Aula 01 – 30/mai.

Introdução a Java Web

Sistemas com clientes baseados em navegadores Internet (browsers) exibem muitas vantagens sobre as aplicações C/S (Cliente/Server) tradicionais, tais como o número virtualmente ilimitado de usuários, procedimentos de distribuição simplificada (instalação e configuração apenas no Servidor), operação em múltiplas plataformas e possibilidade de gerenciamento remoto. Mas tais vantagens requerem a geração dinâmica de conteúdo para web.

Atualmente podemos encontrar aplicações web em todos os lugares, em todas a empresas, de vários tipos diferentes como, por exemplo: blogs, fotoblogs, sites de vendas de todo e qualquer tipo de produtos, pedidos/reservas de passagens de ônibus, avião, hotel e até aplicativos de alto risco e desempenho como aplicações corporativas, bancarias, leilões etc. e etc.

Entrando na parte técnica da coisa, aplicativos web são por natureza APLICAÇÕES DISTRIBUÍDAS. Ou seja, são pedaços de um mesmo programa que de forma combinada executam em diferentes lugares em maquinas separadas denominados clientes e servidores, interconectados através de uma rede comum.

JSP é uma tecnologia JavaEE (java Enterprise Edition) destinada á construção de aplicações para geração de conteúdo dinâmico para web, tais como HTML,XML, e outros, oferecendo facilidades de desenvolvimento.

Atualmente as famílias de produtos disponíveis no atual JEE se resumem assim:

Web Application Technologies

JavaServer Faces 1.2

JavaServer Pages 2.1

JavaServer Pages Standard Tag Library

Java Servlet 2.5

Enterprise Application Technologies

Common Annotations for the Java Platform

Enterprise JavaBeans 3.0

J2EE Connector Architecture 1.5

JavaBeans Activation Framework (JAF) 1.1

JavaMail

Java Data Objects (JDO)

Java Message Service API

Java Persistence API

Java Transaction API (JTA)

Web Services Technologies

Implementing Enterprise Web Services

Java API for XML-Based Web Services (JAX-WS) 2.0

Java API for XML-Based RPC (JAX-RPC) 1.1

Java Architecture for XML Binding (JAXB) 2.0

SOAP with Attachments API for Java (SAAJ)

Streaming API for XML

Web Service Metadata for the Java Platform

Management and Security Technologies

J2EE Application Deployment

J2EE Management

Java Authorization Contract for Containers

Cada um destes pode possuir subdivisões e tecnologias internas, constituindo uma extensa lista de opções de produtos e ferramentas. Mas a boa notícia é que ninguém tem que saber tudo ou dificilmente vai ter que deter o conhecimento de tudo ou utilizar tudo ao mesmo tempo para a construção de uma aplicação web. Provavelmente serão usadas de duas a cinco tecnologias combinadas para construir uma grande, boa, robusta e confiável solução.

Em nossas aulas iremos aprender a construir páginas JSP para processar dados de

formulários HTML, também iremos abordar um pouco sobre Servlets e como construir páginas JSP utilizando eles. Iremos aprender a utilizar uma linguagem dinâmica chamada JavaScript, para validar e deixar nossas aplicações mais dinâmicas. Na segunda aula, iremos aprender sobre escopo de parâmetro, até onde eles são visíveis para as aplicações. Também iremos ver sobre Enterprise JavaBeans, TagLibs e Servlets. Na Terceira aula iremos ver como conectar nossas páginas JSP com o Banco de Dados MySQL através do JDBC e do padrão de projetos DAO. Nas outras aulas iremos construir uma mini aplicação utilizando as conhecimentos adquiridos ao longos de todos os módulos desse mini curso.

Para nossas aulas iremos utilizar o IDE Netbeans versão 6.9.1 e o tomcat 6.0.29.

