Curso de Java

aula 17

Prof. Anderson Henrique

TECH
soluções em ti

JavaEE (Gerenciando Sessões e Cookies)

A melhor forma de entender como funcionam as sessões é pela descrição de um exemplo prático. Suponha que você esteja acessando um site de e-commerce e pretenda comprar alguns produtos.

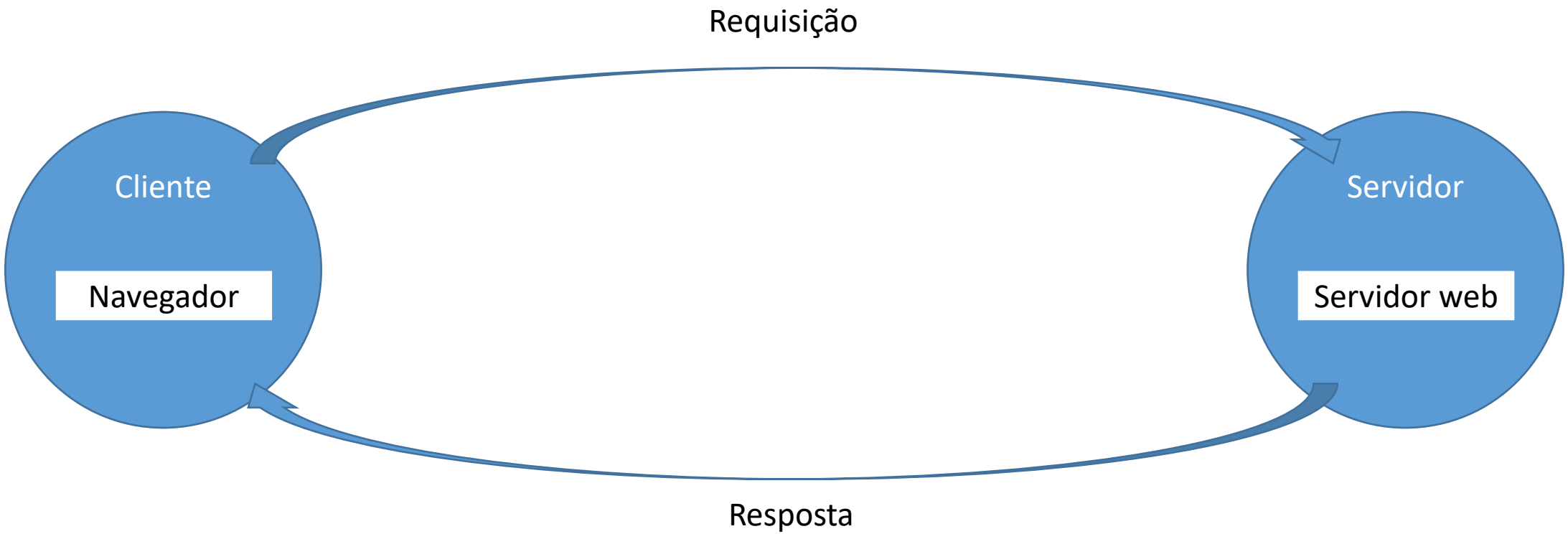
Você já se cadastrou para poder efetuar o login. Então, está escolhendo os produtos e adicionando-os ao carrinho de compras. Mas, por um motivo qualquer você precisa fechar o navegador para executar outra tarefa mais importante.

A pergunta que fica no ar é: será que vou perder todo o trabalho que tive para localizar os produtos e adicioná-los ao carrinho?

É nesse momento que entram as sessões. Elas permitem que seja mantido o estado da “conservação” entre o servidor e o cliente, isto é, a comunicação entre o servidor web/aplicação e o navegador do usuário.

Quando enviamos uma requisição ao servidor, uma sessão temporária é criada para que possa identificar o cliente que efetuou a solicitação, e assim saber a quem deve ser enviada a resposta. Após essa resposta ter sido enviada de volta ao cliente, a sessão temporária é encerrada.

