

BUILDING DYNAMIC WEBSITE

with

Java Servlets Technology

www.catalunha.eng.br

[Start](#) | [Home](#)

Building Dynamic WebSite

Índice

1. [Título](#)
2. [Introdução](#)
 1. [Áreas de atuação e meios de informação na internet](#)
 2. [O que e a quem a internet tem exigido ?](#)
 3. [Ambiente de trabalho](#)
 4. [Processamento no Cliente e no Servidor](#)
 5. [Tecnologias disponíveis](#)
 6. [Desenvolvimento com DNA](#)
 7. [Configurações básicas de um PC](#)
3. [Objetivos](#)
4. Materiais e métodos
 1. Configurações necessárias
 1. Compiladores
 1. [Compilador PHP 4.0](#)
 2. [Compilador JDK](#)
 3. [Package JSDK](#)
 2. Servidor Web
 1. [iMatix Xitame](#)
 2. [Jakarta-TomCat](#)
 3. Banco de dados
 1. [MySQL](#)
 2. [Criar Dumps](#)
 3. [MS ACCESS](#)
 2. SQL
 1. [Características](#)
 2. [Sintaxe](#)
 3. [Tipos java.sql.Types e MySQL](#)
 3. PHP 4.0
 1. [Características](#)
 2. [Sintaxe](#)
 3. [Exemplo PHP 4](#)
 1. GerBDPHP
 1. GerBDPHP.html
 4. Java Technology
 1. [Características](#)
 2. [As Soluções da Sun](#)
 3. [O JavaBeans](#)
 4. [Sintaxe](#)
 5. Servlets
 1. [Características](#)
 2. [Sintaxe](#)

6. JSP
 1. [Características](#)
 2. [Sintaxe](#)
7. [Exemplos Servlets e JSP](#)
 1. GPSServlets
 1. GPSServlets.html
 2. GPSServlets.java
 2. SetGetBean
 1. SetGetBean.html
 2. SetGetBean.java
 3. GerBDJSP
 1. GerBDJSP.html
 2. GerBDJSP.java
5. [Conclusões](#)
6. [Bibliografia](#)

Catalunha - www.catalunha.eng.br

Introdução

[Ant.](#) | [Mapa](#) | [Prox.](#)

Desde 1970 quando a internet rompeu as fronteiras para o mercado internacional, vem representando um dos maiores, senão o maior, veículo de informação hipermídia de todos os tempos.

Devido a sua ampla atuação em diferentes ambientes computacionais e humanos, a interatividade e portabilidade vem exigindo dos profissionais de informática a utilização de ferramentas com as mesmas qualidades.

Desde meados de 1982 quando James Gosling (o pai do Java) escreveu sua primeira versão, não voltada para a internet, mas com as qualidades que ela exigia, tornou-se a ferramenta mais indicada para desenvolvimento neste ambiente multitarefa e multiusuário.

O presente trabalho pretende fornecer uma visão geral desta tecnologia com a utilização de Java Servlets technology para o Desenvolvimento Dinâmico de Sites para a Web.

Catalunha

Áreas de atuação e meios de informação

[Ant.](#) | [Mapa](#) | [Prox.](#)

- **Negócios**

farmacéutica, eletroeletrônica, alimentícia, varejo, metalúrgica, agricultura, têxtil, papel e celulose, finanças, automobilística, construção, administração, economia, contabilidade, . . .

- **Informação**

Jornais, revistas, museus, bibliotecas, livros, reuniões, celular, cursos on line, TV, shopping virtual, . . .

- **Lazer**

Esportes, turismo, jogos, bate papo, terapias, teatros, cinemas, restaurantes, TV e rádios interativos, . . .

Texto



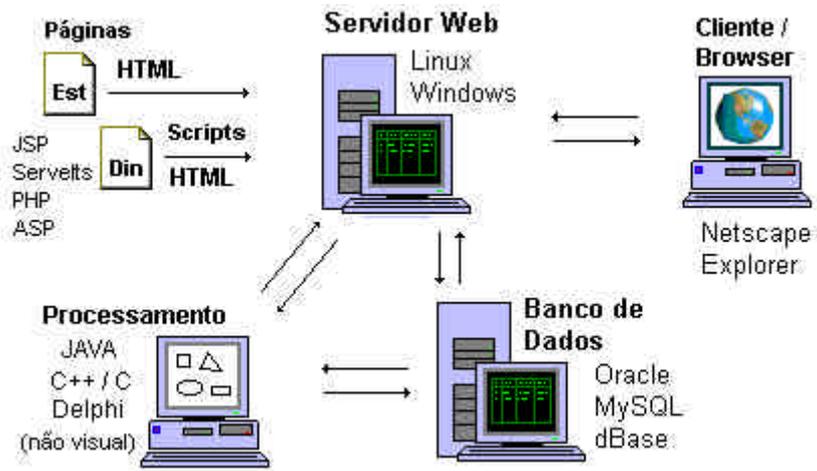
O que e a quem a internet tem exigido?

[Ant.](#) | [Mapa](#) | [Prox.](#)

<ul style="list-style-type: none">• Interatividade Hipermídia, vários tipos de clientes, fácil utilização, muitas informações . . .	WebDesigner
<ul style="list-style-type: none">• Velocidade Usuário exigente, quantidade de informação, número de clientes, níveis de acesso . . .	WebEngineer
<ul style="list-style-type: none">• Segurança Valor da informação, base de dados, valores financeiros, nível de processamento, backup, . . .	WebMaster

Ambiente de trabalho

[Ant.](#) | [Mapa](#) | [Prox.](#)



Tecnologias Cliente e Servidor

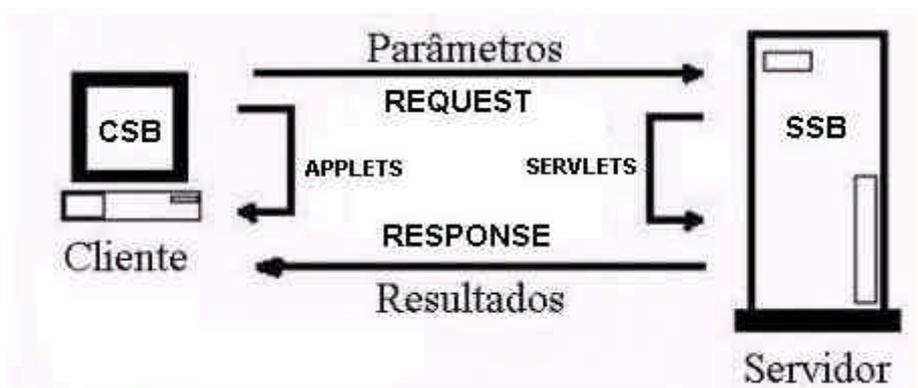
[Ant.](#) | [Mapa](#) | [Prox.](#)

Client Side Building - CSB

Processamento no cliente. Consome tempo de carregamento. Limitação de processamento do cliente. Linguagem dependente de browser. Insegurança por Plug-in. Linguagens disponíveis são JavaScript, Applets ou VBS.

Server Side Building - CSB

Processamento no servidor. Não interessa a linguagem. Retorna ao cliente apenas HTML. Grande capacidade de armazenamento e processamento dos servidores. Deve apresentar total segurança e capacidade de manipulação de informações. Linguagens disponíveis são JSP, Servlets, PHP, PERL, CGI, ASP.



Outros SSB: CGI - PERL, PHP, ASP e . . .

[Ant.](#) | [Mapa](#) | [Prox.](#)

PERL - CGI

São aqueles programas que rodam nos servidores e tem a capacidade de receber dados vindos da Internet, processar estes dados e enviar uma resposta ao cliente. Na maioria das vezes que você preenche um formulário online pela Internet estes dados são processados por um programa localizado no servidor e as respostas enviadas a você. Toda vez que o servidor Web recebe uma requisição CGI ele carrega um novo programa, executa-o, envia a resposta para o cliente e é finalizado. Estes programas são limitados pois o código html tem que vir imbutido dentro dos scripts e são pesados a nível de processamento do servidor.

ASP

Necessitam ser hospedadas no Servidor Web da Microsoft: Windows NT Server 4.0, Internet Explorer 4.0 e Option Pack 4.0 (Internet Information Server IIS). Permite a utilização de todos os comandos do Visual Basic, porém não permite acesso a periféricos e a manipulação de banco de dados, o que é feita através do objeto ADO (ActiveX Data Object). Trabalham com programação orientada a objeto. Possuem tratamento de sessão do servidor (baseado em cookies). Estrutura de programação diferente do C++. Muito popular por possuir uma configuração simples e muitas interfaces já prontas.

Distributed Network Applications Architecture

- **Camada de apresentação**

Seu papel é receber informações do usuário, enviá-las aos componentes de negócio para o processamento, receber os resultados fornecidos pelos componentes de negócio e finalmente, apresentar os resultados para o usuário.

- **Camada de lógica de negócio**

É o coração da aplicação. Nela, o processamento é especificado e as regras de negócios mantidas. Para que você possa entender, regras de negócios são a combinação da validação de dados, verificação de logins, busca em banco de dados e diversos algoritmos de transformação. É a lógica de negócios, colocada nos componentes, que une os ambientes cliente e as camadas de dados.

- **Camada de acesso a dados**

Essa camada permite o acesso a uma variedade de fontes de informações, incluindo dados relacionais e não relacionais, e uma interface de programação fácil de usar que é independente de ferramenta e de linguagem. No modelo DNA, o acesso aos dados é baseado em especificações de padrão aberto e trabalha com as principais plataformas de bancos de dados estabelecidas (SGBD).

Configurações necessárias em um PC

[Ant.](#) | [Mapa](#) | [Prox.](#)

Sistema Operacional

Linux
Windows

Compiladores

JDK e JRE
JSDK
PHP 4

Servidor Web

Apache (jakarta-tomcat)
JSWDK
Xitame

SGBD (SQL)

ORACLE
MySQL
MS ACCESS

Objetivos

[Ant.](#) | [Mapa](#) | [Prox.](#)

- 1 - Fornecer, mesmo que de uma forma superficial, um conhecimento atualizado sobre o Desenvolvimento Dinâmico de Sites para a Web;**
- 2 - Apresentar algumas ferramentas atualmente disponíveis no mercado para Desenvolvimento Dinâmico de Sites para a Web, para entendimento sobre as técnicas SSB e CSB;**
- 3 - Possibilitar a configuração de compiladores, SGBD e Web Server em PC;**
- 4 - Capacitar o desenvolvimento de aplicativos para a Web com interface dinâmica e utilização banco de dados remotos;**

Instalação do PHP4

[Ant.](#) | [Mapa](#) | [Prox.](#)

