

Sistema Servidor Cliente



Educação a Distância
Cruzeiro do Sul Educacional
Campus Virtual

Material Teórico



Programação de aplicações cliente/servidor na Web com JSP 2

Responsável pelo Conteúdo:

Prof. Esp. Marcio Funes

Revisão Técnica:

Prof. Ms. Luiz Carlos Reis

Revisão Textual:

Profa. Ms. Luciene Oliveira da Costa Santos



- Banco de Dados em JSP
- JavaBeans em JSP
- XML em JSP
- JavaScript em JSP
- Ajax em JSP



Objetivo de APRENDIZADO

Nesta quinta unidade, você irá estudar como podemos interagir a acessibilidade e o dinamismo que a tecnologia JavaServer Pages possui com as vantagens que outras tecnologias ligadas a Sistemas Cliente / Servidor podem fornecer. Dessa forma, podemos criar uma solução de Sistemas Distribuídos completa e atender a qualquer demanda que o mercado crescente pode exigir.

Atenção para a importância de realizar todas as atividades propostas dentro do prazo estabelecido para cada unidade, pois, assim, você evitará que o conteúdo se acumule e que você tenha problemas ao final do semestre.

Uma última recomendação: caso tenha problemas para acessar algum item da disciplina ou dúvidas com relação ao conteúdo, não deixe de entrar em contato com seu professor tutor através do botão mensagens

Contextualização

Você está em dia com a Receita Federal?

Parece uma pergunta estranha para um material ligado à tecnologia. Porém, o tema pode nos trazer um excelente exemplo do emprego da tecnologia para administração de declarações no Brasil.

Segundo o site da memória da Receita Federal (<https://goo.gl/ZLqz25>), a primeira disposição no Brasil sobre o imposto de renda ocorreu em outubro de 1843, em pleno início do segundo reinado. Foram estabelecidas quais seriam as porcentagens que deveriam ser pagas de acordo com os vencimentos dos cofres públicos recebidos por qualquer pessoa naquela época.

De 1843 para os tempos atuais, muitas coisas mudaram, por isso tente imaginar a dificuldade que o governo tem para administrar essa grande quantidade de informações que devem ser processadas, verificadas e acertadas quando necessário. Com a chegada da possibilidade de implantação de sistemas cliente / servidor, podemos ver o avanço que o Imposto de Renda obteve. Devido a essa tecnologia, a Receita consegue que cada contribuinte (cliente) possa enviar de seu computador as informações pertinentes às suas declarações diretamente para o servidor da Receita Federal.

Foi-se o tempo em que era necessário salvar suas declarações em um disquete e entregar diretamente na Receita Federal de sua cidade onde cada disquete seria lido e processado. A Receita criou o sistema chamado Receitanet que em sua ultima versão utiliza a tecnologia Java para estabelecer a conexão cliente / servidor com os servidores da receita.

Apesar de ainda serem muitas informações como no passado, a velocidade e as possibilidades que a receita tem de gestão e controle aumentaram muito desde o surgimento do Imposto de Renda no Brasil. Este é um assunto que está presente em nosso cotidiano e que nos mostra também o poder que existe na tecnologia quando bem empregada e administrada.

Fonte: <https://goo.gl/ZLqz25>

Banco de Dados em JSP



Muitos serviços disponíveis na Internet só são possíveis devido à existência de meios para criar e gerenciar banco de dados e aplicá-los a soluções Web. Várias são essas possibilidades devido a diversas tecnologias que permitem a conexão com banco de dados. Vejamos uma solução muito utilizada atualmente que consiste em utilizar os conceitos do JSP.

Estrutura

Para conceber uma solução, seja ela qual for, uma boa estrutura é fundamental. Para conexão com banco de dados e JSP, temos algumas possibilidades.

Para esse tópico, iremos utilizar a estrutura abaixo:

- **NetBeans IDE**: Pacote Java EE 7.2, 7.3, 7.4, 8.0
- **JDK**: Versão 7 ou 8
- **Servidor de banco de dados MySQL**: 5.x
- **Driver MySQL Connector/ J JDBC**: versão 5.x
- **GlassFish Server Open Source Edition**: 3.x ou 4.x

Vamos, agora, entender cada elemento que compõe essa estrutura:

NetBeans IDE

Primeiro, pensamos em qual será o ambiente de desenvolvimento que iremos utilizar, ou seja, escolhemos qual será a IDE (*Integrated Development Environment*) a ser utilizada. O escolhido será NetBeans IDE por ser altamente aceito no mercado.

Download: <https://netbeans.org/downloads/>

JDK – Java Development Kit

O JDK é o Kit de Desenvolvimento Java que contém um conjunto utilitário que permite criar sistemas para a plataforma Java; nele, podemos encontrar a máquina virtual Java (JVM), o compilador Java, APIs do Java e diversas ferramentas e utilitários necessários para programação em Java.

Download: <https://goo.gl/pwl5tf>

MySQL

O MySQL é um sistema de gerenciamento de banco de dados (SGBD) que usa a linguagem SQL como interface. Atualmente mantida pela Oracle, é amplamente utilizada pelo mundo todo tendo a Google e a Cisco como exemplo de clientes (TURNER, 2002).

Download: <http://www.mysql.com/downloads/>

Driver JDBC

O Driver JDBC (*Java Database Connectivity*) possibilita a conexão de uma aplicação Java a um banco de dados. Podemos escolher diferentes drivers jdbc, dependendo de qual banco de dados queremos conectar com nossa aplicação Java.

Download: <http://dev.mysql.com/downloads/connector/j/5.1.html>

GlassFish

Como estamos em um material que fala sobre conexões cliente/servidor era de se esperar que, para conceber soluções Web, utilizando Java, se faz necessário o uso de um servidor Web. O GlassFish é um servidor de aplicação para aplicações Java, ele é um software livre e atualmente mantido pela Sun Microsystems que faz parte da Oracle; por suportar o JDBC, é muito utilizado para conexão com banco de dados (TOOD, 2003).

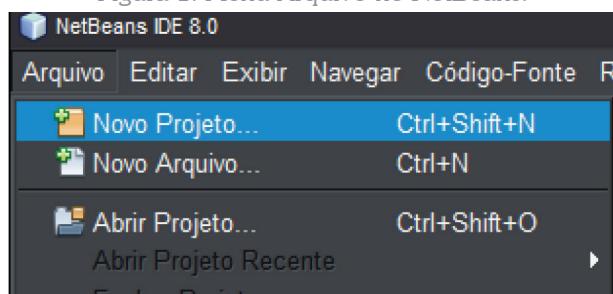
Download: <https://goo.gl/94xK5x>

Testando a conexão com o Banco de Dados

Como o foco é a conexão do banco de dados com JSP, vamos considerar que o banco de dados em MySQL já esteja pronto com algumas tabelas, mas sem os dados, os mesmos virão do que o usuário irá digitar na página JSP. Para testarmos se existe a conexão entre a IDE NetBeans e o MySQL, começamos criando um arquivo chamado por exemplo de “testeBanco.jsp” no NetBeans.

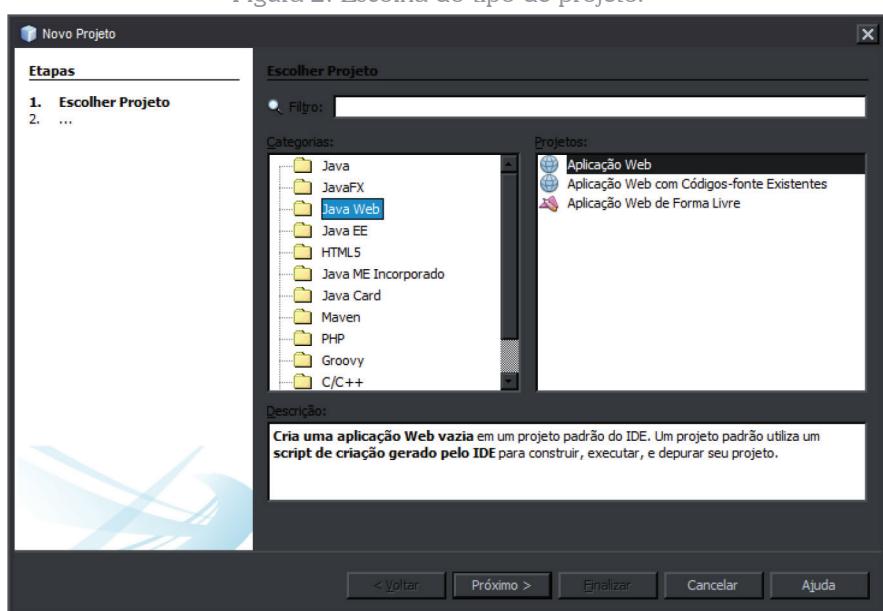
Clique em Arquivo e Novo Projeto e a seguinte tela será exibida:

Figura 1: Menu Arquivo no NetBeans.



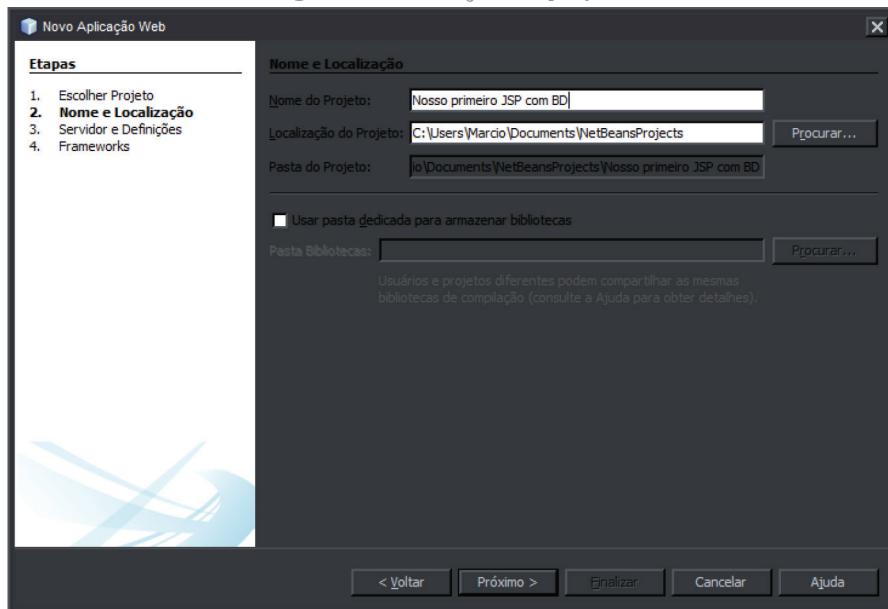
Escolhemos a opção Java Web e na lateral direita escolhemos Aplicações Web e clicamos no botão Próximo:

Figura 2: Escolha do tipo de projeto.



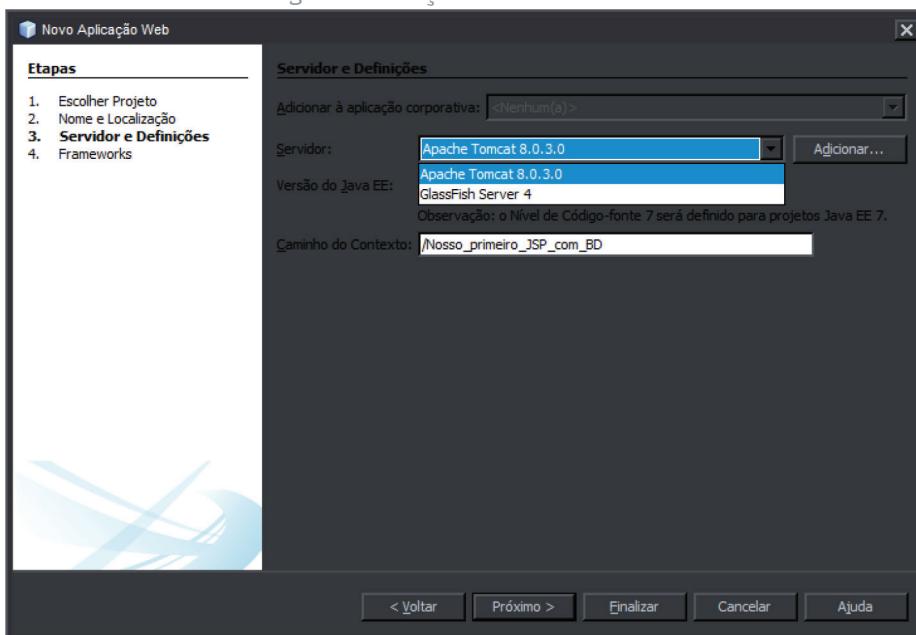
Na tela abaixo, iremos colocar o nome do nosso projeto e clicar em Próximo:

Figura 3: Nomeação do projeto.



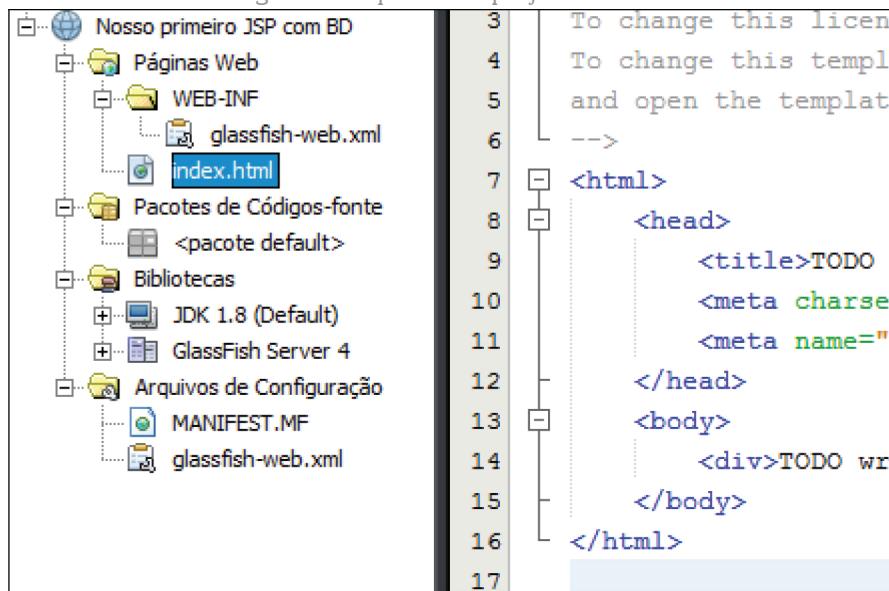
Agora, podemos escolher qual será o servidor de nosso projeto, veja que neste caso podemos escolher entre os dois servidores mais utilizados, o Apache Tomcat e o GlassFish. Vamos selecionar o GlassFish, pois ele foi escolhido no início do projeto. Após a escolha do servidor, você poderá clicar em Finalizar.

Figura 4: Seleção do servidor Web.



Após alguns segundos, o projeto será criado e o NetBeans deverá exibir a tela abaixo (dependendo da versão do NetBeans que você estiver utilizando, o código HTML exibido poderá ser diferente, não se preocupe).

Figura 5: Esquema do projeto e index.html.



The screenshot shows the NetBeans IDE interface. On the left, the project tree is displayed under the title "Nosso primeiro JSP com BD". It contains the following structure:

- Páginas Web
 - WEB-INF
 - glassfish-web.xml
 - index.html**
- Pacotes de Códigos-fonte
 - <pacote default>
- Bibliotecas
 - JDK 1.8 (Default)
 - GlassFish Server 4
- Arquivos de Configuração
 - MANIFEST.MF
 - glassfish-web.xml

On the right, the code editor shows the content of "index.html". The code is as follows:

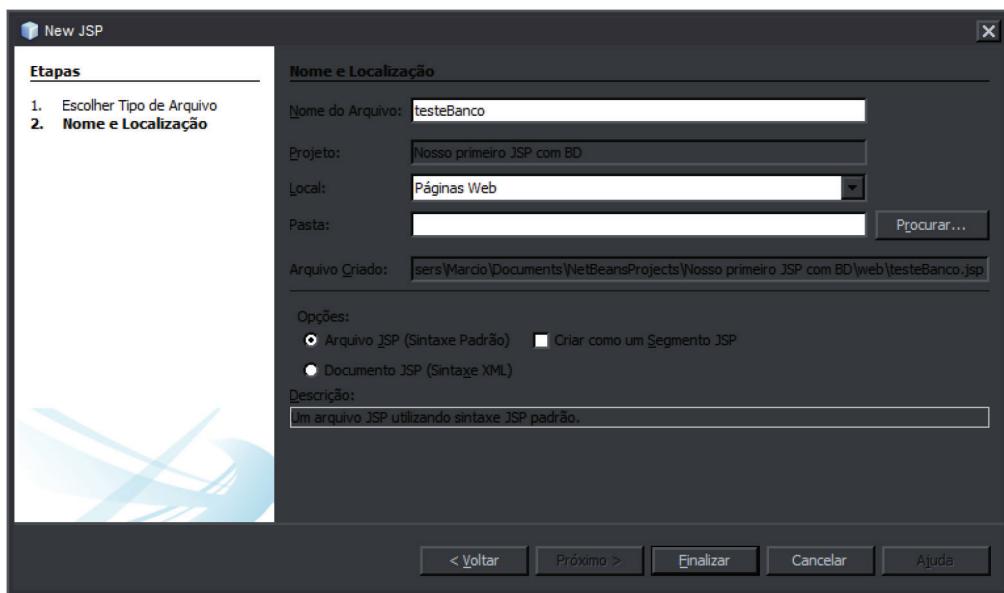
```

3   To change this licen
4   To change this templ
5   and open the templat
6   -->
7   <html>
8       <head>
9           <title>TODO
10          <meta charset="UTF-8">
11          <meta name="viewport" content="width=device-width, initial-scale=1.0">
12      </head>
13      <body>
14          <div>TODO write content here...
15      </body>
16  </html>
17

```

Dê um botão direito sobre o projeto, nosso primeiro JSP com BD, e vamos criar uma página JSP. Para testar nossa conexão com o Banco de Dados, escolha a opção Novo e depois JSP, a tela deverá ser como abaixo:

Figura 6: Criação da página JSP.



Escolha o nome da página JSP. No exemplo, iremos nomear como testeBanco. Clique em finalizar, após alguns segundos, ele irá exibir o conteúdo da página JSP criada que deverá ser semelhante à tela abaixo:

Figura 7: testeBanco.jsp.

```

1  <%--  
2   Document  : testeBanco  
3   Created on :  
4   Author     : Marcio  
5   --%>  
6  
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>Hello World!</h1>  
16  </body>  
17 </html>  
18
  
```

Perceba que agora podemos utilizar o `<% ... %>` característico do JSP em nossa página. Iremos agora testar a conexão:

Figura 8: Código de conexão com o Banco de Dados.

```

1  <%--  
2   Document  : testeBanco  
3   Created on :  
4   Author     : Marcio  
5   --%>  
6   <html>  
7   <%@ page contentType="text/html" language="java" import="java.sql.*"%>  
8   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  
9   <head>  
10  <title>Teste de Conexão</title>  
11  </head>  
12  <body>  
13  <%  
14  Connection con;  
15  Statement stm;  
16  
17  try {  
18      Class.forName("org.gjt.mm.mysql.Driver");  
19      con = DriverManager.getConnection("jdbc:mysql://localhost:3306/teste","root","root");  
20      stm = con.createStatement();  
21      out.println("Conexão efetuada com sucesso");  
22  
23  } catch (Exception e) {  
24      out.println("Não foi possível conectar ao banco" + e.getMessage());  
25  }  
26  %>  
27  
28  </body>  
29 </html>
  
```

Veja no código utilizado na Figura 8, temos o `<%@ page ... import="java.sql.*"%>` que diz ao JSP para realizar a importação necessária para conexão com o Banco de dados. Também podemos perceber no código selecionado em verde-claro como é feito o teste de conexão, temos um Try que recebe na variável “con” o DriverManager entre parênteses, há o endereço de URL que iremos utilizar para nosso Banco (localhost:3306/teste) e ainda colocamos o nome de usuário e senha (“root”, “root”) para que nosso projeto tenha acesso ao banco de dados livremente.

Caso nossa conexão com o banco falhe, temos implementado após o Try o Catch que trata a exceção do teste; caso o teste não seja bem sucedido, o Catch entra em ação e exibe a mensagem “Não foi possível conectar ao banco”.

Este teste de conexão é apenas o primeiro passo para criar aplicações Web que envolvam Banco de Dados. Agora, podemos implementar consultar, inserções e edições de informações que estão em seu banco de dados.



Acesse:

https://netbeans.org/kb/docs/web/mysql-webapp_pt_BR.html#welcomePage

Nesse tutorial disponibilizado pela própria NetBeans em português, podemos encontrar um exemplo prático passo a passo para criação de uma aplicação JSP com banco de dados. Utilize os conhecimentos acima adquiridos e mãos à obra!

JavaBeans em JSP



Iremos agora conhecer mais um integrante da tecnologia Java presente em muitas soluções Web e que permite ganhar maior rapidez no desenvolvimento de soluções, iremos conhecer o JavaBeans.

O JavaBeans, conhecido às vezes apenas por Beans, é um modelo de componentes de software reutilizável de Java. Para que fiquem claros seus conceitos, iremos tomar como base um exemplo que pode ser encontrado no livro *Java: Como Programar*, no capítulo 25, escrito por Deitel e Deitel (2010), pois nele está descrita a seguinte ideia: imagine que estamos construindo uma solução Web que envolva animação e algumas pessoas estão envolvidas no projeto, imagine que para conceber tal solução precisamos criar dois métodos o startAnimation e o stopAnimation, o primeiro inicia a animação e o segundo a interrompe. Queremos agora associar um botão ao startAnimation e outro botão ao stopAnimation, controlando assim a nossa animação. Com o JavaBeans, a IDE de desenvolvimento faz todo o trabalho de associar um evento de pressionar o botão com o método que desejamos chamar.

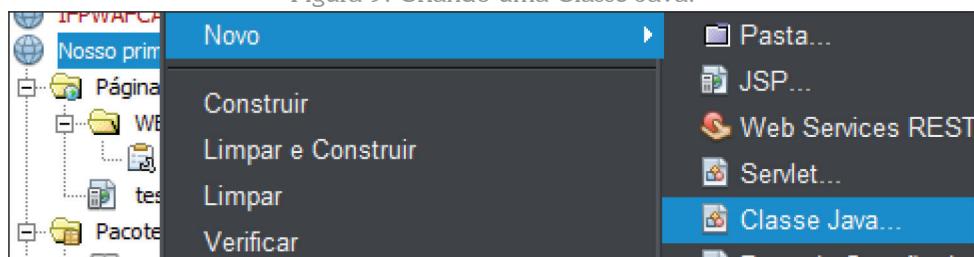
Dessa forma, um programador poderá desenvolver o método startAnimation sem ter o conhecimento do que outro programador está desenvolvendo no método stopAnimation. Quando ambos desenvolverem seus códigos, poderemos então pedir que a IDE associe um botão para cada método, e mais, quando criarmos outra solução que também utilize o startAnimation ou stopAnimation não precisaremos programar esta solução novamente, basta “chamar” o arquivo JavaBeans que contém o startAnimation ou stopAnimation e pedir que associe também a algum botão deste novo projeto, assim vários arquivos JavaBeans poderão ser chamados de projetos diferentes.

Exemplo de JavaBeans

Dando sequência ao primeiro tópico em que conectamos o banco de dados, a nossa IDE, iremos criar um JavaBeans que faça a mesma conexão, porém, teremos a diferença de não implementar a conexão diretamente na página JSP como fizemos anteriormente, pois, agora, quando precisarmos fazer a conexão com o banco de dados que iremos utilizar, basta “chamar” esse JavaBeans e a conexão será feita. Assim, não repetimos os códigos necessários para conectar ao banco várias vezes. Outra vantagem de utilizar o JavaBeans, no caso de conexão com o banco, é a de que, se, por acaso, alterarmos a senha de nosso banco, basta alterar uma vez no JavaBeans e, assim, todos os projetos que o utilizam irão usufruir desta mudança. Quando não utilizamos JavaBeans, teremos que procurar os trechos de código que utilizamos várias vezes para conectar ao banco e alterar uma a nova senha.

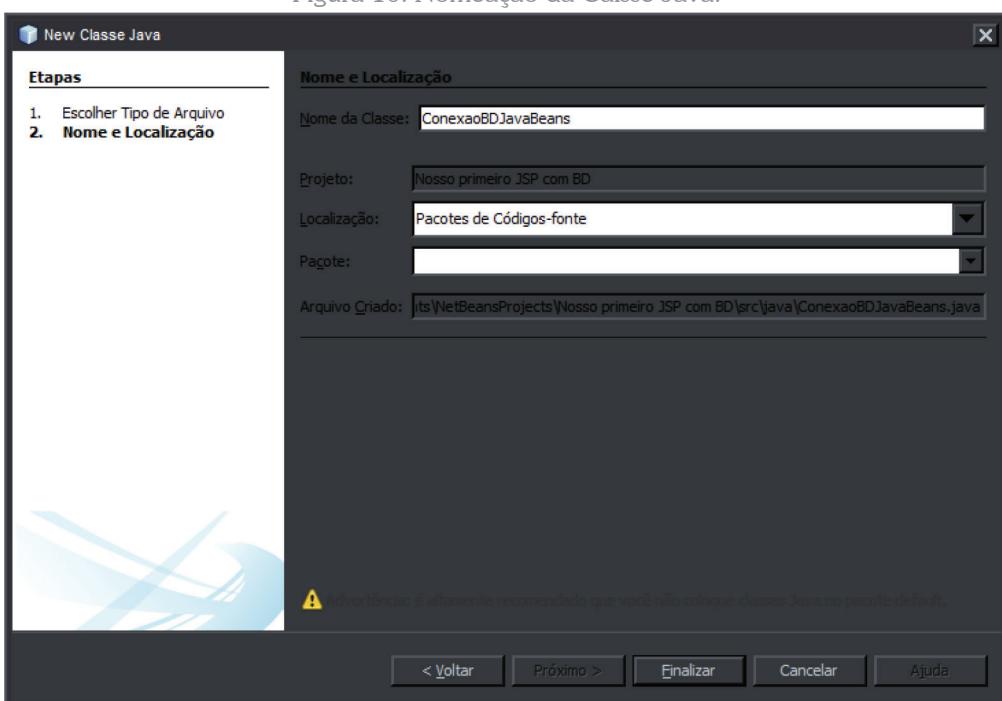
Primeiramente, clique com o botão direito sobre o projeto criado no primeiro tópico e escolha a opção Nova e depois Classe Java.

Figura 9: Criando uma Classe Java.



Na tela que segue, digite o nome dessa nova classe que será nosso JavaBeans, no exemplo abaixo, escolhi o nome ConexaoBDJavaBeans, depois clique em Finalizar.

Figura 10: Nomeação da Calsse Java.



Agora que temos uma nova classe criada, vamos implementar o código abaixo:

Figura 11: Classe Conexão.

```
1  /**
2  * @author Marcio
3  */
4
5 package conexao;
6 import java.sql.*;
7
8 public class ConexaoBDJavaBeans {
9     public Connection con;
10    public Statement stm;
11    private String situacao = "";
12
13    public ConexaoBDJavaBeans() {
14        try {
15            Class.forName("org.gjt.mysql.Driver");
16            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/teste","root","root");
17            stm = con.createStatement();
18            situacao = "Conexão realizada com sucesso";
19        }
20        catch(Exception e){
21            situacao = "não foi possível realizar a conexão" + e.getMessage();
22        }
23    }
24    public String getSituacao(){
25        return situacao;
26    }
27}
```

Veja no código que temos o `package` e o “`import`” característico de uma classe Java e depois o “`public class ConexaoBDJavaBeans`” que contém a conexão com o banco de dados. Foi criada uma variável chamada situação do tipo String para que saibamos qual é a situação de conexão com o banco.

Programamos o Try em que trazemos a utilização do driver de conexão ao banco e também dizemos o endereço, ao qual está o banco (você poderá utilizar o IP ao qual o banco está caso não estiver utilizando o localhost IP/Nome do banco) logo após o usuário e senha do banco.

No “`Catch`” do código tratamos a exceção, caso não aconteça a conexão com o banco, ele irá atribuir a variável “`situacao`” à frase “não foi possível realizar a conexão”. E por fim retornamos o valor da variável situação para sabermos se seu valor foi atribuído pelo “`Try`” ou “`Catch`”.

Este é apenas um simples exemplo de como podemos utilizar o JavaBeans ao longo do desenvolvimento de uma solução Web. Além de permitir mais rapidez e reaproveitamento de código, possibilita que várias pessoas trabalhem em um mesmo projeto sem necessariamente que cada programador só possa realizar seu trabalho após outro a concluir. Essa característica tem muito necessária no mercado atual onde uma equipe atua em um mesmo projeto, principalmente se ele tiver grandes proporções.

XML em JSP

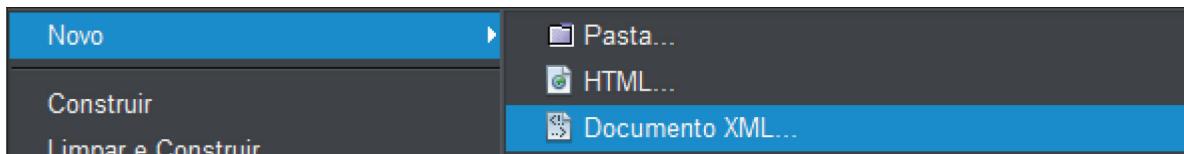


XML do inglês eXtensible Markup Language ou algo como Linguagem de Marcação Extensível é uma linguagem de programação que permite organizar e estruturar dados de forma fácil e simples para o programador. Ele pode ser utilizado para fornecer dados a diversas aplicações e em diferentes computadores, da mesma forma que o HTML consegue organizar de forma hierárquica a estrutura de uma página de internet o XML consegue organizar de forma hierárquica informação como em um banco de dados (KOCHMER, 2002).

Vejamos um exemplo de criação de um arquivo XML utilizando a IDE NetBeans:

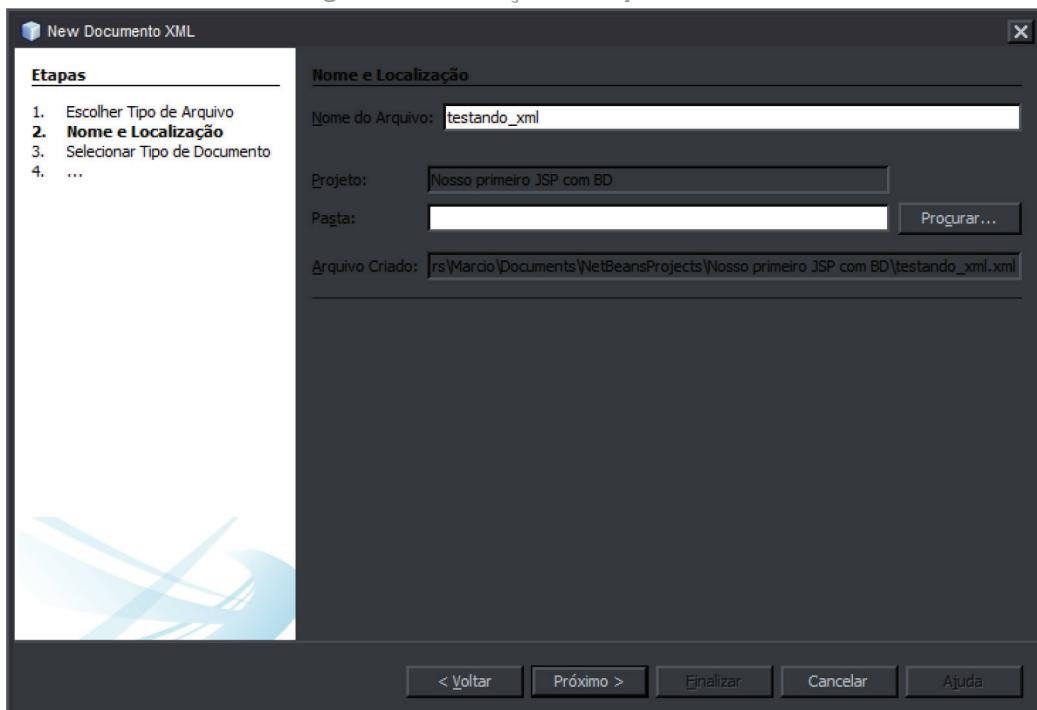
Dê um botão direito do mouse sobre o seu projeto e depois clique em Novo, e depois Documento XML.

Figura 12: Criação de um arquivo XML.



Na tela a seguir, escreva o nome do arquivo XML que desejar:

Figura 13: Nomeação do arquivo XML.



Na tela a seguir, selecione a opção Documento Formado Corretamente e depois Finalizar.

Vejamos um exemplo de XML criado pela própria W3C (http://www.w3schools.com/xml/xml_examples.asp) que mantém e atualiza a linguagem XML:

Figura 14: Exemplo de linguagem XML.

```

1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <breakfast_menu>
4      <food>
5          <name>Belgian Waffles</name>
6          <price>$5.95</price>
7          <description>
8              Two of our famous Belgian Waffles with plenty of real maple syrup
9          </description>
10         <calories>650</calories>
11     </food>
12 </breakfast_menu>
13

```

Vamos entender alguns elementos que compõem o código da Figura 14, perceba que utilizamos a tag `<breakfast_menu>` para iniciar nossa programação, por ser organizado de forma hierárquica a IDE sabe que todas as informações abaixo até fecharmos a tag `</breakfast_menu>` pertencem a “`breakfast_menu`” e assim podemos referenciá-las facilmente quando precisamos fornecer alguma informação deste menu a outra aplicação.

Veja que dentro da tag `<food>` organizamos as informações que desejamos armazenar como: `<name>` (nome), `<price>` (preço), `<description>` (descrição), `<calories>` (calorias), qualquer aplicação de desejarmos ler nesse arquivo XML e a informação que precisarmos.

Vejamos agora um exemplo de como podemos integrar a linguagem XML a JSP extraído do site [tutorialspoint.com](http://www.tutorialspoint.com/jsp/jsp_xml_data.htm); este e outros tutoriais fáceis e educativos você pode encontrar neste link: http://www.tutorialspoint.com/jsp/jsp_xml_data.htm

Primeiro, vamos criar nosso arquivo XML que contém os dados que serão lidos e alterados:

Figura 15: Código da página books.xml.

```

1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <books>
4      <book>
5          <name>Padam History</name>
6          <author>ZARA</author>
7          <price>100</price>
8      </book>
9      <book>
10         <name>Great Mistry</name>
11         <author>NUHA</author>
12         <price>2000</price>
13     </book>
14 </books>

```

Perceba no código acima que temos a tag `<book>` indicando o início da hierarquia que contém as informações: Nome, Autor e Preço de um livro.

Vamos agora visualizar o código da página JSP:

Figura 16: Código da página index.jsp.

```

1   <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
2   <%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>
3
4   <html>
5     <head>
6       <title>JSTL x:parse Tags</title>
7     </head>
8     <body>
9       <h3>Books Info:</h3>
10      <c:import var="bookInfo" url="http://localhost:8080/books.xml"/>
11
12      <x:parse xml="${bookInfo}" var="output"/>
13      <b>The title of the first book is</b>:
14      <x:out select="$output/books/book[1]/name" />
15      <br>
16      <b>The price of the second book</b>:
17      <x:out select="$output/books/book[2]/price" />
18
19    </body>
20  </html>
21

```

Na página JSP da Figura 16, temos o uso de duas Tags Libs definindo o “c” e o “x” como prefixo das definições da java.sun que serão utilizado no <body>.

Perceba que, após o título “Book Info:”, temos o “c:import” definindo uma variável “bookinfo” que servirá de referência ao arquivo book.xml que criamos inicialmente, perceba que a url dessa mesma Tag faz referência ao localhost:8080/book.xml; caso esteja utilizando outro endereço ou outra porta, você deve alterar neste local.

Definido a referência “bookInfo”, agora podemos ler o arquivo XML e trazer apenas as informações que necessitamos. Veja que na tag “x:parse” definimos o XML como sendo a referência bookInfo e criamos uma variável de saída chamada “output”. Agora vamos à leitura do XML.

Na tag “x:out”, selecionamos exatamente a informação no arquivo XML que queremos, “\$output/books/book[1]/name”, através do “book[1], selecionamos a primeira tag <book> que o arquivo XML possui e trazemos dela a informação contida dentro da tag <Name>, o mesmo ocorre quando queremos extrair apenas o preço do segundo livro “\$output/books/book[2]/price”, a saída deste código será esta:

BOOKS INFO:

- The title of the first book is: Padam History
- The price of the second book: 2000

Devido à facilidade que encontramos em criar um arquivo XML, pois é muito similar a forma que o HTML estrutura suas tags, podemos entender por que muitos programadores o utilizam em suas aplicações. Apesar de fazer o papel de banco de dados, não é aconselhável quando temos uma grande quantidade de dados pois os SGBS (Sistema Gerenciador de Banco de Dados) atuais podem fornecer melhor a forma de armazenar e buscar informações.

JavaScript em JSP



Outra linguagem muito utilizada como integração com o JSP é o JavaScript. Ele permite a inserção de diversas funções diretamente em nossas páginas JSP ou HTML ou, se preferir, a criação de um arquivo separado que contenha funções e recursos que podem ser acessados diversas vezes (KURNIAWAN, 2002).

Vejamos uma aplicação bem simples de como podemos criar uma página de Login utilizada amplamente em diversas soluções Web. Criaremos páginas para essa solução, criaremos uma página chamada index.jsp, criaremos também uma página chamada principal.html e por último nosso arquivo em JavaScript, scrip.jp.

Index.jsp

Crie um novo projeto no NetBeans do tipo Java Web, Aplicação Web, coloque o nome que desejar; após a criação, crie uma nova página chamada índice.jsp, clicando com o botão direito sobre seu projeto e depois Novo, JSP.

Nossa tela de Login será simples esteticamente, pois o foco é entender a integração de JSP com JavaScript, porém, será base para que construa seus projetos mais elaborados. Nossa página terá esta aparência:

Figura 17: Tela de Login exibida ao usuário (índex.jsp).

Seja Bem Vindo	
Nome:	<input type="text"/>
Senha:	<input type="password"/>
Entrar	

Agora, vejamos o código da página index.jsp que foi utilizado para criar a tela acima:

Figura 18: Código da página índice.jsp.

```

<html>
    <head>
        <script LANGUAGE="JavaScript" src="Login.js"></script>
    <body>
        <center>
            <form name=login>
                <table width=225 border=1 cellpadding=3>
                    <tr><td colspan=2><center>
                        <p><b>Seja Bem Vindo</b></p>
                    </center></td></tr>
                    <tr><td>
                        <p align="right"><font face="Verdana" style="font-size: 8pt">
                            <b>Nome</b></font></p>
                        <input type="text" name=username size="20">
                    </td><td>
                        <p align="right"><font style="font-size: 8pt" face="Verdana">
                            <b>Senha</b></font></p>
                        <input type="password" name=password size="20">
                    </td></tr>
                    <tr><td colspan=2 align=center>
                        <input type="button" value="Entrar" onClick="Login()">
                    </td>
                </table>
            </form>
        </center>
    </body>
</html>

```

Vamos analisar o código acima, perceba que dentro da tag <head> temos a tag <script> e nela definimos a propriedade LANGUAGE como JavaScript e apontamos através da propriedade “src” o nome do arquivo que contém a programação em JavaScript que será necessária neste código. Neste caso, o arquivo chama-se Login.js.

Mais abaixo, temos a tag <form>, na qual definimos que o nome do formulário será “login”, esse nome será importante para que consigamos no arquivo Login.js saber onde encontrar os elementos desse formulário. Dentro desse formulário temos três elementos principais:

- **Uma caixa de texto Usuário:** <input type="text" name="username" size="20">.
- **Uma caixa de texto Senha:** <input type="text" name="password" size="20">.
- **Um botão Entrar:** <input type="button" value="Entrar" onClick="Login()">.

Perceba que na tag <input>, que é usada para criar o “button”, utilizamos um propriedade chamada “onClick”, que é necessária para dizer ao código que quando o usuário pressionar esse botão ele deverá chamar a função Login(). Por termos definido no início do código que o arquivo que contém as funções JavaScript estão no arquivo Login.js, ao usuário clicar neste botão, o código irá procurar a função Login() dentro do arquivo Login.js.

Login.js

Agora iremos criar o arquivo JavaScript. Dê um botão direito sobre o projeto criado, escolha a opção Novo e depois Arquivo JavaScript, veja o código abaixo:

Figura 19: Código da página script.js.

```

1  function Login() {
2
3      var username = document.login.username.value;
4      username = username.toLowerCase();
5      var password = document.login.password.value;
6      var done = 0;
7
8      password = password.toLowerCase();
9      if (username == "teste" && password == "12345") {
10         window.location = 'principal.html';
11         done = 1;
12     }
13     if (done == 0) {
14         alert("Login ou senha invalido");
15     }
16 }
```

Começamos definindo o nome da função através do “function Login()”, assim “ligamos” a ação do botão da página index.jsp com esse arquivo JavaScript; dentro da função, criamos uma variável chamada “username” que irá receber como valor o que o usuário digitou na página dentro do campo “username” criado no formulário da página index.jsp. Veja agora como atribuir valor à variável username.

document.login.username.value;

O comando document define que iremos fazer referência a uma página. O login que vem logo depois é o nome do formulário e o username é o nome da caixa de texto (<input type="text">) que está dentro do formulário login e, por fim, colocamos o “value” para dizer ao código que queremos o valor que foi digitado na página pelo usuário, o mesmo fazemos com o campo de texto senha.

Também criaremos uma variável chamada “done” para nosso controle de teste no IF. Veja que abaixo criamos dois IFs, no primeiro temos:

```
If(username == "teste" && senha == "12345"){

    window.location = 'Principal.html';

    done = 1;

}
```

Neste IF, definimos que, se a variável “username” for igual a “teste” e a variável senha for igual a “12345”, então executamos a linha window.location = ‘principal.html’; que tem a função de chamar a página principal.html sem abrir uma nova aba no navegador, assim criamos um teste simples de login. Veja que ainda dentro deste IF atribuímos a variável done com valor 1; dessa forma, impedimos que o Segundo IF que informa erro no login seja ativado.

Principal.html

Por último, crie uma simples página HTML, chamada principal.html, clicando com o botão direito em nosso projeto e depois Novo e HTML; coloque apenas um simples texto para que, quando você testar seu projeto, e no Login colocar a palavra “teste” e no campo senha “12345”, permitir que a página principal.html seja aberta na mesma página.

Este é um simples exemplo de utilização para que você entenda como o mecanismo de chamada de um arquivo JavaScript funciona dentro de uma página JSP. Agora é só realizar os teste e melhorar o projeto.

Ajax em JSP



Agora que vimos como podemos integrar a linguagem XML e a linguagem JavaScript em nossa solução em JSP, podemos entender melhor como funciona outra tecnologia Web que também é integrável a JSP, o AJAX.

AJAX é o acrônimo de Asynchronous Javascript and XML, ou seja, é uma tecnologia que utiliza o JavaScript e o XML para desenvolver aplicações Web que tenham funcionalidade similares a uma aplicação de desktop, ou seja, local (HANSEN, 2007).

Muitas vezes, necessitamos de soluções que se aplicam apenas a uma determinada parte de nossa página. Imagine que, ao cadastrar um usuário, em cada novo campo de texto que ele preenche toda a página, necessitasse ser carregada para validar aquele campo. Isso seria muito ruim.

Outro exemplo é quando estamos realizando uma busca no Google e antes mesmo de terminar a palavra o Google já nos traz sugestões daquilo que talvez você queira buscar. Nisso, temos a sensação de que estamos executando alguma função local, apenas em nossa máquina pela rapidez com que a sugestão nos é trazida. Eis o exemplo de chamada assíncrona, ou seja, o AJAX irá fazer chamadas ao servidor sem uma determinada sincronia, ela fará chamadas ao servidor apenas em certas ocasiões.



Para que você tenha mais conhecimento sobre o AJAX, acesse os tutoriais abaixo que, além de gratuitos, permitem o entendimento passo a passo de uma construção integrada entre AJAX e JSP:

<http://www.programming-free.com/2012/08/ajax-with-jsp-and-servlet-using-jquery.html>

<https://www.udemy.com/blog/jsp-ajax/>

Material Complementar

Como complemento desta unidade, sugerimos a leitura completa do capítulo 25 do livro:

DEITEL, H. M.; DEITEL, P. J. **Java**. Como Programar. 8 ed. São Paulo: Prentice Hall, 2010.

Sugerimos, também, a leitura completa do livro:

TODD, N.; SZOLKOWSKI, M. **Javaserver Pages**. O Guia do Desenvolvedor. Rio de Janeiro: Elsevier, 2003.

Boa leitura a todos!

Referências

DEITEL, H. M.; DEITEL, P. J. **Java**. Como Programar. 8 ed. São Paulo: Prentice Hall, 2010.

HANSEN, M. D. **Soa Using Java Web Services**. New Jersey: Prentice Hall, 2007.

KOCHMER, C. **Jsp And Xml**. Integrating Xml And Web Services In Your Jsp Application. Boston: Addison-Wesley, 2002.

KURNIAWAN, B. **Java Para a Web Com Servlets, Jsp e Ejb**. Rio de Janeiro: Ciência Moderna, 2002.

TODD, N.; SZOLKOWSKI, M. **Javaserver Pages**. O Guia do Desenvolvedor. Rio de Janeiro: Elsevier, 2003.

TURNER, J. **Mysql And Jsp Web Applications**. Usa: Sams, 2002.

Anotações





Educação a Distância
Cruzeiro do Sul Educacional
Campus Virtual

www.cruzeirodosulvirtual.com.br
Campus Liberdade
Rua Galvão Bueno, 868
CEP 01506-000
São Paulo SP Brasil
Tel: (55 11) 3385-3000

