispatcher rd = request.getRequestDispatcher("/erro.jsp");
        request.setAttribute("erro", erro);
        rd.forward(request, response);
    }

}

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    processRequest(request, response);
}

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    processRequest(request, response);
}
}

```

**12. No diretório Páginas Web, crie o JSP principal.jsp, que representará a tela principal da nossa aplicação.**

```

7  <%@page import="model.Usuario"%>
8  <%@page contentType="text/html" pageEncoding="UTF-8"%>
9  <!DOCTYPE html>
10 <html>
11 <head>
12     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13     <title>Página Principal</title>
14 </head>
15 <body>
16
17     <%
18         //recupera o usuario da sessao
19         Usuario usuario = (Usuario)session.getAttribute("usuarioAutenticado");
20
21         if(usuario !=null){
22             %>
23             <h1>Bem-vindo, <%= usuario.getLogin() %> !</h1>
24             <%}%>
25             <a href="admin/cadastro_usuario.jsp"> Área restrita</a><br/>
26             <a href="ControleAcesso?acao=Sair">Logout</a>
27         </body>
28 </html>

```

**13. No diretório Páginas Web, crie o JSP logout.jsp, que será exibido quando o usuário desejar sair do sistema.**

```

7  <%@page contentType="text/html" pageEncoding="UTF-8"%>
8  <!DOCTYPE html>
9  <html>
10 <head>
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12     <title>Logout</title>
13 </head>
14 <body>
15     <h1>Obrigado pela visita!</h1>
16     <a href="index.jsp">Logar novamente</a>
17 </body>
18 </html>

```

**14. Vamos testar o funcionamento de nossa aplicação !**

Cadastrando um usuário como administrador do sistema: Login: joao e senha:123

# Área de acesso restrito aos administradores!

## Cadastro de novo usuário!

Login:

Senha:

Perfil: ADMINISTRADOR ▾

[Página Principal](#)

Cadastrando um usuário como usuário comum do sistema: Login marcos e senha: 123

# Área de acesso restrito aos administradores!

## Cadastro de novo usuário!

[cadastrado com sucesso](#)

Login:

Senha:

Perfil: COMUM ▾

[Página Principal](#)

No banco de dados, temos os usuários cadastrados

File Edit View Tools Help					
No limit ▾					
	id [PK]	serial	login character varying	senha character varying	perfil character varying
1	6		joao	123	ADMINISTRADOR
2	7		marcos	123	COMUM
*					

Quando entramos com o login e senha de um usuário cadastrado acessamos o sistema com sucesso, caso contrario uma mensagem de erro e exibida.

Java Image Processing... Java Advanced Image

## Autenticação de Usuário!

# Bem-vindo,joao !

[Área restrita](#)  
[Logout](#)

Login ou Senha Incorreto!

Login:

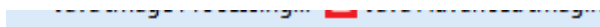
Senha:

Até o momento percebemos que a nossa Autenticação de usuário esta ocorrendo da maneira correta.

## **Mas, e se testarmos a Autorização de acesso a pagina restrita do administrador?**

### **15. Testando a Autorização**

Acesso com o usuário de perfil comum, marcos por exemplo.



# **Bem-vindo,marcos !**

[Área restrita](#)  
[Logout](#)

Se clicarmos no link, Área restrita, perceberemos que na verdade não há restrição.

A situação pode ser pior, de digitarmos a URL correta, podemos acessar diretamente a Área restrita até mesmo sem realizar o login.

Exemplo de URL: [http://localhost:8084/AutenticacaoAutorizacao/admin/cadastro\\_usuario.jsp](http://localhost:8084/AutenticacaoAutorizacao/admin/cadastro_usuario.jsp)

## **Área de acesso restrito aos administradores!**

### **Cadastro de novo usuário!**

Login:

Senha:

Perfil:

[Página Principal](#)

Para resolver este problema precisamos de um controle de AUTORIZAÇÃO e é exatamente isso que faremos com a implementação de FILTROS.

16. Dentro do pacote `AutorizacaoDeAcesso`, implemente um filtro que controle o acesso a área restrita do nosso sistema.

```
20 //imports
21 public class AcessoAdministrativo implements Filter {
22
23     @Override
24     public void init(FilterConfig filterConfig) throws ServletException {
25
26     }
27
28     @Override
29     public void doFilter(ServletRequest request, ServletResponse response,
30                         FilterChain chain) throws IOException, ServletException {
31         //recuperar a sessao
32         HttpSession sessaoUsuario = ((HttpServletRequest)request).getSession();
33         Usuario usuario = (Usuario) sessaoUsuario.getAttribute("usuarioAutenticado");
34
35         if(usuario!=null && usuario.getPerfil().equals(PerfilDeAcesso.ADMINISTRADOR)){
36             chain.doFilter(request, response);
37         }else{
38             ((HttpServletResponse)response).sendRedirect("../acessoNegado.jsp");
39         }
40     }
41
42 }
43
44 @Override
45 public void destroy() {
46
47 }
48
49 }
```

17. Abra o arquivo `web.xml` para declararmos este filtro e definirmos que URL ele deve interceptar.

```
6
7 <!--Declaração dos Filtros -->
8 <filter>
9     <filter-name>AcessoAdministrativo</filter-name>
10    <filter-class>AutorizacaoDeAcesso.AcessoAdministrativo</filter-class>
11 </filter>
12 <filter-mapping>
13     <filter-name>AcessoAdministrativo</filter-name>
14     <url-pattern>/admin/*</url-pattern>
15 </filter-mapping>
16
```



18. No diretório Páginas Web crie o JSP acessoNegado.jsp, que será utilizado pelo nosso filtro.

```
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>
8 <!DOCTYPE html>
9 <html>
10 <head>
11 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12 <title>Acesso Negado</title>
13 </head>
14 <body>
15 <h1>Você não tem permissão de acesso ...</h1>
16 </body>
17 </html>
```

19. Realize o login no sistema um usuário de perfil comum, marcos por exemplo, e tente acessar a área restrita.

## Bem-vindo,marcos !

[Área restrita](#)  
[Logout](#)

Usuário comum tentando  
acessar a área restrita.

Java Image Processing... Java Advanced Imaging veducal - Educação de ... Portal

## Você não tem permissão de acesso ...

Com este filtro resolvemos o acesso à área restrita!!!

20. Realize um novo teste ....

Digite uma URL tentando pular a obrigatoriedade da Autenticação.

Exemplo: <http://localhost:8084/AutenticacaoAutorizacao/principal.jsp>

Perceba que ainda é possível entrar no sistema sem ser autenticado.

[Área restrita](#)[Logout](#)

Para evitar esta situação nós vamos criar outro filtro para controlar a obrigatoriedade da autenticação.

## 21. No pacote `AutorizacaoDeAcesso` crie um novo Filtro com o nome `AcessoLogado`.

```
19 //imports
20 public class AcessoLogado implements Filter {
21
22     @Override
23     public void init(FilterConfig filterConfig) throws ServletException {
24
25     }
26
27     @Override
28     public void doFilter(ServletRequest request,
29                         ServletResponse response, FilterChain chain)
30                         throws IOException, ServletException {
31
32         HttpSession sessaoUsuario = ((HttpServletRequest)request).getSession();
33         Usuario usuarioLogado = (Usuario)sessaoUsuario.getAttribute("usuarioAutenticado");
34
35         if(usuarioLogado !=null){
36             chain.doFilter(request, response);
37         }else{
38             ((HttpServletResponse)response).sendRedirect("naoAutenticado.jsp");
39         }
40     }
41
42     @Override
43     public void destroy() {
44
45     }
46 }
```

**22. Abra o arquivo web.xml e adicione a identificação deste filtro mais a URL que o mesmo irá interceptar.**

```
18
19 <filter>
20     <filter-name>AcessoLogado</filter-name>
21     <filter-class>AutorizacaoDeAcesso.AcessoLogado</filter-class>
22 </filter>
23 <filter-mapping>
24     <filter-name>AcessoLogado</filter-name>
25     <url-pattern>/principal.jsp</url-pattern>
26 </filter-mapping>
27
```

**23. No diretório Paginas Web crie um JSP chamado naoAutenticado.jsp**

```
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>
8 <!DOCTYPE html>
9 <html>
10 <head>
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12     <title>JSP Page</title>
13 </head>
14 <body>
15     <h1>Você deve estar logado no sistema!</h1>
16 </body>
17 </html>
18
```

**24. Tese novamente a aplicação tentando burlar a restrição de acesso e a obrigatoriedade da autenticação.**