Servidor Tomcat e IDE Netbeans– instalação

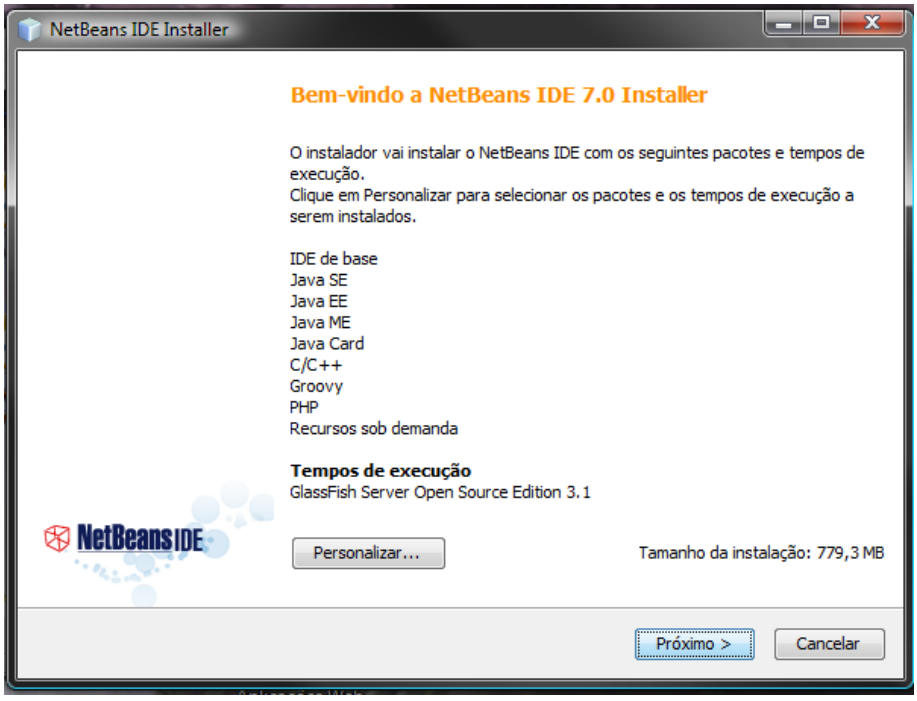
O JSP necessita rodar sobre um servidor, mais precisamente ele é instalado sobre um servidor e assim, a aplicação JSP pode ser aberta por qualquer navegador de internet.

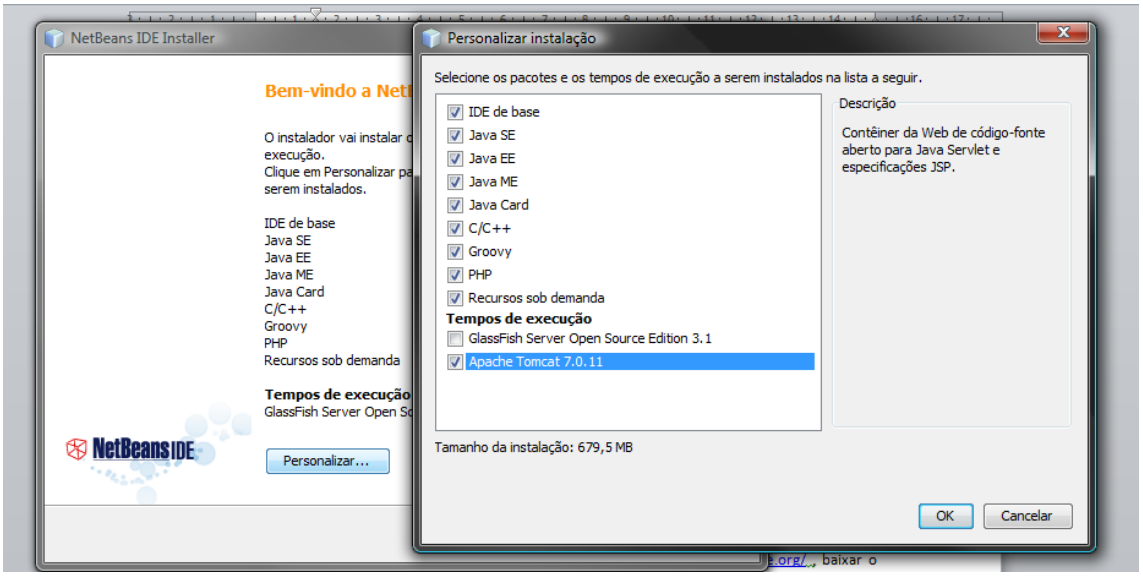
Iremos agora mostrar passo a passo a instalação do Tomcat e do Netbeans.

