Exemplo de uso de um objeto no contexto de sessão Jakarta java 17 Tomcat 11

Aqui está um exemplo completo de uso de objetos em contexto de sessão usando Jakarta EE com Java 17 e Tomcat 11:

1. Classe Model (Objeto para armazenar na sessão)

java

```java
package com.exemplo.model;

import java.io.Serializable;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

public class UsuarioSessao implements Serializable {
    private static final long serialVersionUID = 1L;

    private String nome;
    private String email;
    private LocalDateTime loginTime;
    private List<String> atividades;

    public UsuarioSessao() {
        this.atividades = new ArrayList<>();
        this.loginTime = LocalDateTime.now();
    }

    public UsuarioSessao(String nome, String email) {
        this();
        this.nome = nome;
        this.email = email;
        adicionarAtividade("Usuário criado");
    }

    public void adicionarAtividade(String atividade) {
        atividades.add(LocalDateTime.now() + " - " + atividade);
    }

    // Getters e Setters
```

```
    public String getNome() { return nome; }
    public void setNome(String nome) { this.nome = nome; }

    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }

    public LocalDateTime getLoginTime() { return loginTime; }
    public void setLoginTime(LocalDateTime loginTime) { this.loginTime = loginTime;
}

    public List<String> getAtividades() { return atividades; }
    public void setAtividades(List<String> atividades) { this.atividades = atividad
es; }
}
```

2. Servlet para Login (Armazenar na sessão)

java

```
package com.exemplo.servlet;

import com.exemplo.model.UsuarioSessao;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import java.io.IOException;

@WebServlet("/login")
public class LoginServlet extends HttpServlet {

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

        String nome = request.getParameter("nome");
        String email = request.getParameter("email");

        // Criar objeto usuário
        UsuarioSessao usuario = new UsuarioSessao(nome, email);
        usuario.adicionarAtividade("Realizou login");

        // Obter ou criar sessão
        HttpSession sessao = request.getSession(true);
```

```
        // Armazenar objeto na sessão
        sessao.setAttribute("usuario", usuario);
        sessao.setAttribute("ultimoAcesso", System.currentTimeMillis());

        // Configurar tempo de sessão (30 minutos)
        sessao.setMaxInactiveInterval(30 * 60);

        response.sendRedirect("dashboard");
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        request.getRequestDispatcher("/login.jsp").forward(request, response);
    }
}
```

3. Servlet para Dashboard (Ler da sessão)

java

```
package com.exemplo.servlet;

import com.exemplo.model.UsuarioSessao;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import java.io.IOException;

@WebServlet("/dashboard")
public class DashboardServlet extends HttpServlet {

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

        HttpSession sessao = request.getSession(false);

        // Verificar se existe sessão e usuário
        if (sessao == null || sessao.getAttribute("usuario") == null) {
            response.sendRedirect("login");
            return;
```

```
        }

        // Recuperar objeto da sessão
        UsuarioSessao usuario = (UsuarioSessao) sessao.getAttribute("usuario");
        usuario.adicionarAtividade("Acessou dashboard");

        // Atualizar na sessão
        sessao.setAttribute("usuario", usuario);

        // Disponibilizar para a JSP
        request.setAttribute("usuario", usuario);
        request.getRequestDispatcher("/dashboard.jsp").forward(request, response);
    }
}
```