1. Acesso o endereço <http://www.php.net> e procure a opção download, siga as instruções até conseguir fazer o download do arquivo **php-4.0.0-Win32.zip** ou superior
2. Descompactar este arquivo na pasta c:\php4
3. Copiar o arquivo php-ini.dist para c:\windows
4. Renomear este arquivo para php-ini
5. Abrir o arquivo php-ini no item "Paths and Directories" e fazer as seguintes alterações:
 1. Procurar a linha **include_path = ; UNIX: "/path1:/path2" Windows: "\path1;\path2"** e alterar para **include_path = "c:\php4" ; UNIX: "/path1:/path2" Windows: "\path1;\path2"**
 2. Procurar a linha **extension_dir = ./ ; directory in which the loadable extensions (modules) reside** e alterar para **extension_dir = "c:\php4" ; directory in which the loadable extensions (modules) reside**
6. Ainda no arquivo php-ini no item ";Windows Extensions" fazer as seguintes alterações:
 1. retire o ponto e virgula, que é um comentário da linha **;extension=php_mysql.dll** que passará a ser **extension=php_mysql.dll**, assim este item passará a ser recolhido pelo inicializador do windows;
7. Copiar os arquivos MSVCRT.DLL e PHP4TS.DLL. da pasta de origem c:\php4 para a pasta c:\windows\system
8. Os passos a seguir são opcionais e dependem de cada máquina
 1. Acionar a barra de tarefas Start\Run\regedit, este comando mostrará o editor de registros do windows;
 2. Siga o caminho [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\w3sv Map]] se for necessário crie a chave que estiver faltando usando o Edit\New\Key;
 3. Logo em seguida acione Edit\New\String Value e preencha **Value name** com **.php** e **Value data** com **c:\php4\php4isapi.dll**
9. Pronto o PHP4 já esta configurado;

Instalação do Java Developer Kit 1.3

[Ant.](#) | [Mapa](#) | [Prox.](#)

1. Acesse o endereço <http://www.sun.com> e faça download do **JDK1.3.exe** ou superior, este é o pacote principal para desenvolvimento java;
2. Click no pacote e siga as instruções de instalação, instale de preferencia no diretorio raiz. Exemplo **c:\jdk1.3**;
3. Acessar ao arquivo Autoexec.bat e inclua as seguinte configuração e o seu JDK estará configurado;

```
rem *** Inicio Config JDK 1.3 ***  
SET PATH=%PATH%;C:\JDK1.2.2\BIN  
SET CLASSPATH=%CLASSPATH%;c:\jdk1.3\lib\tools.jar;  
rem *** Fim Config JDK 1.3***
```

Instalação do Java Server Developer Kit 2.1

[Ant.](#) | [Mapa](#) | [Prox.](#)

Considerando que a instalação do JDK 1.3 já foi realizada:

1. Acesse o endereço <http://www.javasoft.com> e faça download do **jdk2_1-win.zip** ou superior, este é o pacote principal para desenvolvimento de java servlets que ainda vem separado do JDK;
2. Descompacte este arquivo no diretório raiz do JDK, ficando **c:\jdk1.2.2\jdk2.1**
3. Acessar ao arquivo Autoexec.bat e incluir a seguinte configuração e o seu JSDK estará habilitado

```
rem *** Inicio Config JSDK2.1 ***
SET CLASSPATH=%CLASSPATH%;c:\jdk1.3\jdk2.1\servlet.jar
rem *** Fim Config JSDK2.1 ***
```

Instalação do Web Server Xitame

[Ant.](#) | [Mapa](#) | [Prox.](#)

1. Acesso o endereço <http://www.tucows.com> e procure a opção Web Servers, siga as instruções até conseguir fazer o download do arquivo **bw3224d7.exe** ou superior
2. Descompactar o arquivo em uma pasta c:\temp
3. Crie uma pasta com o nome **c:\http** que será utilizada posteriormente
4. Execute o arquivo Setup.exe
5. Informe os campos devidamente, anotando o login e senha para posterior administração
6. Clique na barra de tarefas do windows Start\programs\Internet toos\Xitame
7. Clique em Setup e surgirá uma janela no seu navegador, escrita Xitami administrator
8. Clique em configuration, surgirá uma tela com alguns campos a preencher
9. No campo "Main HTML directory: " preencha com o nome da pasta que você criou anteriormente, **c:\http**, esta será a pasta residente de todas as suas páginas *.html e *.php
10. Logo abaixo o campo "Default HTML page: " estará preenchido com "index.htm" e "default.htm" coloque mais uma opção "index.php"; pois quando você acessar ao navegador ele iniciará automaticamente a página com estes nomes
11. Os demais campos deixar sem preencher, e retornar ao topo da página;
12. Procure no campo superior a esquerda um link com o nome "Filters", de um clique
13. Preencha o campo "File extension (.xxx):" com **.php**, e logo em seguida o campo "Filter command or script:" com **c:\php4\php.exe** ; Isto quer dizer que quando o navegador encontrar um arquivo php ele irá buscar na pasta informada a sua versão compilada para ser executada no browser;
14. Salve as alterações e saia do browser;
15. Pronto o Web Server Xitame já esta configurado.

Instalação do jakarta-tomcat Web Server

[Ant.](#) | [Mapa](#) | [Prox.](#)

Considerando que a instalação do JDK 1.3 já foi realizada:

1. Acesse o endereço <http://www.sun.com> e faça download do **jakarta-tomcat.zip** versão mais recente, este é o servidor muito utilizado para suporte ao java servlet
2. Descompacte este arquivo na pasta raiz do JDK, ficando **c:\jdk1.2.2\jakarta-tomcat**
3. Para iniciar o servidor basta clicar no arquivo **c:\jdk1.2.2\jakarta-tomcat\bin\startup.bat**
4. Caso apareça uma mensagem "Out of environment space" acesse ao item properties da janela do DOS e na aba memory no item initial environment altera para a capacidade máxima
5. Para encerrar o servidor basta clicar no arquivo **c:\jdk1.2.2\jakarta-tomcat\bin\shutdown.bat**
6. Para configurar uma pasta de trabalho inicie o servidor
7. Digite <http://localhost:porta/admin>. Exemplo: <http://127.0.0.1:8080/admin>
8. No item Add Context no campo Path: digite **/curso** e no campo Document base: digite **webapps/curso**
9. Click em Add Context e logo em seguida em View Alls Contexts para conferir a inclusão
10. Vá para a pasta **c:\jdk1.2.2\jakarta-tomcat\webapps**
11. Crie uma pasta com o nome **curso**, em seguida a subpasta **Web-inf** e nesta a pasta **classes**
12. O caminho ficou **c:\jdk1.2.2\jakarta-tomcat\webapps\curso\Web-inf\classes**
13. Na pasta **c:\jdk1.2.2\jakarta-tomcat\webapps\curso** ficarão as pastas html e JSP
14. Na pasta **c:\jdk1.2.2\jakarta-tomcat\webapps\curso\Web-inf\classes** ficarão as package e class java

Instalação do SGBD MySQL

[Ant.](#) | [Mapa](#) | [Prox.](#)

1. Acesso o endereço <http://www.mysql.org> e siga as instruções até conseguir fazer o download dos arquivos **mysql-shareware-3.22.34-win.zip** e **MySQLWinAdmn-104.zip** ou superior
2. Descompactar o arquivo **mysql-shareware-3.22.34-win.zip** em uma pasta c:\temp
3. Execute o arquivo Setup.exe e informe os campos devidamente
4. Descompactar o arquivo **MySQLWinAdmn-104.zip** na pasta de instalação já criada .\mysql\bin
5. Para habilitar o MySql siga a pasta .\mysql\bin e de um duplo clique no arquivo **mysqld-shareware.exe** , não acontecerá nada pois as operações são a nível de sistema, mas o MySql estará habilitado;
6. Para utilizar o MySql siga a pasta .\mysql\bin e de um duplo clique no arquivo **MySQLWinAdmn.exe** , surgirá uma interface de manipulação de banco de dados MySql
7. Esta pronta a instalação

Criar arquivo SQL padrão - Dumps

[Ant.](#) | [Mapa](#) | [Prox.](#)

Qualquer aplicativo que utilize estrutura SQL como gerenciador de banco de dados precisa converter a base de dados para o formato SQL padrão, para que qualquer outro SGBD possam acessá-los.

Considerando que a instalação do MySQL já foi realizada

1. Crie uma pasta temporária no seguinte caminho `.\mysql\bin\` ficando `.\mysql\bin\temp`
2. Abrir o prompt do DOS e seguir até a pasta criada `.\mysql\bin\`, estas operações a seguir não aceitam comandos pelo windows, devendo serem feitos pelo DOS
3. No prompt do DOS digite o nome do arquivo conversor **mysqldump** com os seguintes parâmetros `--tab= [nome_do_diretorio_criado]` `[nome_da_base_de_dados]` . Como Exemplo: `mysqldump --tab= temp cliente`
4. O sistema criará dois arquivos com o nome da base de dados e com a extensão ***.sql** e ***.txt** com a base de dados no formato SQL universal
5. É só compactar estes dois arquivos e enviar ao provedor

Configuração do ODBC do windows

[Ant.](#) | [Mapa](#) | [Prox.](#)

Considerando que a instalação do MC ACCESS já foi realizada

1. Acione no painel de Controle do Windows 95/98 um ícone para ODBC(32 bits).
2. clique em System DSN
3. Em seguida pressione “Add” para fazer aparecer “Create new data source”
4. Realce o “Microsoft Access driver ” e pressione “finish”
5. Isso faz aparecer um painel denominado “ODBC Microsoft Access 97 Setup”
6. No campo Data Source Name digite um nome para a fonte de dados
7. Em seguida pressione “Select” e selecione o arquivo de base de dados ja criada pelo MS ACCESS, clique em “OK”
8. Se o Database aponta para o diretório correto você associou com êxito o nome da fonte de dados e o ODBC de 32 bits
9. Pressione “OK” para concluir a configuração

Características da SQL

[Ant.](#) | [Mapa](#) | [Prox.](#)

A sigla SQL são as iniciais de Structured Query Language que significa uma linguagem estruturada em perguntas, que se dedica exclusivamente ao gerenciamento de banco de dados.

É simples e fácil de usar. Ela se opõe a outras linguagens no sentido de que uma consulta SQL especifica a forma do resultado e não o caminho para chegar a ele.

Ela é um linguagem declarativa em oposição a outras linguagens procedurais. Isto reduz o ciclo de aprendizado daqueles que se iniciam na linguagem.

Em meados de 1986 a IBM adotou sua estrutura como padrão em seus projetos; isto vez com que a linguagem fosse rapidamente difundida e se tornasse componente essencial nos SGBD (Sistema Gerenciador de Banco de Dados).

Sintaxe básica SQL

[Ant.](#) | [Mapa](#) | [Prox.](#)

Uma API baseada na JDBC tem três níveis de suporte a SQL: mínimo, básico e estendido. Neste item estudaremos a gramática mínima necessária de uma API compatível com a JDBC, seguindo as regras de sintaxe:

- Itens entre [] são opcionais;
- Itens entre < > são obrigatórios;
- Reticências . . . indicam elementos que podem ser repetidos;
- Símbolo barra | indica componentes alternativos;
- As chaves { } agrupam componentes alternados.

Considera-se para título de exemplo que se esta trabalhando com uma tabela chamada **Aluno.tbl** com os seguintes campos: **Nome** do tipo texto, **AreaGraduação** do tipo inteiro e **Nota** do tipo ponto flutuante. Os comandos estão em ordem alfabética:

Comando: ALTER TABLE

Função: Adicionar uma ou mais colunas à estrutura da tabela na sua fonte de dados.

Estrutura: ALTER TABLE <nome_tabela> ADD <nome_coluna>
<tipo_coluna> | ADD (<nome_coluna> <tipo_coluna> [,
<nome_coluna><tipo_coluna>]...)

Exemplo: ALTER TABLE Aluno ADD Email texto

Comando: CREATE TABLE

Função: Cria fisicamente uma tabela no local descrito pelo DataBaseName

Estrutura: CREATE TABLE <nome_tabela> (<nome_coluna> <tipo_coluna> [,
<nome_coluna> <tipo_coluna>]...)

Exemplo: CREATE TABLE Aluno (Nome texto)

Comando: DELETE

Função: Elimina as linhas que correspondem à condição de busca

Estrutura: DELETE FROM <nome_tabela> WHERE <nome_coluna>
<condicao> <valor>

Exemplo: DELETE FROM Aluno WHERE AreaGraduação = 1

Comando: DROP TABLE

Função: Elimina uma tabela

Estrutura: DROP TABLE <nome_tabela>

Exemplo: DROP TABLE Aluno

Comando: INSERT INTO

Função: Insere uma única linha em uma tabela

Estrutura: INSERT INTO <nome_tabela>(<nome_coluna> [, <nome_coluna>]
...)] VALUES (<valor> [,<valor>]....)

Exemplo: INSERT INTO Aluno(nome,fone) VALUES ('Catalunha','0000000')

Comando: SELECT

Função: Recupera um conjunto selecionado de colunas a partir de uma ou mais tabelas

Estrutura: SELECT [ALL|DISTINCT] <nome_coluna>| * FROM
<nome_tabela> WHERE <condição>

Exemplo: SELECT * FROM Aluno

Comando: UPDATE

Função: Atualiza (edita) as colunas de uma tabela

Estrutura: UPDATE <nome_tabela> SET <nome_tabela> = <valor>| NULL
[,<nome_tabela> = <valor> | NULL] WHERE <condição>

Exemplo: UPDATE Aluno SET Nome = 'Catalunha' WHERE AreaGraduação = 1

As possíveis cláusulas utilizadas

ALL : recupera as linhas do conjunto de resultados independentes de duplicidade. Utilizada geralmente no comando WHERE.

DISTINCT : elimina as linhas duplicadas das linhas do conjunto. Utilizada geralmente no comando WHERE.

WHERE : Permite montar um filtro e exige uma lista de condições de busca

AND : Permite agrupar condições. Utilizada geralmente no comando WHERE.

OR : Exclui um conjunto de condição de outro. Utilizada geralmente no comando WHERE.

LIKE : É semelhante à comparação com a exceção de que o lado direito deve ter um

modelo de valor. Uma cadeia de caracteres que pode conter qualquer caractere.
Exemplo: `WHERE <nome_coluna> LIKE 'CA%'` . Isto lista todos os registros que tenham um nome com as duas primeiras letras CA (Catalunha, Carlos, etc)

NULL : Selecciona registros para os quais temos alguma coluna nula (NULL).
Utilizada geralmente no comando `WHERE`.

ORDER BY : Esta cláusula permite determinar a ordem dos resultados. ASC (ascendente) ou DESC (descendente) além de poder especificar o nome das colunas a ser ordenadas. Utilizada geralmente no comando `SELECT`

GROUP BY : É utilizada para fazer agrupamento de valores. Você pode escolher a coluna a ser utilizada. Utilizadas geralmente no comando `SELECT`

Tipos de dados java.sql.Types e MySQL

[Ant.](#) | [Mapa](#) | [Prox.](#)

Os diferentes SGBD (Sistemas Gerenciadores de Banco de Dados) possuem seus próprios tipos de dados. No caso do MySQL alguns tipos permitem uma ampla manipulação de dados e são aceitos pelos padrões java.sql.Types. Os mesmos podem ser vistos na tabela a seguir:

Tipo de Campo	Abrangência	Aceita Default	Valor Default
_BIGINT	bigint(255)	DEFAULT	'0',
_BLOB	blob,		
_CHAR	char(1)	DEFAULT	'9',
_DATE	date	DEFAULT	'0000-00-00',
_DATETIME	datetime	DEFAULT	'0000-00-00 00:00:00',
_DECIMAL	decimal(255,30)	DEFAULT	'0.00000000000000000000000000000000',
_DOUBLE	double(255,30)	DEFAULT	'0.00000000000000000000000000000000',
_FLOAT	float(255,30)	DEFAULT	'0.00000000000000000000000000000000',
_FLOAT_4	float(10,2)	DEFAULT	'9999.00',
_FLOAT_8	double(16,4)	DEFAULT	'9999.0000',
_INT	int(255)	DEFAULT	'9999',
_LONGBLOB	longblob,		
_LONGTEXT	bigint(255)	DEFAULT	'9999',
_MEDIUMBLOB	mediumblob,		
_MEDIUMINT	mediumint(255)	DEFAULT	'9999',
_MEDIUMTEXT	mediumtext,		
_SMALLINT	smallint(255)	DEFAULT	'0',
_TEXT	text,		
_TIME	time	DEFAULT	'00:00:00',
_TIMESTAMP	timestamp(14),		
_TINYBLOB	tinyblob,		
_TINYINT	tinyint(255)	DEFAULT	'0',
_TINYTEXT	tinytext,		
_VARCHAR	varchar(255)	DEFAULT	'9999',
_YEAR	year(4)	DEFAULT	'0000'

Outras clausulas possíveis a um campo são: **AUTO_INCREMENT**, **PRIMARY**, **KEY**

Como a linguagem Java é multiplataforma uma padronização de alguns tipos foi definida para permitir uma melhor utilização por diferentes tipos de usuários. Os tipos são apresentados na tabela a seguir: (para mais informações sobre esta tabela gentileza consultar a página oficial da SUN ou o manual do JDK)

ARRAY	BIGINT	BINARY	BIT	BLOB	CHAR
CLOB	DATE	DECIMAL	DISTINCT	DOUBLE	FLOAT
INTEGER	JAVA_OBJECT	LONGVARBINARY	LONGVARCHAR	NULL	NUMERIC
OTHER	REAL	REF	SMALLINT	STRUCT	TIME
TIMESTAMP	TINYINT	VARBINARY	VARCHAR		

Características do PHP - "Personal Home Page"

[Ant.](#) | [Mapa](#) | [Prox.](#)

Baseada em C++;

PHP fica embutido no script HTML;

SGBD (dBase, Interbase, mSQL, mySQL, Oracle, Sybase, PostgreSQL e vários outros);

Protocolos como IMAP, SNMP, NNTP, POP3 e HTTP;

Seu módulo de compilação é pequeno e rápido;

Criada no outono de 1994 por Rasmus Lerdorf, aperfeiçoado pelo projeto Zend 2000 - PHP4;

Sintaxe básica do PHP

[Ant.](#) | [Mapa](#) | [Prox.](#)

- Elementos de script
- Manipulação de dados
- Estruturas de controle
- Declaração de funções
- Definindo objetos
- Passagem de parâmetros
- Conexão com o servidor
- Seleção do banco de dados
- Execução de query SQL
- Trabalha com o resultado

Edição em HTML:

`<?php comandos ? >`

`<script language="php" > comandos </script >`

`<? comandos ?>`

`<% comandos %>`

`// ou /* */`

Manipulação de dados:

Tipos variáveis: Variant

*Operador aritmético: + , - , * , / , %*

Operador string: . (ponto)

*Operador de atribuição: = , += , -= , *= , /= , %= , .=*

Operador lógico: and , or , xor , ! , & , && , ||

Operador comparação: == , != , < > , <= , >=

Declaração de variáveis: \$catalunha e \$CATALUNHA

Estruturas de controle:

(condição) ? (se_verdadeiro) : (se_falso)

```
if ( condição ) {  
    bloco 1  
} else {  
    bloco 2  
}
```

```
switch ( variável ) {  
    case 0: {  
        bloco 1  
        break; }  
    ...  
    case N: {  
        bloco N  
        break; }  
    default : {
```

```
    bloco  
    break;}  
}
```

```
while (condição) {  
    bloco  
};
```

```
do {  
    bloco  
}while (condição);
```

```
for (inicialização ; condição ; incremento) {  
    bloco  
};
```

Declaração de funções:

```
function nome_da_função([arg1, arg2, . . .]) {  
    bloco;  
    return Valor;  
}
```

```
$Var1 = "Valor";  
function Teste() {  
    global $Var1;  
    print $Var2;  
  
}  
Teste();
```

Definindo objetos:

```
class Nome_da_classe {  
    var $variavel1;  
    var $variavel2;  
    function funcao1 ($parametro) {  
        bloco;  
    }  
}
```

```
$variavel = new $nome_da_classe;
```

```
$variavel -> funcao1(Valor)
```

Passagem de parâmetros:

```
Arquivo.php?Parâmetro1=Valor& . . .  
&ParâmetroN=ValorN
```

Conexão com o servidor

```
$conexao = mysql_connect("Local", "Usuario",  
"Senha");
```

Seleção do banco de dados

```
mysql_select_db("DataBase", $conexao);
```

Execução de query SQL

```
$cria = "CREATE TABLE exemplo ( nome  
CHAR(40))";
```

```
mysql_query($cria, $conexao);
```

```
$consulta = "SELECT nome FROM exemplo";
```

```
$resultado = mysql_query($consulta, $conexao);
```

Trabalha com o resultado

```
print "Nome: " . mysql_result($resultado,0,"nome");
```

Páginas Dinâmicas com PHP 4

[Ant.](#) | [Mapa](#) | [Prox.](#)

1. Declaração de variáveis, data do sistema e navegador

1. [Hello.php](#) ([Hello.php](#))

2. Gerenciamento de banco de dados com PHP 4

1. [GerBDPHP.php](#) ([GerBDPHP.php](#))

X-Powered-By: PHP/4.0.0 Content-type: text/html

EMPRESA X

Administração OnLine

[Primeiro](#) [Anterior](#) [Próximo](#) [Último](#) [Inserir](#) [Apagar](#) [Atualizar](#)

Dados cadastrais

Nome:

Emai:

```

<html>
<head>
<title>Cip-Con/Objetivo - Administra&ccedil;&atilde;o OnLine</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
</head>
<?
////////// FUNCOES //////////
function AtualizaCamposPag($AtualReg){
    global $resultado;
    global $Nome,$Email;
    $Nome = mysql_result($resultado,$AtualReg,"Nome");
    $Email = mysql_result($resultado,$AtualReg,"Email");
}
?>

<?
////////// ALGORITIMO //////////
$conexao = mysql_connect("localhost", "catalunha", "");
mysql_select_db("dbcatalunha", $conexao);
$query = "SELECT * FROM Cliente ORDER BY Indice";
$resultado = mysql_query($query,$conexao);
$num_reg = mysql_num_rows($resultado);

$botao = "Inserir";
switch($F){
    case "_FIRST":{
        if ($num_reg > 0){
            $record = 0;
            AtualizaCamposPag($record);
            $botao = "Inserir";
            $delete = "Cliente.php?F=_DELETE&Record=" . "$record";
            $action = "Cliente.php?F=_SAVE&Record=" . "$record";
        }
        break;
    }
    case "_PRIOR":{
        if ($num_reg > 0){
            $num_reg = mysql_num_rows($resultado)-1;
            $record--;
            if ($record < 0){
                $record = $num_reg;
            }
            AtualizaCamposPag($record);
            $botao = "Inserir";
            $delete = "Cliente.php?F=_DELETE&Record=" . "$record";
            $action = "Cliente.php?F=_SAVE&Record=" . "$record";
        }
        break;
    }
    case "_NEXT":{
        if ($num_reg > 0){
            $num_reg = mysql_num_rows($resultado)-1;
            $record++;
            if ($record > $num_reg){
                $record = 0;
            }
            AtualizaCamposPag($record);
            $botao = "Inserir";
            $delete = "Cliente.php?F=_DELETE&Record=" . "$record";
            $action = "Cliente.php?F=_SAVE&Record=" . "$record";
        }
    }
}

```

```

    break;
}
case "_LAST":{
    if ($NumReg > 0){
        $NumReg = mysql_num_rows($resultado)-1;
        AtualizaCamposPag($NumReg);
        $Record = $NumReg;
        $Botao = "Inserir";
        $Delete = "Cliente.php?F=_DELETE&Record=" . "$Record";
        $Action = "Cliente.php?F=_SAVE&Record=" . "$Record";
    }
    break;
}
case "_NEW":{
    $Botao = "Confirmar";
    $Action = "Cliente.php?F=_INSERT";
    break;
}
case "_INSERT":{
    $Inserir = "INSERT INTO Cliente";
    $Inserir .= "(Nome,Email)";
    $Inserir .= "VALUES";
    $Inserir .= "(";
    $Inserir .= ("\"$tfNome\"\",\"$tfEmail\"");
    mysql_query($Inserir,$conexao);

    $consulta = "SELECT * FROM Cliente ORDER BY Indice";
    $resultado = mysql_query($consulta, $conexao);
    $NumReg = mysql_num_rows($resultado)-1;
    AtualizaCamposPag($NumReg);
    $Record = $NumReg;
    $Botao = "Inserir";
    $Delete = "Cliente.php?F=_DELETE&Record=" . "$Record";
    $Action = "Cliente.php?F=_SAVE&Record=" . "$Record";
    break;
}
case "_DELETE":{
    if ($NumReg > 0){
        $Indice = mysql_result($resultado,$Record,"Indice");
        mysql_query("DELETE FROM Cliente WHERE Indice=$Indice",$conexao);
        if ($NumReg > 0){
            $consulta = "SELECT * FROM Cliente";
            $resultado = mysql_query($consulta,$conexao);
            $NumReg = mysql_num_rows($resultado)-1;
            $Record++;
            if ($Record > $NumReg){
                $Record = 0;
            }
            AtualizaCamposPag($Record);
            $Botao = "Inserir";
            $Delete = "Cliente.php?F=_DELETE&Record=" . "$Record";
            $Action = "Cliente.php?F=_SAVE&Record=" . "$Record";
        }else{
            $Botao = "Inserir";
            $Delete = "Cliente.php";
            $Save = "Cliente.php";
        }
    }
    break;
}
case "_SAVE":{
    if ($NumReg > 0){

```

```

    $Indice = mysql_result($resultado,$Record,"Indice");
    $Atualizar = "update Cliente set ";
    $Atualizar .= "nome='$stfNome',Email='$stfEmail'";
    $Atualizar .= " where Indice = $Indice";
    mysql_query($Atualizar,$conexao);
    $consulta = "SELECT * FROM Cliente ORDER BY Indice";
    $resultado = mysql_query($consulta, $conexao);
    AtualizaCamposPag($Record);
    $Botao = "Inserir";
    $Delete = "Cliente.php?F=_DELETE&Record=" . "$Record";
    $Action = "Cliente.php?F=_SAVE&Record=" . "$Record";
}
break;
}
default :{
    $Nome=$Email="";
    $Nota="";
    $Record = -1;
    $Delete = "Cliente.php";
    $Action = "Cliente.php?F=_INSERT";
    $Botao = "Inserir";
    break;
}
}
?>

```

```

<body bgcolor="#FFFFFF">
<p align="center"><b><font color="#FF0000" size="6">EMPRESA X</font><
<b><font color="#0000FF" size="3">Administra&ccedil;&atilde;o OnLin
<form method="post" action="<?print "$Action" ?>" enctype="multipart/
<table border="0" align="center">
<tr align="center" valign="top">
<td>
<div align="center">
<input type="button" name="Button" value="Primeiro" onclick:
</div>
</td>
<td>
<div align="center">
<input type="button" name="Button" value="Anterior" onclick:
</div>
</td>
<td>
<div align="center">
<input type="button" name="Submit2" value="Pr&oacute;ximo" (
</div>
</td>
<td>
<div align="center">
<input type="button" name="Submit3" value="&Uacute;ltimo" o
</div>
</td>
<td>
<div align="center"> <? if ($Botao == "Inserir") { ?>
<input type="button" name="btInserir" value="Inserir" oncli
<? }else{ ?>
<input type="submit" name="btConfirmar" value="Confirmar" >
<? } ?> </div>
</td>
<td>
<div align="center">

```

```

        <input type="button" name="Submit4" value="Apagar" onclick=
    </div>
</td>
<td>
    <input type="submit" name="Submit5" value="Atualizar" >
</td>
</tr>
</table>
<b>Dados cadastrais</b><br>
<table align="center">
<tr>
<td width="64">
    <div align="right">Nome: </div>
</td>
<td width="346">
    <input type="text" name="tfNome" size="40" value="<?print "$N
</td>
</tr>
<tr>
<td width="64">
    <p align="right">Emai: </p>
</td>
<td height="27" width="346">
    <input type="text" name="tfEmail" size="40" value= <?print "$!
</td>
</tr>
</table>
<div align="left"></div>
</form>
</body>
</html>

<?
    mysql_close($conexao);
?>

```

Características da Linguagem Java

[Ant.](#) | [Mapa](#) | [Prox.](#)

- Uma linguagem de Programação Orienta a Objeto
- Independente de Plataforma
- Não utiliza ponteiros
- Tem alocação dinâmica de memória em tempo de execução
- Utiliza Multithreading
- Mesma sintaxe usada em diferentes tecnológicas
- Alguns usuários java
 - Nokia (telecomunicações via celular)
 - Banco Itaú
 - Hong Kong Telecom (rede de TV interativa)
 - J. P. Morgan (banco de investimentos)
 - Fannie Mac (maior empresa de hipotecas dos EUA)
 - CSX (maior companhia ferroviária dos EUA)

Java™ - Products & APIs

[Ant.](#) | [Mapa](#) | [Prox.](#)

- **Java Development Kit - JDK**
- **Java Runtime Environment - JRE**
- **Java Foundation Classes - JFC**
- **Java Server Web Development Kit - JSWDK**
- **Java Servlet Development Kit - JSDK**
- **Java DataBase Connecty - JDBC**
- **Enterprise JavaBeans - JBeans**
- **Java Remote Method Invocation - JRMI**
- **Java Naming Directory Interface - JNDI**
- **Java Interface Definition Language - JIDL**
- **Java Message Service - JMS**
- **Java Operation System - JOS**
- **Java Electronic Commerce - JEC**
- **InfoBus**
- **Java PC**
- **Java Help**
- **Java Card**
- **Java Blend**
- **Java Mail**
- **Java Media APIs**
- **Embedded Java**
- **Java Check**
- **Java Plug-in**
- **HotSpot**

JDK

O JDK1.3 é a última versão do KIT de desenvolvimento Java. Contém as ferramentas para compilação do código Java (javac), documentação (javadoc), visualizador de applets (appletviewer) e outros aplicativos. Esta versão traz melhorias na performance e uma maior funcionalidade. Inclui novos pacotes, alguns dos que serão discutidos abaixo, como Internacionalização, Segurança, JavaBeans, JAR, RMI, JDBC, JNI, Object Serialization e melhorias nas classes AWT, IO, NET e Math. O JDK1.3 traz as mais novas tecnologias prometidas pela Sun e que podem concretizar a Java como uma revolução não só em linguagem de programação mas como uma plataforma completa. Algumas das funcionalidades incluídas nesta versão são os pacotes Java 2D, Accessibility, Drag and Drop, Application Services, Extensions Framework, Collections, Input Methods, Version Identification, Weak References, Java™ interface definition language (IDL), Java Virtual Machine Debugger Interface (JVMDI), o Java Servlet standard extension, e Javadoc Doclets. A inclusão de novas tecnologias como JavaBeans, Invocação de métodos remotamente (RMI), Object Serialization, Java Sound, Java Archive (JAR), Java Native Interface (JNI), Java Security e melhorias na performance também são oferecidos nesta nova versão.

JRE

O JRE tem a função de executar aplicações Java standalone (standalone são aplicações executadas fora de uma rede). Ele contém a máquina java virtual, as classes java e suporte para arquivos. Ele não tem compilador, debugger ou outros tipos de aplicativos. Uma boa alternativa para quem desenvolve aplicações deste tipo é utilizar esta ferramenta em conjunto com um arquivo em lote (batch). Depois de gerados os bytecodes (.class), criamos um arquivo batch que faz a chamada ao programa JRE que,

por sua vez, executa a aplicação.

JFC

Um dos grandes problemas apresentados pela portabilidade da linguagem Java são os componentes gráficos de interface. No início a Sun teve que utilizar apenas objetos que fossem encontrados em todas as plataformas como botões, janelas, caixas de texto. Alguns componentes particulares de alguma plataforma não puderam ser incluídos na AWT padrão, como por exemplo as Guias e ícones do Windows. Para resolver este problema a Sun lançou a JFC, um pacote estendido da AWT que apresenta várias classes que facilitam o desenvolvimento de interfaces.

JSWDK

Java Web Server é um servidor baseado na plataforma java. Ele é fácil de usar, extensível, fácil para administrar, independente de plataforma, e uma solução para aumentar e simplificar o desenvolvimento e gerenciamento de sites Web em sua Internet e Intranet. Para isso ele oferece recursos de segurança embutidos como autenticação, criptografia e integridade. Recursos: SSL, Secure area sandboxes, listas de controle de acesso e assinaturas digitais. E você pode estendê-lo aumentando a funcionalidade de seu servidor, escrevendo aplicações de rede que usam Servlets, Page Compilation, Presentation Templates e Session Tracking.

JSDK

O JavaServer Toolkit tem a função de simplificar a tarefa de desenvolver aplicações baseadas no servidor. O toolkit oferece melhorias que facilitam o desenvolvimento de aplicações Java no lado do servidor, os chamados servlets. "Goodbye CGI Say Hello To Java Servlets" Com esta frase a Sun define a função do Java Servlets. Que tem como principal característica rodar em qualquer servidor. Este kit proporciona algumas ferramentas necessárias para você criar Servlets. O kit inclui o servlet engine para você rodar e testar os servlets, os fontes javax.servlet.*, a documentação da API (javax.servlet.*, Sun.servlet.*), e suporte para os servidores de rede Netscape, Microsoft e Apaches.

JDBC

Incluído a partir da versão 1.1 do JDK, proporciona aos programadores uma interface uniforme para acessar bancos de dados relacionais, e provê uma base comum na qual podem ser construídas ferramentas de alto-nível e interfaces. Esta API oferece ao desenvolvedor a possibilidade de trabalhar com uma grande variedade de banco de dados relacionais. É uma ótima solução para o ambiente corporativo, visto que a portabilidade é uma característica vital para o funcionamento de sistemas na rede.

JavaBeans

Consiste na definição de uma API que torna mais fácil para o desenvolvedor criar, utilizar e gerenciar várias plataformas. Agora rodando no lado do servidor, os componentes JavaBeans tem a promessa de fazer o mesmo sucesso que obtiveram no lado cliente.

JRMI

Projetado para criar objetos java que podem invocar métodos de outra máquina virtual. Isto quer dizer que podemos fazer com que uma applet possa invocar métodos

que estejam em outra applet. Objeto Serialization permite que programas serializem objetos em uma stream de bytes que podem ser usadas depois para construir objetos equivalentes.

JNDI

Esta API oferece acesso aos serviços de nome e diretório em intranets e internet, possibilitando o acesso a uma variedade de informações sobre usuários, máquinas, redes, serviços e aplicações.

JIDL

Proporciona a interoperabilidade com CORBA, o padrão da indústria para computação heterogênea. A Java IDL inclui um compilador IDL para Java e um ORB que suporta IIOP. O compilador idltojava gera o cliente portátil e a espinha dorsal do servidor que trabalham com qualquer CORBA-compliant Object Request Broker (ORB), implementação que inclui o Java IDL ORB distribuído com o JDK™ 1.2. Um ORB permite que aplicações Java distribuídas invoquem operações transparentes em serviços de rede remotos que usam o Internet Inter-ORB Protocol (IIOP) desenvolvido pela Object Management Group. Você precisa do idltojava se você pretende definir e implementar serviços ORB na linguagem Java.

JMS

Esta API tem a finalidade de auxiliar a integração de operações em várias companhias, utilizando a filosofia de extranet. As classes facilitam a criação de um canal de comunicação entre as aplicações que rodam nestas empresas.

JavaOS

JavaOS é um pequeno e eficiente sistema operacional que executa o ambiente Java diretamente sobre o hardware sem precisar de qualquer outro sistema operacional, trazendo as vantagens da plataforma Java para um SO. JavaOS pode ser utilizado por uma grande variedade de produtos, que vão desde computadores de rede até Palmtops. Existem três tipos de produtos disponíveis: JavaOSTM for Business, JavaOSTM for Consumers e JavaOSTM for Network Computers.

Java Electronic Commerce

A Sun entra no mercado de comércio eletrônico com este framework. A idéia é ter um ponto-de-venda virtual acessível por qualquer browser habilitado java, permitindo a utilização de carteira eletrônica. (Java Wallet)

Info Bus

InfoBus habilita a troca dinâmica de dados entre componentes JavaBeans definindo um pequeno número de interfaces para cooperação de Beans e especificação do protocolo para uso dessas interfaces. Os protocolos são baseados em uma noção de um ônibus de informação. Todos os componentes que implementam estas interfaces podem andar no ônibus. Como um membro do ônibus qualquer componente pode trocar dados com qualquer outro componente de modo estruturado, inclusive arrays, tabelas e informações de um banco de dados.

JavaPC

O software JavaPC permite que você converta os PCs da sua empresa em network

computers. Se você deseja migrar para a plataforma Java, esta é uma solução que pode ser avaliada.

JavaHelp

O software JavaHelp é um API e um sistema de ajuda independente de plataforma e extensível, escrito completamente em Java. O sistema de JavaHelp dispõe capacidades para navegar, pesquisar, e exibir informações. Os arquitetos da informação e desenvolvedores podem usar este software para gerar ajuda on-line para componentes, applets e aplicações. Autores também podem usar o software de JavaHelp para gerar a documentação on-line de sua rede e Intranet.

Java Card

Esta API fornece a possibilidade de conexão com os smartcards, dispositivos que deverão fazer parte de nosso dia-a-dia em pouco tempo, sendo um dos principais candidatos a dinheiro eletrônico.

Java Blend

Java Blend é um software desenvolvido pela Sun Microsystems que permite simplificar o desenvolvimento de aplicações de banco de dados. Contém uma ferramenta e um ambiente que facilita a construção de aplicações empresariais que integram objetos Java e banco de dados. De fato, desenvolvedores não precisam conhecer SQL ou entender esquemas de banco de dados. Java Blend oferece algoritmos de caching sofisticados e outros recursos que fazem a Sun acreditar que: "Todos os programadores Java são agora programadores de banco de dados!"

Java Mail

O JavaMail API oferece classes abstratas que modelam um sistema de correio. Com esta API voce pode criar mail baseado em java ou aplicação de mensagem. Esta API oferece suporte aos protocolos IMAP e SMTP.

Java Media APIs

Esta família de APIs permite ao desenvolvedor criar aplicações e applets multimídia. Estas APIs incluem: 2D, 3D, Advanced Imaging, Java Media framework, Sound, Speech and Telephony. O controle de arquivos AVI e WAV, dentre outros, é oferecido pela Java Media Framework. Trabalhar com desenhos em 2D e 3D, que alguns dizem ser concorrente do VRML, também é possível, além da manipulação de outras classes e objetos.

Embedded Java

Uma API e ambiente de aplicação para dispositivos dedicados. Você pode usar EmbeddedJava para criar uma variedade de produtos como telefones móveis, pagers, controle de processo, networking routers e switches. EmbeddedJava permite que os fabricantes de dispositivos tirem vantagem da portabilidade e flexibilidade de Java nos seus produtos. A API foi projetada para ser facilmente portátil para qualquer sistema operacional de tempo real, e roda em uma grande variedade de microprocessadores.

Java Check

JavaCheck é uma ferramenta para o desenvolvedor certificar que suas aplicações e applets serão compatíveis com todas as plataformas, isto quer dizer 100% Pure Java. Basicamente voce tem uma aplicação Java que testa a portabilidade de sua applet ou aplicação standalone.

Java Plug-in

Este plug-in contém a implementação da Sun do Java Runtime Environment (JRE). Quando você instala este plug-in (que é compatível com Internet Explorer 3.02 ou superior e Netscape Navigator 3.0 ou superior) a Java Virtual Machine usada para a interpretação das applets passa a ser a JVM do plug-in e não a que está incluída em seu browser. Assim você poderá acessar os novos recursos disponibilizados pela Sun nas novas versões da linguagem java como o SWING e novos eventos da AWT 1.1

HotSpot

A estratégia do HotSpot é atacar dois pontos que consomem cerca de 39% do tempo de execução de uma aplicação Java, o Garbage Collector e a Sincronização. Apesar dos benefícios que os dois pontos trazem para o desenvolvedor, o tempo de execução ainda é grande. Além disso, cerca de 60% do tempo de execução e uma applet é usado no processo de interpretação do bytecode. O HotSpot utiliza um novo algoritmo chamado Generational Garbage Collection, que divide o processo de limpeza da memória em fatias de tempo. Ao invés de varrer toda a memória à procura de objetos para limpar, o que causa um grande retardo na execução, ele varre pequenos espaços de cada vez. Na verdade este processo demora poucos milissegundos e com isso ganha-se performance. Em grande parte das situações é preferível um aumento de velocidade de execução que toda a memória limpa. Para melhorar a sincronização, o HotSpot transforma o que eram linhas múltiplas de código de sincronização numa única instrução. A Sun ainda não divulgou como isso está sendo implementado. Talvez o mais sensacional no HotSpot seja o seu funcionamento durante a interpretação e compilação. Enquanto o HotSpot interpreta os bytecodes, analisa também como cada programa Java executa e imediatamente usa esta informação para aperfeiçoar os métodos críticos que consomem mais tempo. Uma grande vantagem é que ele tem todas as informações de tempo de execução à sua disposição, o que é impossível para um compilador estático. Depois de otimizar, o compilador gera uma versão em código de máquina deste método, que é então armazenada num buffer de memória na máquina nativa. A cada nova interpretação ele otimiza outros métodos, até que seu programa Java esteja rodando a 220 Km/h e deixando C++ literalmente comendo poeira. A velocidade da Java foi desde o início o maior alvo de críticas à linguagem. Mas o HotSpot vem com a promessa de fazer da Java a linguagem mais completa da atualidade.

A tecnologia JavaBeans

[Ant.](#) | [Mapa](#) | [Prox.](#)

Uma tendência crescente no campo da Engenharia de Software é a idéia dos componentes reutilizáveis, elementos de um programa que podem ser usados com mais de um pacote de software.

É um conjunto de classes independente de arquitetura e de plataforma para a criação e uso de componentes de software Java.

O JavaBeans foi projetado para ser compacto, pois freqüentemente os componentes serão usados em ambientes distribuídos, onde componentes inteiros podem ser transferidos por uma conexão de baixa largura de banda com a internet.

Sintaxe básica da linguagem Java

[Ant.](#) | [Mapa](#) | [Prox.](#)

- Especificando package e import
- Tipos de variáveis
- Estruturas de controle
- Declaração de funções
- Definindo classes

Especificando package e import

Constantemente estamos criando classes, instanciando objetos, em nossos projetos Java e importando outros já desenvolvidos.

Para isto utilizamos package que é um método de organizar grupos de classes na forma de pastas. Um pacote contém qualquer número de classes que se relacionam pelo objetivo, pelo escopo ou pela herança.

Quando precisamos utilizar classes previamente desenvolvidas temos que importá-las para a classe atual. A grande força do Java é ter um grande número de classes já prontas.

Na verdade, para sermos tecnicamente corretos, esse comando não importa todas as classes de um pacote; ele importa apenas as classes que foram declaradas como public e, mesmo assim, importa apenas as classes a que o próprio código faz referência.

```
package Nome_do_pacote;
```

```
import java.io.*;
```

```
import java.net.*;
```

Tipos de variáveis:

Tipos de variáveis: byte, short, int, long, float, double, char, boolean, String

*Operador aritmético: + , - , * , / , %*

Concatenador de strings: +

*Operador de atribuição: = , += , -= , *= , /= , %=*

Operador lógico: and , or , xor , ! , & , && , ||

Operador comparação: == , != , < > , <= , >=

Estruturas de controle:

(condição) ? (se_verdadeiro) : (se_falso)

```
if ( condição ) {  
    bloco 1  
} else {  
    bloco 2  
}
```

```
switch ( variável ) {  
    case 0: {  
        bloco 1  
        break; }  
    ...  
    case N: {  
        bloco N  
        break; }  
    default : {
```

```
    bloco  
    break;}  
}
```

```
while (condição) {  
    bloco  
};
```

```
do {  
    bloco  
}while (condição);
```

```
for (inicialização ; condição ; incremento) {  
    bloco  
};
```

Declaração de funções:

```
escopo tipo_retorno nome_funcao ([tipo_arg arg1, ...])  
{  
    bloco;  
    return Valor;  
}
```

Definindo classes:

```
escopo class nome_da_classe [extends super_classe] {
```

```
tipo_variável nome_variável;  
escopo tipo_retorno nome_função([tipo_arg arg1, . . .  
]) {  
    bloco;  
    return Valor;  
}
```

```
variavel = new nome_da_classe();
```

```
variavel.nome_da_função(argumento);
```

Java Servlet Developer Kit - JSDK

[Ant.](#) | [Mapa](#) | [Prox.](#)

É uma API Java que compilada atende requisições de páginas HTML e qualquer outra Java Technology. Trabalha com o servidor por multi-thread. Trata cookies com muita segurança. Implementa manipulador de sessão. Possui estrutura orientada por objeto o que aumenta a modularização do sistema.

Applets x Servlets

- Navegador deve suportar Java e na versão correta da applet;
- Demandam tempo de carregamento;
- Difícil comunicação com outro aplicativo Java ou Banco de Dados;

CGI x Servlets

- Sobrecarrega o servidor;
- Tem funções limitadas;
- Difícil programação;

PHP x Servlets

- São limitados ao ambiente da página HTML;
- Tipos de dados são variant reduzindo o processamento;
- Diferentes comandos para cada SGBD;

ASP x Servlets

- São limitadas a plataforma Windows;
- Estrutura de código diferente da filosofia C++;
- Difícil portabilidade para outros ambientes;

Independência de plataforma

Os servlets podem rodar em qualquer plataforma sem serem reescritos ou até mesmo compilados novamente. Os Scripts PERL, por serem interpretados, também possuem esta funcionalidade mas são muito lentos. As aplicações CGI que necessitam de alta performance são escritas em C.

Performance

Um novo programa CGI é carregado para cada requisição ao servidor. Isto quer dizer que se você tiver 10 requisições simultâneas, você tem 10 programas iguais na memória. Os servlets são carregados apenas uma vez e para cada nova

requisição o servlet gera um novo thread. O método `init()` do servlet, assim como nas applets, ocorre apenas na primeira vez que a classe é carregada. É geralmente no método `init()` que, por exemplo, estabelecemos uma conexão ao Banco de Dados. Cada uma das threads geradas pode usar a mesma conexão aberta no método `init()`. Este tipo de tratamento aumenta em muito o tempo de performance do servlet, já que você faz a conexão ao BD apenas uma vez e todos as outras requisições usam esta conexão.

Extensibilidade

Com Java você pode criar aplicações muito mais modulares, tirar todo o proveito da orientação a objetos e utilizar o grande número de APIs disponíveis pela Sun ou terceiros.

Facilidade

Programar em Java é muito mais fácil que programar em C ou PERL.

Sintaxe básica de um Servlet

[Ant.](#) | [Mapa](#) | [Prox.](#)

- Importando classes essenciais
- Declarando a classe HttpServlet
- Criando o método Init()
- Criando o método destroy()
- Criando o método doGet()
- Criando o método doPost()
- Criando o método service()
- Construindo a página HTML

Importando classes essenciais

As duas classes essenciais são:

```
import javax.servlet.*;
```

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

Declarando a classe HttpServlet

A criação da classe é feita herdando de uma superclasse todas as características da tecnologia servlet.

```
public class NomeDaNovaClasse extends  
HttpServlet{  
  
}
```

Criando o método Init()

Inicializa o Servlet e os logs de inicialização. Este método é chamado apenas uma vez e automaticamente assim que o Servlet é carregado pela primeira vez, ficando depois alocado na memória do servidor. Nenhuma requisição será processada até que este método esteja concluído. Neste método geralmente é feita a conexão ao Banco de Dados. Desta forma há apenas uma conexão ao BD e todos as requisições dos clientes usam esta mesma conexão para o acesso aos dados. Esta é uma das principais diferenças entre os Servlets e as aplicações CGI. As aplicações CGI inicializam um novo programa para cada requisição. Se houver 10 requisições simultâneas haverá 10 programas sendo executados ao mesmo tempo, cada um abrindo uma conexão com o banco de dados. Imagine isso numa página com 5000 visitas diárias. Vale ressaltar aqui que a conexão ao banco de dados é feito da mesma forma que em uma aplicação Java, isto quer dizer que não é necessário nenhum aprendizado pois você reaproveita todo o seu conhecimento. Os Servlets podem se tornar na forma mais fácil de migração de aplicações console para aplicações para a Internet. E o melhor, multiplataforma.

```
public void init(ServletConfig ConfigInicial)
throws ServletException{
    super.init(ConfigInicial);
}
}
```

Criando o método destroy()

Finaliza o Servlet, liberando os recursos alocados e destruindo os logs do Servlet. Este método, assim como o init, é chamado apenas uma vez e automaticamente pelo serviço de rede cada vez que o Servlet for removido da memória. Para este método ser chamado novamente é preciso que o Servlet seja recarregado. É neste método que fechamos a conexão ao Banco de Dados.

```
public void destroy() {
}
}
```

Sobre o método doGet() e doPost()

Uma consideração importante é quanto ao método usado no formulário. Grande parte das aplicações para Internet utilizam dois tipos de método o GET e POST. A diferença básica entre eles é que o método GET anexa na URL da página os

parâmetros passados pela conexão http enquanto o método POST oculta estes dados do usuário. As aplicações CGI tratam de forma separada cada um destes métodos, isto quer dizer que uma alteração no método de um formulário pode fazer o CGI não funcionar corretamente. Os dados do formulário serão enviados através de uma conexão http para o Servlet que irá receber e responder a requisição. A classe `HttpServlet` tem vários métodos para tratar os vários tipos de requisição (POST, GET). Para formulários com requisições do tipo POST pode-se utilizar o método `doPost(HttpServletRequest, HttpServletResponse)`, para o GET existe o `doGet(HttpServletRequest, HttpServletResponse)` e assim por diante. Este tipo de implementação oferece a possibilidade de utilizar os vários tipos de requisições mas também prende o Servlet ao formulário ou vice-versa, o formulário HTML não poderá ser alterado devido à incompatibilidade com o Servlet. Quem já programou CGI's utilizando Perl, C ou até mesmo ASP sabe do que estou dizendo. Java mais uma vez facilitou isso para os desenvolvedores e implementou o método `service(HttpServletRequest, HttpServletResponse)` que aceita todos os tipos de requisição como GET e POST e a repassa para o método especializado em tratar aquela requisição. Isto facilita em muito o desenvolvimento.

Criando o método `doGet()`

```
public void doGet(HttpServletRequest Req,
  HttpServletResponse Res) throws
  IOException, ServletException{

  }
```

Criando o método `doPost()`

```
public void doPost(HttpServletRequest Req,
  HttpServletResponse Res) throws
  IOException, ServletException{

  }
```

Criando o método `service()`

Esta é uma versão específica do método `Servlet.service`, que aceita parâmetros da conexão HTTP. Os pedidos padrão do HTTP são suportados despachando aos métodos de Java especializados para executá-los. Toda vez que uma requisição é feita ao servlet este método é executado e caso haja mais de uma conexão simultânea, vários threads são criados automaticamente. É responsabilidade do desenvolvedor do servlet sincronizar o acesso a todos os recursos compartilhados, tais como conexões de rede ou as variáveis da classe. O objeto `ServletRequest` contém informações sobre a requisição, incluindo os parâmetros fornecidos pelo cliente. O objeto `ServletResponse` é usado para retornar as informações para o cliente. Este método não é executado até que o método `init` seja finalizado.

```
public abstract void service(ServletRequest
    Req, ServletResponse Res) throws
    ServletException, IOException{

    String VariavelLocal =
    Req.getParameter("VariavelHtml");

}
```

Comentário:

Quaisquer dos métodos citados acima recebem como parâmetro dois valores: `HttpServletRequest` e `HttpServletResponse`. Estes dois métodos são responsáveis pela comunicação entre o Servlet e o cliente HTML, fazendo a comunicação através de uma conexão http. É através da interface `HttpServletRequest` que o Servlet tem acesso à informações enviadas pelo cliente como por exemplo os dados do formulário, o tipo do método HTTP usado (GET,POST) e até mesmo os cookies instalados. A interface `HttpServletResponse` por sua vez permite ao Servlet enviar os dados para o cliente, manipular o protocolo HTTP, especificar informações de cabeçalho da conexão. Esta interface possui métodos que permitem adicionar um Cookie no cliente ou redirecioná-lo para um outro endereço. Vale complementar que a interface `HttpServletRequest` é uma extensão da interface `ServletRequest` e o mesmo acontece com a interface `HttpServletResponse` que estende a interface `ServletResponse`.

Construindo a página HTML

Uma forma comum de resposta para o cliente é criar um objeto da Classe

ServletOutputStream e depois a utilizar para enviar os dados. Porém para cada envio de um pequeno texto, uma nova conexão para o cliente é criada.

```
ServletOutputStream Out = Res.  
getOutputStream();
```

```
Out.print("Building Dynamic WebSite -  
www.catalunha.eng.br");
```

Uma forma de melhorar a performance do Servlet é adicionar todos os string num buffer e depois que ele estiver todo preenchido abrir uma conexão com o cliente e mandar os dados de uma só vez. Utilizamos este método como uma forma de aumentar a performance do Servlet.

```
StringBuffer PagHtml = new StringBuffer();
```

```
PagHtml.append("Building Dynamic WebSite -  
www.catalunha.eng.br");
```

```
Res.setContentLength(PagHtml.length());
```

```
Res.getOutputStream().print(PagHtml.toString());
```

Características do JSP - JavaServer Pages

[Ant.](#) | [Mapa](#) | [Prox.](#)

É uma script Java para desenvolvimento de aplicações Web dinâmica utilizando HTML, . . . Em segundo plano todo arquivo JSP é convertido em um arquivo Servlet e enviado ao cliente. As outras tecnologias Java são acionadas pelo JSP via tags específicos. Veio para atender a facilidade de manipulação do HTML. Mais poderoso e flexível do que os outros SSB.



Sintaxe básica do JSP - JavaServer Page

[Ant.](#) | [Mapa](#) | [Prox.](#)

- Elementos de script
- Trabalhado com um JavaBean
- Tipos de variáveis
- Estruturas de controle
- Declaração de funções
- Definindo classes
- Passagem de parâmetros

Elementos de script

Elemento	Expression
Sintaxe	<%= expression %>
Descrição	Expressão é avaliada e enviada para a saída padrão
	Exemplo
Comentários	<%= "Catalunha" %> <%= Resultado %>

Elemento	Scriptlet
Sintaxe	<% código %>
Descrição	Contém código que será avaliado pelo script
	Exemplo
Comentários	<% R = 2 * A; %> .

Elemento	Declaration
Sintaxe	<%! código %>
Descrição	Declaração de variáveis globais e métodos válidos no script da página
	Exemplo
Comentários	<%! String Status = ""; %> <%! public int Soma(int a, int b){ ... }%>

Elemento	page Directive
Sintaxe	<%@ page prop="valor" %>
Descrição	Informa as configurações gerais do JSP
Comentários	Os valores default estão em negrito: <ul style="list-style-type: none"> • import="<i>package.class</i>" • contentType="<i>MIME-Type</i>" • isThreadSafe="true false" • session="true false" • buffer="<i>size</i>kb none" • autoflush="true false" • extends="<i>package.class</i>" • info="<i>message</i>" • errorPage="<i>url</i>" • isErrorPage="true false" • language="java"
Elemento	include Directive
Sintaxe	<%@ include file="URL" %>
Descrição	Inclue arquivo, texto ou código no código fonte do JSP
Comentários	Exemplo <%@ include file="Cabecalho.htm"; %>
Elemento	Comment client
Sintaxe	<!-- comentário -->
Descrição	Comentário que é ignorado pelo JSP, mas enviados ao cliente
Comentários	<!-- Autor: Catalunha -->
Elemento	Comment page
Sintaxe	<%-- comentário --%>
Descrição	Comentário que é ignorado pelo JSP, não é enviado ao cliente
Comentários	Os comentários de código são // ou /* */. Este são ocultos a página
Elemento	jsp:include
Sintaxe	<jsp:include pag="URL" flush="true"/>
Descrição	Inclue a página no tempo de carregamento
Comentários	Se o buffer de descarga for muito grande pode incluir as páginas com outra directiva

Elemento	<code>jsp:useBean</code>
Sintaxe	<code><jsp:useBean prop="valor" /></code>
Descrição	Localiza e inicializa um Java Bean.
Comentários	<p>Os valores default estão em negrito: <code>id="name"</code></p> <ul style="list-style-type: none"> • <code>scope="page request session application"</code> • <code>class="package.class"</code> • <code>type="package.class"</code> • <code>beanName="package.class"</code>
Elemento	<code>jsp:setProperty</code>
Sintaxe	<code><jsp:setProperty prop="valor" /></code>
Descrição	Set as propriedades do JavaBeans, implicitamente ou se declaradas pelo valor do parâmetro request
Comentários	<p>As propriedades são</p> <ul style="list-style-type: none"> • <code>name="beanName"</code> • <code>property="propertyName *"</code> • <code>param="parameterName"</code> • <code>value="value"</code>
Elemento	<code>jsp:getProperty</code>
Sintaxe	<code><jsp:getProperty name="propertyName" value="val"/></code>
Descrição	Recupera a propriedade de saída do JavaBeans
Comentários	<p>As propriedades são</p> <ul style="list-style-type: none"> • <code>name="beanName"</code> • <code>property="propertyName"</code>
Elemento	<code>jsp:forward</code>
Sintaxe	<code><jsp:forward page="URL"/></code>
Descrição	Desvia o fluxo para outra página.
Comentários	<p>Exemplo</p> <pre><jsp:forward page= "InfoGeral.htm"/></pre>

Elemento	jsp:plugin
Sintaxe	<jsp:plugin prop="value" > ... </jsp:plugin>
Descrição	Faz download de Java plugins para browser do cliente para executar Applet ou JavaBeans
Comentários	

Trabalhando com um JavaBean

A maior inserção de métodos em JSP é oriunda dos JavaBeans. Após serem definidos e testados basta agora informar a página para utilizá-lo com o tag visto na tabela anterior

Os campos mais importantes são:

id = identifica o Bean, dando-lhe um nome para ser usado no resto da página;

scope = Informa ao Bean qual a abrangência de sua atuação.

scope="page" = ações locais a página, valor assumido por default

scope="request" = atribuídas ao método request

scope="session" = abrangência a toda a sessão de carregamento da página

scope="application" = ações refletem em toda a aplicação

class = informa o caminho onde encontrar o bean seu package e class

Uma vez o Bean declarado você tem acesso a seus métodos pelo seguinte tag:

```
<jsp:setProperty id="beanInstanceName" property="*" />
<jsp:setProperty id="beanInstanceName" property="MetodoName"
value="StringValue" />
<jsp:setProperty id="beanInstanceName" property="MetodoName"
value= <%= expression %> />
```

Para solicitar retorno de algum método do Bean você pode usar o seguinte tag:

```
<jsp:getProperty id="beanInstanceName" property="MetodoName" />
```

Tanto para o tag set ou get do jsp o nome do método no JavaBeans tem que vir com estas palavras e com o a primeira letra em maiúscula. O exemplo abaixo ilustra bem a tabalho com JavaBeans

BeanSimples.jsp

```
<HTML>
<HEAD>
<TITLE>Utilizando JavaBeans in JSP</TITLE>
</HEAD>

<BODY>
<jsp:useBean id="teste" class="SetGetBean.SimplesBean"/>
<jsp:setProperty name="teste"
                 property="envia_msg"
                 value="Saudações colegas de curso" />

Message:<jsp:getProperty name="teste" property="busca_msg"

</BODY>
</HTML>
```

SimplesBean.java

```
package SetGetBean;
public class SimplesBean {
    private String mensagem = "Sem mensagem";

    public String getBusca_msg() {
        return(mensagem);
    }

    public void setEnvia_msg(String msg) {
        this.mensagem = msg;
    }
}
```

Passagem de parâmetros:

Arquivo.jsp?Parâmetro1=*Valor*& . . .

Páginas Dinâmicas com Servlets e JSP

[Ant.](#) | [Mapa](#) | [Prox.](#)

1. **Métodos Get, Post e Service de um Servlets**
 1. [GPSServlet.java](#)
2. **Métodos setProperties e getProperties de um JSP Bean**
 1. [SetGetBean.jsp](#)
3. **Passagem de parâmetros para um JSP Bean**
 1. [ParamBean.jsp](#)
4. **JSP aciona um Servlet**
 1. [EtoServlet.java](#)
5. **Gerenciamento de banco de dados com JSP**
 1. [GerBDJSP.jsp](#)
6. **Criador de tabelas interativo com JSP**
 1. [CriarBD.jsp](#)
7. **Gerenciador de Data e Hora com JSP**
 1. [DataBean.jsp](#)

Acionar um Servlet pelo método **Get** ou **Post** do formulário enviando-lhe uma mensagem.
O Servlet responderá pelo método **doGet** ou **doPost** ao método acionador correspondente ou **service** indiferentemente do método acionador.

Mensagem:

WebEngineer: catalunha@catalunha.eng.br

```
<html>
<head>
<title>Building Dynamic WebSite - www.catalunha.eng.br</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body bgcolor="#FFFFFF">
<p>&nbsp;</p>
<p align="center"><font size="5">Acionar um Servlet pelo m&eacute;todo
<b>Get</b>
    ou <b>Post</b> do formul&aacute;rio enviando-lhe uma mensagem.<br>
    O Servlet responder&aacute; pelo m&eacute;todo <b>doGet</b> ou
<b>doPost</b>
    ao m&eacute;todo acionador correspondente ou <b>service</b>
indiferentemente
    do m&eacute;todo acionador.</font></p>
<form name="form1" method="post"
action="http://127.0.0.1:8080/curso/servlet/GPSServlet.ServletGPS" >
    <div align="center">Mensagem:
        <input type="text" name="Mensagem">
        <br>
        <br>
        <input type="submit" name="Submit" value="Enviar">
    </div>
</form>
<p>&nbsp;</p>
<p><font size="2">WebEngineer: <a
href="mailto:catalunha@catalunha.eng.br">catalunha@catalunha.eng.br</a></fo
nt></p>
</body>
</html>
```

```

//Definindo um pacote para a classe
package GPSServlet;

//Importando as classes necessárias as funções chamadas
import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

//Classe principal estendida da classe generica de tratamento de servlets
HttpServlet
public class ServletGPS extends HttpServlet {
    private String Msg;
    private int NumAcessos = 0;

    //Método que é executado toda vez que um servlet é carregado
    public void init(ServletConfig servletConfig) throws ServletException{
        super.init(servletConfig);
        Msg = "Nenhuma mensagem foi enviada";
    }

    //Método que é executado toda vez que um servlet é descarregado
    public void destroy(){
        Msg = "Fim do Servlet";
    }

    //Método que atende ao acionamento do método get do formulário
    public void doGet(HttpServletRequest Req, HttpServletResponse Res)throws
    IOException, ServletException{

        Res.setContentType("text/html");
        PrintWriter out = Res.getWriter();

        //Parâmetro fornecido pelo formulário chamador
        Msg = Req.getParameter("Mensagem");

        //HTML que será gerado como resposta
        //Cada linha é enviada ao cliente em uma conexão
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Building Dynamic WebSite -
www.catalunha.eng.br</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<p><h1>Método acionado foi o: <b> GET </b></h1></p>");
        out.println("<p><h1>A mensagem enviada é: <b> " + Msg + "</b></h1></p>");
        out.println("<p><h2>Número de acessos: " + ++NumAcessos + "</h2>");
        out.println("</body>");
        out.println("</html>");

    }

    //Método que atende ao acionamento do método post do formulário
    public void doPost(HttpServletRequest Req, HttpServletResponse Res)throws
    IOException, ServletException{

        Res.setContentType("text/html");
        PrintWriter out = Res.getWriter();

```

```

//Parâmetro fornecido pelo formulário chamador
Msg = Req.getParameter("Mensagem");

//HTML que será gerado como resposta
//Cada linha é enviada ao cliente em uma conexão
out.println("<html>\n");
out.println("<head>\n");
out.println("<title>Building Dynamic WebSite -
www.catalunha.eng.br</title>\n");
out.println("</head>\n");
out.println("<body>\n");
out.println("<p><h1>Método acionado foi o: <b> POST </b></h1></p>\n");
out.println("<p><h1>A mensagem enviada é: <b> " + Msg + "</b></h1></p>
\n");
out.println("<p><h2>Número de acessos: " + ++NumAcessos + "</h2>\n");
out.println("</body>\n");
out.println("</html>\n");
}

//Método que atende ao acionamento do método get ou post do formulário
indiferentemente
public void service(HttpServletRequest Req, HttpServletResponse Res)throws
IOException, ServletException{

    Res.setContentType("text/html");

    //Parâmetro fornecido pelo formulário chamador
    Msg = Req.getParameter("Mensagem");

    //HTML que será gerado como resposta
    //Todo o texto é armazenado em um buffer de saída
    StringBuffer buffer = new StringBuffer();
    buffer.append("<html>\n");
    buffer.append("<head>\n");
    buffer.append("<title>Building Dynamic WebSite -
www.catalunha.eng.br</title>\n");
    buffer.append("</head>\n");
    buffer.append("<body bgcolor=\"#CCCCCC\">\n");
    buffer.append("<p><h1>Método acionado foi o: <b>" + Req.getMethod() +
"</b></h1></p>\n");
    buffer.append("<p><h1>A mensagem enviada é: <b> " + Msg + "</b></h1></p>
\n");
    buffer.append("<p><h2>Número de acessos: " + ++NumAcessos + "</h2>\n");
    buffer.append("</body>\n");
    buffer.append("</html>\n");

    //O buffer será enviado ao cliente em uma única conexão. Processo mais
rápido que o anterior
    Res.setContentLength(buffer.length());
    Res.getOutputStream().print(buffer.toString());
}
}

```

Mensagem:

Saudações colegas de curso

```
<%--Directiva jsp:useBean que irá associar o Bean ao JSP--%>
<jsp:useBean id="teste" class="SetGetBean.SimplesBean" />

<%--Directiva jsp:setProperty que irá enviar valores iniciais aos métodos do
Bean--%>
<jsp:setProperty name="teste" property="envia_msg" value="Saudações colegas de
curso"/>

<html>
<head>
<title>Building Dynamic WebSite - www.catalunha.eng.br</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body bgcolor="#FFFFFF">
<div align="center"><font size="5">Mensagem: </font> <p>
<%--Directiva jsp:getProperty que irá buscar o resultado do método de um
Bean--%>
<h1><jsp:getProperty name="teste" property="busca_msg" /></h1> </div>
</body>
</html>
```

```
//Definindo um pacote para a classe
package SetGetBean;

//Classe principal
public class SimplesBean{
    private String mensagem = "Sem mensagem";

    //Método que envia a mensagem ao objeto acionador
    public String getBusca_msg(){
        return mensagem;
    }

    //Método que busca a mensagem do objeto acionador
    public void setEnvia_msg(String msg){
        this.mensagem = msg;
    }
}
```

EMPRESA X

Administração OnLine - *100% Java*

Gerenciamento de cadastros:

Código:

Nome:

Email:

WebEngineer: catalunha@catalunha.eng.br

```

<!--
Projeto: Simples Gerenciador de Banco de Dados utilizando JSP
Autor: Prof. Catalunha
Data: 21/06/2000
-->
<%--Importando classes necessárias as funções utilizadas --%>

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

<%!

//Variáveis de gerenciamento do banco de dados
Connection Conexao;
Statement stmt;
ResultSet Resultado;

// Variáveis para gerenciamento da pagina
String Delete, Action, Botao, Path = "Cliente.jsp";

// Variáveis de campos da tabela
String Indice, Nome="", Email="";

//Metodos para receber os dados do formulário
public void AtualParam( HttpServletRequest Req){
    try{
        Indice = "9999";
        Nome = Req.getParameter("tfNome");
        Email = Req.getParameter("tfEmail");
    }catch(Exception e){
        System.out.println("Erro leitura de parâmetros");
        e.printStackTrace();
    }
}

//Método para atualizar os registros da tabela a página
public void AtualReg()throws IOException, SQLException{
    try{
        Indice = Resultado.getString(1);
        Nome = Resultado.getString(2);
        Email = Resultado.getString(3);
    }catch(Exception e){
        System.out.println("Erro atribuição de valores do ResultSet");
        e.printStackTrace();
    }
}

%>
<%--Directiva jsp:useBean para associar o Bean ao JSP --%>
<jsp:useBean id="Conectar" class="ConexaoBD.ConexaoBD" scope="page"/>

<%
//Configuração dos parâmetros para conexão da base de dados
String
    driver = "sun.jdbc.odbc.JdbcOdbcDriver",
    url = "jdbc:odbc:dbcatalunha",
    usuario= "catalunha",
    senha = "senha";

Conexao = Conectar.AbreConexao(driver,url,usuario,senha);
if(Conexao == null){
    Conectar.FechaConexao(Conexao);
}

%>

```

```

    <jsp:forward page="/ClienteJSP/ErroConexao.htm"/>
<%
}
//Confirma a criação de uma variável de conexão
stmt = Conexao.createStatement
(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPDATABLE);
%>

<%
int Registro=1;
String Operacao = request.getParameter("Registro");
if(Operacao != null){
    Registro = Integer.valueOf(Operacao).intValue();
}
Operacao = request.getParameter("Oper");
if(Operacao == null){
    Operacao = "";
}
try{
String Consulta = "Select * from Cliente";
Resultado = stmt.executeQuery(Consulta);
int NumReg = 0;
if(Resultado.next()){
    Resultado.last();
    NumReg = Resultado.getRow();
}else if(!Operacao.equalsIgnoreCase("Novo") && !Operacao.equalsIgnoreCase
("Inserir")){
    Operacao = "";
}
if(Operacao.equalsIgnoreCase("Primeiro")){
    Resultado.first();
    Registro = Resultado.getRow();
    AtualReg();
    Botao = "Inserir";
    Delete = Path + "?Oper=Deletar&Registro=1";
    Action = Path + "?Oper=Salvar&Registro=1";
}else if( Operacao.equalsIgnoreCase("Anterior")){
    Registro--;
    Resultado.last();
    if((Registro < 1) || (Registro > Resultado.getRow())){Registro =
NumReg;}
    Resultado.first();
    Resultado.absolute(Registro);
    AtualReg();
    Botao = "Inserir";
    Delete = Path + "?Oper=Deletar&Registro=" + Integer.toString
(Registro);
    Action = Path + "?Oper=Salvar&Registro=" + Integer.toString
(Registro);
}else if( Operacao.equalsIgnoreCase("Proximo") ){
    Registro++;
    Resultado.last();
    if((Registro > Resultado.getRow())){Registro = 1;}
    Resultado.first();
    Resultado.absolute(Registro);
    AtualReg();
    Botao = "Inserir";
    Delete = Path + "?Oper=Deletar&Registro=" + Integer.toString
(Registro);
    Action = Path + "?Oper=Salvar&Registro=" + Integer.toString
(Registro);
}
}

```

```

    }else if( Operacao.equalsIgnoreCase("Ultimo") ){
        Resultado.last();
        Registro = Resultado.getRow();
        AtualReg();
        Botao = "Inserir";
        Delete = Path + "?Oper=Deletar&Registro=" + Integer.toString
(Registro);
        Action = Path + "?Oper=Salvar&Registro=" + Integer.toString
(Registro);
    }else if( Operacao.equalsIgnoreCase("Novo") ){
        Indice=Email="";
        Nome = "Favor preencher os campos devidamente";
        Botao = "Confirmar";
        Action = Path + "?Oper=Inserir";
    }else if( Operacao.equalsIgnoreCase("Inserir") ){
        AtualParam(request);
        Consulta = "INSERT INTO Cliente (Nome,Email) VALUES ('" + Nome +
"' ,'" + Email + "')";
        stmt.executeUpdate(Consulta);
        Consulta = "Select * from Cliente";
        Resultado = stmt.executeQuery(Consulta);
        Resultado.last();
        Registro = Resultado.getRow();
        AtualReg();
        Botao = "Inserir";
        Delete = Path + "?Oper=Deletar&Registro=" + Integer.toString
(Registro);
        Action = Path + "?Oper=Salvar&Registro=" + Integer.toString
(Registro);
    }else if( Operacao.equalsIgnoreCase("Deletar") ){
        Resultado.first();
        Resultado.absolute(Registro);
        AtualReg();
        Consulta = "DELETE FROM Cliente WHERE Indice=" + Indice;
        stmt.executeUpdate(Consulta);
        Consulta = "Select * from Cliente";
        Resultado = stmt.executeQuery(Consulta);
        if(Resultado.next()){
            Resultado.first();
            if(Registro > 1){
                Resultado.absolute(--Registro);
            }else {
                Resultado.absolute(Registro);
            }
            AtualReg();
            Botao = "Inserir";
            Delete = Path + "?Oper=Deletar&Registro=" + Integer.toString
(Registro);
            Action = Path + "?Oper=Salvar&Registro=" + Integer.toString
(Registro);
        }else{
            Indice=Nome=Email="";
            Botao = "Inserir";
            Delete = Path;
        }
    }else if( Operacao.equalsIgnoreCase("Salvar") ){
        Resultado.first();
        Resultado.absolute(Registro);
        AtualParam(request);
        Consulta = "UPDATE Cliente SET Nome='" + Nome + "',Email='" + Email +
"' WHERE Indice = " + Resultado.getString(1);

```

```

        stmt.executeUpdate(Consulta);
        Consulta = "Select * from Cliente";
        Resultado = stmt.executeQuery(Consulta);
        Resultado.first();
        Resultado.absolute(Registro);
        AtualReg();
        Botao = "Inserir";
        Delete = Path + "?Oper=Deletar&Registro=" + Integer.toString
(Registro);
        Action = Path + "?Oper=Salvar&Registro=" + Integer.toString
(Registro);
    }else{
        Indice=Nome=Email="";
        Registro = 1;
        Botao = "Inserir";
        Action = Path;
        Delete = Path;
    }
}catch(SQLException sql){
    Nome = "Erro de SQL";
    System.out.println("Houve uma SQLException: " + sql);
}catch(Exception e){
    Nome = "Erro qualquer";
    System.out.println("Houve uma Exceção");
    e.printStackTrace();
}
}

```

```
%>
```

```

<html>
<head>
<title>Building Dynamic WebSite - www.catalunha.eng.br</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body bgcolor="#FFFFFF">
<div align="center">
<p><b><font color="#FF0000" size="7">EMPRESA X</font></b><br>
    <b><font color="#0000FF" size="5">Administra&ccedil;&atilde;o OnLine - <i>
100%
    Java</i></font></b></p>
</div>
<form method="post" action="<%=Action%>" name="form1" >
    <table border="0" align="center" width="663">
        <tr>
            <%if(Botao == "Inserir"){%>
            <td height="42">
                <input type="button" name="Submit" value="Primeiro" onClick =
"location='<%= Path %>?Oper=Primeiro'">
            </td>
            <td height="42">
                <input type="button" name="Button" value="Anterior" onClick =
"location='<%= Path %>?Oper=Anterior&Registro=<%= Integer.toString(Registro) %
>'">
            </td>
            <td height="42">
                <input type="button" name="Submit22" value="Pr&oacute;ximo" onClick=
"location='<%= Path %>?Oper=Proximo&Registro=<%= Integer.toString(Registro) %
>'">
            </td>
            <td height="42">

```

```

        <input type="button" name="Submit3" value="&Uacute;ltime" onClick =
"location='<%= Path %>?Oper=Ultimo'">
    </td>
    <td height="42">
        <input type="button" name="Submit4" value="Apagar" onClick =
"location='<%= Delete %>'">
    </td>
    <td height="42">
        <input type="submit" name="Submit5" value="Atualizar" >
    </td>
<%=}%>
<%if(Botao == "Inserir"){%>
    <td height="42">
        <input type="button" name="btInserir" value="Inserir" onClick =
"location='<%= Path %>?Oper=Novo'">
        <%=}%else{%>
            <input type="submit" name="btConfirmar" value="Confirmar" >
            <input type="button" name="Submit2" value="Cancelar"  onClick =
"location='<%= Path %>?Oper=Primeiro'">
        </td>
        <%=}%>
    </tr>
</table>
<p align="left"><b>Gerenciamento de cadastros:</b></p>
<table align="center">
    <tr>
        <td width="57">C&oacute;digo:</td>
        <td width="226"><%= Indice %></td>
    </tr>
    <tr>
        <td width="57">Nome:</td>
        <td width="226">
            <input type="text" name="tfNome" size="40" value="<%= Nome %>">
        </td>
    </tr>
    <tr>
        <td width="57" height="31">Email:</td>
        <td width="226" height="31">
            <input type="text" name="tfEmail" size="40" value="<%= Email %>">
        </td>
    </tr>
</table>
</form>
<p align="left"><font size="2"><i><br>
    </i>WebEngineer: <a
href="mailto:catalunha@catalunha.eng.br">catalunha@catalunha.eng.br</a></font>
</p>
</body>
</html>

<%
Conectar.FechaConexao(Conexao);
%>

```

```

package ConexaoBD;

import java.sql.*;

public class ConexaoBD{

    public java.sql.Connection AbreConexao(String driver,String url, String
usuario, String senha){
        java.sql.Connection Con = null;
        try{
            System.out.print("Iniciando a conexao com o BD...");
            Class.forName(driver);
            Con = java.sql.DriverManager.getConnection(url,usuario,senha);
            if(!VerificaErroSql(Con.getWarnings())){
                System.out.println("OK!");
            }
        }catch(SQLException sql){
            System.out.println("Houve uma SQLException: " + sql);
        }catch(ClassNotFoundException fnf){
            System.out.println("Houve uma ClassNotFoundException: " + fnf);
        }
        return Con;
    }

    public boolean FechaConexao(java.sql.Connection Con){
        boolean fc = false;
        try{
            if(Con != null){
                System.out.print("Fechando a conexao com o BD...");
                Con.close();
            }else{
                System.out.println("Nao existe conexao aberta");
            }
            if(!VerificaErroSql(Con.getWarnings())){
                System.out.println("OK!");
                fc = true;
            }
        }catch(SQLException sql){
            System.out.println("Houve uma SQLException: " + sql);
        }
        return fc;
    }

    private static boolean VerificaErroSql(SQLWarning warn) throws
SQLException {
        boolean rc = false;
        if (warn != null) {
            System.out.println("\n *** Warning ***\n");
            rc = true;
            while (warn != null){
                System.out.println("SQLState: " + warn.getSQLState());
                System.out.println("Message: " + warn.getMessage());
                System.out.println("Vendor: " + warn.getErrorCode());
                System.out.println("");
                warn = warn.getNextWarning();
            }
        }
        return rc;
    }
}

```

Conclusões:

[Ant.](#) | [Mapa](#) | [Prox.](#)

O conhecimento aqui ministrado mostrou que as atuais ferramentas para Desenvolvimento Dinâmico de Sites para a Web não são Plug-in-Play; exigindo do desenvolvedor um conhecimento amplo sobre os multiformes ambientes que seu sistema estará submetido.

Mostrou a superioridade da tecnologia Java para solução dos problemas apresentados pelo mercado mundial, frente ao melhor atendimento ao cliente.

Foi enfática a necessidade de soluções de problemas utilizando sistemas orientados a objetos, devido a capacidade de reorganização e implementação de aplicativos com pequenas modificações em caráter específico.

Catalunha

www.catalunha.eng.br

Bibliografia:

[Ant.](#) | [Mapa](#) | [Início](#)

- Revistas:
 - Exame informática
 - Developers Magazine
 - PC Master
 - Revista do Linux
- Sites
 - <http://www.java.com>
 - <http://www.javabr.com.br>
 - <http://www.javasite.com.br>
 - <http://www.php.net>
 - <http://www.mysql.org>
 - <http://www.xitami.com>
 - <http://www.uol.com.br>