O processo deve se repetir no caso de outra solicitação ser efetuada, veja a ilustração a seguir:


Requisição
Cliente
Navegador
Servidor
Servidor web
Resposta

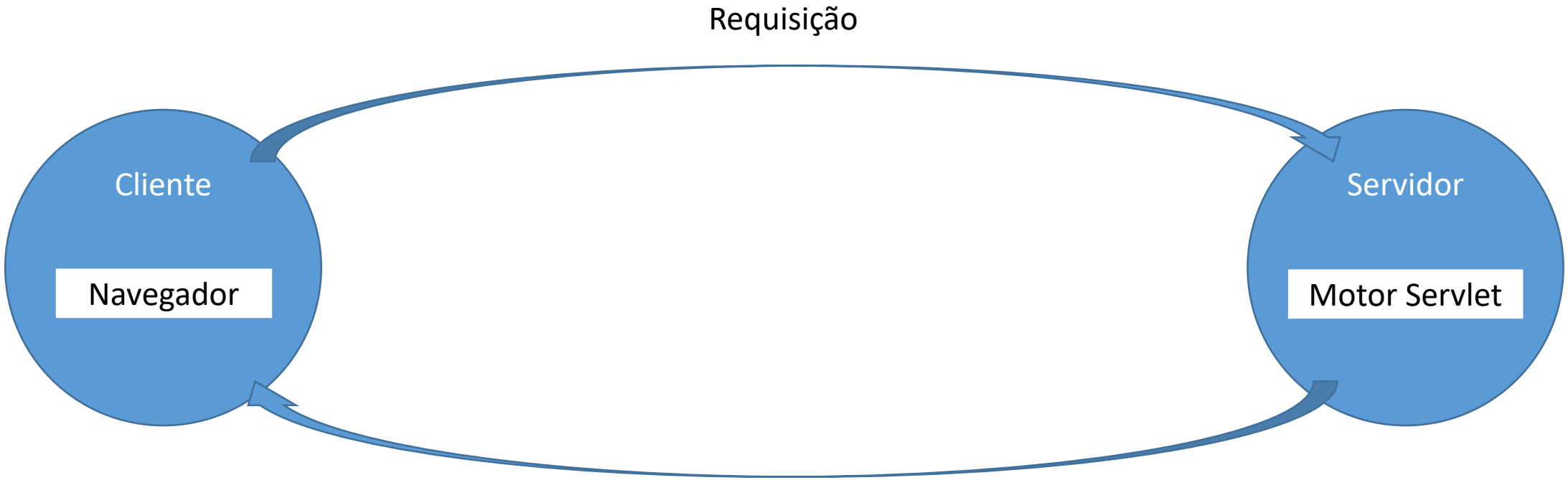
Para páginas estáticas, isso não representa um problema, mas em aplicações web, que acessam banco de dados e geram páginas de forma dinâmica, com um alto grau de interação com o usuário, essa característica de temporariedade não é algo com o qual se possa conviver.

É necessária uma maneira de o Servlet registrar essa sessão criada para conservação com o cliente e mantê-la ativa até que a aplicação encerre ou que um determinado tempo limite para expiração seja alcançado.