4. Servlet para Atualizar (Modificar na sessão)

java

```
package com.exemplo.servlet;

import com.exemplo.model.UsuarioSessao;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import java.io.IOException;

@WebServlet("/atualizar-perfil")
public class AtualizarPerfilServlet extends HttpServlet {

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

        HttpSession sessao = request.getSession(false);

        if (sessao == null || sessao.getAttribute("usuario") == null) {
            response.sendRedirect("login");
            return;
        }

        // Recuperar da sessão
        UsuarioSessao usuario = (UsuarioSessao) sessao.getAttribute("usuario");
```

```
        // Atualizar dados
        String novoNome = request.getParameter("nome");
        String novoEmail = request.getParameter("email");

        if (novoNome != null && !novoNome.isEmpty()) {
            usuario.setNome(novoNome);
            usuario.adicionarAtividade("Nome atualizado para: " + novoNome);
        }

        if (novoEmail != null && !novoEmail.isEmpty()) {
            usuario.setEmail(novoEmail);
            usuario.adicionarAtividade("Email atualizado para: " + novoEmail);
        }

        // Atualizar na sessão (não é necessário re-set, pois é o mesmo objeto)
        // Mas podemos chamar setAttribute novamente para garantir
        sessao.setAttribute("usuario", usuario);

        response.sendRedirect("dashboard");
    }
}
```

5. Servlet para Logout (Remover da sessão)

java

```
package com.exemplo.servlet;

import com.exemplo.model.UsuarioSessao;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import java.io.IOException;

@WebServlet("/logout")
public class LogoutServlet extends HttpServlet {

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

        HttpSession sessao = request.getSession(false);
```

```
        if (sessao != null) {
            // Opção 1: Remover apenas o atributo específico
            // sessao.removeAttribute("usuario");

            // Opção 2: Invalidar a sessão inteira (mais comum)
            sessao.invalidate();
        }

        response.sendRedirect("login");
    }
}
```

6. Listener para Monitorar Sessões

java

```
package com.exemplo.listener;

import com.exemplo.model.UsuarioSessao;
import jakarta.servlet.annotation.WebListener;
import jakarta.servlet.http.HttpSession;
import jakarta.servlet.http.HttpSessionEvent;
import jakarta.servlet.http.HttpSessionListener;
import java.util.concurrent.atomic.AtomicInteger;

@WebListener
public class SessaoListener implements HttpSessionListener {

    private static final AtomicInteger sessoesAtivas = new AtomicInteger(0);

    @Override
    public void sessionCreated(HttpSessionEvent se) {
        sessoesAtivas.incrementAndGet();
        System.out.println("Sessão criada. Total ativas: " + sessoesAtivas.get());
    }

    @Override
    public void sessionDestroyed(HttpSessionEvent se) {
        sessoesAtivas.decrementAndGet();

        HttpSession sessao = se.getSession();
        UsuarioSessao usuario = (UsuarioSessao) sessao.getAttribute("usuario");

        if (usuario != null) {
            System.out.println("Sessão do usuário " + usuario.getNome() + " foi des
```

```
truída");
        }

        System.out.println("Sessão destruída. Total ativas: " + sessoesAtivas.get
());
    }

    public static int getSessoesAtivas() {
        return sessoesAtivas.get();
    }
}
```

7. Páginas JSP

login.jsp

```
jsp

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
    <h2>Login</h2>
    <form action="login" method="post">
        <label>Nome:</label>
        <input type="text" name="nome" required><br><br>

        <label>Email:</label>
        <input type="email" name="email" required><br><br>

        <input type="submit" value="Entrar">
    </form>
</body>
</html>
```

dashboard.jsp

```
jsp

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ page import="com.exemplo.model.UsuarioSessao" %>
```

```
<%@ page import="java.util.List" %>
<!DOCTYPE html>
<html>
<head>
    <title>Dashboard</title>
</head>
<body>
    <%
        UsuarioSessao usuario = (UsuarioSessao) request.getAttribute("usuario");
        if (usuario == null) {
            response.sendRedirect("login");
            return;
        }
    %>

    <h2>Bem-vindo, <%= usuario.getNome() %>!</h2>

    <p><strong>Email:</strong> <%= usuario.getEmail() %></p>
    <p><strong>Login realizado em:</strong> <%= usuario.getLoginTime() %></p>
    <p><strong>ID da Sessão:</strong> <%= session.getId() %></p>

    <h3>Atualizar Perfil</h3>
    <form action="atualizar-perfil" method="post">
        <label>Novo Nome:</label>
        <input type="text" name="nome" value="<%= usuario.getNome() %>"><br><br>

        <label>Novo Email:</label>
        <input type="email" name="email" value="<%= usuario.getEmail() %>"><br><br>

        <input type="submit" value="Atualizar">
    </form>

    <h3>Histórico de Atividades</h3>
    <ul>
        <%
            List<String> atividades = usuario.getAtividades();
            for (String atividade : atividades) {
        %>
            <li><%= atividade %></li>
        <%
            }
        %>
    </ul>

    <p><a href="logout">Sair</a></p>
```

```
</body>
</html>
```

