

6.

Programação para Web

Sérgio Donizetti Zorzo

Resumo

Este capítulo trata da programação para web, apresentando uma visão global das aplicações na arquitetura www, das linguagens de marcação, e linguagens de programação nessa arquitetura. As linguagens PHP e Java são apresentadas como típicas para a programação de aplicações na web, desenvolvendo uma mesma aplicação nas diversas abordagens apresentadas.

Abstract

This chapter deals with web programming presenting a broad view of the applications for the WWW, the markup languages, and programming languages for this architecture. Java and PHP are presented as two representatives languages used for developing web applications. A case study is also presented developing the same application using different approaches.

6.1. Introdução

Para as aplicações web mais simples pode-se projetar uma arquitetura em que clientes utilizam um navegador, em sua máquina local, para fazer solicitações a um servidor web, que localiza o arquivo solicitado e o devolve ao cliente. Essa situação utiliza arquivos estáticos, ou seja, documentos que nunca sofrem alteração, independente da forma ou localização da solicitação.

No entanto, a maioria das aplicações exige que os dados sejam fornecidos por servidores web de forma dinâmica. O conteúdo dinâmico exige que o servidor web faça algum processamento adicional na solicitação feita. Além da identificação da solicitação, o servidor web vai depender de valores, transferidos do cliente para o servidor, que são parâmetros para a geração da resposta dinamicamente.

Os servidores web mais antigos não incluíam qualquer mecanismo embutido para gerar respostas dinamicamente, mas permitiam que fossem ativados trechos de código nativo para tratar tais solicitações, baseado no padrão CGI (Common Gateway Interface). Quando o servidor HTTP recebe uma solicitação de execução de uma CGI é gerado um processo, que gera a resposta desejada para, em seguida, enviar o resultado ao servidor HTTP, que o transmite ao solicitante. Esse mecanismo gera um overhead associado com a criação e comunicação com esse processo de tratamento da solicitação do cliente, pois os processos necessitam recursos e memória local para sua execução. Além disso, os processos foram projetados para atender a uma única solicitação.

Diversas tecnologias foram apresentadas para dar suporte ao conteúdo dinâmico de criação de páginas do lado do servidor. A linguagem PHP e a tecnologia Java são tecnologias para a construção de páginas dinâmicas, mas com abordagem e abrangência distintas. Apresentamos a descrição dessas duas tecnologias para a construção de sites dinâmicos. Na primeira seção é apresentado a linguagem de marcação HTML, com ênfase nos formulários e formas de envio dos dados de cliente para o servidor web. Em seguida, é apresentado a linguagem PHP com MySQL, detalhando uma aplicação exemplo. Na seção 4 é apresentada a plataforma Java, desde a concepção de applets, servlets, como a linguagem de script JSP e a conceituação de JavaBeans. Nas conclusões é feito uma avaliação dessas diferentes formas de desenvolvimento de aplicações web.

6.2. HTML

HTML (Hyper Text Markup Language) é uma linguagem de marcação que descreve a estrutura, o conteúdo e a apresentação de um documento e sua relação com outros documentos. Os documentos hipertexto são descritos por uma linguagem de marcação, que possibilitam a representação de uma informação e o vínculo a outros tipos de recursos, como texto, áudio, vídeo, gráficos, entre outros, permitindo que diferentes tipos de informações sejam exibidos de forma simultânea e que esses recursos diferentes se complementem. As *tags* são os elementos de marcação que compõem a estrutura de um documento hipertexto. Esse documento hipertexto é composto por dois tipos de texto: as informações que serão

exibidas e as *tags* que definirão como será exibida a informação contida no documento hipertexto.

Um documento HTML deve ser descrito pelos elementos de estruturação e pelos elementos que definem e organizam o seu conteúdo. Os elementos básicos da estrutura do documento incluem:

`<!DOCTYPE... >` - escolha da linguagem de marcação e o tipo da DTD (Document Type Declaration) correspondente.

`<html> ... </html>` - início e fim do documento

`<head> ... </head>` - seção de cabeçalho do documento em que se detalha algumas informações para os navegadores, a saber:

`<meta> ... </meta>` - incorpora metainformação do documento, que serão extraídas pelos navegadores. Ex:

`<meta http-equiv = "expires" content = "Tue 12 Oct 2004 18:00 GMT" />`

`<meta http-equiv = "refresh" content = "http://www.dc.ufscar.br/novo.html" />`

`<meta name = "descripton" content = "Programação Web" />`

`<link />` - permite a definição de vínculos do documento atual para outros documentos e recursos.

`<link rel="stylesheet" href= "estilo.css" type= "text/css" />`

`<style> ... </style>` - permitem que sejam definidas regras de folhas de estilo (CSS-Cascading Style Sheets) para aplicação no documento atual.

`<title> ... </title>` - indica o título do documento que será apresentado na barra superior.

`<script> ... </script>` - são utilizadas para códigos script, que é um programa, dentro do documento HTML.

`<body> ... </body>` - permite a definição de todo o conteúdo do documento.

No corpo do documento a sua marcação estrutural inclui os seguintes elementos:

`<!-- coment -->` - comentários

`<div> ... </div>` - divide o documento em blocos, sem formatar o conteúdo.

` ... ` - divide a linha em blocos, sendo utilizado em associação com folhas de estilo.

`<a> ` - define um link clicável, estabelecendo uma navegabilidade para outro ponto do documento ou para outro documento.

` ` define o nome de um link

` vá para o inicio ` (outra parte do documento)

` vá para documento novo ` (do mesmo diretório)

 saiba sobre O dia de Java

<p> texto </p> - bloco de texto como um parágrafo, sendo que ocorre uma quebra de linha antes e depois dessas *tags*.

 - inserção de imagem no documento.

<dl> ... </dl> - lista de termos e definições com o uso das tags <dt> ... </dt> para os termos e <dd> .. </dd> para as respectivas definições.

<dl> <dt> termo1 </dt> <dd> definição do termo1 </dd>

<dt> termo2 </dt> <dd> definição do termo2 </dd> </dl>

 ... - lista não ordenada de termos, definidos por ...

 ... - lista ordenada de termos, definidos por ...

<table> ... </table> - tabela com informações tabuladas por linhas e colunas, onde o título de cada coluna é descrito por <th> ... </th>, as linhas delimitadas por <tr> ... </tr> e cada elemento da tabela descrito por <td> ... </td>. Os atributos width, border, align, valign, colspan e rowspan definem a largura da tabela e da borda, alinhamento horizontal e vertical dentro da célula, o número de colunas e linhas que a célula ocupará, entre outros.

<table border=4> <th colspan=2> COLUNA 1 E 2 </th>

<tr> <td> LINHA1,COLUNA1 </td><td> LINHA1,COLUNA2</td></tr>

<tr> <td> LINHA2, COLUNA1 </td><td> LINHA2,COLUNA2</td></tr>

<th rowspan=3> 3 LINHAS </th>

<tr> <td> LINHA UM </td></tr>

<tr> <td> LINHA DOIS </td> </tr>

<tr><td> LINHAS TRES</td></tr>

</table>

TESTE DE TABELAS	
COLUNA 1 E 2	
LINHA1,COLUNA1	LINHA1,COLUNA2
LINHA2, COLUNA1	LINHA2,COLUNA2
3 LINHAS	LINHA UM
	LINHA DOIS
	LINHAS TRES

<form> ... </form> - mecanismo de envio de informação, permitindo interatividade do usuário. Os atributos name, action, method e enctype definem o identificador do formulário, nome da aplicação que receberá os dados do formulário, o método de envio, e o formato de envio dos dados, respectivamente.

Há diversos métodos de envio, como PUT, HEAD, DELETE, OPTIONS, TRACE, mas os métodos GET e POST são utilizados com mais frequência.

O método GET é usado para requisitar informações do servidor, mas pode ser empregado no envio de um conjunto dos dados do formulário junto com a URL (endereço e nome da aplicação especificada pelo atributo action). Os dados são separados da URL pelo caractere ? e são agrupados por nome e valor, separados entre si pelo caractere &.

`<form action="http://www.dc.ufscar.br/~zorzo/receber.php" method="GET" >`

marcações dos campos de entrada

`</form>`

O método POST envia o conjunto de dados do formulário para a aplicação que irá fazer o tratamento como um bloco de dados separado da requisição. Esse método é utilizado quando o bloco de dados ultrapassa o tamanho máximo da URL ou quando não é seguro anexar os dados no final da URL.

Um formulário consiste de um ou mais campos de entrada de dados, descritos pela *tag* `<input>`, onde o atributo `type` permite a criação de campos de texto, senhas, caixa e lista de seleção, menus, botão de opção, entre outros.

`<input type=... name=... value=... size= ... maxlength=...>`, onde o atributo `name` associa um identificador ao campo de dados, `value` especifica o valor inicial do campo de dados, `size` determina o tamanho do campo, e `maxlength` especifica a quantidade máxima de caracteres que podem ser digitados nos campos de entrada. O atributo `type` pode ser:

`text` - cria um campo de entrada de texto de uma linha apenas.

`password` - cria campo de entrada de texto como o `text`, mas os caracteres digitados não aparecem, sendo substituídos por `*`.

`checkbox` - cria campo de caixa de seleção, sendo possível marcar ou desmarcar a seleção feita.

`radio` - cria um botão de opção. Para criar um grupo de botões, atribui-se o mesmo nome com valores distintos para os diversos botões.

`reset` - cria um botão que ao ser clicado restaura os valores iniciais do formulário. O valor do atributo `value` irá ser exibido no botão do formulário.

`submit` - cria um botão que ao ser clicado envia os dados do formulário para a aplicação especificada no campo `action` do formulário. O valor do atributo `value` irá ser exibido no botão do formulário.

