Java JSTL

Curso prático de linguagem Java

Rogério Laureano Martins

Minha gratidão a Deus por me permitir compartilhar o que tenho aprendido. Minha gratidão à minha esposa Beatriz e ao meu filho João por terem paciência. Minha gratidão aos professores e youtubers que de alguma forma procura passar o conhecimento pessoal às pessoas.

Rogério Laureano Martins

Linguagem Java

A linguagem Java é uma poderosa ferramenta de programação. Neste livro temos temos diversos conceitos desta linguagem.

Com o Java, o desenvolvimento de 4 tipos de programas são bastante populares.

Standalone → Programas populares para rodar em desktop. Semelhantes a Media Player, antivírus etc.

Web → Páginas HTML dinâmicas

Aplicativos corporativos → Aplicações para empresas como bancos, varejo, indústria etc.

Mobile → Aplicativos para equipamentos mobile como telefone celular, tablets etc.

Para tanto, forma criadas 4 plataformas de desenvolvimento para o java:

Java SE → Plataforma de desenvolvimento java que inclui todos os recursos de IO, Expressões regulares etc.

Java EE → Plataforma para desenvolvimentos corporativos e Web. É um pedaço importante do Java SE.

Java ME → Plataforma "micro" do Java. Utilizada principalmente para Mobile.

Java FX → Plataforma especial utilizada principalmente para aplicativos Web mais ricos.

História do Java

Em 1991 o projeto Java se iniciava com uma equipe formada por James Gosling, Mike Sheridan e Patrick Naughton. O time de desenvolvimento se denominou Green Team. Por isso o primeiro nome foi Greentalk. Posteriormente a linguagem mudou de nome para Oak, que era uma árvore típica dos Estados Unidos, França, Romênia etc. Como já existia uma empresa chamada Oak, mudou-se o nome para Java. Este nome foi dado em homenagem a ilha de Java, que tem uma história com a produção de café. Por isso também que o símbolo do java é uma chícara de café.

Features Java

O projeto Java, foi feito com o intuito de criar uma linguagem portável, segura e simples. Abaixo a lista dos principais features:

- Programa Simples → Os desenvolvedores pretenderam tirar a dificuldade da linguagem. É muito semelhante a linguagem C. Porém, sem as complicações de ponteiros. Este, por exemplo, podem ser usados. Porém, são internos no Java.

- Orientação por objeto → O Java é uma linguagem orientada por objetos. Possibilita a criação de Objetos, Classes, Heranças, Polimorfismo e todos os conceitos de orientação por objetos.

- Portável → O bytecode gerado é o mesmo em qualquer interpretador Java

- Multiplataforma → Por rodar em uma máquina virtual, "JVM" é portátil e roda em diversos sistemas operacionais.

- Segura → Por estar na JVM, tem mais uma camada de segurança independente do sistema operacional

- Robusta → Ferramente robusta no desenvolvimento

- Archetecture neutral → Por exemplo, os tamanhos dos tipos primitivos são reservados sempre o mesmo tamanho de bits.

- Linguagem interpretada → Ao rodar o aplicativo criado em Java, é necessário que se cria um bytecode. Ou seja, é feito uma "compilação" no Java para criar este bytecode. Após isso o Java interpreta o bytecode e executa o programa.

- Multithreaded → O Java permite programas com diversas threads em paralelo

- Distribuída → Facilmente distribuido

- Dinâmica → Linguagem bastante dinâmica e versátil

Preparação para programar em Java

Para executar qualquer programa java, você precisa

- Instale o JDK se você não o tiver instalado, faça o download do JDK e instale-o.
- Defina o caminho do diretório jdk
- Crie o programa java