1- Vá para o site : http://netbeans.org/

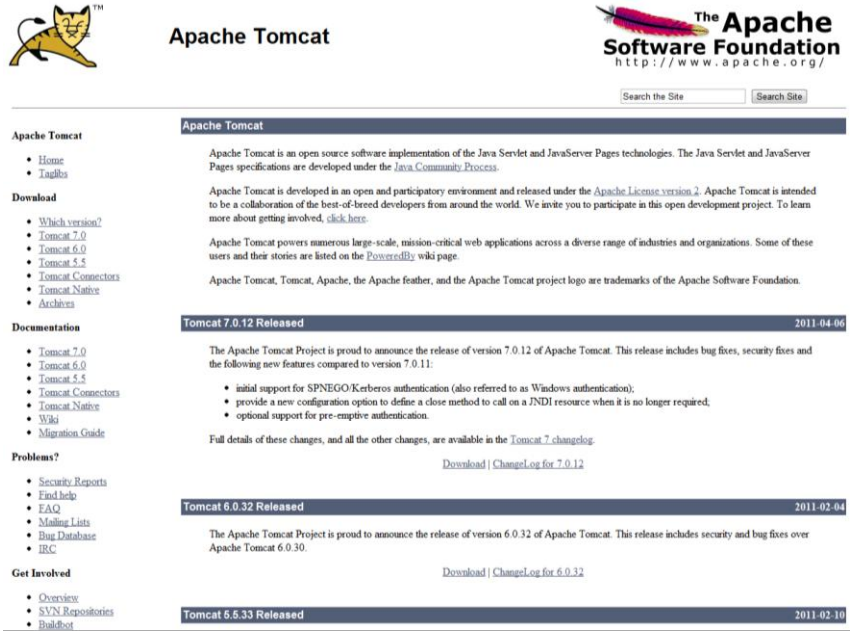
2- Clique em Download FREE e no botão download na coluna JAVA na página seguinte.

3- Após baixar o Netbeans, siga as instruções de instalação normalmente. O Apache Tomcat 6.0.26 está incluído nas opções de download "Java" e "Tudo", mas não é instalado por padrão nessas opções. Para instalar o Apache Tomcat da opção de download Java ou "Tudo", inicie o instalador e selecione Apache Tomcat 6.0.26 na caixa de diálogo Instalação personalizada.