8. web.xml (opcional - configurações adicionais)

```
xml

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
         https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
         version="6.0">

    <display-name>Exemplo Sessao Jakarta</display-name>

    <!-- Configuração de timeout da sessão (em minutos) -->
    <session-config>
        <session-timeout>30</session-timeout>
        <cookie-config>
            <http-only>true</http-only>
            <secure>false</secure>
        </cookie-config>
        <tracking-mode>COOKIE</tracking-mode>
    </session-config>
</web-app>
```

Características importantes:

1. **Serializable**: O objeto armazenado na sessão deve implementar `Serializable`
2. **HttpSession**: Interface principal para trabalhar com sessões
3. **Métodos principais**:
   - `getAttribute()` : Recuperar objeto
   - `setAttribute()` : Armazenar/atualizar objeto
   - `removeAttribute()` : Remover objeto
   - `invalidate()` : Invalidar sessão
4. **Escopo**: Objetos na sessão ficam disponíveis enquanto a sessão existir
5. **Timeout**: Configurável via `setMaxInactiveInterval()` ou `web.xml`

Este exemplo demonstra um fluxo completo de login, manipulação de dados na sessão e logout usando Jakarta EE com Tomcat 11.

Use xhtml

Aqui está um exemplo completo usando XHTML (Facelets) com Jakarta EE, Java 17 e Tomcat 11:

1. Dependências Maven (pom.xml)

xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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>com.exemplo</groupId>
    <artifactId>sessao-jsf</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>war</packaging>

    <properties>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <!-- Jakarta EE Web API -->
        <dependency>
            <groupId>jakarta.platform</groupId>
            <artifactId>jakarta.jakartaee-web-api</artifactId>
            <version>10.0.0</version>
            <scope>provided</scope>
        </dependency>

        <!-- Mojarra (JSF Implementation) -->
        <dependency>
            <groupId>org.glassfish</groupId>
            <artifactId>jakarta.faces</artifactId>
            <version>4.0.1</version>
        </dependency>
```

```
    </dependencies>

    <build>
        <finalName>sessao-jsf</finalName>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <version>3.3.2</version>
            </plugin>
        </plugins>
    </build>
</project>
```

2. Classe Model (Objeto para sessão)

java

```
package com.exemplo.model;

import java.io.Serializable;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.List;

public class UsuarioSessao implements Serializable {
    private static final long serialVersionUID = 1L;

    private String nome;
    private String email;
    private LocalDateTime loginTime;
    private List<String> atividades;
    private boolean loggedIn;

    public UsuarioSessao() {
        this.atividades = new ArrayList<>();
        this.loginTime = LocalDateTime.now();
        this.loggedIn = false;
    }

    public void adicionarAtividade(String atividade) {
        String timestamp = LocalDateTime.now().format(DateTimeFormatter.ofPattern
("dd/MM/yyyy HH:mm:ss"));
        atividades.add(timestamp + " - " + atividade);
    }
```

```
    // Getters e Setters
    public String getNome() { return nome; }
    public void setNome(String nome) { this.nome = nome; }