- Compile e execute o programa java

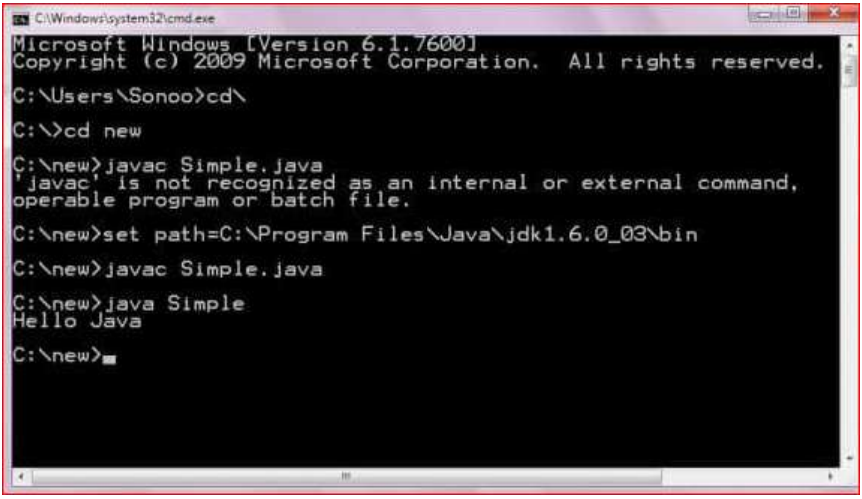
É necessário definir o caminho para o uso de ferramentas como javac, java, etc.

Se você estiver salvando o arquivo de origem Java dentro do diretório JDK / bin, não será necessário definir o caminho, pois todas as ferramentas estarão disponíveis no diretório atual.

No entanto, se você tiver seu arquivo Java fora da pasta JDK / bin, será necessário definir o caminho do JDK.

Existem duas maneiras de definir o caminho em Java:
1 – Temporário
2 – Permanente

1) Como definir o caminho temporário do JDK no Windows
Para definir o caminho temporário do JDK, você precisa seguir as seguintes etapas:
Abra o prompt de comando
Copie o caminho do diretório JDK / bin
Escreva no prompt de comando: defina path = copied_path
Por exemplo:
definir caminho = C: \ Arquivos de Programas \ Java \ jdk1.6.0_23 \ bin
Vamos ver na figura abaixo:



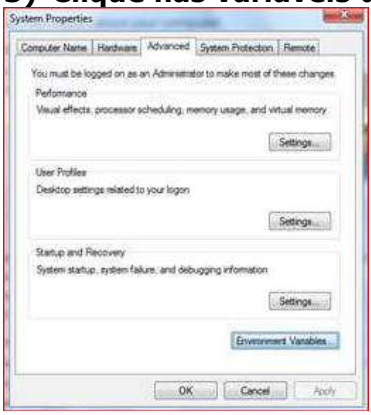
2) Como definir o caminho permanente do JDK no Windows

Para definir o caminho permanente do JDK, siga estas etapas:
Vá para Propriedades do MyComputer -> guia avançado -> variáveis de ambiente -> nova guia da variável do usuário -> caminho de gravação no nome da variável -> caminho de gravação da pasta bin no valor da variável -> ok -> ok -> ok
Por exemplo:

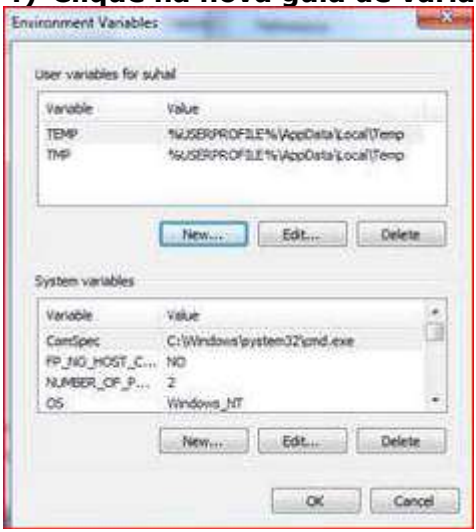
1) Vá para propriedades do MyComputer

2) Clique na aba avançada

3) Clique nas variáveis de ambiente

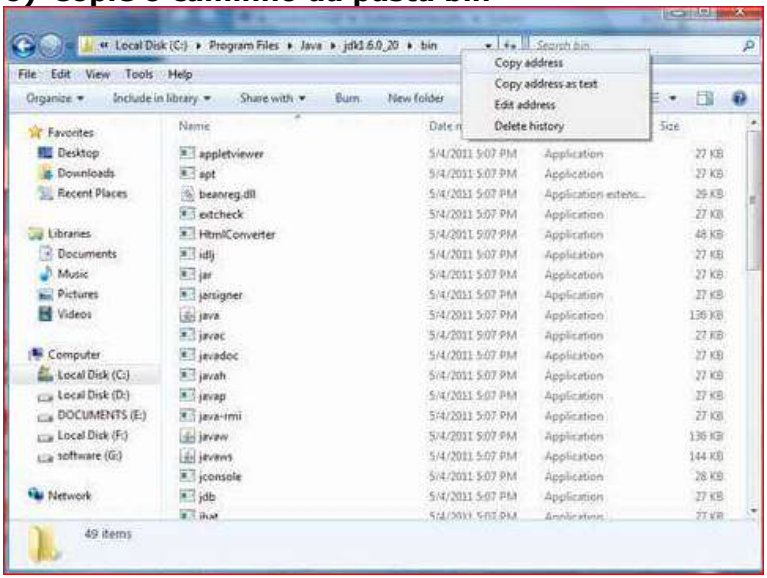


4) Clique na nova guia de variáveis do usuário

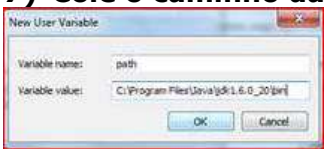


5) Escreva o caminho no nome da variável

6) Copie o caminho da pasta bin

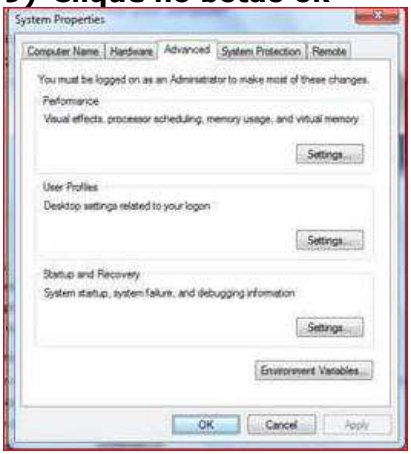


7) Cole o caminho da pasta bin no valor da variável



8) Clique no botão ok

9) Clique no botão ok



Agora seu caminho permanente está definido. Agora você pode executar qualquer programa de java a partir de qualquer unidade.

Configurando o caminho Java no Linux OS

Definir caminho no sistema operacional Linux é o mesmo que definir o caminho no sistema operacional Windows. Mas, aqui usamos a ferramenta de exportação em vez de definir. Vamos ver como definir o caminho no sistema operacional Linux:

```
exportar PATH = $ PATH: /home/jdk1.6.01/bin/
```

Aqui, instalamos o JDK no diretório inicial em Raiz (/ home).

FrameWorks

A programação contemporânea em Java, se utiliza bastante de frameworks. Já vi várias definições do que seja um framework. De maneira didática, vou arriscar uma definição minha: Framework é um conjunto de bibliotecas interligadas, que permitem uma solução mais simples de problemas comuns. Esta definição que fiz talvez não seja muito aceita por alguns profissionais. Vou tentar outra: Framework é um conjunto de classes que permite ao programador desenvolver de maneira diferente e tem o objetivo de resolver problemas comuns. Esta próxima definição será a última que vou arriscar: Framework é um conjunto de programas feitos por alguns programadores, que tinham o intuito de facilitar a vida de outros programadores. Esta última com certeza é a menos ortodoxa das definições. Talvez esteja incompleta. Acho melhor dar exemplos.

Digamos que o objetivo do programa seja uma página Web com o dizer "Olá mundo". A outra premissa deste exemplo, é que vamos

chamar a codificação em JSP de “Java puro” ou “Java raiz”. Veja... é só uma premissa OK?