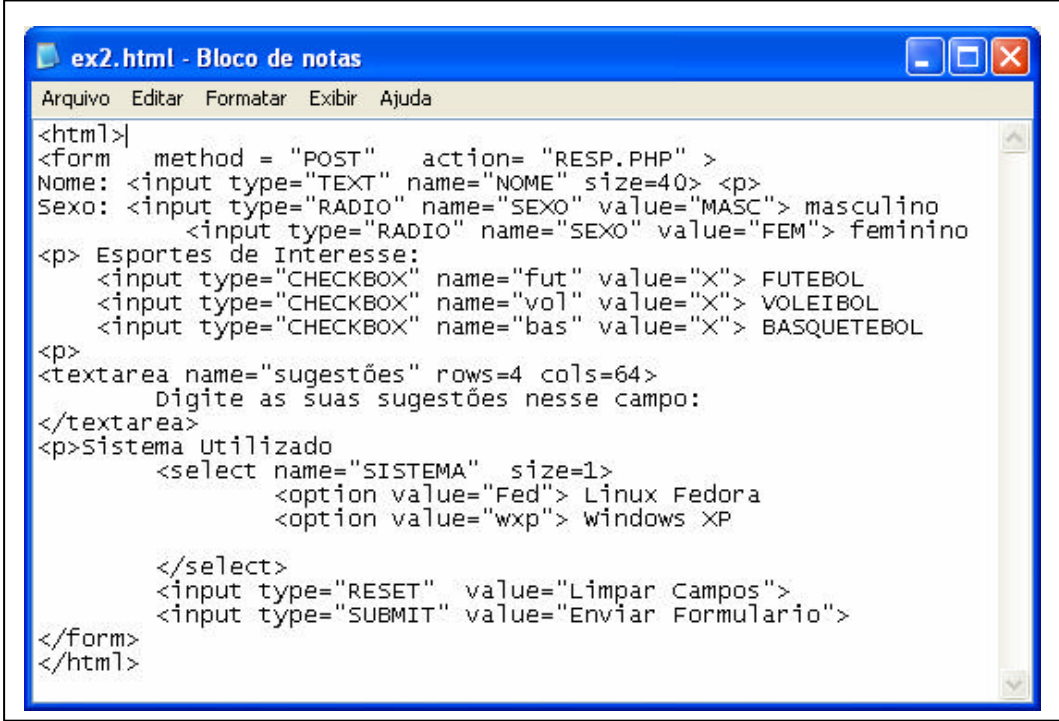
Outras *tags* podem compor os campos de um formulário, como `textarea`, que permite criar campo de texto de múltiplas linhas, e `select`, permitindo criar uma lista de opções.

`<textarea col=... row=... name=... wrap=...> texto inicial </textarea>`, onde o texto inicial será exibido com o numero de linhas e colunas especificado, podendo ser alterado pelo usuário para entrada de novos dados. O atributo `wrap`, com valores `soft`, `hard`, `off`, especifica a forma de quebra de linha que será apenas visual, físico ou não haverá quebra de linha.

`<select name= ... size=... multiple>` exibe no formulário uma lista de seleção, onde `size` determina o número de itens visíveis, e `multiple` se for permitido a seleção múltipla. Os itens serão descritos pela *tag* `option`, onde

`<option value= ... > texto </option>` exibe o texto com um item de seleção e associa o valor fornecido no parâmetro.

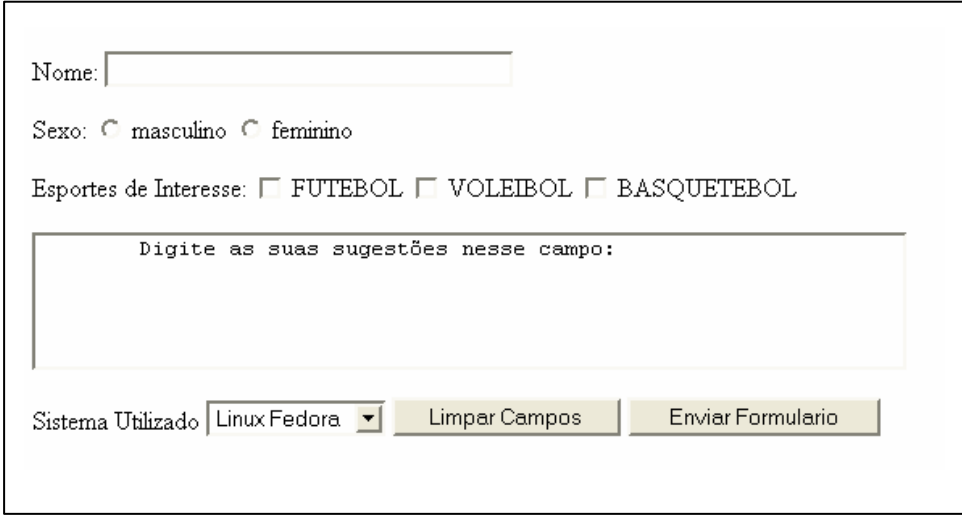
O documento HTML contendo o formulário



```
<html>
<form method = "POST" action= "RESP.PHP" >
Nome: <input type="TEXT" name="NOME" size=40> <p>
Sexo: <input type="RADIO" name="SEXO" value="MASC"> masculino
      <input type="RADIO" name="SEXO" value="FEM"> feminino
<p> Esportes de Interesse:
      <input type="CHECKBOX" name="fut" value="X"> FUTEBOL
      <input type="CHECKBOX" name="vol" value="X"> VOLEIBOL
      <input type="CHECKBOX" name="bas" value="X"> BASQUETEBOL
<p>
<textarea name="sugestões" rows=4 cols=64>
      Digite as suas sugestões nesse campo:
</textarea>
<p>Sistema utilizado
      <select name="SISTEMA" size=1>
        <option value="Fed"> Linux Fedora
        <option value="wxp"> windows XP

      </select>
      <input type="RESET" value="Limpar Campos">
      <input type="SUBMIT" value="Enviar Formulario">
</form>
</html>
```

será visível como:



Nome:

Sexo: ☐ masculino ☐ feminino

Esportes de Interesse: ☐ FUTEBOL ☐ VOLEIBOL ☐ BASQUETEBOL

Sistema Utilizado

6.2. PHP

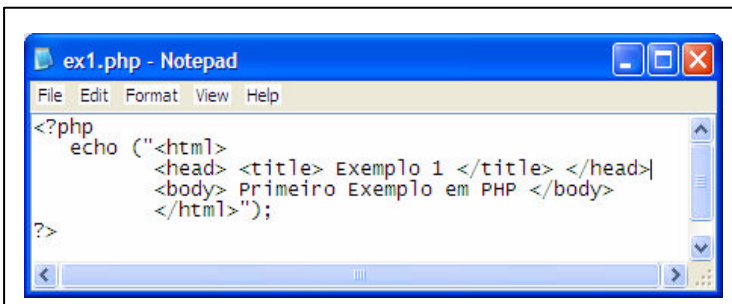
PHP (Hypertext Preprocessor) é uma linguagem de programação interpretada, voltada para o desenvolvimento de aplicações web, que pode ter seus trechos de códigos inseridos em uma página html. A sintaxe da linguagem é semelhante à sintaxe de C, Java e Perl, mas com simplificações que a torna mais atrativa para os desenvolvedores iniciantes. PHP é uma linguagem de script do lado do servidor, com o objetivo de permitir a escrita de páginas web dinâmicas.

PHP pode ser utilizado na maioria dos sistemas operacionais, incluindo Linux, e pela maioria dos servidores web, incluindo Apache e IIS da Microsoft. PHP oferece suporte a muitos gerenciadores de banco de dados incluindo MySQL, PostgreSQL, Oracle, entre outros, além do suporte ODBC (Open Database Connection), que permite o acesso a qualquer banco de dados nesse padrão.

O resultado de um programa escrito em PHP pode ser obtido pelo acesso, por um navegador web, a uma página identificada como PHP, disponibilizada por um servidor web ativo e que esteja conectado a um interpretador PHP.

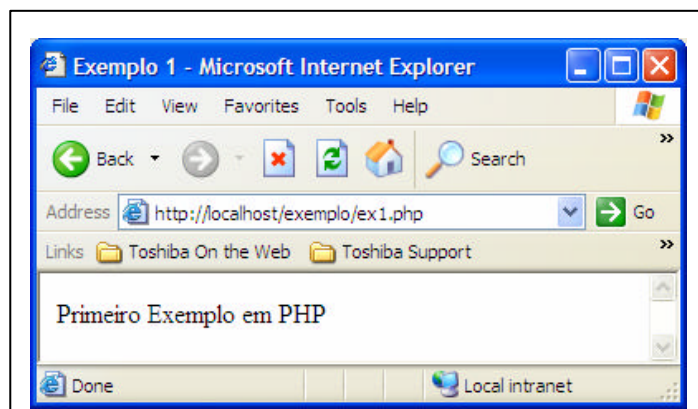
O interpretador PHP ignora todo o texto do arquivo, limitando-se a interpretar o código PHP que esteja entre tags que identificam o seu início e fim. Isso permite que num arquivo com código HTML possa ter diversos trechos de código PHP. O início e final do código PHP são descritas pelas tags `<?php?>` ou `<script language="php"> ... </script>`.

O arquivo `ex1.php` a seguir ilustra um exemplo em php, utilizando a função `echo` para exibir o conteúdo de seu parâmetro. Esse arquivo deve ser armazenado no diretório exemplo, como subdiretório de `httpdocs` do Apache, que é o diretório raiz desse servidor web. O diretório raiz do servidor web é determinado pelo seu arquivo de configuração.

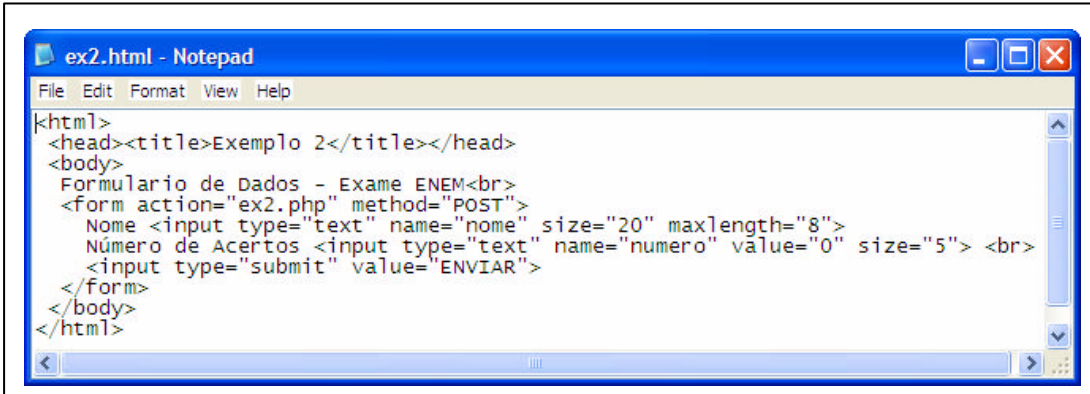


```
<?php
echo ("<html>
    <head> <title> Exemplo 1 </title> </head>|
    <body> Primeiro Exemplo em PHP </body>
</html>");
?>
```

e exibido por um navegador como

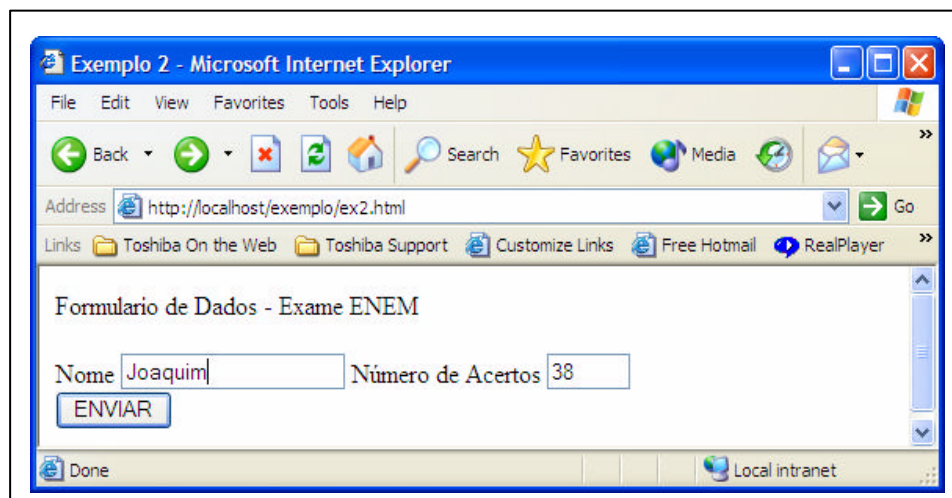


O tratamento de formulários será feito por um arquivo PHP, que deve ter acesso aos valores transmitidos pelo usuário. Em PHP faremos uso das variáveis \$_POST["nome do campo"] ou \$_GET["nome do campo"] ou ainda \$_REQUEST["nome do campo"], se o método de transmissão dos dados for post, get ou qualquer um deles, respectivamente. O “nome do campo” deve ser o mesmo utilizado no atributo name das tags do formulário.

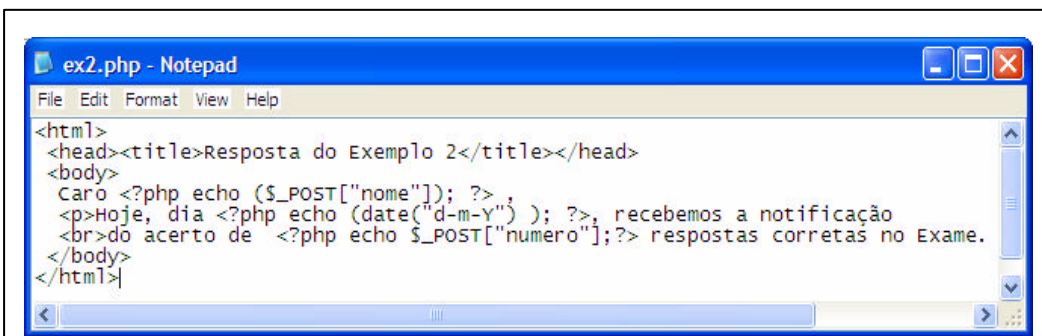


```
<html>
<head><title>Exemplo 2</title></head>
<body>
Formulario de Dados - Exame ENEM<br>
<form action="ex2.php" method="POST">
  Nome <input type="text" name="nome" size="20" maxlength="8">
  Número de Acertos <input type="text" name="numero" value="0" size="5"> <br>
  <input type="submit" value="ENVIAR">
</form>
</body>
</html>
```

visível como

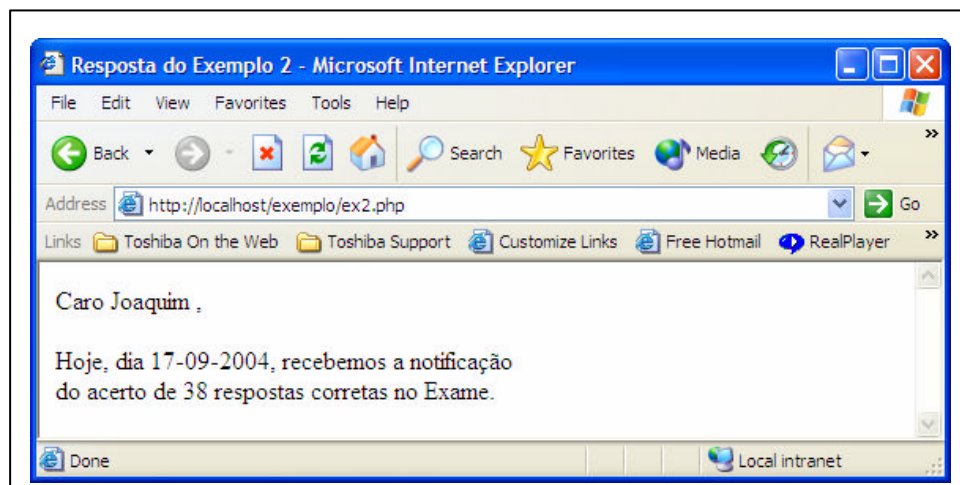


será tratada pelo arquivo ex2.php



```
<html>
<head><title>Resposta do Exemplo 2</title></head>
<body>
Caro <?php echo ($_POST["nome"]); ?>
<p>Hoje, dia <?php echo (date("d-m-Y")); ?>, recebemos a notificação
<br>do acerto de <?php echo $_POST["numero"];?> respostas corretas no Exame.
</body>
</html>
```


com resposta



6.2.1. Tipos de Dados.

PHP suporta os tipos de dados primitivos boolean, integer, float ou double e string, e os tipos de dados array e classe, além dos tipos especiais resource e NULL.

Um valor boolean pode ser TRUE ou FALSE, sendo considerado FALSE os seguintes valores: inteiro 0, float 0.0, string "" ou "0", objeto sem elementos, e o tipo NULL.

Inteiro é um valor especificado por sequência de dígitos, com ou sem sinal. Pode ainda ser representado em octal ou em hexadecimal, se for precedido por 0 ou por 0x respectivamente.

Números de ponto flutuante são representados por uma sequência de dígitos contendo o ponto decimal (.) e/ou o sinal e de expoente. O número de bytes de armazenamento é que determina o tipo de dados float ou double.

String é representada por uma sequência de caracteres entre apostrofes (') ou aspas ("), sendo que neste último caso a sequência de caracteres será interpretada para resultar em um conteúdo a ser representado.

```
$var1='Programação web'; // variável do tipo String contém a cadeia Programação web
```

```
$var2="<<$var1>>"; // variável do tipo String contém a cadeia <<Programação web>>
```

Array é uma sequência de pares ordenados que relaciona valores a chaves e pode ser utilizado como um array usual, uma tabela de um banco de dados e outras funcionalidades. Um array pode ser criado como o construtor array com uma sequência de pares chave => valor, onde a chave pode ser uma constante string ou integer e o valor constante de qualquer tipo. Se omitir a chave, ele tomará o valor seguinte ao da última chave, como é o caso da chave 4 de valor 200 do exemplo a seguir.

```
$arr1 = array (1=> 'Maria', 3=> 'Ana', 2=>'Rosa', 'outra'=> 100, 200);
```

```
echo "ultimo valor =" . $arr1['outra']; // será impresso ultimo = 100
```

```
$arr2 = array (10,20,30,40); // equivalente a array (0=>10, 1=>20, 2=>30, 3=>40);
```

O valor de um elemento de array pode ser outro array, possibilitando obter uma estrutura multidimensional, como estruturas de árvores.

Classe é um agregado de dados e métodos que os manipulam, sendo instanciada por objetos explicitamente pelo operador new. As instâncias podem acessar os métodos definidos na classe ou em suas classes ascendentes.

Um recurso é um tipo de dado especial, o qual possibilita que variáveis desse tipo possam manter uma referência de recurso externo.

NULL é um tipo de dados que mantém apenas um valor (NULL), que denota que uma dada variável não tem valor.

A conversão de um tipo de dados para outro pode ser feito explicitamente por meio de moldes (casting) ou pelo uso de funções (settype (nomevar, nome do tipo)). Os moldes permitidos são (int) ou (integer), (bool) ou (boolean), (float), (double) ou (real), (string), (array) e (object) para inteiro, booleano, ponto flutuante, string, array e object respectivamente. Algumas funções podem auxiliar o desenvolvedor na manipulação dos tipos de variáveis, como a função gettype (mixed var) retorna o tipo da variável var, settype(mixed var, string type), que converte a variável var para o tipo type, is_bool(mixed var), que verifica se o tipo da variável var é boolean, entre outros. Outra funcionalidade muito empregada na tipagem de dados é isset(mixed var), que verifica se a variável var existe.

6.2.2. Variáveis, Operadores e Expressões.

As variáveis em PHP são identificadas por sequência de caracteres iniciando com \$, como \$i, \$cont, etc. O tipo da variável é determinado pelo interpretador PHP em tempo de execução pelo tipo de valor que a mesma contém.

O escopo de visibilidade das variáveis é local ao arquivo, corpo da função ou classe em que foi definido. O escopo pode ser modificado para global, incluindo a palavra reservada global antes do nome da variável, como em uma declaração usual. A variável também pode ser alterada para static, existindo apenas no escopo local, mas não perdendo o seu valor quando a execução termina do bloco de código que a contém. Algumas variáveis foram pré-definidas, como \$_GET, \$_POST, \$_COOKIE, \$_SESSION, que são úteis para obter os valores postados via método GET, POST, cookies ou em sessões.

O conceito de variável em PHP é o mesmo das linguagens procedurais, mas não é feita a declaração inicial em que se estabelece a ligação entre o identificador da variável com o tipo de dados que será mantido. Uma variável mantém um valor de dado, que é de tipo específico, e que pode ser alterado na próxima atribuição.

Expressões são formas de expressar um dado valor, que em alguns casos resultam em alteração de valores armazenados em memória. PHP denota expressões com sintaxe e semântica muito semelhante a C++ e Java. A tabela 6.1 a seguir descreve a precedência e a semântica dos operadores.

Tabela 6.1. Precedência de Operadores

new	criação de objetos
[colchete (elemento de array)
! ~ ++ -- (int) (float) (string) (array) (object) @	Não lógico, inversão de bits, incremento e decremento, conversão de tipos e controle de erro
* / %	Multiplicação, divisão e resto de divisão
+ - .	Soma, subtração e concatenação
<< >>	Deslocamentos binários
< <= > >=	Maior, maior ou igual, menor, menor ou igual
== != === !==	Igual e diferente
&	AND binário
^	XOR binário
	OR binário
&&	AND lógico
	OR lógico
? :	operador ternário
= += -= *= /= .= %= &= = ^= <<= >>=	operador de atribuição
print	impressão
and	AND lógico
xor	XOR lógico (de menor precedência)
or	OR lógico (de menor precedência)
,	vírgula

Os operadores de igualdade / desigualdade e o operador new são não associativos, enquanto que os operadores unários e os de atribuição, além do print e colchete são associativos da direita para a esquerda, enquanto que os demais são associativos da esquerda para a direita.

O operador @ precedendo qualquer expressão indica que será ignorada qualquer mensagem de erro que possa ser gerada na avaliação daquela expressão. A mensagem de erro será sempre gravada na variável global \$php_errormsg, se o recurso track_errors estiver habilitado.

Os operadores de concatenação (.) e de concatenação com atribuição (.=) de string permitem a manipulação de cadeias de uma maneira muito simples.

O operador + aplicado a arrays resulta em um array obtido pela união das chaves, sem sobrescrever chaves repetidas. A identidade de arrays pode ser verificada por == ou ===, sendo que o último exige que os elementos estejam na mesma ordem, enquanto que a desigualdade de arrays pode ser verificada por != ou <>.

6.2.3. Estruturas de Controle.

A sintaxe das instruções em PHP segue a das linguagens C++ e Java. Uma instrução ou comando pode ser uma atribuição, uma chamada de função ou método, uma repetição ou a execução condicional de instruções. Toda instrução termina com a marca do ponto e vírgula (;), sendo que as instruções podem estar agrupadas em uma seqüência, denotando uma única instrução, se estiverem precedidas por { e sucedidas por }. O agrupamento de comandos pode ser feito de uma forma alternativa justapondo os comandos e finalizando com uma palavra chave de cada bloco, como endif, endwhile, endfor, etc.

O comando if se apresenta de forma incompleta, completa ou permitindo novas seleções de execução pela cláusula elseif.

```
if (expressão) {comandos}
if (expressão) {comandos} else {comandos}
if (expressão) {comandos} elseif (expressão) {comandos} else {comandos}
```

O comando switch é similar a uma seqüência de comandos condicionais.

```
switch (expr) { case valor: coms1; break; case valor: coms2; break; default:comsk;}
```

Os comandos de repetição da linguagem incluem while, repeat e for.

```
while (expressão) {comandos} ou while (expressão) com1; com2; ...comk; endwhile
do {comandos} while (expressão)
for (expressão1; expressão2; expressão3) comando
for (expressão1; expressão2; expressão3) com1; com2; ...comk; endfor
```

O comando foreach é aplicado a arrays, com a execução de um comando para cada um de seus elementos. No primeiro caso \$var vai assumir o valor do elemento corrente do array, enquanto que no segundo caso a chave do elemento atual será atribuída para \$chave.

```
foreach (expressão-array as $var ) comando
foreach (expressão-array as $chave => $var ) comando
```

O comando break interrompe a execução de um comando de repetição ou de seleção múltipla. O comando continue é empregado dentro dos blocos de execução e é

empregado para interromper a execução do corpo atual e continuar no início da próxima interação.

As funções em PHP devem seguir a sintaxe:

```
function nome ($param1, $param2, ... $paramk) { corpo da função }
```

sendo que no corpo da função pode ter qualquer código válido PHP, que inclui a definição de classes e de outras funções, mas deve necessariamente ter o comando `return $valor;` que finaliza a execução da mesma, retornando o valor fornecido.

A transferência de parâmetros pode ser por valor, indicado apenas pelo identificador do parâmetro, ou por referência, quando o identificador estiver precedido por `&`. A forma de transferência de parâmetros pode ser determinada na definição ou na ativação da função. Os parâmetros podem ter um valor padrão, descrito na definição da função, e que será utilizado no caso em que não foi transmitido esse argumento na ativação da função. O número de parâmetros pode ser variável, fazendo uso de funções para auxílio na determinação dos mesmos. Essas funções incluem `func_num_args()` que determinam o número de argumentos fornecidos na ativação da função, e `func_get_arg(n)` que fornece o (n+1)-ésimo argumento transferido para a função. O trecho de código a seguir ilustra a definição de uma função sem parâmetros e com a sua ativação com três argumentos, sendo obtido o valor do último argumento por `func_get_arg($nargs-1)`, onde `$nargs` mantém o número de argumentos transferidos.

```
function func( ) {  
    $nargs = func_num_args();  
    echo "ultimo argumento = ". func_get_arg($nargs-1);  
}  
$a=45; func ($i, 23,$a); // será impresso: ultimo argumento = 45
```

PHP possibilita criar funções variáveis, em que o nome de uma função pode ser modificado. Todo identificador seguido de parêntesis é considerado função para a linguagem, e se esse identificador for uma variável, então será avaliada podendo ativar a execução de diferentes funções.

```
$func='f1'; $func(); // ativa a execução de f1  
$func='f2'; $func(25,$a); // ativa a execução de f2
```

6.2.4. Objetos e Classes.

Objetos são instâncias de classes, que contém a definição dos dados e dos métodos que os manipulam. Uma classe é um tipo de dados, definido pela sintaxe ilustrada no exemplo da classe conta, onde temos a variável `saldo` e as funções `saldo()` e `credito()` que manipulam o saldo.

```

<?php
class conta {
    var $saldo ;
    function saldo () { return $this->saldo; }
    function credito ( $valor) { $this->saldo += $valor; }
}
class conta_numerada extends conta {
    var $numero;
    function numero () { return $this->numero; }
    function atualizanumero( $nro) { $this->numero=$nro; }
    function conta_numerada() { $this->numero=0; $this->saldo=0; }
}
$nc = new conta_numerada;
echo "conta " . $nc->numero; echo " possui saldo de " . $nc -> saldo();
$nc -> atualizanumero(2222); $nc -> credito (1000);
echo "<br>conta " . $nc->numero; echo " alterou saldo para " . $nc -> saldo();
?>

```

As classes podem ser definidas como extensões de uma classe base. É uma boa prática definir classes genéricas que podem ser utilizadas em todos os seus projetos, e adaptar essas classes com as necessidades específicas de cada projeto.

A classe derivada ou estendida tem todas as variáveis e funções da classe base e mais aquelas que foram definidas na classe derivada. No exemplo apresentado, a classe `conta_numerada` herda todas as características da classe `conta`. Em PHP só é possível implementar a herança simples.

PHP 5 apresenta um novo modelo de objetos, mas permanecerá com suas características de linguagem procedural. As alterações apresentadas evidenciam a aproximação dos conceitos e sintaxe de orientação a objetos existentes em Java e C++. As principais alterações incluem a propriedade de visibilidade da variável e da função de uma classe, que passam a ser pública, privada, ou protegida. As classes e métodos (funções) em PHP 5 podem ser abstratos, podendo também ser definida uma interface que será implementada por classes concretas. Os métodos podem ser finais, não sendo permitido a sua reescrita nas classes derivadas e, é possível implementar construtores e destrutores em uma classe. Os objetivos das alterações apresentadas visam o desempenho das aplicações que utilizam a orientação a objetos em sua implementação.

6.2.5. Acesso a Banco de Dados.

Um dos recursos mais poderosos e simples da linguagem é a capacidade de se conectar a um banco de dados. PHP 5 apresenta o SQLite, que é um gerenciador de banco de dados implementado por uma biblioteca de funções, podendo ativar suas funcionalidades nos programas escritos em PHP 5. A biblioteca SQLite lê e escreve diretamente para e do arquivo do banco de dados no disco. MySQL não estará mais sendo o gerenciador de banco

de dados padrão em PHP 5, mas pode ainda ser utilizado, como os demais gerenciadores de banco de dados.

Conexões persistentes são os links da linguagem de consulta estruturada (SQL) mantém com o banco de dados, quando a execução de um comando SQL termina. Quando uma conexão persistente é solicitada, PHP verifica se já existe uma conexão prévia para o mesmo usuário numa determinada máquina com a mesma senha.

A função `mysql_pconnect()` abre uma comunicação persistente, onde será priorizado a escolha de uma conexão já existente com o mesmo servidor, usuário e senha, ou será aberta uma nova conexão. A função `mysql_connect()` é semelhante, mas sempre abrirá uma nova conexão, que ficará aberta para uso futuro e fechada pela função `mysql_close()`.

A manipulação dos dados em MySQL pode ser analisada em quatro fases:

1ª) conexão com o servidor: `mysql_connect(servidor, usuário, senha)` (ou `mysql_pconnect`) em que o servidor é localhost e não houve o cadastro de usuário e senha. `mysql_connect` retorna um recurso, que será utilizado nas próximas transações.

2ª.) ligação com o banco de dados: `mysql_select_db (banco, recurso)` faz a abertura do banco fornecida com a conexão fornecida. Caso não seja fornecido o recurso será utilizado o da última conexão feita.

3ª.) acesso aos dados: `mysql_query(consulta, conexão)` realiza a transação, fornecida pela string consulta contendo uma expressão SQL, ao banco de dados da conexão fornecida. Na consulta realizada no exemplo a seguir, todos os registros da tabela cadaluno serão armazenados numa memória de resultados.

A memória de resultados pode ser manipulada por diversas funções de MySQL, como `mysql_fetch_array($resultado, tipo)`, que devolve um array com a próxima tupla de resultado e que pode ser manipulado pelo índice dos campos, `mysql_data_seek (resultado, posição)`, que posiciona a memória de resultados na posição fornecida, `mysql_numrows(resultado)` que fornece o número de tuplas obtidas, `mysql_free_result (resultado)` que libera o espaço da memória de resultados, entre outros. O tipo da consulta a memória de resultados pode ser `MYSQL_NUM` ou `MYSQL_ASSOC`, associando os valores de forma numérica ou pelos nomes dos campos.

4º.) fechamento da conexão: `mysql_close(recurso)` encerra a conexão estabelecida com o comando `mysql_connect` com o recurso fornecido. Caso não seja fornecido o argumento, será encerrada a última conexão aberta.

O exemplo a seguir ilustra o uso de MySQL no banco de dados “alunos”, manipulando a tabela “cadaluno”, que possui três campos de dados: Nro (integer), Nome (varchar[30]) e Nota (float).

```
<?php
$conexao = mysql_connect("localhost", "", "")
or die("Erro na conexao " . mysql_error() . "<br>");
echo "Conexao realizada <br>";
$abert = mysql_select_db("alunos")
```

```

        or die("Erro na abertura do banco de dados". mysql_error(). "<br>");
$consulta = "select * from cadaluno";
$resultado = mysql_query($consulta, $conexao)
        or die("Falha na consulta " . mysql_error(). "<br>");
echo "<center> <H1> ALUNOS </H1>";
echo "<table border=1>";
while ( $linha = mysql_fetch_array($resultado,MYSQL_ASSOC) ) {
    echo "<tr>";
    foreach ($linha as $valor) {
        echo "<td> $valor </td>";
    }
    echo "</tr>";
}
echo "</table>";
mysql_free_result($resultado);
mysql_close($conexao);
?>

```

6.3. Java

Java é uma linguagem orientada a objetos, independente da plataforma e segura. A orientação a objetos é empregada no processo de desenvolvimento de software, onde um programa é concebido como um grupo de objetos que trabalham juntos, comunicando-se e cooperando para solucionar o problema proposto. A neutralidade de plataforma é a capacidade de um programa ser executado sem modificações em diferentes ambientes computacionais, tanto de software como de hardware. Os programas em Java são compilados para um formato denominado bytecode e que podem ser executados por diferentes máquinas virtuais Java, presentes em diferentes plataformas de software e de hardware.

Java cuida do gerenciamento da alocação e desalocação de espaço em memória, não sendo necessário que o programador tenha essa preocupação. A linguagem não permite o uso de ponteiros, minimizando a geração e o tempo gasto com a depuração de erros. Java só possui herança simples na programação orientada a objetos.

A sintaxe da linguagem Java é oriunda da linguagem C e a versão atual da linguagem, denominada Java Tiger, apresenta algumas alterações sintáticas, como número de parâmetros variáveis, tipo enumerado, entre outros.

A linguagem Java é a linguagem empregada nas aplicações que utilizam os conceitos de applets, que são executados nas máquinas cliente. Mas as aplicações são limitadas a utilizar apenas um subconjunto da linguagem, de forma a garantir a segurança das aplicações. Nos servlets e JSP é também empregado Java como linguagem base para as aplicações.

6.3.1. Applets

Um applet é um programa escrito em Java que é referenciado por uma página HTML, da mesma forma que referenciamos imagens e arquivos de som ou vídeo. Quando uma página

que contém um applet é transferida do servidor web para a máquina do usuário, a Máquina Virtual Java residente no navegador será ativada para executar o código do applet.

Uma desvantagem dessa estratégia é que será necessário aguardar o término da transferência do código do applet, residente no servidor, para a máquina do usuário quando então será iniciada a sua execução. Por outro lado, há a vantagem de não ter que aguardar o servidor realizar todos os cálculos para transferir a página estática ao cliente. Essa análise deve ser fundamentada também pela classe de aplicação, que pode evidenciar uma das características apresentadas anteriormente. Applets funcionam bem em conexões rápidas ao servidor web, como em redes locais, mantendo uma cópia única do código de acesso a todos os usuários. Quando o código for alterado, seus usuários passam a utilizar a nova versão independente de notificação de atualização.

Os applets foram concebidos para serem executados em máquinas clientes de forma segura. Um applet pode exibir informações e obter dados de entrada do usuário, mas não pode ter acesso a arquivos locais e a dados do computador cliente, tanto para leitura como para escrita. Os applets não podem se comunicar com qualquer outra página da Internet além daquela que o contém. Os privilégios de segurança de um applet são especificados e podem ser negociados com o usuário, de forma que possa transmitir informações de cliente para o servidor de forma segura e confiável. Applets ou miniaplicativos Java foram projetados para serem pequenos, confiáveis e que possam ser executados em qualquer navegador web que possua a Máquina Virtual Java.

Um arquivo HTML pode ter vários applets, sendo que no local em que está presente a tag `<applet ...></applet>` será criada uma janela para iniciar a execução. Uma applet pode ter os seguintes atributos: `archive`, que fornece o nome do arquivo compactado .jar do applet; `alt`, que especifica o texto alternativo se o navegador não puder exibir o applet; `code`, fornecendo o nome da classe incluindo a extensão .class codebase, que fornece o endereço onde está localizada o arquivo contendo o applet; `align`, `height`, `hspace`, `vspace`, e `width` que tratam da apresentação do applet; e `object` onde é fornecido o nome do arquivo que contém o applet de forma serializado. A tag `<applet>` também permite o envio de valores parametrizados.

Um applet é programado como uma classe que herda todas as características da classe Applet, disponível na biblioteca `java.applet.*`, ou seja, herdando as características da classe pai. Applets não possuem métodos `main()`, mas métodos que são ativados quando ações específicas ocorrem em sua execução. Os cinco métodos mais importantes na execução de um applet: `init`, `start`, `stop`, `destroy` e `paint`. O método `init()` é ativado pelo navegador quando o applet é carregado para execução, sendo que as ações típicas são inicializações de variáveis, carregamento de sons para reproduzir ou imagens a exibir, bem como o desenho da tela de apresentação. O método `start()` é ativado logo após o término do método `init()` e toda vez que o usuário retornar à página que contém o applet. O método `stop()` é ativado toda vez que a área de exibição do applet deixa de ser visível, e como o método `start()` esse método pode ocorrer inúmeras vezes dentro do ciclo de vida do applet. O método `destroy()` é ativado no encerramento do applet, quando for descarregado da página e assim é possível realizar a liberação final de todos os recursos utilizados durante a sua execução. O método `paint()`, que é o modo como um applet apresenta algo na tela, ocorre

toda vez que for visualizado o trecho da página web que o contém, podendo ocorrer inúmeras vezes dentro de seu ciclo de vida.

O exemplo a seguir apresenta um applet que possui apenas o método `init()` que cria uma interface de entrada de dados e ativa a execução de ações por cliques de mouse em um dos três botões desenhados. A funcionalidade dos botões permite que seja realizada a soma ou multiplicação dos dados de entrada e exibindo o resultado escolhido.

```
import java.awt.*;
import java.applet.*;

public class exapplet extends Applet {
    java.awt.Button button1 = new java.awt.Button();
    java.awt.Button button2 = new java.awt.Button();
    java.awt.Button button3 = new java.awt.Button();

    java.awt.TextField textField1 = new java.awt.TextField();
    java.awt.TextField textField2 = new java.awt.TextField();
    java.awt.TextField textField3 = new java.awt.TextField();

    java.awt.Label label1 = new java.awt.Label();
    java.awt.Label label2 = new java.awt.Label();

    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        setForeground(java.awt.Color.black);
        setFont(new Font("Dialog", Font.PLAIN, 12));

        label1.setText("DADOS DE ENTRADA");
        label1.setAlignment(java.awt.Label.CENTER);
        label1.setFont(new Font("Dialog", Font.BOLD, 12));
        label1.setBounds(10,10,150,30);
        add(label1);

        button1.setLabel("Somar");
        button1.setFont(new Font("Dialog", Font.BOLD, 12));
        button1.setBounds(10,100,80,30);
        add(button1);

        button2.setLabel("Multiplicar");
        button2.setFont(new Font("Dialog", Font.BOLD, 12));
        button2.setBounds(110,100,80,30);
        add(button2);
```

```
button3.setLabel("Limpar");
button3.setFont(new Font("Dialog", Font.BOLD, 12));
button3.setBounds(210,100,80,30);
add(button3);
```

```
textField1.setText("0");
textField1.setBounds(60,50,80,30);
add(textField1);
```

```
textField2.setText("0");
textField2.setBounds(220,50,80,30);
add(textField2);
```

```
label2.setText("RESULTADO");
label2.setAlignment(java.awt.Label.CENTER);
label2.setFont(new Font("Dialog", Font.BOLD, 12));
label2.setBounds(10,170,70,30);
add(label2);
```

```
textField3.setText("0");
textField3.setBounds(110,170,180,30);
add(textField3);
```

```
SymMouse aSymMouse = new SymMouse();
button1.addMouseListener(aSymMouse);
button2.addMouseListener(aSymMouse);
button3.addMouseListener(aSymMouse);
this.addMouseListener(aSymMouse);
}
```

```
class SymMouse extends java.awt.event.MouseAdapter {
    public void mouseClicked(java.awt.event.MouseEvent event) {
        Object object = event.getSource();
        if (object == button1)
            button1_MouseClicked(event);
        else if (object == button2)
            button2_MouseClicked(event);
        else if (object == button3)
            button3_MouseClicked(event);
        else if (object == exapplet.this)
            exapplet_mouseClicked(event);
    }
}
```

```

void button1_MouseClicked(java.awt.event.MouseEvent event) { // botão somar
    int valor1, valor2, parcial;
    valor1= Integer.parseInt (textField1.getText()) ;
    valor2 = Integer.parseInt (textField2.getText());
    parcial = valor1 + valor2;
    textField3.setText( String.valueOf(parcial) );
}

void button2_MouseClicked(java.awt.event.MouseEvent event) { //botão multiplicar
    int valor1, valor2, parcial;
    valor1= Integer.parseInt (textField1.getText()) ;
    valor2 = Integer.parseInt (textField2.getText());
    parcial = valor1 * valor2;
    textField3.setText(String.valueOf(parcial) );
}

void button3_MouseClicked(java.awt.event.MouseEvent event) { //botão que limpa
    textField1.setText("0");
    textField2.setText("0");
    textField3.setText("0");
}

void exapplet_mouseClicked(java.awt.event.MouseEvent event) {
    button1.setVisible(true);
    button2.setVisible(true);
    button3.setVisible(true);
}
}

```

6.3.2. Servlets

Os Servlets Java são executados num servidor web por um módulo especial, o motor de servlet, que otimiza a sua execução com o mínimo de recursos do servidor web. Há diversos servidores web que aceitam servlets, devendo ser observado as especificidades de cada um.

Os servlets, por serem escritos em Java, são independentes de plataforma podendo ser criados, compilados e executados em plataformas diferentes.

Servlet é uma API Java que compilada atende requisições de páginas HTML e compatível com a plataforma Java. Os servlets trabalham com o servidor por multithread, implementando sessões e tratando cookies com segurança. Além disso, os servlets são escritos utilizando-se o paradigma orientado a objetos, que possibilita a modularização do sistema.

Um servlet Java é uma subclasse da classe `HttpServlet`, que faz parte do pacote `javax.servlet`, devendo ser importado os pacotes `javax.servlet` e `javax.servlet.http`. A classe `HttpServlet` implementa a interface `Servlet`, que estabeleceu arquitetonicamente o funcionamento dos servlets, estendendo a funcionalidade do servidor web com os recursos avançados de processamento. A classe `HttpServlet` inclui alguns métodos que representam o ciclo de vida de um servlet, com comportamento estabelecido na interface.

O método `init(ServletConfig config)` é automaticamente executado para tornar o servlet ativo, sendo que seu argumento é fornecido pelo servidor que está executando o servlet. Um mesmo servlet pode ser ativado por diversos usuários, mas o método `init()` só será realizado apenas uma vez e novas solicitações do mesmo servlet não ativarão o método `init()` enquanto o servlet estiver ativo. Geralmente é nesse método que é feita a conexão com o banco de dados, permitindo que as demais requisições dos clientes façam uso dessa conexão para o acesso aos dados.

O método `destroy()` é chamado quando um servlet é finalizado pelo servidor. O servidor executa esse método após o decurso de um período de tempo da última requisição do servlet e será executado apenas uma vez. Geralmente, é no método `destroy()` que é realizado o fechamento da conexão ao banco de dados da aplicação.

Os métodos `doGet()` e `doPost()` respondem às requisições GET e POST dos clientes, respectivamente. Esses métodos recebem como argumento um objeto `HttpServletRequest` e um objeto `HttpServletResponse` que permitem a interação entre cliente e o servidor.

O servidor web que executa o servlet cria um objeto que implementa a interface `HttpServletRequest` e passa para o método `service`, que por sua vez, passa-o para o método `doGet` ou `doPost`. Esse objeto contém a solicitação do cliente, que podem ser manipulados por alguns métodos, descritos a seguir:

`String getParameter (String nome)` – retorna o valor referente ao parâmetro enviado ao servlet, onde `nome` é o nome do parâmetro.

`Enumeration getParameterNames()` – retorna todos os nomes dos parâmetros enviados para o servlet, como parte de uma requisição POST.

`String [] getParameterValues(String nome)` – retorna um array de Strings com os valores do parâmetro fornecido.

`Cookie [] getCookies()` – retorna um array de objetos `Cookie` do cliente e que foram armazenados pelo servidor web.

O servidor web que executa o servlet cria um objeto que implementa a interface `HttpServletResponse` e passa para o método `service`, que por sua vez, passa-o para o método `doGet` ou `doPost`. Esse objeto contém a resposta para o cliente, que podem ser manipulados por alguns métodos, descritos a seguir:

`void addCookie (Cookie cookie)` – adiciona um cookie no cabeçalho da resposta para o cliente.

`ServletOutputStream getOutputStream()` – obtém um fluxo de saída baseado em bytes.

`PrintWriter getWriter()` – obtém um fluxo de saída baseado em caractere.

void setContentType (String tipo) – especifica o tipo de documento de resposta que será produzido.

O exemplo a seguir ilustra a aplicação de soma ou multiplicação de dois valores inteiros, sendo ativado por uma pagina html contendo um formulário que será tratado pelo servlet abaixo pelo método POST.

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class somap extends HttpServlet {
    // método para inicializar o servlet
    public void init(ServletConfig ConfInicial) throws ServletException {
        super.init(ConfInicial);
    }

    public void doPost(HttpServletRequest solicitacao, HttpServletResponse resposta)
        throws ServletException, IOException {
        int valor;
        int numero1 = Integer.parseInt(solicitacao.getParameter("num1"));
        int numero2 = Integer.parseInt(solicitacao.getParameter("num2"));
        PrintWriter saida = resposta.getWriter();
        if (solicitacao.getParameter("Botao").equals("Somar")) {
            valor = numero1 + numero2;
            saida.println(
                "<HTML> <HEAD> <TITLE><H2>Exemplo Soma</TITLE> </HEAD> " +
                "<BODY > <H1 ALIGN=CENTER>Soma de dois numeros</H1> " +
                "Soma = " + valor );
            saida.println(
                " <P>vamos novamente somar dois numeros" +
                "<FORM method=post "+"ACTION=http://localhost:8080/examples/servlet/somap >"
                + "<P>Numeros: ");
            saida.println("<INPUT TYPE=TEXT NAME=num1> "+
                "<INPUT TYPE=TEXT NAME=num2><BR><BR>"+
                "<INPUT TYPE=SUBMIT VALUE=Somar NAME=Botao>"+
                "<INPUT TYPE=SUBMIT VALUE=Multiplicar NAME=Botao>" +
                "</FORM> </BODY> </HTML>");
        }
        else
        {
            valor = numero1 * numero2;
            saida.println("Multiplicacao = " + valor);
        }
    }
}
```

```
//método chamado quando o servlet é terminado
public void destroy() {
    System.out.println("Fim do servlet");
}
}
```

O método destroy é apenas ilustrativo, pois está exibindo uma mensagem no console da máquina servidora e não traz nenhuma informação adicional para o cliente.

Os servlets Java estendem a funcionalidade do servidor web, que recebe solicitações de atendimento e encaminha para o contêiner de Servlet. O contêiner Servlet faz um encadeamento das solicitações para o atendimento enquanto ativa a execução do método apropriado no servlet, o qual gera resposta para o servidor http, que se responsabiliza de transmitir ao cliente solicitante.

6.3.3. JSP

Para utilizar a tecnologia JavaServer Pages (JSP) é necessário que o servidor web esteja integrado a um contêiner JSP. O servidor Apache Tomcat possui contêiner JSP disponível, embora outros servidores também o possuam.

JSP é uma tecnologia também baseada em Java para o desenvolvimento de páginas dinâmicas na web. JSP é implementada através de servlets, mas o programador não precisa mais ter a preocupação de gerar todo o código HTML, como era feito em servlets. JSP permite a separação da parte dinâmica, escrita por scripts especiais, das partes estáticas, escritas como HTML usual. Páginas JSP são arquivos texto com sufixo .jsp e residentes no contêiner JSP do servidor web, substituindo as páginas HTML tradicionais. As páginas JSP incorporam o código Java no código HTML estático.

Esse modelo de programação enfatiza o projeto de aplicações centradas em componentes, como JavaBeans que é escrito em Java, que é um objeto independente que mantém o registro de uma coleção de propriedades e comportamentos. JSP possui tags especiais para acessar JavaBeans e manipular suas propriedades. O projeto de desenvolvimento de software baseado em componentes favorece o reuso, separando a apresentação da implementação, onde alterações de apresentação podem ser realizadas sem que haja a necessidade de realizar qualquer mudança na implementação.

Um processo especial trata as solicitações de clientes para páginas JSP. O contêiner JSP é um processo que executa a Máquina Virtual Java para fazer o tratamento de múltiplas solicitações de um mesmo JSP ao mesmo tempo, através de encadeamentos destas solicitações. O encadeamento gerenciado pelo contêiner contém informações similares aos processos, mas de forma otimizada e de forma que o compartilhamento de recursos torna-se facilitado.

As solicitações para o servidor web de páginas JSP são enviadas ao contêiner JSP, que ativa o compilador para tratá-las. O compilador faz a análise da página JSP, buscando as tags JSP e substituindo-as por código Java, gerando um servlet. Após a construção do código fonte do servlet correspondente à página JSP, o compilador Java realiza a compilação. O

código do servlet (em bytecode) é então carregado e submetido para execução no contêiner do servlet. A resposta gerada é enviada para o servidor web, que se encarrega de enviá-la ao cliente que submeteu a solicitação.

Esses passos são realizados no primeiro acesso a uma página JSP, pois o contêiner JSP armazena a tradução feita, bem como o código gerado após a compilação. Nos próximos acessos a essa página o tempo de resposta será otimizado. Num primeiro momento o contêiner verifica se a página JSP possui um servlet correspondente, obtido anteriormente por esse processo de tradução. Caso a página JSP acessada não foi alterada, após o último acesso, não será necessário fazer a tradução novamente.

Após isso, o contêiner analisa se o servlet JSP está carregado, uma vez que ele pode ter sido descarregado. Caso o servlet não esteja carregado, o seu carregamento ocorre imediatamente, ou se já foi carregado anteriormente, este passo não ocorre. Em ambos os casos, o servlet correspondente à página JSP é ativada para gerar a resposta, aguardando no encadeamento do contêiner do servlet. A resposta gerada é enviada para o servidor web, que se encarrega de enviá-la ao cliente que submeteu a solicitação.

JavaServer Pages (JSP) possui quatro categorias de tags de marcação: diretivas, elementos de criação de scripts, comentários e ações.

As diretivas informam ao contêiner JSP como deve ser o processamento do documento, não afetando o tratamento das solicitações de clientes. Os elementos de criação de scripts são usados para embutir instruções de programação, que serão executadas a cada ativação da página que os contém. Os comentários são usados para adicionar documentação a uma página JSP, que podem ser feitos por diferentes estilos. As ações permitem a transferência de controle entre páginas e a interação com componentes JavaBeans residentes no mesmo servidor.

6.3.3.1 Diretivas JSP

As diretivas não produzem visualização aos usuários no acesso à página, mas alteram as propriedades globais da página JSP, influenciando sua tradução em um servlet. Podem ser utilizadas para indicar a linguagem de criação de scripts, incluir o conteúdo de uma outra página, indicar que a página usa uma biblioteca de tags personalizadas, entre outros. A sintaxe básica de uma diretiva é:

```
<% @ diretiva atributo1="valor1" atributo2="valor2" atributo 3=... %>
```

onde diretiva pode ser page, include ou taglib.

A diretiva de inclusão (include) é utilizada como o atributo file para incluir o conteúdo de outro arquivo na página JSP.

A diretiva de página (page) é empregada para definir propriedades dependentes da página. Essa diretiva é aplicada à página JSP resultante pela inclusão de arquivos por outras diretivas. Exemplos dessas diretivas de página são dadas por:

```
<% @ page import="java.lang.*, javax.servlet.*" %>
```


fornecendo a lista de pacotes a serem importados pela página JSP.

```
<% @ page contentType = "text/html" %>
```

indicando o tipo de conteúdo da página resultante ao usuário.

A diretiva de biblioteca de tags é utilizada para notificar o contêiner JSP que essa página se apoia em uma ou mais biblioteca de tags personalizadas.

```
<% @ taglib uri = "url da biblioteca de tags" prefix=mtag %>
```

indica que será utilizada a biblioteca de tags armazenada no endereço indicado pelo atributo uri e será empregado o prefixo mtag para as tags dessa biblioteca.

6.3.3.2 Elementos de Criação de Scripts

Os elementos de criação de scripts permitem que trechos de código de programa, escrito em Java, sejam incorporados à página JSP. Esses trechos de código podem ou não produzir saída para o usuário da página. JSP oferece três tipos de elementos de criação de scripts: declarações, expressões e scriptlets.

As declarações permitem a definição de variáveis e métodos específicos para a página, e que poderão ser manipulados por outros elementos de criação de scripts da página.

A sintaxe `<%! declaração; declaração; %>` permite especificar diversas declarações numa mesma tag. As variáveis declaradas por essa tag tem o mesmo significado da criação de variáveis de instâncias no servlet equivalente, que foi gerado pelo processo de tradução da página JSP. Os métodos e variáveis declarados por essa tag podem ser acessados por todos elementos de criação de scripts da página. Exemplos ilustrativos são apresentados a seguir.

```
<%! private int x = 0, y = 0; private String curso = "poscc"; %>
```

```
<%! public long fatorial (long x) {  
    if (x==0) return 1;  
    else return (x * fatorial (x-1));  
} %>
```

As expressões em JSP são definidas pela sintaxe `<%= expressão-java %>`, com semântica de exibição do valor da avaliação da expressão na saída para o usuário.

As expressões são linhas de código individual e produzirão uma saída, que é um valor de um tipo de dados em Java. Esse valor será convertido em cadeia de caracteres (String) antes de ser adicionado à saída da página, empregando-se implicitamente o método `toString()` fornecido pela biblioteca `java.lang.Object()`.

```
<%= fatorial(7) %>
```

```
<%= (idade<18)? "menor": "maior"%>
```

indicando que será exibido o valor do resultado da aplicação da função fatorial(7), que é um valor do tipo long, convertido para String. Em seguida será exibida a cadeia “maior” se a variável idade estiver com o valor menor que 18.

Os scriptlets são empregados para a criação de scripts de objetivos gerais e de forma abrangente e não adicionam conteúdo à saída da pagina JSP.

A sintaxe de scriptlet <% Código Java %> permite que sejam tratados os parâmetros do formulário de entrada, chamadas a banco de dados e outros processamentos. As expressões e os scriptlets são usados pelo contêiner JSP para criar o método jspService() do servlet correspondente à pagina JSP. O método jspService() é o responsável pela geração da saída da página JSP, que será visível pelo usuário que acessou essa página.

Os scriptlets permitem que sejam incluídos blocos de instruções em Java, observando a estrutura de controle da linguagem. As estruturas de controle de seleção, repetição e exceções serão detalhadas a seguir.

A estrutura de controle de seleção de Java baseia-se nos testes lógicos true/false para controlar a execução do código, selecionando as cláusulas opcionais else if e else. O trecho da página a seguir ilustra o emprego do scriptlet de seleção, onde o código html estático é selecionado para exibição.

```
<% if (x<0) { %>
    <br>Não será possível calcular o fatorial, pois o número é negativo.<br>
<% } else if (x>20) { %>
    <br>Valores maiores do que 20 causam um erro de overflow.
<% } else { %>
    <br><%= x %>! = <%= fatorial(x) %>
<% } %>
```

Pode-se empregar as três estruturas de controle de repetição de Java nos scriptlets. As estruturas de controle for, while e do-while são ilustradas a seguir, onde é o scriptlet exibe uma tabela dos valores da função fatorial.

```
<table> <th>Inteiro</th><th>Fatorial(inteiro)</th>
<% for (long x = 0; x <= 20; x++) { %>
<tr><td><%= x %></td><td><%= fatorial(x) %></td></tr>
<% } %></table>
```

Utilizando o loop while, teremos o mesmo resultado com o trecho de código.

```
<%! public long x=0; %>
<table> <th>Inteiro</th><th>Fatorial(inteiro)</th>
<% while ( x <= 20) { %>
<tr><td><%= x %></td><td><%= fatorial(x) %></td></tr>
```

```
<% x++; } %></table>
```

e que pode ser obtido com a utilização da estrutura do-while.

```
<%! public long x=0; %>

<table> <th>Inteiro</th><th>Fatorial(inteiro)</th>

<% do { %>

<tr><td><%= x %></td><td><%= fatorial(x) %></td></tr>

<% while (++x<=20) ; } %></table>
```

O código HTML é usado para criar a tabela e seu cabeçalho, enquanto que o conteúdo de cada célula da tabela é gerado por uma expressão JSP, tendo sido controlado por um scriptlet.

O tratamento de exceções também é permitido que seja empregado nos scriptlets de controle de execução. O tratamento de exceções possibilita o controle de todos os problemas ocorridos durante a execução de um programa, fazendo o tratamento adequado. O emprego dessa estrutura de controle se dá em dois momentos, na definição da função em que se levanta a exceção e na ativação da função em que se dá o tratamento da exceção. A definição da função fatorial passa a ter o código

```
<%! public long fatorial(long x) throws IllegalArgumentException
{   if ((x < 0) || (x > 20))
        throw new IllegalArgumentException("Fora do dominio");
    else if (x == 0) return 1;
    else return x * fatorial(x-1);
} %>
```

e o tratamento da exceção se dá no trecho de código

```
<% try {
        out.print( "O fatorial é: " + fatorial( 21 ) );
    }
    catch (IllegalArgumentException e) {
        out.print(e.getMessage());
    } %>
```

6.3.3.3 Comentários JSP

Os comentários em JSP podem ser feitos de três formas diferentes: comentários de conteúdo, comentários JSP e comentários em Java.

Comentários de conteúdo são aqueles que serão transmitidos para o usuário que acessou a página JSP. O comentário de conteúdo será comentário na página exibida como resposta para o usuário, ou seja, não será visível. Mas poderá ser visualizada pela exibição do código fonte da página. Esse tipo de comentário é útil para indicar a forma de uso das informações da página JSP. O comentário de conteúdo

```
<!-- O tipo long de Java tem tamanho de 64 bits, então 20! =  
      <%= fatorial(20) %> é o limite máximo. -->
```

será exibido como resposta a página

```
<HTML>  
<HEAD>  
  Fora do dominio  
<!-- O tipo long de Java tem tamanho de 64 bits, então 20! =  
      2432902008176640000 é o limite máximo. -->  
</BODY>  
</HTML>
```

O comentário JSP é empregado para documentar a página JSP, não sendo enviado para a página visualizada pelo usuário. O corpo desse trecho será ignorado pelo contêiner JSP durante o processo de tradução da página JSP para um servlet. A sintaxe para o comentário JSP é `<%-- comentário --%>`, podendo ser empregado em qualquer momento do trecho do código.

O terceiro tipo de comentário são os comentários empregados pela própria linguagem de programação Java. Esse tipo de comentário não será transmitido para o usuário.

```
<%! float f; //comentario da linguagem  
    /* que tambem pode ser dessa forma */  
    } %>
```

6.3.3.4 Ações JSP

As ações são processadas para cada solicitação recebida do usuário ao acessar uma página JSP. As ações podem transferir o controle de uma página para outra, especificar applets e interagir com componentes JavaBeans no servidor. As ações podem ou não gerar conteúdo dinâmico e todas as tags personalizadas são consideradas ações. As ações a seguir ilustram a sua utilização.

```
<jsp:forward page= "http://www.dc.ufscar.br"/>  
<jsp:include page= "http://www.dc.ufscar.br/latosensu/material/2.jsp"/>  
<jsp:useBean name= "id" scope="page" />  
<jsp:getProperty name= "id" property="nome-da-propriedade" />  
<jsp:setProperty name= "id" property="nome-da-propriedade" value=valor />
```

onde é feito o deslocamento para outra página, a inclusão de uma página e as três últimas em que se usa e manipula propriedade de instâncias JavaBeans.

6.3.3.5 Objetos Implícitos em JSP

O contêiner JSP disponibiliza alguns objetos internos para o autor da página JSP. Esses objetos são chamados de implícitos, porque sua disponibilidade em uma página JSP é automática. O programador pode assumir que esses objetos estão presentes e acessíveis através dos elementos de criação de scripts JSP.

Os objetos implícitos podem ser agrupados em quatro categorias: objetos relacionados ao servlet da página JSP correspondente, objetos de entrada e saída dos dados da página, objetos fornecendo informação a respeito do contexto dentro do qual uma página JSP está sendo processada e objetos resultantes de erros.

Os objetos `page`, que representa o próprio servlet, e o objeto `config`, que armazena os parâmetros de inicialização dos servlets, são os objetos relacionados ao servlet da página JSP.

Os objetos de entrada e saída das páginas são os objetos `request`, que representa os dados de entrada, `response`, que representa os dados de saída, e `out`, representando o fluxo de saída associado com o objeto `response`.

O objeto `request` pode ser manipulado por métodos para acessar parâmetros da solicitação, para recuperar cabeçalhos de solicitação e por outros métodos de interface. Os métodos `getParameterNames()`, `getParameter(name)` e `getParameterValues(name)` permitem que se faça a ligação dos dados do usuário com a execução da página web.

- `getParameterNames()` Retorna os nomes de todos os parâmetros da solicitação
- `getParameter(name)` Retorna o primeiro valor de um único parâmetro da solicitação
- `getParameterValues(name)` Recupera todos os valores para um único parâmetro
- `getMethod()` Retorna o método de HTTP (GET, POST) para a solicitação
- `getQueryString()` Retorna a cadeia de consulta que segue o URL de solicitação

O objeto `response` permite que se defina o tipo MIME, a adição de cookies, enquanto que o objeto `out` é uma instância da classe `JspWriter`, com os métodos de impressão usual, como `print(String)` e `println(String)`. Possui ainda os métodos `getBufferSize()`, `clearBuffer()`, `flush()`, `close()`, dentre outros.

Os objetos contextuais fornecem informação a respeito do contexto em que a página JSP está sendo processada, de uma forma geral, da aplicação e de sessão. O objeto `session` fornece o contexto para a solicitação da qual a página está respondendo, o objeto `application` fornece um contexto no servidor, enquanto que o objeto `page-Context` se concentra no contexto da própria página JSP, fornecendo acesso a todos os objetos implícitos JSP que estão disponíveis e gerenciando os seus atributos.

O exemplo `ex1.jsp`, só possui uma tag de ação e que faz a inclusão de arquivo `ex2.jsp`, esvaziando o buffer de saída da página atual antes da inclusão de `ex2.jsp`.

```
<html>
<head><title> usando a Ação forward</title></head>
<body>
    <jsp:forward page="ex2.jsp?numero=32" />
</body>
```

</html>

A pagina ex2.jsp emprega os conceitos apresentados anteriormente

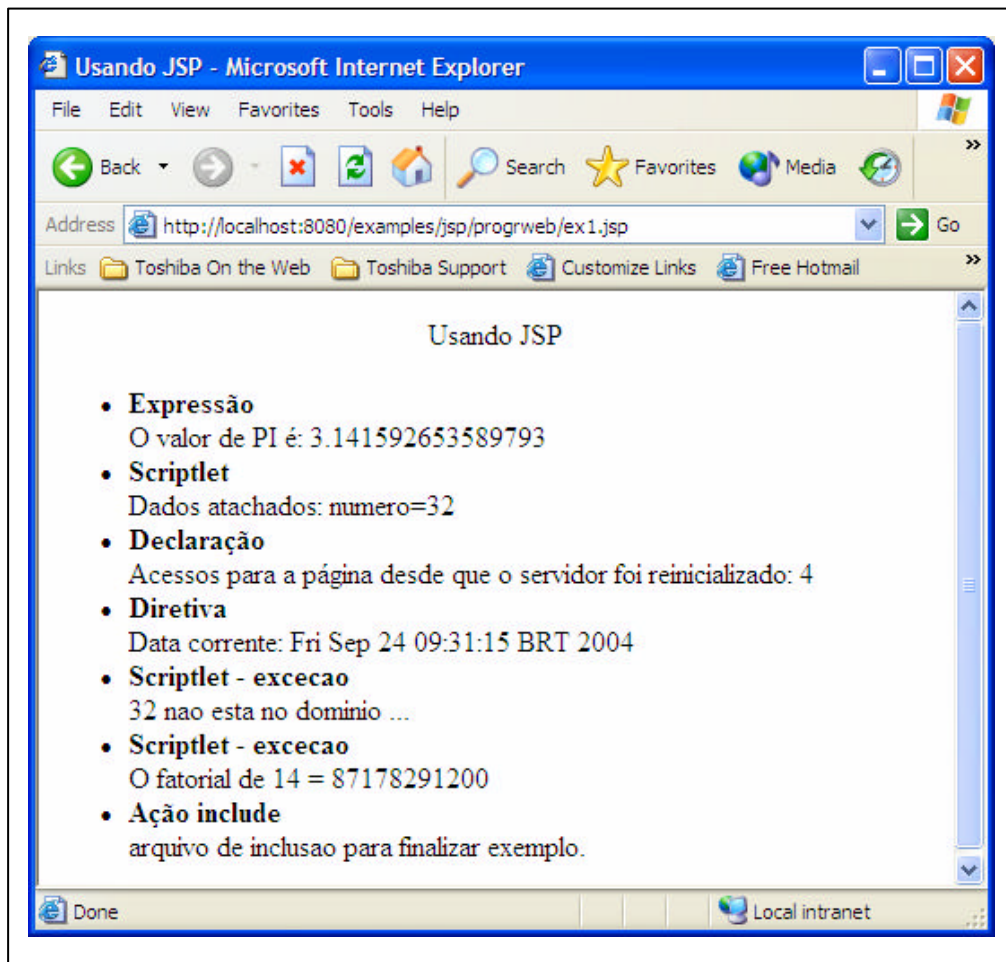
```
<html>
<head><title>Usando JSP</title> </head>
<body><center>Usando JSP</center>
<%-- usando as tags de JSP – comentário de conteúdo --%>
<ul>
<li><b>Expressão</b> <br>O valor de PI é: <%= Math.PI %>
<li><b>Scriptlet</b><br>
    <% out.println("Dados atachados: " + request.getQueryString()); %>
<li><b>Declaração</b><br>
    <%! private int contAcesso = 0 ; %>
Acessos para a página desde que o servidor foi reinicializado: <%= ++contAcesso %>
<li><b>Diretiva</b><br>
<% @ page import = "java.util.*" %>
    Data corrente: <%= new Date() %>
<%! public long num=0 ; %>
<%! public long fatorial(long x) throws IllegalArgumentException {
    if ((x < 0) || (x > 20))
        throw new IllegalArgumentException (x + " nao esta no dominio ...");
    else if (x == 0) return 1;
    else return x * fatorial(x-1);
    } %>
<li><b> Scriptlet - execucao </b><br>
<% try {
    num = Long.parseLong(request.getParameter("numero"));
    out.print("O fatorial e: " + fatorial(num));
    }
    catch (Exception e) {
        out.print(e.getMessage());
    }
%>
<li><b> Scriptlet - execucao </b><br>
<%-- Ativação da função fatorial,lançando uma exceção --%>
<% try {
    out.print("O fatorial de 14 = " + fatorial(14));
    }
    catch (IllegalArgumentException e) {
        out.clearBuffer();
        out.print("Sinto muito, 34 eh um valor fora do dominio");
    } %>
<li><b>Ação include</b><br>
<jsp:include page="ex3.jsp" flush="true" />
```

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

e o arquivo ex3.jsp contendo apenas uma mensagem de finalização.

```
<html>
<head><title> Arquivo de Inclusao</title></head>
<body>
    arquivo de inclusao para finalizar exemplo.
</body>
</html>
```

e será visualizado como



6.3.4 JavaBeans

JavaBeans são componentes de software escritos em Java, sendo que os próprios componentes são denominados beans. Esses componentes devem atender a especificação da API de JavaBeans, que foi criada com a finalidade de criar componentes de software independentes e reutilizáveis. Os beans são componentes de software que encapsulam tanto estado como comportamento, sendo que há diferentes tipos de componentes, como componentes visuais, de dados, de serviço, etc.

JSP precisa apenas de três tags simples para permitir interação com JavaBeans. Essas três tags permitirão inserir um JavaBeans na página JSP (<jsp:useBean>), acessar suas propriedades (jsp:getProperty>) e alterar suas propriedades(jsp:setProperty>).

Para criar uma instância de um JavaBean utilizamos a sintaxe, onde id é o identificador de instância para o bean e class especifica o nome da classe do próprio JavaBean, que deve estar em um diretório pré-determinado e conhecido pela aplicação.

```
<jsp:useBean id="nome bean" class="nome classe" />
```

ou

```
<jsp:useBean id="nome bean" class="nome classe" >
```

```
    código de inicialização
```

```
<jsp:useBean>
```

Para acessar as propriedades de um JavaBean criado usamos a sintaxe

```
<jsp:getProperty name="nome bean" property="nome propriedade" />
```

onde name é o identificador da instância da classe do JavaBean criado na aplicação e property é o nome da propriedade. A ativação dessa tag resulta no valor da propriedade escolhida, convertida para String.

Para modificar as propriedades de um JavaBean é necessário três atributos: name, que identifica a instância da classe de JavaBean criado na aplicação, property, que especifica a propriedade da instância do JavaBean que será definida, e value, fornecendo o valor da propriedade. A sintaxe utilizada é:

```
<jsp:setProperty name="nome bean" property="nome da propriedade"
    value="novo valor da propriedade"/>
```

Um JavaBean é um componente de software autocontido e reutilizável, e são implementados como classes Java que obedecem as especificações de componentes JavaBeans. Essa classe que irá definir um JavaBean ou simplesmente bean deve obedecer as seguintes observações:

A classe que implementa deve ter apenas um construtor sem argumentos.

Um bean deve fornecer propriedades que permitam a personalização do bean. Para cada propriedade é necessário definir métodos de obtenção e de alteração de seu valor.

Especificamente, se for criado a propriedade valor na classe então deverá ter os métodos getValor e setValor, acessados pelas tags <jsp:getProperty name="valor"/> e <jsp:setProperty name="valor" value="novo valor da propriedade"/>.

O método de obtenção não deve ter parâmetros e deve retornar um objeto do tipo da propriedade. O método de alteração deve ter um único parâmetro do tipo da propriedade e sempre retornará o valor void.

O exemplo a seguir detalha a criação de uma classe para ser um JavaBean e em seguida a sua utilização por uma página JSP.

```
public class SomaBean {  
    int numero1, numero2;  
    public SomaBean() {  
        numero1 = 0;  
        numero2 = 0;  
    }  
    public int getNumero1() {  
        return numero1;  
    }  
  
    public int getNumero2() {  
        return numero2;  
    }  
    public void setNumero1(int numero1) {  
        this.numero1 = numero1;  
    }  
    public void setNumero2(int numero2) {  
        this.numero2 = numero2;  
    }  
    public int getSoma() {  
        return (numero1 + numero2);  
    }  
}
```

que será utilizada pela página JSP

```
<html>  
<body>  
    <jsp:useBean id="soma" class="SomaBean" />  
    <jsp:setProperty name="soma" property="numero1" value="2" />  
    <jsp:setProperty name="soma" property="numero2" value="52"/>  
    <br>A soma  
    de <jsp:getProperty name="soma" property="numero1" />
```

```
com <jsp:getProperty name="soma" property="numero2"/>  
resulta em <jsp:getProperty name="soma" property="soma"/>  
</body>  
</html>
```

6.4. Conclusão

Os aplicativos são executados em um servidor web, sendo que cada aplicativo web pode ser visto como um conjunto de páginas estáticas, servlets, JavaBeans, páginas JSP ou PHP e qualquer outro recurso que seja utilizado pelo aplicativo e empacotados como uma aplicação web completa. Há arquiteturas, como J2EE, que buscam empacotar todos os arquivos relacionados à aplicação de forma a facilitar a distribuição para outros servidores, bem como o seu reuso.

A elaboração de páginas dinâmicas com o emprego da tecnologia Java e em PHP, podem evidenciar alguns fatos e contribuir para a análise da abrangência e segurança das aplicações desenvolvidas nas duas plataformas. Pode-se também analisar aspectos de manutenção e validação dentro do ciclo de vida de um sistema de software. Por outro lado, evidencia-se também o tempo de formação de programadores nas duas plataformas de desenvolvimento.

6.5. References

- Bond, M. et al. (2003) “Aprenda J2EE com EJB, JSP, Servlets, JNDI, JDBC e XML”. Pearson Education do Brasil Ltda, São Paulo, Brasil.
- Deitel, H.M. and Deitel, P.J (2000) “Java Como Programar”. Artmed Editora Ltda. São Paulo, Brasil.
- Fields, D.K. and Kolb, M.A. (2000) “Desenvolvendo na Web com JavaServer Pages”. Editora Ciência Moderna, Rio de Janeiro, Brasil.
- Horstmann, C. (2002) “Big Java”. Artmed Editora Ltda. São Paulo, Brasil.
- Hughes, S. & Zmievski, A. (2001) “PHP – Guia do Desenvolvedor”. Editora Berkeley. São Paulo, Brasil.
- Mecenas, I. (2003) “Java2 – Fundamentos, Swing e JDBC”. Editora Alta Books Ltda. Rio de Janeiro, Brasil.
- Muto, C.A. (2004) “PHP & MySQL – Guia Avançado”. Brasport Livros e Multimídia. Rio de Janeiro, Brasil.
- Niederauer, J. (2004) “PHP para quem conhece PHP”. Novatec Editora Ltda. São Paulo, Brasil.

<http://www.java.com>

<http://www.javabr.com.br>

<http://www.soujava.com.br>

<http://apache.jakarta.org>

<http://www.php.net>

<http://www.phpbrasil.com>

<http://www.mysql.com.br>

<http://www.mysql.org>