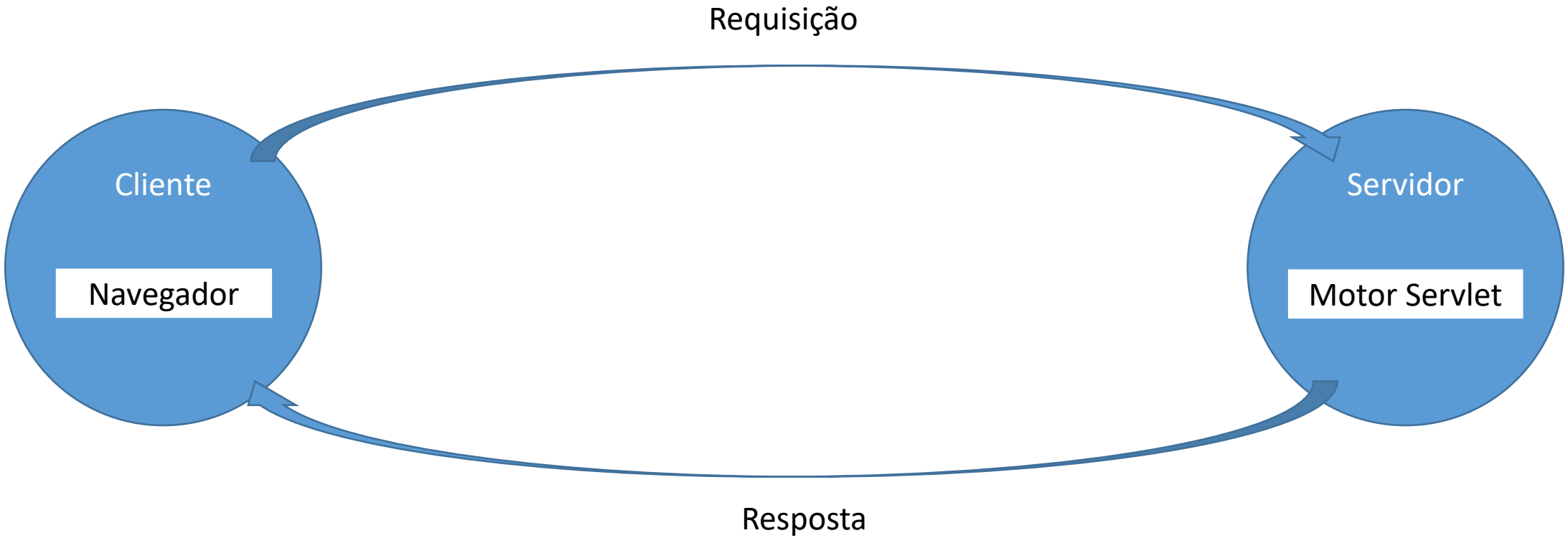
O navegador (cliente) lança uma requisição de página JSP ou Servlet ao servidor web, que é repassado ao motor Servlet (o Container). Este, por sua vez, verifica se já existe um identificador de sessão, e, em caso negativo, cria um, junto com um objeto de sessão para armazenamento de dados.

A partir de então, esse identificador de sessão é utilizado em cada requisição para criar uma relação entre o cliente e o servidor.

Primeira requisição



Requisições subsequentes
SessionID=EC64B2D5



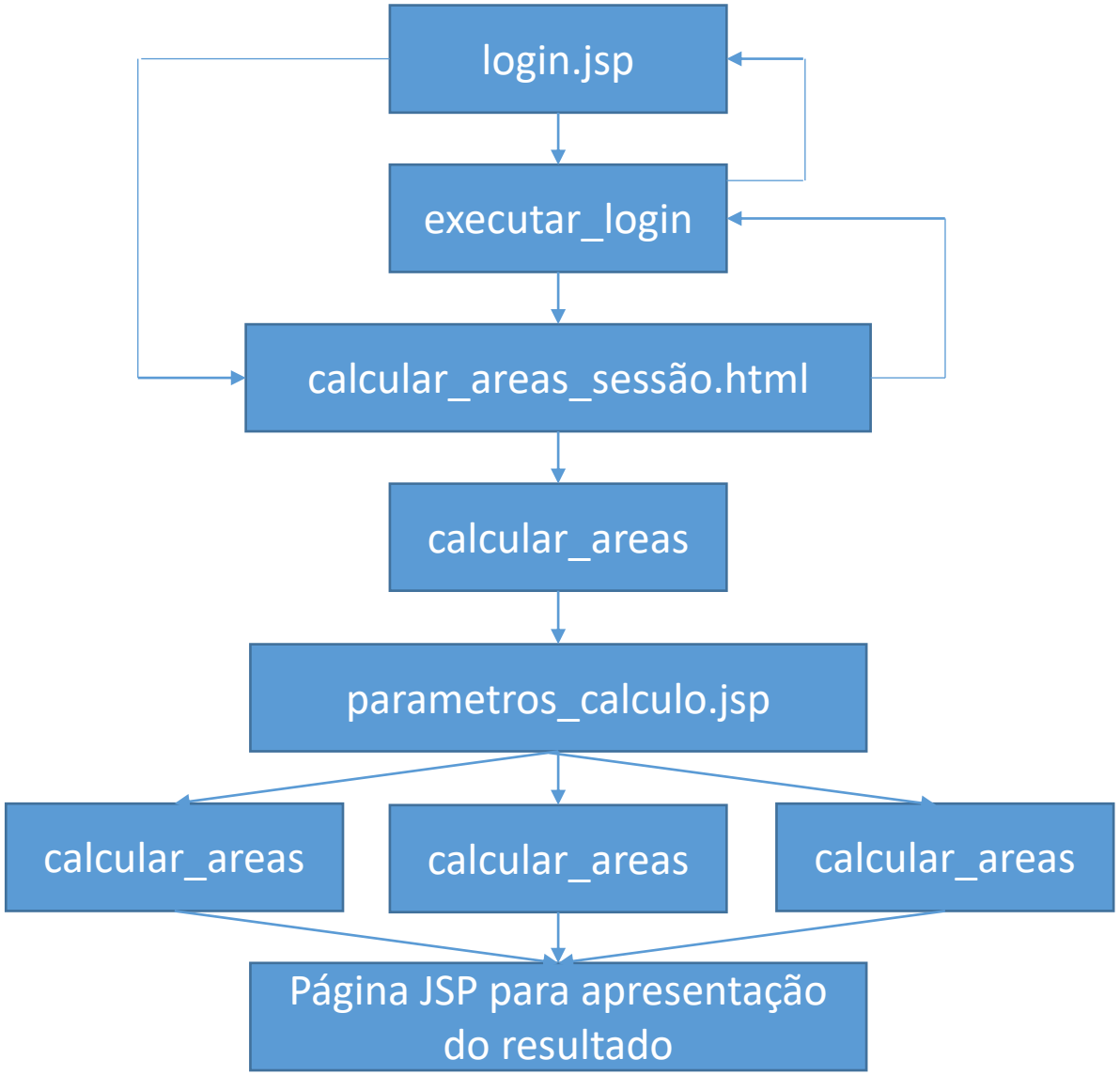
O armazenamento do identificador de sessão é feito pelo Servlet por meio de cookies, mas, se o navegador estiver configurado para não permitir esse recurso, não será possível registrar essa sessão.