Código de olá mundo em JSP:

```
<html>
<body>
<%
  out.print("Ol&aacute; mundo");
%>
</body>
</html>
```

Então podemos utilizar o conjunto de bibliotecas JSTL. Bom... talvez para alguns profissionais o JSTL não seja um framework. Mas serve de exemplo:

```
<html>
<body>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c" %>
<c:out value="Ol&aacute; mundo"/>
</body>
</html>
```

Uma outra maneira agora com o framework JSF

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">

<h:head />
<h:body>
   <h:outputText value="Ol&aacute; mundo" />
</h:body>
```

```
</html>
```

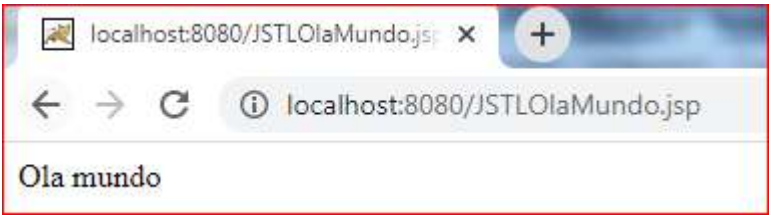
Neste livro está descrito o framework JSTL.

JSTL

Para programar com JSTL, é necessário que tenha o arquivo do JSTL. Em nosso caso uso o jstl-1.2.jar que pode ser baixado da internet. O JSTL tem comandos próprios que tem o intuito de facilitar o desenvolvimento. Assim, abaixo temos diversos exemplos da utilização do JSTL. E como toda linguagem começamos com o programa "Olá mundo".

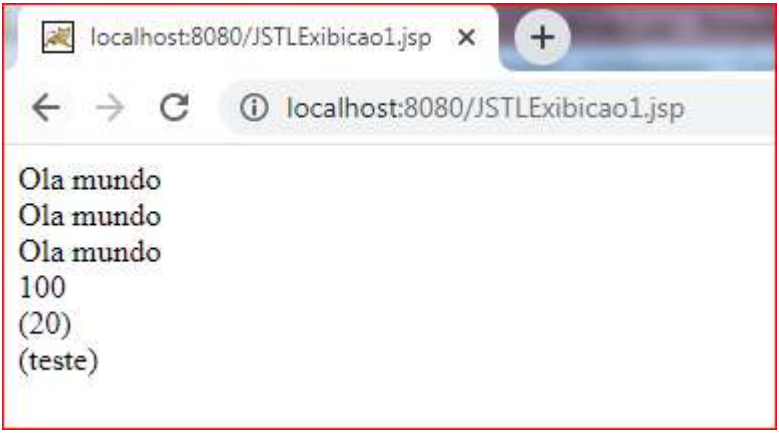
Exibição de mensagem

```
<html>
<head>
<meta charset="utf-8">
</head>
<body>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c" %>
<c:out value="Ola mundo"/>
</body>
</html>
```



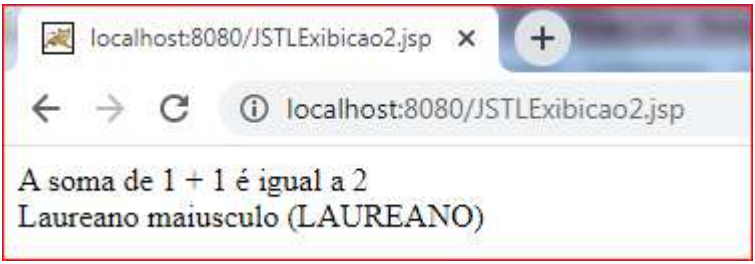
Uma outra maneria de fazer a saída do dado é colocar comandos de java da seguinte forma '${<COMANDOS>}' ou "${<COMANDO>}".