    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }

    public LocalDateTime getLoginTime() { return loginTime; }
    public void setLoginTime(LocalDateTime loginTime) { this.loginTime = loginTime;
}

    public List<String> getAtividades() { return atividades; }
    public void setAtividades(List<String> atividades) { this.atividades = atividad
es; }

    public boolean isLoggedIn() { return loggedIn; }
    public void setLoggedIn(boolean loggedIn) { this.loggedIn = loggedIn; }
}
```

3. Managed Bean com Escopo de Sessão

```
java

package com.exemplo.beans;

import com.exemplo.model.UsuarioSessao;
import jakarta.annotation.PostConstruct;
import jakarta.enterprise.context.SessionScoped;
import jakarta.faces.application.FacesMessage;
import jakarta.faces.context.FacesContext;
import jakarta.inject.Named;
import java.io.Serializable;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

@Named
@SessionScoped
public class SessaoBean implements Serializable {
    private static final long serialVersionUID = 1L;

    private UsuarioSessao usuario;
    private String nomeLogin;
    private String emailLogin;
    private String novoNome;
    private String novoEmail;
```

```
@PostConstruct
public void init() {
    usuario = new UsuarioSessao();
}

public String login() {
    if (nomeLogin != null && !nomeLogin.trim().isEmpty() &&
        emailLogin != null && !emailLogin.trim().isEmpty()) {

        usuario.setNome(nomeLogin);
        usuario.setEmail(emailLogin);
        usuario.setLoggedIn(true);
        usuario.setLoginTime(LocalDateTime.now());
        usuario.adicionarAtividade("Login realizado");

        // Limpar campos de login
        nomeLogin = null;
        emailLogin = null;

        // Adicionar mensagem de sucesso
        FacesContext.getCurrentInstance().addMessage(null,
            new FacesMessage(FacesMessage.SEVERITY_INFO,
            "Login realizado com sucesso!", null));

        return "dashboard?faces-redirect=true";
    }

    FacesContext.getCurrentInstance().addMessage(null,
        new FacesMessage(FacesMessage.SEVERITY_ERROR,
        "Por favor, preencha todos os campos", null));

    return null;
}

public String logout() {
    usuario.adicionarAtividade("Logout realizado");
    usuario.setLoggedIn(false);

    // Invalidar sessão
    FacesContext.getCurrentInstance().getExternalContext().invalidateSession();

    return "login?faces-redirect=true";
}

public String atualizarPerfil() {
    boolean atualizado = false;
```

```
            if (novoNome != null && !novoNome.trim().isEmpty()) {
                usuario.setNome(novoNome);
                usuario.adicionarAtividade("Nome atualizado para: " + novoNome);
                novoNome = null;
                atualizado = true;
            }

            if (novoEmail != null && !novoEmail.trim().isEmpty()) {
                usuario.setEmail(novoEmail);
                usuario.adicionarAtividade("Email atualizado para: " + novoEmail);
                novoEmail = null;
                atualizado = true;
            }

            if (atualizado) {
                FacesContext.getCurrentInstance().addMessage(null,
                    new FacesMessage(FacesMessage.SEVERITY_INFO,
                    "Perfil atualizado com sucesso!", null));
            }

            return null;
        }

        public String getTempoSessao() {
            if (usuario.getLoginTime() != null) {
                long minutos = java.time.Duration.between(usuario.getLoginTime(), Local
    DateTime.now()).toMinutes();
                return String.format("%d minutos", minutos);
            }
            return "0 minutos";
        }

        // Getters e Setters
        public UsuarioSessao getUsuario() { return usuario; }
        public void setUsuario(UsuarioSessao usuario) { this.usuario = usuario; }

        public String getNomeLogin() { return nomeLogin; }
        public void setNomeLogin(String nomeLogin) { this.nomeLogin = nomeLogin; }

        public String getEmailLogin() { return emailLogin; }
        public void setEmailLogin(String emailLogin) { this.emailLogin = emailLogin; }

        public String getNovoNome() { return novoNome; }
        public void setNovoNome(String novoNome) { this.novoNome = novoNome; }

        public String getNovoEmail() { return novoEmail; }
```

```
    public void setNovoEmail(String novoEmail) { this.novoEmail = novoEmail; }
}
```

4. Managed Bean para Informações da Sessão

java

```
package com.exemplo.beans;

import jakarta.enterprise.context.RequestScoped;
import jakarta.faces.context.FacesContext;
import jakarta.inject.Named;
import jakarta.servlet.http.HttpSession;
import java.io.Serializable;

@Named
@RequestScoped
public class SessaoInfoBean implements Serializable {

    public String getSessionId() {
        FacesContext facesContext = FacesContext.getCurrentInstance();
        HttpSession session = (HttpSession) facesContext.getExternalContext().getSe
ssion(false);
        return session != null ? session.getId() : "Não disponível";
    }

    public int getSessionMaxInactiveInterval() {
        FacesContext facesContext = FacesContext.getCurrentInstance();
        HttpSession session = (HttpSession) facesContext.getExternalContext().getSe
ssion(false);
        return session != null ? session.getMaxInactiveInterval() / 60 : 0;
    }

    public long getSessionCreationTime() {
        FacesContext facesContext = FacesContext.getCurrentInstance();
        HttpSession session = (HttpSession) facesContext.getExternalContext().getSe
ssion(false);
        return session != null ? session.getCreationTime() : 0;
    }
}
```

5. Páginas XHTML

login.xhtml

xhtml

```
<?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="jakarta.faces.html"
      xmlns:f="jakarta.faces.core"
      xmlns.ui="jakarta.faces.facelets">

<h:head>
    <title>Login - Sistema de Sessão</title>
    <style type="text/css">
        .container {
            width: 400px;
            margin: 50px auto;
            padding: 20px;
            border: 1px solid #ccc;
            border-radius: 5px;
        }
        .form-group {
            margin-bottom: 15px;
        }
        label {
            display: block;
            margin-bottom: 5px;
            font-weight: bold;
        }
        input[type="text"], input[type="email"] {
            width: 100%;
            padding: 8px;
            border: 1px solid #ddd;
            border-radius: 4px;
        }
        .btn {
            background-color: #007bff;
            color: white;
            padding: 10px 15px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            width: 100%;
        }
        .btn:hover {
            background-color: #0056b3;
```

```
        }
        .messages {
            margin-bottom: 15px;
            color: red;
        }
    </style>
</h:head>

<h:body>
    <div class="container">
        <h2>Acesso ao Sistema</h2>

        <h:form id="loginForm">
            <h:messages globalOnly="true" styleClass="messages"/>

            <div class="form-group">
                <label for="nome">Nome:</label>
                <h:inputText id="nome" value="#{sessaoBean.nomeLogin}"
                             required="true" requiredMessage="Nome é obrigatório"/>
            </div>

            <div class="form-group">
                <label for="email">Email:</label>
                <h:inputText id="email" value="#{sessaoBean.emailLogin}"
                             required="true" requiredMessage="Email é obrigatório"/>
            </div>

            <h:commandButton value="Entrar" action="#{sessaoBean.login}"
                             styleClass="btn"/>
        </h:form>
    </div>
</h:body>
</html>
```

dashboard.xhtml

xhtml

```
<?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="jakarta.faces.html"
      xmlns:f="jakarta.faces.core"
      xmlns.ui="jakarta.faces.facelets">
```

```
<h:head>
    <title>Dashboard - Sistema de Sessão</title>
    <style type="text/css">
        body { font-family: Arial, sans-serif; margin: 0; padding: 20px; background
-color: #f4f4f4; }
        .container { max-width: 800px; margin: 0 auto; background: white; padding:
20px; border-radius: 5px; }
        .header { background: #007bff; color: white; padding: 10px; margin: -20px -
20px 20px -20px; border-radius: 5px 5px 0 0; }
        .info-box { background: #e9ecef; padding: 15px; border-radius: 4px; margin-
bottom: 20px; }
        .info-item { margin-bottom: 10px; }
        .label { font-weight: bold; color: #495057; }
        .atividades { list-style: none; padding: 0; }
        .atividades li { background: #f8f9fa; padding: 8px; margin-bottom: 5px; bor
der-left: 3px solid #007bff; }
        .form-group { margin-bottom: 15px; }
        .form-group input { width: 100%; padding: 8px; border: 1px solid #ddd; bord
er-radius: 4px; }
        .btn { background-color: #28a745; color: white; padding: 10px 15px; border:
none; border-radius: 4px; cursor: pointer; }
        .btn-danger { background-color: #dc3545; }
        .btn:hover { opacity: 0.9; }
        .nav { display: flex; justify-content: space-between; align-items: center;
}
    </style>
</h:head>

<h:body>
    <div class="container">
        <div class="header">
            <div class="nav">
                <h2>Sistema de Gerenciamento de Sessão</h2>
                <h:form>
                    <h:commandButton value="Sair" action="#{sessaoBean.logout}"
                                     styleClass="btn btn-danger"/>
                </h:form>
            </div>
        </div>

        <h:messages globalOnly="true" style="margin-bottom: 15px; color: green;"/>

        <!-- Informações do Usuário -->
        <div class="info-box">
            <h3>Informações do Usuário</h3>
            <div class="info-item">
                <span class="label">Nome:</span> #{sessaoBean.usuario.nome}
```

```
        </div>
        <div class="info-item">
            <span class="label">Email:</span> #{sessaoBean.usuario.email}
        </div>
        <div class="info-item">
            <span class="label">Login realizado em:</span>
            #{sessaoBean.usuario.loginTime}
        </div>
        <div class="info-item">
            <span class="label">Tempo de sessão:</span>
            #{sessaoBean.tempoSessao}
        </div>
    </div>

    <!-- Informações Técnicas da Sessão -->
    <div class="info-box">
        <h3>Informações da Sessão</h3>
        <div class="info-item">
            <span class="label">ID da Sessão:</span>
            #{sessaoInfoBean.sessionId}
        </div>
        <div class="info-item">
            <span class="label">Timeout (minutos):</span>
            #{sessaoInfoBean.sessionMaxInactiveInterval}
        </div>
    </div>

    <!-- Formulário de Atualização -->
    <h:form>
        <h3>Atualizar Perfil</h3>

        <div class="form-group">
            <label for="novoNome">Novo Nome:</label>
            <h:inputText id="novoNome" value="#{sessaoBean.novoNome}"
                         placeholder="Digite o novo nome"/>
        </div>

        <div class="form-group">
            <label for="novoEmail">Novo Email:</label>
            <h:inputText id="novoEmail" value="#{sessaoBean.novoEmail}"
                         placeholder="Digite o novo email"/>
        </div>

        <h:commandButton value="Atualizar Perfil"
                         action="#{sessaoBean.atualizarPerfil}"
                         styleClass="btn"/>
    </h:form>
```

```
        <!-- Histórico de Atividades -->
        <h3>Histórico de Atividades</h3>
        <ul class="atividades">
            <ui:repeat value="#{sessaoBean.usuario.atividades}" var="atividade">
                <li>#{atividade}</li>
            </ui:repeat>
        </ul>
    </div>
</h:body>
</html>
```

template.xhtml (Template Base)

xhtml

```
<?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="jakarta.faces.html"
      xmlns:f="jakarta.faces.core"
      xmlns.ui="jakarta.faces.facelets">

<h:head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title><ui:insert name="title">Sistema de Sessão</ui:insert></title>
    <h:outputStylesheet library="css" name="style.css" />
</h:head>

<h:body>
    <div class="wrapper">
        <header>
            <h1>Sistema de Gerenciamento de Sessão</h1>
        </header>

        <main>
            <ui:insert name="content">
                Conteúdo padrão
            </ui:insert>
        </main>

        <footer>
            <p>&copy; 2024 - Sistema de Sessão Jakarta EE</p>
        </footer>
    </div>
```

```
</h:body>
</html>
```

6. Configuração do JSF (faces-config.xml)

```
xml

<?xml version="1.0" encoding="UTF-8"?>
<faces-config xmlns="https://jakarta.ee/xml/ns/jakartaee"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
                                  https://jakarta.ee/xml/ns/jakartaee/web-facesconf
ig_4_0.xsd"
              version="4.0">

    <application>
        <locale-config>
            <default-locale>pt_BR</default-locale>
        </locale-config>

        <resource-bundle>
            <base-name>com.exemplo.messages</base-name>
            <var>msg</var>
        </resource-bundle>
    </application>

    <navigation-rule>
        <from-view-id>/login.xhtml</from-view-id>
        <navigation-case>
            <from-outcome>success</from-outcome>
            <to-view-id>/dashboard.xhtml</to-view-id>
            <redirect/>
        </navigation-case>
    </navigation-rule>
</faces-config>
```

7. Configuração do web.xml

```
xml

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
```

```
            https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
            version="6.0">

    <display-name>SessaoJSF</display-name>

    <!-- Configuração do JSF -->
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>jakarta.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.xhtml</url-pattern>
    </servlet-mapping>

    <!-- Configuração da Sessão -->
    <session-config>
        <session-timeout>30</session-timeout>
        <cookie-config>
            <http-only>true</http-only>
            <secure>false</secure>
        </cookie-config>
        <tracking-mode>COOKIE</tracking-mode>
    </session-config>

    <!-- Página de boas-vindas -->
    <welcome-file-list>
        <welcome-file>login.xhtml</welcome-file>
    </welcome-file-list>
</web-app>
```

8. Filter para Controle de Acesso

java

```
package com.exemplo.filters;

import com.exemplo.beans.SessaoBean;
import jakarta.faces.application.ResourceHandler;
import jakarta.inject.Inject;
import jakarta.servlet.*;
import jakarta.servlet.annotation.WebFilter;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
```

```
import java.io.IOException;

@WebFilter("/*")
public class AutenticacaoFilter implements Filter {

    @Inject
    private SessaoBean sessaoBean;

    @Override
    public void doFilter(ServletRequest request, ServletResponse response,
                         FilterChain chain) throws IOException, ServletException {

        HttpServletRequest req = (HttpServletRequest) request;
        HttpServletResponse res = (HttpServletResponse) response;

        String path = req.getRequestURI().substring(req.getContextPath().length());

        // Permitir acesso a recursos estáticos e página de login
        boolean isResource = path.startsWith("/jakarta.faces.resource")
                          || path.startsWith("/resources")
                          || path.startsWith("/javax.faces.resource");

        boolean isLoginPage = path.equals("/login.xhtml")
                           || path.equals("/");

        if (isResource || isLoginPage) {
            chain.doFilter(request, response);
            return;
        }

        // Verificar se usuário está logado
        if (sessaoBean != null && sessaoBean.getUsuario() != null
            && sessaoBean.getUsuario().isLoggedIn()) {
            chain.doFilter(request, response);
        } else {
            res.sendRedirect(req.getContextPath() + "/login.xhtml");
        }
    }

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

    @Override
    public void destroy() {}
}
```

Características importantes do JSF com SessionScoped:

1. **@SessionScoped**: Mantém o bean durante toda a sessão do usuário
2. **Serializable**: Beans de sessão devem ser serializáveis
3. **Injeção de dependências**: Usa CDI para injeção
4. **Navegação**: Retorno de strings nas actions para navegação
5. **Validação**: Integrada com JSF validation

Este exemplo demonstra o uso completo de objetos em sessão com XHTML/JSF, incluindo:

- Login com armazenamento de dados na sessão
- Atualização de dados na sessão
- Visualização de informações da sessão
- Logout com invalidação da sessão
- Filtro de autenticação
- Template com Facelets