Para contornar esse problema, é possível utilizar uma alternativa, denominada reescrita de URL, na qual o identificador de sessão é incluído no endereço URL da página. Vimos uma ilustração do processo envolvido na comunicação com gerenciamento de sessão.

JavaEE (Exemplo prático sessões)

A partir do exemplo prático que criamos (Cálculo de Áreas), vamos desenvolver uma nova versão que possui uma tela de login e que gerencia sessões.

A estrutura desse novo exemplo é apresentada no gráfico a seguir:



Tudo começa pela tela de login, na qual o usuário entra com a sua identificação e senha de acesso. Essa tarefa é executada pela página JSP denominada **login.jsp** que monta o formulário de entrada de dados.

Essa página chama um Servlet denominado **executar_login** para fazer a verificação de usuário e senha de acesso. Se ambos estiverem corretos, o Servlet **pagina_calculos** é invocado. No caso de um deles estar errado, volta-se à tela de login.

O servlet cria a sessão para o usuário que efetuou o login e, então, abre a página **calcular_areas_sessao.html**.

Essa página é, na verdade, um formulário que permite a seleção da figura geométrica cuja área se deseja calcular. Para isso, a ação executada pelo formulário é um Servlet **calcularAreas**. Essa página permite ainda que o usuário saia da aplicação.

Nesse último caso, o Servlet **executar_login** é chamado novamente para encerrar a sessão, conforme veremos mais à frente.

O Servlet **calcularAreas** recupera o tipo de figura selecionado no formulário **calcular_areas_sessao.html** e, então, chama a página JSP denominada **parametros_calculo.jsp**, que executa e monta o formulário para entrada de dados dos parâmetros da figura geométrica, de acordo com a opção escolhida pelo usuário.

Essa mesma página executa um Servlet adequado à seleção da figura geométrica do usuário. Cada um dos Servlets calcula a área da respectiva figura geométrica. O resultado é apresentado pela página **resultado_area.jsp**.

Vamos ao código da página login.jsp, mostrado a seguir:

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Login</title>
    </head>
    <body>
        <%@page import="javax.servlet.http.*"%>
        <%
            HttpSession sessao = request.getSession();
            if(sessao.getAttribute("usuario_logado") == null){
                sessao.setAttribute("usuario_logado", "false");
            }

            if(sessao.isNew() || sessao.getAttribute("usuario_logado").equals("false")){
                out.println("<form action='executar_login' method='post'>");
                out.println("Usuário: <input type='text' name='usuario'> <br><br>");
                out.println("Senha: <input type='password' name='senha'> <br><br>");
                out.println("<input type='submit' value='Entrar'>");
                out.println("</form>");
            }else{
                out.println("<h2>Olá "+session.getAttribute("nome_usuario")+"!</h2>");
                out.println("<a href='calcular_areas_sessao.html'>Calcular Áreas</a>");
            }
        %>
    </body>
</html>
```

Como esse código faz uso da classe **HttpSession** para poder tratar as sessões, é necessário referenciar o pacote **javax.servlet.http**. Primeiramente, o código instancia um objeto denominado **sessao** e atribui a ele o identificador de sessão aberta, caso haja alguma.

Note que para isso, utilizamos o método **getSession( )** do objeto implícito **request**, que permite recuperar a sessão atualmente associada à requisição recebida ou então criar uma nova sessão se nenhuma existir.

Este último caso ocorre no primeiro acesso à página.

De posse desse objeto de sessão, podemos recuperar ou definir o valor de um atributo de sessão, operação executada logo em seguida por uma estrutura condicional **if**. O código verifica se já existe um atributo denominado **usuario_logado**. Se não houver, ele o cria, atribuindo-lhe a cadeia de caracteres “false”.

Seguindo com o processo, é efetuada outra verificação para determinar se a sessão é nova, utilizamos o método **isNew( )**. Em qualquer desses casos, um formulário é montado para que o usuário digite sua identificação e senha de acesso

No caso de o usuário já ter acessado a página e efetuado o login anteriormente, uma mensagem de boas vindas é apresentada e um link para se acessar o menu de opções de figuras geométricas é mostrado.

Caso o usuário esteja acessando a página pela primeira vez, o Servlet **executar_login** é invocado após ele digitar sua identificação e senha de acesso. O código da classe Java que implementa esse Servlet é apresentado a seguir:

```
public class executar_login extends HttpServlet {

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

            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Cálculo de áreas de figuras geométricas</title>");
            out.println("</head>");
            out.println("<body>");

            if(request.getParameter("usuario").equals("jsp") &&
                    request.getParameter("senha").equals("1!2@3#")){
                sessao.setAttribute("usuario_logado", "true");
                sessao.setAttribute("nome_usuario", request.getParameter("usuario"));
                out.println("<h2>Bem Vindo(a) "+request.getParameter("usuario")+"</h2");
                out.println("<br><br>");
                out.println("<a href='calcular_areas_sessao.html'>Calcular áreas</a>");
            }else{
                out.println("<p>Usuário ou senha inválidos</p>");
                out.println("<a href='login.jsp'>Tentar novamente</a>");
            }
            out.println("</body>");
            out.println("</html>");
    }
```

TECH soluções em ti

Como o formulário criado pela página login.jsp utiliza o método **POST** para envio de dados, definimos o método **doPost( )** de forma que ele recupere a sessão vinculada à requisição e verifica se o nome do usuário e a senha de acesso estão corretos.

Nesse exemplo, em virtude da simplicidade, essas informações estão armazenadas no próprio código, mas, em um ambiente real, elas deveriam ser recuperadas de um banco de dados.

Se o nome do usuário (parâmetro usuário) for “jsp” e a senha (parâmetro senha) for “1!2@3#”, o atributo **usuario_logado** é ajustado com o valor **true** e uma tela de boas vindas é exibida. Essa tela contém um link para acesso ao formulário em que será selecionada a figura geométrica cuja área se deseja calcular.

Se houver erro no nome do usuário ou sua senha, uma mensagem informando a ocorrência e um link para voltar à tela de login são apresentados.

O formulário para seleção do tipo de figura geométrica é similar ao que já fizemos no exemplo anterior.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cálculo de Área de Figuras Geométricas</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
            initial-scale=1.0">
    </head>
    <body>
        <h2>Cálculo de Área de Figuras Geométricas</h2>
        <form method="get" action="CalculoAreas">
            <input type="radio" name="TipoFigura" value="1">Retângulo
            <input type="radio" name="TipoFigura" value="2">Circunferência
            <input type="radio" name="TipoFigura" value="3">Triângulo
            <input type="submit" value="Confirmar">
        </form>
        <br><br>
        <a href="executar_login">Sair</a>
    </body>
</html>
```

A diferença em relação ao código anterior é a existência de um link para sair da aplicação, o que força uma chamada ao Servlet **executar_login**. Neste ponto é necessária uma pequena modificação, implementando o método **doGet( )** no nosso Servlet.

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

            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Cálculo de áreas de figuras geométricas</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<p>Sessão encerrada...</p>");
            out.println("<p>Obrigado e até logo!</p>");
            out.println("</body>");
            out.println("</html>");

            HttpSession sessao = request.getSession();
            sessao.setAttribute("usuario_logado", null);
            sessao.invalidate();
    }
}
```

Esse código cria uma página HTML contendo uma mensagem de despedida e invoca o método **invalidate( )** para finalizar a sessão criada anteriormente.

Já o código **CalculoAreas** é o mesmo apresentado anteriormente. Os demais códigos de páginas JSP e classes Java Servlets também são os mesmos.

Vamos ao teste, compile e execute o arquivo login.jsp...

Note que após logado, se você acessar outro site e depois voltar para a página **login.jsp**, receberá como mensagem "**Olá jsp**", no qual aparece o nome do usuário que estiver atualmente logado.

JavaEE (Session ID e cookies)

Vimos anteriormente que, no processo de gerenciamento de sessões, um identificador de sessão (**Session ID**) é gerado automaticamente e retornado com a resposta ao cliente. Um arquivo texto é criado no cliente contendo essa informação, de forma que ela se torne persistente para futuro envio de volta ao servidor nas próximas requisições.

O conteúdo desse arquivo é muito simples, formado por uma cadeia de caracteres que representa o nome de um atributo e por um valor para esse atributo. Esse é o conhecido **cookie**.

Entretanto, os navegadores permitem que a gravação de cookies seja desabilitada. Nesse caso, não será possível gerar sessões? Existe uma solução para esse impasse, e ela se chama **reescrita de URL**.

Seu princípio de funcionamento é a anexação do Session ID ao final da URL enviada ao servidor de aplicação. Utilizar essa técnica é importante no desenvolvimento de uma aplicação web, pois devemos levar em consideração que pode haver usuários que não consiga executá-la por terem desativado o recurso de cookies no navegador.

Vamos criar um novo Servlet com o nome index, clique com o botão direito sobre o pacote e selecione a opção **Novo -> Servlet** e ative a opção de criação do descritor de implantação.

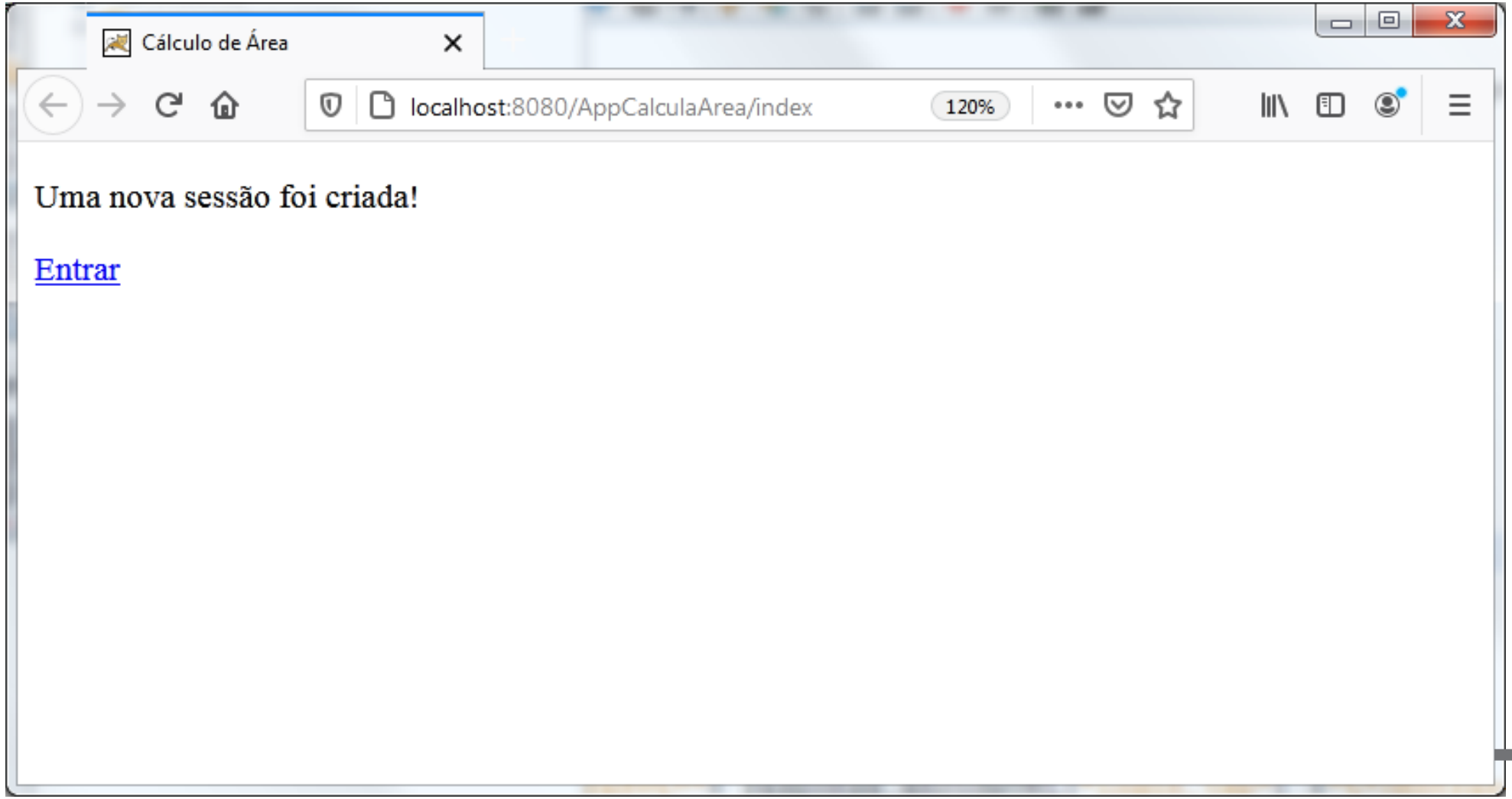
```java
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        PrintWriter out;
        out = response.getWriter();
        HttpSession sessao = request.getSession();

        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Cálculo de Área</title>");
        out.println("</head>");
        out.println("<body>");

        if(sessao.isNew())
            out.println("<p>Uma nova sessão foi criada!</p>");
        else
            out.println("<p>Olá. Você voltou à página!</p>");

        out.println("<a href=\""+ response.encodeURL("login.jsp") +"\">Entrar</a>");
        out.println("</body>");
        out.println("</html>");
    }
```

Compile e execute a classe Java index (Servlet), apresentará a mensagem “Uma nova sessão foi criada!” com um link abaixo “Entrar”. Note como a URL é formada.

Ela apresenta um complemento que consiste em um ponto e vírgula na palavra **jsessionid** seguida por um sinal de igual e em uma cadeia de caracteres representando a Session ID gerada no momento da primeira solicitação do cliente.

Se você tentar voltar para o index, apresentará a mensagem “Olá. Você voltou para a página!” com um link **Entrar** logo abaixo.

A novidade mais relevante é o uso do método **encodeURL**(), que faz parte da interface HttpServletResponse. Ele é responsável por criar um link para a página **login.jsp** com reescrita de URL, e enviar como resposta ao cliente. Ao executar esse arquivo:



Clique no link **Entrar** e você verá a tela de login logo em seguida.

JavaEE (Como destruir sessões)

Ao se criar uma sessão, um objeto que a identifica é criado, e isso ocupa recursos do computador. Quando o navegador é fechado pelo usuário, esse objeto de sessão é automaticamente destruído da memória.

No entanto, nosso código precisa ser capaz de destruir a sessão a qualquer momento, quando ela não for mais necessária. Veremos como isso é possível.

Podemos especificar um tempo limite, em minutos, para que uma sessão possa ficar inativa. Ao fim desse tempo, conhecido também como *timeout*, a sessão é automaticamente finalizada, e os recursos alocados pelo *Container* para o seu armazenamento, liberados.

Essa configuração pode ser feita de duas maneiras. A primeira por meio do próprio arquivo descritor de implantação **web.xml**. Entre os pares de tags **<session-timeout>** e **</session-timeout>** da seção **<session-config>** e **</session-config>**, permite a configuração desse tempo de inatividade (medido em minutos).

```
<session-config>
    <session-timeout>
        30
    </session-timeout>
</session-config>
```

Podemos configurar o *timeout* para uma sessão específica a partir do nosso código, utilizando o método **setMaxInactiveInterval( )**, da interface **HttpSession**. Devemos passar como parâmetro a esse método um valor numérico que represente o tempo em segundos, ou seja, precisamos multiplica-lo por 60 para ter o valor em minutos.

Veja a seguir a alteração feita na classe Java index para definir um timeout de 10 minutos de inatividade:

```
if(sessao.isNew()){
    sessao.setMaxInactiveInterval(10*60);
    out.println("<p>Uma nova sessão foi criada!</p>");
}else{
    out.println("<p>Olá. Você voltou a página!</p>");
}
```

JavaEE (Como trabalhar com cookies)

Já vimos o conceito de cookies, mas não trabalhamos com eles diretamente, apenas por meio de sessões. Nesse caso, o próprio Container se encarregou de gerenciá-los.

Então, vejamos como podemos criar nossos próprios cookies dentro das aplicações. Para podermos gerenciar manualmente os cookies, existe a classe **Cookie** do pacote **javax.servlet.http**.

A declaração e instanciação de um objeto cookie é feita por meio de uma linha de código similar à apresentada a seguir:

Cookie meucookie = new **Cookie**(“nome_cookie”, “valor_cookie”);

O primeiro parâmetro representa o nome do cookie e o segundo, o valor que deve ser atribuído a ele. O nome deve ser formado apenas por caracteres alfanuméricos, **sem espaço ou sinais gráficos**. O valor pode conter qualquer caractere da tabela ASCII.

Depois de ter instanciado o objeto cookie, podemos adicioná-lo ao objeto de resposta, que deve ser enviado ao navegador do usuário utilizando-se o método **addCookie( )**, presente na interface **HttpServletResponse**. Se, por outro lado, desejarmos recuperar o valor de um determinado cookie, será necessário utilizar o método **getCookies( )** da interface **HttpServletRequest**.

Vamos criar duas páginas JSP para entender como tudo funciona. A primeira será responsável pela criação de alguns cookies que será exibidos pela segunda página. Código JSP, **criar_cookies.jsp**, a seguir:

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Criação de Cookies</title>
    </head>
    <body>
        <%@page import="javax.servlet.http.*" %>
        <%
            Cookie cookieLivros;

            cookieLivros = new Cookie("Codigo7042","Arquitetura_de_Computadores");
            response.addCookie(cookieLivros);

            cookieLivros = new Cookie("Codigo1093","Algoritmos_e_Linguagens");
            response.addCookie(cookieLivros);

            cookieLivros = new Cookie("Codigo5411","Desenvolvimento_Web_Com_Java");
            response.addCookie(cookieLivros);
        %>
    </body>
</html>
```

A segunda página (**listar_cookies.jsp**) deve ter o seguinte código:

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Listar Cookies</title>
    </head>
    <body>
        <h2>Visualizando Cookies</h2>
        <%@page import="javax.servlet.http.* "%>
        <%
            Cookie meuscookies[] = request.getCookies();
            for(Cookie lista: meuscookies){
                out.println("Nome: "+lista.getName()+" | "+" Valor: "
                                                +lista.getValue()+"<br>");
            }
        %>
    </body>
</html>
```

É importante notar que o método **getCookies( )** retorna um vetor (array), uma vez que podemos ter vários cookies.

Prosseguiremos no próximo slide... Com JEE (JSP, Servlets, JavaBeans e Integração com o Banco de Dados)

Professor: Anderson Henrique

Programador nas Linguagens Java e PHP