```
<html>
<head>
<meta charset="utf-8">
</head>
<body>
<%@ taglib
  uri="http://java.sun.com/jsp/jstl/core" prefix="c"
%>
<c:out value="Ola mundo"/><br>
<c:out value="${'Ola mundo'}"/><br>
<c:out value='${"Ola mundo"}'/><br>
<c:out value="${10 * 10}"/><br>
<c:out value="${''.format('(%d)',10 + 10)}"/><br>
<c:out value='${"".format("(%s)","teste")}'/>
</body>
</html>
```



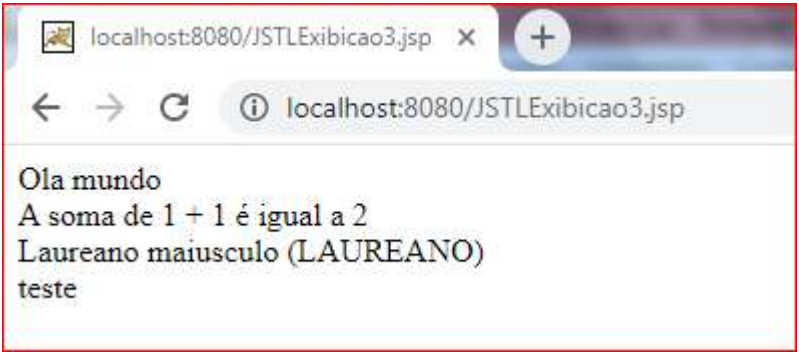
Outro tipo de exibição por expressões.

```
<html>
<body>
${"".format("A soma de 1 + 1 &eacute; igual a %d",1+1)}<br>
${"".format("Laureano maiúsculo (%s)","Laureano".toUpperCase())}
<br>
</body>
</html>
```



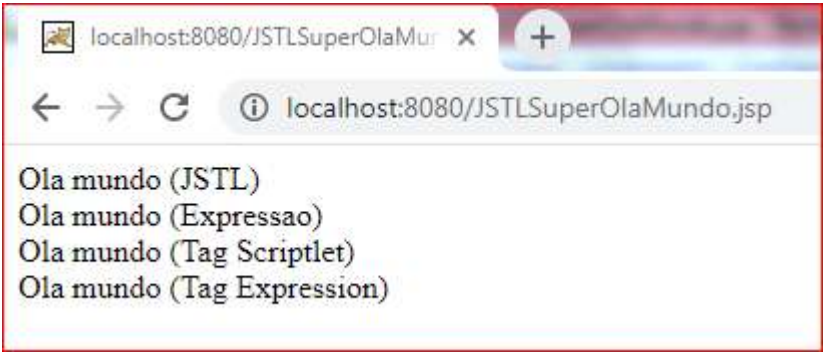
Podemos também utilizar vários tipos de exibição em um mesmo fonte de arquivo. No caso abaixo temos JSTL, expressões e JSP.

```
<html>
<body>
<!-- JSTL -->
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<c:out value="Ola mundo"/> <br>

<!-- EXPRESSÕES -->
${"".format("A soma de 1 + 1 &eacute; igual a %d",1+1)}<br>
${"".format("Laureano maiusculo (%s)","Laureano".toUpperCase())}
<br>

<!-- JSP -->
<%
out.print("teste");
%>

</body>
</html>
```



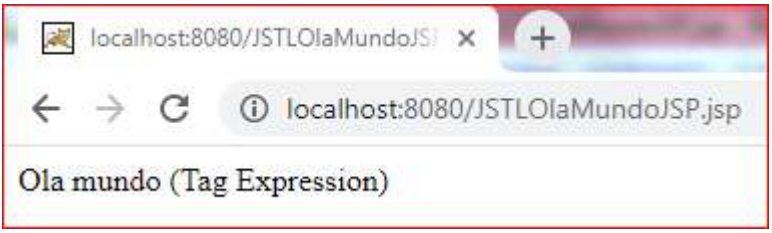
Assim segue abaixo 4 dos tipos possíveis de "Olá mundo" que podemos fazer com JSP

```
<html>
<body>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c" %>
<!-- JSTL -->
<c:out value="Ola mundo (JSTL)"/> <br>