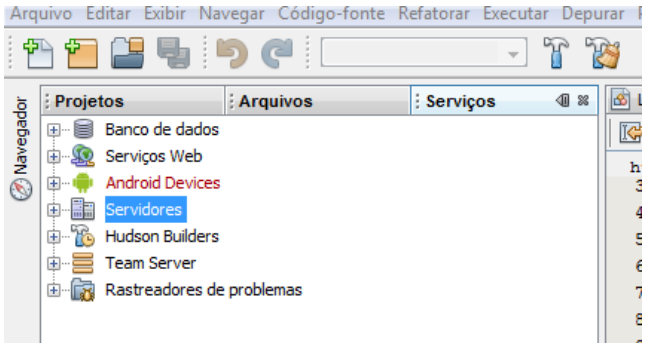


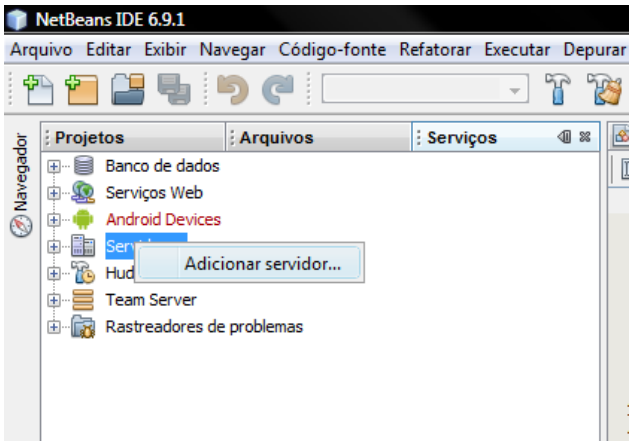


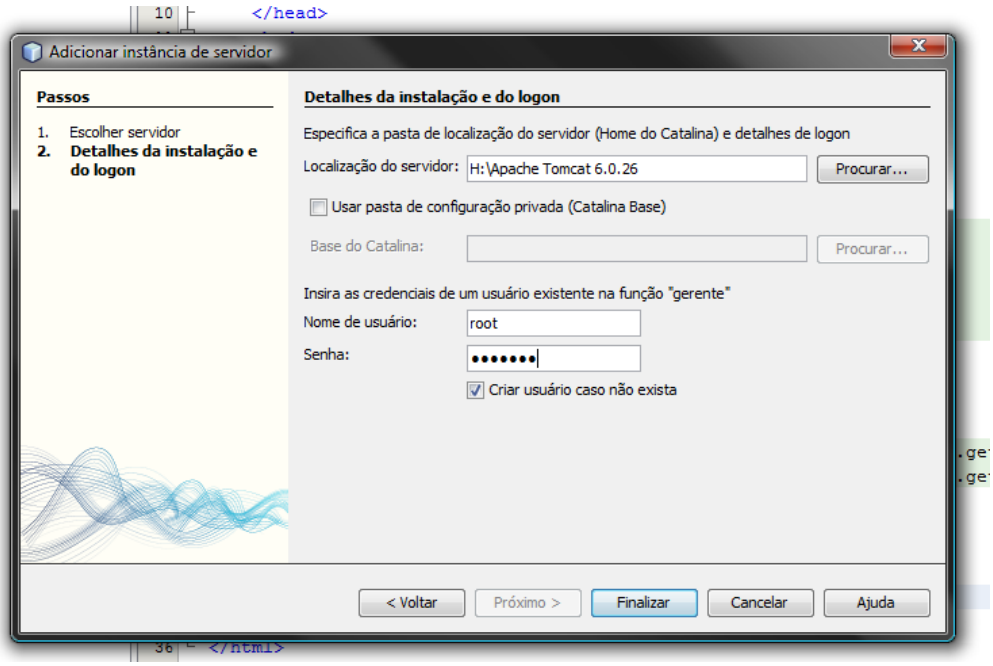
4- Ou você pode ir até o site da Apache: http://tomcat.apache.org/ , baixar o tomcat 6.0 (não baixe o 7.0 pois esse só é compatível com a versão 7.0 do Netbeans e estamos utilizando a 6.9.1). Descompacte a pasta zip baixada e coloque no diretório C: ou em outro lugar de sua preferencia. Após isso, entre na interface Netbeans para criar a conexão com esse servidor.



5- Uma vez aberto o Netbeans, vá para a aba “Serviços”. Clique com o botão direito em “Adicionar Servidor”. Escolha Tomcat 6.0 , clique em “Próximo” e na próxima janela, procure pela pasta descompactada, crie um usuário chamado root e crie uma senha própria.







O Tomcat e Netbeans já estão configurados e prontos para o uso.

Elementos do JSP

O JSP é composto de cinco elementos:

Scripting: possibilitam o uso de código Java capaz de interagir com outros elementos do JSP e do conteúdo do documento, por exemplo, Scriptlet.

Ações: tags pré-definidas incorporadas em páginas JSP, cujas ações em geral se realizam com base nas informações do pedido enviado ao servidor.

Diretivas: mensagens enviadas ao web container para especificar configurações de página, inclusão de recursos e bibliotecas.

Taglibs: bibliotecas de tags que podem ser criadas e usadas pelos programadores para personalizar ações e evitar a repetição de código.

Conteúdo Fixo: marcações fixas HTML ou XML que determinam uma parcela do conteúdo do documento.

Exemplos: (na ordem acima)

```
<% ...algum comando Java ....%>
<%=  alguma expressão em Java que retorne algum valor%>
```

```
<jsp:include page="url_da_pagina">  - Ação de Inclusão
<jsp:forward page="url_da_página"> - Ação de encaminhamento
```

```
<%@ page import="Java.io.*" %> - diretiva de importação
<%@ page isErrorPage="true|false" ou errorPage="url" %> - direcionamento de erros
<%@ include ... – inclusão de arquivos
<%@ taglibs .... – inclusão de taglibs
```

```
<html>
        <head>...</head>
        <body>
        Conteudo fixo
        </body>
</html>
```

```
<c:forEach: ...> </c:forEach> , <c:if ...>...</c:if>
Taglibs...
```

Primeiros Passos

Scriptlet ou Scripting

Um scriptlet é um pedaço de código Java embutido em um código JSP semelhante a um código HTML.

O scriptlet sempre está dentro de tags <% %> ou <%=%>. Eles definem uma expressão/código Java dentro de uma Página HTML. A diferença de cada um é:

<% %>: usado para instruções java, por exemplo: <% String str = "oi"; %>

<%= %>: usado para expressões que retornam um valor a ser inserido na página

HTML, por exemplo: <%= str %> (esse scriptlet irá mostrar o conteúdo de "str" onde for colocado na página html.

Expression Language

Similares aos scriptlets <%= %>. São usadas para expressões que retornem valores, JavaBeans (Visto na próxima aula) e parâmetros. Sua sintaxe é: ${...}

Classes em JSP

Os scriptlet nos dão poder de usar qualquer código Java numa página HTML. Desse modo, iremos trabalhar com classes e afins do mesmo modo que numa aplicação desktop (vista no módulo Java Básico). Exemplo:

```
<%@ page import="java.sql.*,travel.*" %>

<html>
        <body>
                <% String str = "Oi Mundo!";  %>
                <%= str %>
        </body>
</html>
```

Parâmetros

As páginas de uma aplicação WEB precisam trocar informações entre si. Para isso elas utilizam muitas vezes formulários. Quando se usa um formulário em HTML, se coloca seu nome, o endereço da página receptora e o método a ser enviados os dados (GET, menos seguro, com os parâmetros visíveis e POST, mais seguro. Com os parâmetros ocultos).

```
        <form name="nome" action="pagina" method="método">

                ...

        </form>
```

Cada objeto do formulário – text, texbox, button, submit....entre outros – deve vir com um nome. Na página receptora, esse nome será muito importante. Nela usaremos um comando chamado "getParameter()" onde ele presisa do nome do paramêtro para retornar seu valor. É importante resultar que o parâmetro retornado é sempre uma String, e caso originalmente ele fossem um número, ele deve ser convertido. Veja abaixo o exemplo:

```
        <% String nome = request.getParameter("nome"); %>
```

Uma vez capturado o parâmetro da página anterior, e atribuído seu valor a variável, seu valor já pode ser utilizado para a páginas.

Necessidade do JavaScript

Muitas vezes o usuário informa dados inválidos ou deixam campos obrigatórios

vazios antes de pressionarem o botão Enviar. Isso pode causar problemas ao processar os dados.

Para assegurar que os dados sejam validos e que todos os campos obrigatórios foram informados antes de o botão Enviar ser pressionado, você poderá utilizar Javascript nas páginas HTML.

O Javascript não faz parte da "família" Java, ele é um tópico pleno em si e separado. É outra linguagem de programação, muito boa, que torna as páginas mais dinâmicas. Ela não necessita de um servidor próprio como o Java, o php, o asp entre outros. Por isso veremos Javascript apenas como uma forma de validar os campos dos formulário. E para dar mais dinamismo as aplicações.

Não há necessidade de um compilador, ou kit para o Javascript, ele é tão nativo quanto o HTML.

Como colocar Funções de JavaScript em HTML

Antes ou dentro das etiquetas <head>...<./head> de maneira interna a pagina ou em um arquivo próprio separado, de maneira externa a página, mas colocando um link para ligar a HTML e a JavaScript.

Maneira Interna, usando as etiquetas SCRIPT:

```
<script language="JAVASCRIPT">

        ...

</script>
```

Maneira externa, ao criar um arquivo.js, faça o link para a página.

```
    <script language="JavaScript" src="lugar onde está e nome"></script>
```

Ex:

```
    <script language="JavaScript" src="js/validacoes.js"></script>
```

Validação de formulários

Para tornar o código de validação acessível, deve-se criar uma função usando o comando "function". Dentro dessa função criada é onde o código irá ser colocado. Para acessar o campo do formulário é preciso chegar ate ele pela hierarquia de objetos do

JavaScript:

```
Document.forms.<nome do formulário>.<nome do campo>
```

Ex:

```
Document.forms.cadastro.nome
```

Para acessar o valor do campo, usa-se: “value”.

Exemplo de um código de validação:

```
Function validar(){
        var nome = Document.forms.cadastro.nome;
        if(<condição de erro>){
                alert(“mensagem”);
                nome.focus();
                return;
        }
        document.forms.principal.submit();
}
```

Seguindo sempre esse modelo: criamos um objeto que representa o campo do formulário (no exemplo, nome). Assim poderemos acessar o campo mais facilmente, deixando o código mais limpo. Depois disso, é realizada a verificação necessária. Caso ela atenda ao erro, damos um alerta usando o comando “alert”, focamos a atenção para o objeto, e por fim, terminamos o código. Caso ele não esteja com problema e passe sem problemas pelo “if”, o formulário poderá ser enviado usando o comando submit().

```
Function validar(){
        var nome = Document.forms.cadastro.nome;
        if(nome.value == “”){
                alert(“O campo nome não foi preenchido”);
                nome.focus();
```

```
                return;

            }

            document.forms.principal.submit();

        }
```

Várias outras funções podem ser utilizada, desde simples verificações de campos vazios, até mesmo expressões regulares complexas. Para números, podemos usar : isNaN() , que verifica se o conteúdo do campo não é um número – is Not a Number. E também podemos utiliza length para verificar se a quantidade de letras no campo obedece o máximo ou o mínimo desejado.

Extra- Validando por JSP (Descobrindo o porque o JavaScript é melhor e mais fácil)

Crie um novo Projeto. Na página index digite os códigos para criar um formulário pequeno onde se tem nome e idade:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%  //verificar se é a primeira vista a página ou se é o retorno do erro.
            String msg = request.getParameter("msg");
            String msgOut = "";
            if (msg != null && (msg.equals("1") || msg.equals("2")) ) {
                msgOut = (msg.equals("1")) ? "Forneça um nome!" : "Forneça idade válida!" ;
            }
%>

<html>
    <body>
        <form action="form4proc.jsp" method="post">
            <table border="0">
                <tr><td><b>Formulário 4</b></td></tr>
                <tr bgcolor="#dfdfdf">
                    <td colspan="2">
                        <font color="red"><%= msgOut %></font>
                    </td>
                </tr>
                <tr bgcolor="#dfdfef">
                    <td>Nome:</td>
                    <td><input type="text" name="nome" value="${param.nome}"></td>
                </tr>
                <tr bgcolor="#efdfdf">
                    <td>Idade:</td>
                    <td><input type="text" name="idade" value="${param.idade}"></td>
                </tr>
                <tr>
                    <td></td>
                    <td>
                        <input type="submit" value="OK">
                        <input type="reset" value="Limpar">
                    </td>
                </tr>
            </table>
        </form>
```

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

Crie uma nova página JSP para receber esses parâmetros e validar.

```
<% //capturo os parâmetros
            int erro = 0, idade = -1;
            String nome = request.getParameter("nome");
%>

<% //defino o erro
            if (nome == null || nome.length() == 0) erro = 1;
            else {
                try {
                    idade = Integer.parseInt(request.getParameter("idade"));
                } catch (NumberFormatException exc) {
                    erro = 2;
                }
                if (idade < 0)  erro = 2;
            }
%>

<% //redireciono para para a página correta utilizando diretiva forward
            if (erro > 0) {
%>
                 <jsp:forward page="index.jsp">
                        <jsp:param name="msg" value="<%= erro%>"/>
                    </jsp:forward>
<%
            }
            String pagina = (idade < 18) ? "form4menor.jsp" : "form4maior.jsp";
%>
            <jsp:forward page="<%= pagina%>">
                <jsp:param name="nome" value="${param.nome}"/>
                <jsp:param name="idade" value="${param.idade}"/>
            </jsp:forward>
```

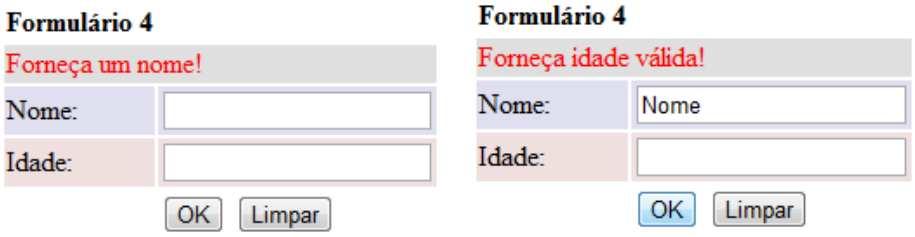
Veja:

Formulário 4

Nome:	
Idade:	

OK Limpar

Ao clicar em OK, o usuário é redirecionado para a página de validação, caso gere um erro, ele verifica qual o erro foi gerado e volta a página index. A página SEMPRE ira verificar se existem os parâmetros de erro. Caso eles existam, ele mostra uma mensagem de acordo com o erro, senão nada acontece. No caos de um erro, ele retorna e mostra um erro acima do formulário.



Caso não ocorra erro, ele usa a diretiva forward para redirecionar o usuário para a determinada pagina, passando os parâmetros nome e idade usando a diretiva param.

Usando o JavaScript é bem mais fácil de se fazer isso, por que o JavaScript "reconhece" os elementos do html, eu consigo ler, modificar, excluir, limpar um valor de um campo ou outro objeto na pagina sem a necessidade de ficar passando parâmetros. Parâmetros, isso por que o JavaScript é uma linguagem dinâmica. Muitos Programadores usam outras linguagens dinâmicas como o Python e o Ruby para fazer validações entre outras coisas e deixam a parte mais pesada e a comunicação com o banco de dados com o Java.

Projetos feitos em aula:

- Soma

- Calculadora

- Calculadora Classes

- calculadora validando

Exercícios Complementares de fixação:

1- Refaçam em casa os exercícios feitos em aula.
2- Crie um formulário que cadastre os tipos de objetos indicados abaixo. Não é preciso cadastrar, apenas faça a estrutura do formulário HTML (Capriche e coloque seus conhecimentos adquiridos no módulo de HTML e CSS), envie para uma página de resposta e mostre os dados dos objetos "cadastrados". Faça esse exercício utilizando uma primeira versão sem o uso de classes (como no projeto calculadora) e uma segunda versão com o uso de classes (como no projeto Calculadora Classes).
    a. Aluno
        i. Nome: String
        ii. RA: Integer
        iii. Curso: String
        iv. Ano: Integer
        v. Período: String
        vi. Endereço: String
        vii. Cidade: String

viii. Estado: String
ix. Telefone: String

b. Produto
   i. Codigo: Integer
   ii. Descrição: String
   iii. Quantidade em estoque: Integer
   iv. Valor: Float
   v. Quantidade mínima em estoque: Integer
   vi. Quantidade máxima para compra:Integer
   vii. Fornecedor: String

c. Fornecedor
   i. Cnpj: Integer
   ii. Nome: String
   iii. Endereço: String
   iv. Cidade: String
   v. Estado: String
   vi. Telefone: String
   vii. E-mail:String

d. Cliente
   i. Nome: String
   ii. RG: Integer
   iii. CPF: Integer
   iv. Idade : Integer
   v. Endereço: String
   vi. Cidade: String
   vii. Estado: String
   viii. Telefone: String
   ix. E-mail: String

e. Escola
   i. Cnpj: Integer
   ii. Nome: String
   iii. Endereço: String
   iv. Cidade: String
   v. Estado: String
   vi. Telefone: String
   vii. E-mail: String

f. Curso
   i. Codigo: Integer
   ii. Nome: String
   iii. Descrição: String
   iv. Coordenador: String
   v. CargaHorária: Integer

3- Crie uma página para gerar as notas finais de um aluno. Crie um formulário nela para que receba o Código do aluno, seu nome e suas três notas. Calcule a média ponderada do aluno usando os pesos: prova1 (25%), prova2 (25%) e prova3 (50%). Mostre na tela o código e nome do aluno assim como suas notas e média. Caso a média seja menor que 5, imprima a abaixo dos dados : “reprovado” , caso contrario “aprovado”. Mostre a mensagem e as médias nas cores vermelham para reprovado e azul para reprovado.
4- Validar usando JavaScript TODOS os formulários dos exercícios anteriores. Verifique se os campos foram preenchidos, caso haja necessidade, verifique se os campos são números (para valores numéricos).
5- Idem ao exercício 4, mas validando via JSP.

Exercícios de Pesquisa:

1. Defina: JSP, Java e TomCat.
2. Para que surgiu a JSP?
3. Qual(is) linguagem pode ser utilizada para se criar páginas JSP?
4. Antes de surgir a JSP era possível um desenvolvedor Java criar alguma aplicação para a Internet? Justifique.
5. O que é necessário para o desenvolvedor trabalhar com a JSP?
6. Defina e cite exemplos, que não sejam os da apostila, sobre : ações, diretivas, Scripting, taglibs.
7. O que é J2SE? Porque precisamos dele para trabalhar com a JSP?
8. Explique o funcionamento da JSP.
9. NA JSP, qual comando deveu utilizar para recebermos parâmetros passados através de tags de formulários?
10. Monte uma página JSP com uma tabela contendo os principais operadores condicionais, matemáticos e lógicos Java