<!-- Expressões -->
${"Ola mundo (Expressao)"}<br>

<!-- JSP TAG scriptlet -->
<%
  out.print("Ola mundo (Tag Scriptlet)<br>");
%>

<!-- JSP TAG Expression -->
<%= "Ola mundo (Tag Expression)<br>" %>

</body>
</html>
```



Também podemos utilizar dentro da propriedade "value" das tags JSTL a Tag Expression do JSP:

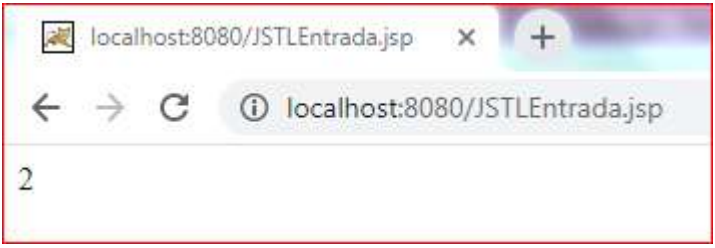
```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c" %>
<html>
<body>
<!-- JSTL -->
<c:out value='<%= "Ola mundo (Tag Expression)" %>' />
</body>
</html>
```



Entrada

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c" %>
```

```
<html>
<body>
<c:set var="Variavel" scope="session"
value="${1+1}"/>
<c:out value="${Variavel}"/>
</body>
</html>
```



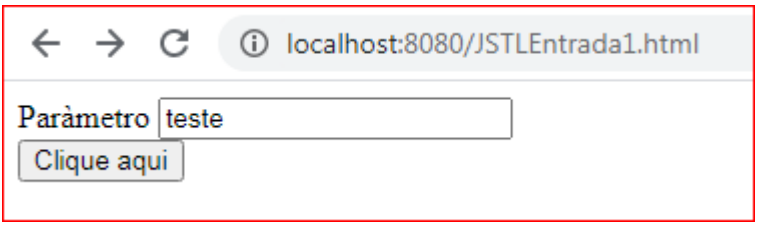
Exemplo com interação do usuário (JSTLEntrada1.html)

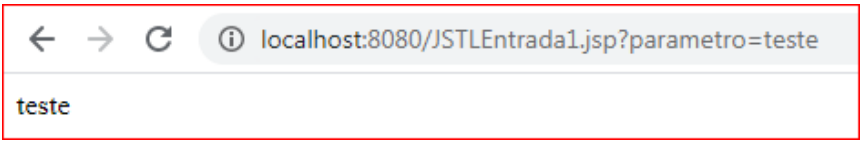
```
<html>
<body>
<form action="JSTLEntrada1.jsp">
Par&agrave;metro
<input type=text name="parametro"><br>
<input type=submit value="Clique aqui">
</form>
</body>
</html>
```

(JSTLEntrada1.jsp)

```
<html>
<body>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c" %>
<!-- JSTL -->
<c:set var="variavel" scope="session"
value="${param.parametro}"/>
<c:out value="${variavel}"/>
```

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





Operações Numéricas

Segue tabela com os sinais e funções das expressões numéricas.

Sinal	Funcionalidade
+	Possibilita a soma entre dois números
-	Possibilita a subtração entre dois números
*	Possibilita a multiplicação entre dois números
/	Possibilita a divisão entre dois números
%	Possibilita a recuperar o resto de uma divisão.
++[var]	Pré-soma. Onde [var] é a variável
[var]++	Pós-soma. Onde [var] é a variável
--[var]	Pré-subtração. Onde [var] é a variável
[var]--	Pós-subtração. Onde [var] é a variável
[var]+=[val]	Soma implícita. Onde [var] é a variável e [val] o valor
[var]-=[val]	Subtração implícita. Onde [var] é a variável e [val] o valor
[var]*=[val]	Multiplicação implícita. Onde [var] é a variável e [val] o valor

Sinal	Funcionalidade
[var]/=[val]	Divisão implícita. Onde [var] é a variável e [val] o valor
~	Subtrai 1 e inverte sinal. Exemplos: (10 vira -11) e (-10 vira 9)
>>	Regressão geométrica
<<	Progressão geométrica
pow	Possibilita a potenciação entre dois números
sqrt	Possibilita a raíz entre dois números
cbrt	Possibilita a raíz cúbica
()	Agrupador de expressão numérica
abs	Permite arredondar em valor absoluto
round	Permite arredondar para o valor fracionado mais próximo
ceil	Permite arredondar para cima os números fracionados
floor	Permite arredondar para baixo os números fracionados
abs	Retorna o valor absoluto positivo de um número
acos	Função retorna o arco cosseno do parâmetro x; isto é, o valor cujo cosseno é x. O parâmetro de entrada x deve estar entre -1 e 1. O valor de retorno é em radianos, entre 0 e pi
asin	Função retorna o arco seno do parâmetro x; isto é, o valor cujo cosseno é x. O parâmetro de entrada x deve estar entre -1 e 1. O valor de retorno é em radianos, entre 0 e pi
atan	Função retorna o arco tangente do parâmetro x; isto é, o valor cujo cosseno é x. O parâmetro de entrada x deve estar entre -1 e 1. O valor de retorno é em radianos, entre 0 e pi
atan2	Função que retorna tangente entre dois números
cos	Função retorna o cosseno de f em radianos. O valor de retorno é de -1 a 1.
cosh	Função retorna o cosseno hiperbólico de x, definido matematicamente como (ex + e-x) / 2. Se o valor de x for muito grande (se ocorrer um estouro), a função falhará.

Sinal	Funcionalidade
exp	A função retorna o valor de e - a base dos logaritmos naturais - aumentada para a potência x (ou seja, ex).
sin	Função retorna o seno de f em radianos. O valor de retorno é de -1 a 1.
sinh	Função retorna o cosseno hiperbólico de x, definido matematicamente como (ex + e-x) / 2. Se o valor de x for muito grande (se ocorrer um estouro), a função falhará.
tan	Função retorna a tangente de x em radianos.
tanh	Função retorna a tangente hiperbólica de x, definida matematicamente como seno (x) / cosseno (x).

Exemplo de adição entre dois números
Arquivo (JSTLopNumerica1.html):

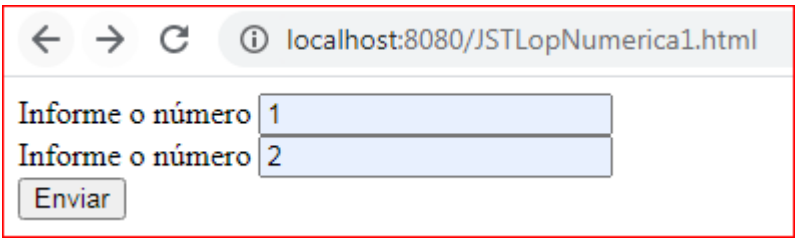
```
<html>
<body>
<form action="JSTLopNumerica1.jsp">
Informe o n&uacute;mero
 <input type="text" name="A"><br/>
Informe o n&uacute;mero
 <input type="text" name="B"><br/>
<input type="submit" value="Enviar">
</form>
</body>
</html>
```

Arquivo (JSTLopNumerica1.jsp)

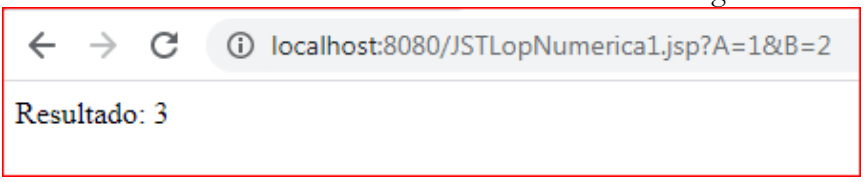
```
</html>
<html>
<body>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c" %>
<!-- JSTL -->
<c:set var="A" scope="session" value="${param.A}"/>
```

```
<c:set var="B" scope="session" value="${param.B}"/>
<c:set var="C" scope="session" value="${A+B}"/>
<c:out value="Resultado: "/>
<c:out value="${C}"/>
</body>
</html>
```

O sistema exibe a tela



O usuário informa os números e o sistema exibe a mensagem abaixo



Exemplo de adição com números fracionados: