Trabalho escrito sobre "Tecnologias de Personalização para WebSites de E-commerce"

Aluno: Carlin
Email: carlin@dsc.ufpb.br
Orientador: Francilene P. Garcia
Semestre: 2000.2
UFPB – Campina Grande

Índice

1 Apresentação	**3**
2. Personalização de *WebSites*	**4**
2.1 Visão Geral	4
2.2 Referências	4
3. A tecnologia ASP (*Active Server Pages*)	**5**
3.1 Introdução	5
3.2 Arquitetura	6
3.3 Características	7
3.4 ASP X CGI	7
3.5 ASP & Banco de Dados	8
3.6 Back-End	8
3.7 Front-End	9
3.8 Conclusão Parcial	9
3.9 Referências	9
4. A tecnologia JSP (Java *Server* Pages)	**10**
4.1 Introdução	10
4.2 Arquitetura	11
4.3 Características	12
4.3.a Modelo 1	12
4.3.b modelo 2	13
4.4 JSP X CGI	13
4.5 JSP & Banco de Dados	14
4.6 Back-End	14
4.7 Front-End	14
4.8 Conclusão Parcial	15
4.9 Referências	15
5. A tecnologia PHP (Personal Home Page Tools)	**16**
5.1 Introdução	16
5.2 Arquitetura	16
5.3 Características	17
5.4 PHP X CGI	18
5.5 PHP & Banco de Dados	18
5.6 Back-End	19
5.7 Front-End	19
5.8 Conclusão Parcial	19
5.9 Referências	20
6. A tecnologia XML (eXtensible Markup Language)	**21**
6.1 Introdução	21
6.2 Arquitetura	21
6.3 Características	23
6.4 Vantagens	24
6.5 XML X HTML	24
6.6 Conclusão Parcial	25
6.7 Referências	25
7. Avaliação Comparativa	**26**
7.1 Introdução	26
7.2 Tabela de Comparação	26
8. Conclusão	**27**
9. Links Interessantes	**29**
10. Anexos	**30**
10.1 Anexo 1 – ASP acessando Banco de Dados usando ODBC	31
10.2 Anexo 2 – JSP acessando Banco de Dados usando ODBC	31
10.3 Anexo 3 – PHP acessando Banco de Dados	32
10.4 Glossário	33

Apresentação

Já faz algum tempo que estamos presenciando uma grande mudança na *Internet*, mudanças essas que estão ocorrendo principalmente em todos os grandes *WebSites* de E-commerce, que estão tentando interagir com seus clientes para descobrir e armazenar suas preferências e características.

A tecnologia que permite este tipo de recurso é chamada de, desenvolvimento de páginas *Web* dinâmicas ou personalização de *WebSites*, que consiste em inserir porções de código (*scripts*) nas páginas HTML (*Hiper Text Markup Language*), e processá-las no *WebServer* antes de serem enviadas para os usuários, separando desta forma a geração da interface da geração do conteúdo. Isto permite ao *designer* mudar toda a interface do *site*, sem precisar se preocupar com o conteúdo, já que este será gerado dinamicamente.

No contexto de um estudo sobre tecnologias para o desenvolvimento de páginas *Web* dinâmicas e particularmente para o desenvolvimento de aplicativos para ambientes de comércio eletrônico via *Internet*, iremos destacar algumas tecnologias como ASP (*Active Server Pages*), JSP (*Java Server Pages*), PHP (*Personal Home Page Tools*) e XML (*eXtensible Markup Language*), que deverão ser analisadas como uma alternativa na concepção deste projeto.

Neste trabalho, serão apresentados os principais aspectos que devem ser considerados para uma análise da relação custo/benefício em relação à aplicação de cada uma destas tecnologias abordadas. Estes aspectos incluem uma breve introdução, juntamente com algumas características técnicas, suas vantagens e desvantagens, uma descrição de suas funcionalidades em situações de acesso a bancos de dados, bem como um comparativo geral entre todas estas tecnologias.

Personalização de *WebSites*

2.1 – Visão Geral

Quando se lança qualquer *site* na *Internet*, é necessário determinar o público alvo que, deve ser oconsumidor final. Com *sites* de *E-commerce* não é diferente, pois além deste fator, teremos agora que encontrar maneiras de agradar ao consumidor.

Pesquisas americanas apontam que, para a sobrevivência da empresa moderna, principalmente em mercados saturados como Europa e Estados Unidos, é necessário primeiramente manter a base de clientes (consumidores) e, em segundo lugar conquistar novos(1).

Desta forma, ao realizar uma segunda compra, o consumidor quer ter vantagens extras por comprar um produto da mesma empresa mais de uma vez, ou em maior quantidade. Para isto, é necessário que a empresa saiba identifica-lo (nome, dados), e quando ele quiser saber o preço, que este já venha com desconto, com brinde ou outro tipo de vantagem que ele conquistou por ter comprado anteriormente, afinal de contas a empresa sabe quem ele é, e o julga um cliente muito querido e especial.

Algumas sugestões de como conseguir agradar seu consumidor, é através da captura de seu perfil, que poderá ser feito via *Email*, em salas de bate-papo, em fóruns de discussão, na navegação por *WebSites* de produtos concorrentes, no preenchimento de formulários eletrônicos, em promoções *online*, ou com a contratação de empresas especializadas.

A personalização provê aos comerciantes e publicitários, a habilidade para fazer com que os *Websites* de anúncios e promoções de produtos, sejam personalizados, para isso existem várias técnicas para alcançar este objetivo, as mais comuns são:

- ✓ Baseadas no seu comportamento passado (filtragem por histórico ou filtragem de conteúdo);
- ✓ Baseadas em conclusões tiradas de outros clientes cadastrados (filtragem colaborativa);
- ✓ Ou pode ser feita a união destas técnicas para gerar uma forma híbrida;

Este *E-commerce* de oportunidade (personalizado), serve para fazer comércio B2C em vez do comércio massificado tradicional, por isso neste trabalho, nós iremos discutir as diferentes tecnologias para prover personalização de *WebSites,* e também forneceremos as características, arquiteturas, comparações com CGI, compatibilidade com bancos de dados, *back-end, front-end*, de cada uma delas, finalizando com um quadro comparativo e com uma demonstração através de cenários pré estabelecidos.

2.2 – Referências:

1 - http://www.br-business.com.br/ - 10/09/2000;

2- Data Mining and Personalization Technologies – Philip S. Yu – IBM T. J. Watson Research Center – New York - 2000

A Tecnologia ASP (*Active Server Pages*)

3.1 - Introdução:

Quando a *Microsoft* lançou a versão 3 do seu servidor *Web* o IIS (*Internet Information Server*), ela criou um novo ambiente de programação, baseado em *scripts*, específico para o desenvolvimento de páginas *Web* dinâmicas, ASP (*Active Server Pages*).

ASP é uma combinação de *scripts* desenvolvidos em linguagem *VBScript* ou *Jscript* preferencialmente, que, unidos a *tags* HTML, podem ser executados tanto no cliente como no servidor, e podem ser acessados por todos os *browsers*.

Páginas ASP ficam preferencialmente armazenadas no servidor *Web* da *Microsoft*, o IIS, *Internet Information Server* versão 3 ou superior (existe um componente desenvolvido nos EUA pela Chili Soft www.chilisoft.com que permite a execução de páginas ASP em ambiente Unix ou Linux), e podem ser usados para desenvolver aplicações que serão utilizadas na *Internet* ou em uma *Intranet*.

Para obter mais informações sobre a utilização do ASP em plataformas que não sejam *Windows*, uma boa dica é pesquisar no *site* da empresa Halcyon no endereço www.halcyonsoft.com/.

Por ser executado no servidor, o ASP mostrou-se uma alternativa bastante interessante para os *WebDesigners* que utilizavam programas CGI (*Common Gateway Interface*) para executar tarefas como, recuperar informações vindas de um formulário e gravar em um banco de dados.

Usando programas CGI, para cada requisição vinda do usuário era criado um processo no *WebServer*(1), ou então o programa seria totalmente executado na máquina do cliente, por isso eles se tornavam lentos.

ASP veio suprir esta necessidade, de forma que, os programas feitos anteriormente usando CGI, agora já poderiam ser reescritos, e com isso se tornarem mais rápidos e menos dispendiosos para o *WebServer*, já que cada requisição vinda do cliente será gerenciada como uma *thread*.

3.2 - Arquitetura

Sistemas desenvolvidos utilizando ASP, quase sempre irão seguir uma arquitetura de desenvolvimento em n-camadas, onde, a primeira seria o *browser*, a segunda o *WebServer* e as outras camadas que poderão existir, irão variar de acordo com cada aplicação.

Seguindo este modelo da figura 1, o usuário acessará a página ASP e fará uma requisição, o *WebServer* recebe a requisição vinda da página ASP e faz a interpretação do código naquele instante (*on-the-fly*), com isso, ele retorna o conteúdo gerado dinamicamente para quem o solicitou. Desta forma o usuário não terá acesso ao código fonte, pois quem executou toda a operação foi o servidor, o usuário recebeu como resposta apenas HTML puro.

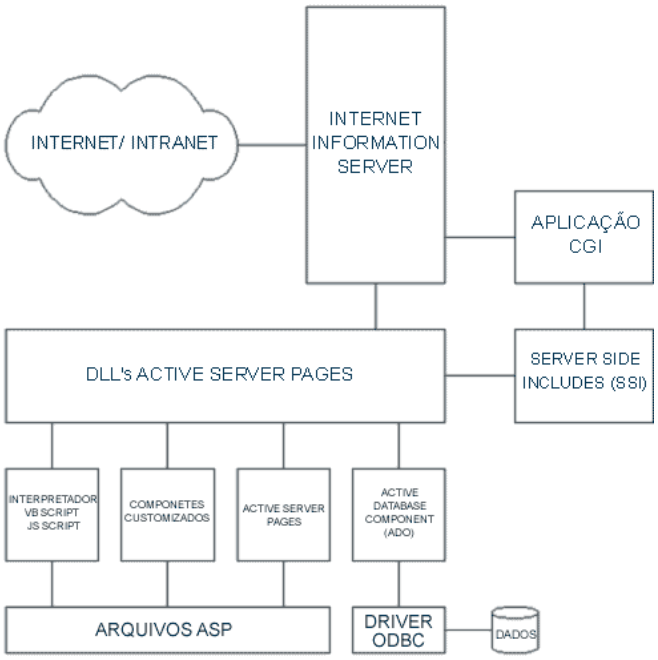

fig-1: Uma Arquitetura ASP

3.3 - Características

As páginas ASP podem ser escritas utilizando as seguintes linguagens de *script*: *Jscript* (o *JavaScript* da *Microsoft*), *VBScript* e outras linguagens como REXX, PERL (*Practical Extraction and Report Language*), e Python, desde que elas possuam seus respectivos *plug-ins*.

Segundo Lutz Prechelt, em seu *papper* "An Empirical Comparison of Seven Programming Languages" (2), chegou à conclusão que é necessário um estudo mais detalhado entre as linguagens de programação, que evidencie as virtudes e defeitos de cada uma delas, de maneira que, não nos deixemos influenciar por vendedores de software e nem pela influência de outros programadores tendenciosos.

Se a aplicação não possuir *scripts* para serem executados no lado cliente, todos os *browser*s do mercado podem acessar páginas feitas em ASP, inclusive o *Netscape*, não sendo necessário à instalação de nenhum *plug-in*, pois após o servidor processar a requisição ASP, o resultado que o cliente receberá será código HTML puro, como vemos no exemplo abaixo (figura 2):

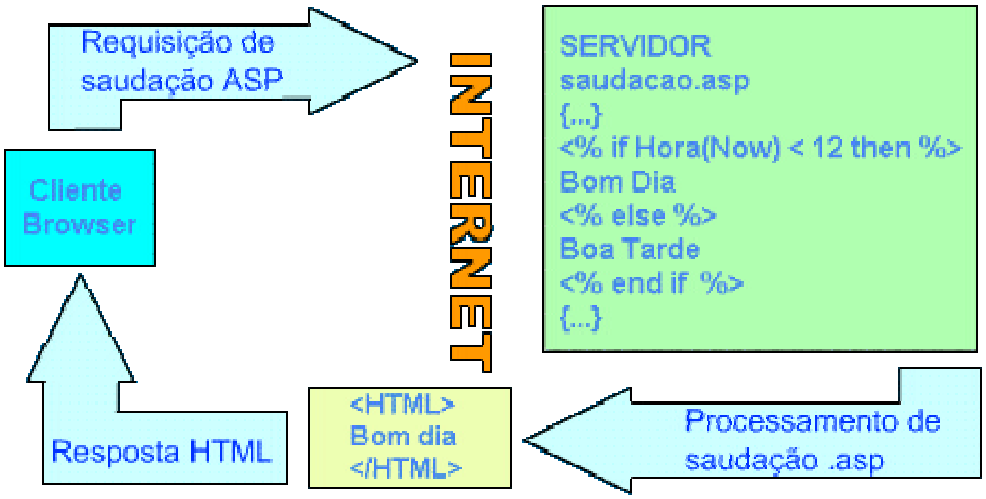

fig-2: Uma Requisição ASP

A relação entre os formulários HTML e ASP é muito importante, porque a partir deles nós podemos disparar as ações, nas quais iremos requisitar uma página ASP, com isso podemos construir campos ou passar parâmetros de uma página para outra.

Aplicações ASP podem armazenar dados que são mantidos durante toda uma sessão, assim, quando um usuário fornecer seu nome em uma página todas as demais poderão obter este dado automaticamente. Este recurso é ideal para aplicações de E-commerce.

3.4 - ASP X CGI

ASP fornece todos os recursos das aplicações CGI de uma forma simples e robusta, pois diferentemente do CGI, não cria um processo no servidor para cada pedido vindo do usuário, além de ser mais muito fácil a criação de conexões entre o *browser* e os dados, que normalmente estão em formatos incompatíveis com HTML (ex: bancos de dados).

Usando ASP ao invés de CGI, um servidor pode atender a um grande número de pedidos vindo dos usuários de forma mais rápida e usando menos memória(3). Além disso, criar páginas ASP é em geral muito mais fácil do que criar aplicações CGI(4).

3.5 - ASP & Banco de Dados

Usando ASP será possível visualizar, atualizar e adicionar informações nos servidores de banco de dados que possuam um *driver* padrão ODBC (*Open DataBase Connectivity*), que é a maneira mais simples de acessar um banco de dados quando usamos algum dos sistemas operacionais da *Microsoft* (*Windows*). Na figura 3, ilustramos um acesso à banco de dados usando ASP.

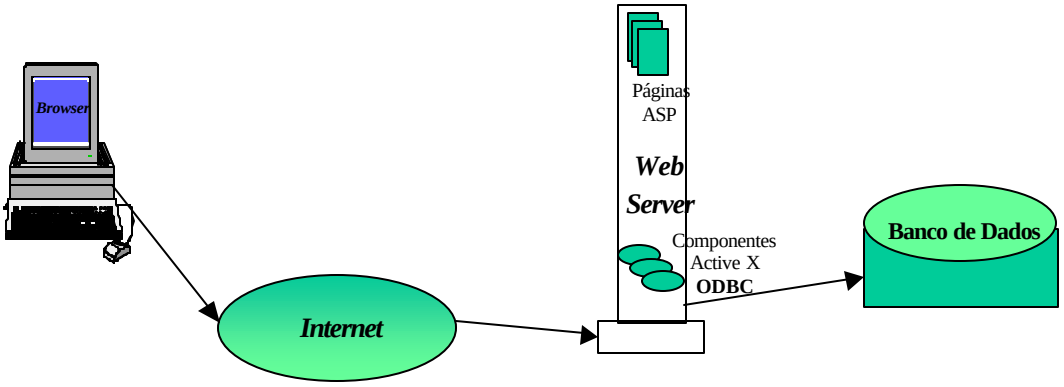

fig-3: ASP acessando BD

A integridade dos dados fica sobre a responsabilidade do MTS (*Microsoft Transaction Server*), que coordena as transações (bem ou mal sucedidas) em conjunto, nos diversos bancos de dados existentes, desde que possuam *driver* ODBC.

O IIS fornece o pacote ADO (*ActiveX Data Objects*), que é uma coleção de objetos utilizados para recuperação, alteração, inclusão e exclusão de registros em bancos de dados ODBC e OLE DB. O ADO é escrito em páginas ASP(11), executado no servidor *Web*, e retorna as informações dos bancos de dados em formato HTML. A página retornada pode ser visualizada por qualquer *browser* em qualquer plataforma, pois o código é todo executado pelo Servidor.

3.6- Back-End

Toda aplicação desenvolvida utilizando ASP possui seu código fonte preservado, já que todo o processamento acontece no servidor e, se tentarmos visualizar pelo *browser* o que veremos será puramente HTML (7).

A depuração do ASP, permite que se localizem os problemas, fazendo também interativamente os testes das suas aplicações, que podem ser escritas em qualquer linguagem de *script* por ele suportado.

A execução das aplicações fora do processo(10), em um espaço de memória separado, permite desenvolver componentes que serão iniciados e finalizados rapidamente, assim, os códigos que não foram bem executados, não irão afetar todo o trabalho do servidor.

A JVM (*Java Virtual Machine*)(6) do *Windows*, irá interagir com os componentes Java do servidor que suportam a versão 1.1 do JDK (*Java Development Kit*), desta forma, eles serão instanciados pelo ASP e gerenciados como se fossem nativos do MTS.

A maneira que ASP fornece para expandir a sua capacidade, é através do desenvolvimento de componentes externos, onde eles poderão ser escritos em C++ ou Visual Basic e são chamados de COM (*Component Object Model*)(6).

Por ser parte integrante do IIS, possui toda a sua segurança integrada com o *Windows NT Server*, sendo bastante trivial para o administrador do servidor restringir o acesso a páginas ASP, usando os esquemas de autenticação do IIS (senha básica da *Web*, senha do NT ou certificados de cliente), possibilitando também prover segurança aos dados transmitidos usando o protocolo SSL (*Secure Socket Layer*)(7).

3.7 - Front-End

Possibilita ao usuário trabalhar com sessões, ou seja, quando alguém visita uma página no *site*, ASP cria uma sessão e imediatamente pode diferenciar aquele usuário de todos os outros que estão no *site*, desta forma nada que for armazenado na sessão daquele usuário, pode ser recuperado ou manipulado por uma página, ou pelas próximas páginas que ele visitar(5).

Por possuir todo o seu processamento executado no servidor, as páginas ASP, poderão ser visualizadas através de qualquer *browser* existente no mercado, pois após sua execução no servidor o usuário receberá apenas a parte que possui código HTML.

3.8 – Conclusão Parcial

ASP é a ferramenta para confecção de páginas dinâmicas da *Web* oferecida pela *Microsoft*, que é a empresa que atualmente domina o mercado mundial de software.

Como toda a família de produtos *Microsoft*, ASP também possui o suporte das empresas *Microsoft Solution Providers*, que estão espalhadas pelo mundo, além de listas de discussões, fóruns, tutoriais, e um *site* completo de informações sobre seus sub-produtos, com isso, ASP fornece uma significativa vantagem em relação à outras ferramentas, quanto a resolução rápida dos problemas.

Por utilizarem a maioria dos servidores *Web* existentes no mercado e apresentarem uma maior confiabilidade e melhor performance que os programas CGI, ASP tornou-se a linguagem mais poderosa para desenvolvimento de páginas ativas para *Windows* NT(9), com uma infinidade de objetos já prontos, possibilitando ao usuário de maneira fácil e rápida o desenvolvimento de outros objetos utilizando para isso, a linguagem *Visual Basic* ou a linguagem C++.

3.9 - Referências

1 - http://www.br-business.com.br/ - 15/09/2000

2 - http://www.lemon.com.br - 16/09/2000

3 – An Empirical Comparison of Seven Programming Languages – Lutz Prechelt - Computing Practices – October 2000

4 - http://www.aspbrasil.com.br - 03/10/2000

5 - http://www.*Microsoft*.com/BRASIL/mspress - 30/09/2000

6 - http://www.*Microsoft*.com - 18/09/2000 (inglês)

7 - http://www.ccuec.unicamp.br/ - 23/09/2000

8 - http://*Web*masterguide.com - 16/09/2000 (inglês)

9 - http://www.ultimateasp.com/bulletin/default.asp - 18/09/2000 (inglês)

10 - http://www.asptoday.com - 21/09/2000 (inglês)

11 - http://www.asp101.com - 01/10/2000 (inglês)

A Tecnologia JSP (Java *Server* Pages)

4.1- Introdução

JSP (*Java Server Pages*), é uma tecnologia desenvolvida pela *Sun* MicroSystems para ser uma opção mais fácil de desenvolvimento de aplicações dinâmicas para a *Internet*, que tenham sua execução feita no servidor, utilizando a linguagem de programação Java, a tecnologia *Servlet* e algumas características similares as do ASP (*Active Server Pages*) da *Microsoft*(3).

A tecnologia JSP, não veio para substituir a tecnologia *Servlet*, mas sim para oferecer uma maneira mais fácil de se trabalhar com ela, sem a necessidade do usuário ser um programador experiente em Java, haja visto que, usando *Servlet* todo o código HTML fica embutido no programa escrito em Java, exigindo do programador um bom conhecimento das duas linguagens (Java e *Servlet*)(2).

Existe uma vantagem crucial da tecnologia JSP sobre a tecnologia baseada em *Servlet*, que fez o JSP adquirir um importante papel no ciclo de desenvolvimento de *Web*Pages dinâmicas:

- ✓ O problema mais grave do *Servlet*, era que o programador de Java, teria que escrever o comando out.println() para exibir cada linha HTML, por isso criar e manter páginas usando *Servlet* resultava em uma péssima produtividade, que diminuía ainda mais com o aumento de conteúdo na página(2).

Para solucionar este problema surgiu o JSP, que ao invés de precisar embutir o código HTML dentro do programa Java, você poderia colocar "pedaços" de código Java em seu código HTML, desta forma, é o servidor que fará toda criação do *Servlet*, com isso, o JSP foi então considerado uma opção de alta produtividade para um programador de Java escrever um *Servlet*, já que os códigos HTML eram exibidos normalmente sem precisar do comando out.println(),

Agora com o JSP ao invés de ter HTML dentro do código Java, o que existirá será código Java dentro do HTML, ou seja, facilitou bastante mas ainda não estava muito usual, porque não existia uma forma simples de acessar componentes externos, até que após algum tempo de amadurecimento da tecnologia, alguns programadores passaram a utilizar JSP com *JavaBeans*.

Com o uso dessa prática, o tamanho das páginas JSP foi reduzido, pois muita coisa relacionada à lógica do negócio ficou fora dela, sendo acessada somente através dos "ganchos" oferecidos pelos componentes *JavaBeans.*

Além da facilidade de criação e manutenção das páginas *Web*, outras fortes razões para se usar JSP, é sua portabilidade e seu poder, resultante da união do Java (lado servidor), com as funcionalidades do HTML.

As páginas JSP, podem ser criadas e mantidas através das ferramentas de editoração de HTML/XML e são compostas tipicamente de :

- Componentes HTML/XML estáticos;
- *Tags* especiais JSP;
- Opcionalmente, pedaços de códigos escritos em linguagem Java chamados de "*scriptlets*";
- JSP + *JavaBeans*, são usados para separar a lógica do negócio da camada de apresentação.

4.2- Arquitetura

Antes do aparecimento do JSP, os programadores que usavam *Servlet* tiveram que passar pela "dor de cabeça" de formatar suas próprias declarações de impressão, isto conduzia a muitos erros de HTML, dificultando também a manutenção do código escrito em Java.

A proposta do JSP era prover um método de desenvolvimento de *Servlet* centrado na apresentação, ou seja, uma página JSP não é nada mais que um *Servlet* automatizado.

Páginas JSP são processadas no servidor, onde são convertidas em *Servlet* para aceitar as requisições HTML e responder neste mesmo formato. Quando acontece esta conversão o arquivo inteiro é convertido em código Java puro, sem as *tags* "<%" e "%>" típicas do JSP. Esta conversão é dividida em duas fases(1):

- ✓ **Fase de Tradução:** É feita normalmente apenas uma vez, a menos que os *scripts* JSP sejam alterados (neste caso a tradução será repetida). Assumindo que não existam erros de sintaxe nos *scripts*, o resultado será a implementação de uma página JSP que, utiliza a interface *Servlet*, como mostrado (na figura 4) abaixo:

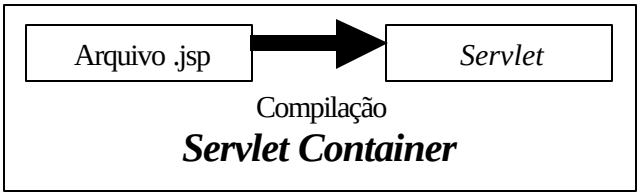

fig-4: *Servlet Container*.

Na especificação do JSP 1.1, esta fase é tipicamente chamada pela máquina JSP ao receber uma requisição de entrada pela primeira vez, onde, com isso, as páginas JSP são pré-compiladas em arquivos ".class", que servem para reduzir o atraso no Start-Up que ocorre quando as páginas JSP respondem pela primeira vez a uma requisição vinda do cliente.

Os detalhes de localização dos arquivos de origem e dos arquivos ".class", bem como seu armazenamento, são feitos de acordo com cada implementação.

- ✓ **Fase de Processamento de Pedido:** É nesta fase que se processa a requisição vinda do cliente, que na maioria das vezes ocorre através do preenchimento de um formulário disponível na página HTML.

Java é considerado lento em comparação com outras linguagens como *Visual Basic* e C++(3), mas este atraso é um preço que alguns programadores aceitam pagar em virtude da independência de plataforma, pois um programa feito em Java após ser compilado, ficará sendo composto apenas de *bytes* genéricos, podendo ser executado em múltiplas plataformas, pelo simples fato de serem interpretados em tempo de execução pelo JVM (*Java Virtual Machine*) específico de cada Sistema Operacional.

Estes códigos compostos agora de *bytes* genéricos podem ser transportados para qualquer outro Sistema Operacional compatível com JVM sem nenhuma alteração. Os códigos só perderão esta portabilidade total se durante a programação forem usados algum dos recursos oferecidos pela GUI (*Graphic User Interface*).

O *Servlet* é mais rápido que programas escritos em Java, como *applets* e aplicações *stand-alone*, porque ele é processado no servidor, e não possui qualquer interface GUI que não seja HTML. As requisições vindas para o *Servlet* só serão interpretadas na primeira vez que o cliente pedir, isso demora alguns segundos, mas após a primeira chamada ser processada todas as demais serão muito mais rápidas pois o interpretador do código ficará gravado no servidor.

4.3 - Características

Servlets e JSP possuem muitas características comuns, e podem ser usadas para desenvolver páginas de conteúdos dinâmicos, por causa disso pode haver alguma confusão na hora da escolha.

É aconselhável usar *Servlet* como uma extensão da tecnologia de *WebServer(1)*, incluindo a implementação de componentes especializados em controlar serviços como autenticação, validação de Banco de Dados e outros.

A máquina JSP é um *Servlet* especializado executando sobre uma máquina *Servlet*, e só lida com dados textuais, por isso continuaremos usando o *Servlet* para comunicação com as aplicações e com os *applets*.

Use JSP para desenvolver aplicações *Web* que possuam conteúdo dinâmico, sendo também usadas nos locais de extensões proprietárias em servidores que possuam muitas características para manuseio de conteúdo repetido

Usando *Servlet* para desenvolvimento de páginas, o conteúdo dinâmico será parte intrínseca dele, e ficará embutido nos modelos estáticos da apresentação da interface com o usuário, por isso a menor alteração na interface resultará uma recompilação de todo o *Servlet*, resultando assim em um forte acoplamento entre a apresentação e o conteúdo (isso gera códigos frágeis e inflexíveis).

As características de JSP dependem do modelo de acesso adotado em cada aplicação, de forma que, existem dois modelos filosóficos de arquitetura para tecnologia JSP(1) que são bastante defendidos, e diferem essencialmente na localização em que o processamento da requisição é executado.

4.3.a - Modelo 1:

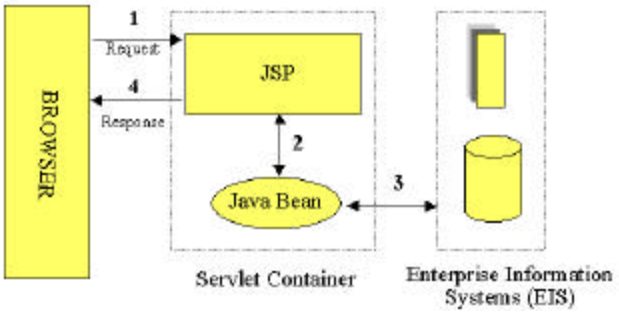

fig-5: Modelo 1 (Arquitetura JSP, *JavaBeans*)

De acordo com a figura (figura 5) acima, a requisição vem através do *browser*, passando direto para o JSP, que é responsável por manda-la pré-formatada para o *Bean* que busca as informações requeridas no banco de dados e as devolve ao *Servlet*, que por sua vez envia de volta ao *browser*. Esta apresentação é tipicamente dividida em 03 (três) camadas, pois todos os acessos aos dados são executados usando *Bean*s.

Embora o modelo 1 de arquitetura seja satisfatório para aplicações simples, pode não ser muito útil para aplicações mais complexas, pois o uso indiscriminado desta arquitetura, significará o aumento de *scriptlets* embutido nas páginas JSP, especialmente se existir um aumento significante de pedidos de processamento para serem executados simultaneamente.

Enquanto isto possa não parecer um grande problema para programadores de Java, com certeza será levado em consideração se suas páginas JSP são criadas e mantidas por *WebDesigners*, que normalmente se preocupam com essas questões em grandes projetos.

O outro lado ruim desta arquitetura, é que cada página JSP terá que ser individualmente responsável por gerenciar o estado das aplicações e também verificar a autenticação e a segurança.

4.3.b - Modelo 2:

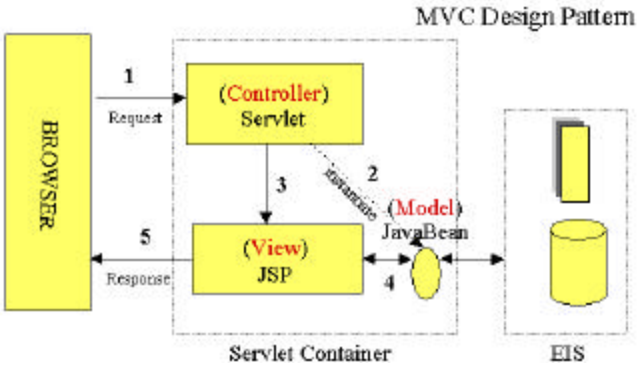

fig-6: Modelo 2 (Arquitetura *Servlet*, JSP, *JavaBean*)

Como vemos na figura (figura 6) acima, a arquitetura do modelo 2, é uma implementação feita no lado servidor do *design* pattern *View*/*Controller*, onde o processamento é dividido entre a apresentação e o *front-end.*

O *front-end* ou *Controller*s, não manipula qualquer ação de apresentação, ao contrário, processam as requisições vindas pelo HTTP, ficando responsável pela criação de qualquer *Bean* ou objeto usado pelos componentes de apresentação, decidindo também de acordo com as ações do usuário, que componentes da apresentação serão mandados como requisição. O *front-end* pode ser implementado como um *Servlet* ou como uma página JSP.

A apresentação ou *View* é feita com páginas JSP, que geram respostas HTML/XML, de acordo com o tipo da interface do usuário que será usada (*browser*, celular, PDA).

A vantagem desta arquitetura é que ao tirar o processamento lógico dos componentes de apresentação, diminui as responsabilidades deles, pois a criação de qualquer objeto ou *Bean* passa a ser do *Controller*, que extrairá o conteúdo dinâmico, para inserção no seu modelo estático.

Conseqüentemente, a separação da apresentação, do conteúdo, será fundamental para a criação de papéis e responsabilidades dos *WebDesigners* e da equipe de programação, além disso um outro benefício, é que o *front-end* apresenta um único ponto de entrada para a aplicação, isto facilitará a segurança, uma apresentação uniforme, e seu gerenciamento.

4.4 - JSP X CGI

O JSP também fornece todos os recursos das aplicações CGI de uma forma simples e robusta, pois diferentemente do CGI, ele não cria um processo no servidor para cada pedido do usuário, além de ser mais fácil criar conexões entre o *browser* e os dados em formatos normalmente incompatíveis com HTML (ex: bancos de dados).

Usando JSP ao invés de CGI, quando vários usuários requisitam ao mesmo tempo páginas JSP, cria-se um processo para manipular cada requisição, estes processos são gerenciadas pelo ***Web Server Process***(1) e podem ser comparadas com os criados pela arquitetura ASP (ASP.dll), sendo uma forma muito mais eficiente que utilizar CGI, que cria um processo inteiro para cada usuário que faz uma requisição.

4.5 - JSP & Banco de Dados

Ao invés de usar conexões ODBC, o JSP usa para acesso a um banco de dados as conexões JDBC (*Java DataBase Connectivity*), que são fornecidas pela linguagem Java, onde o Banco de Dados só precisa possuir um *driver* compatível com JDBC para que se concretize a conexão, possibilitando assim a manipulação dos dados a partir de páginas JSP(1).

Mesmo se um banco de dados não possuir um *driver* JDBC, a conexão poderá ser realizada desde que o mesmo possua um *driver* ODBC, pois existem alguns programas que traduzem JDBC-ODBC e vice-versa (a *Sun MicroSystems* desenvolveu uma "ponte" JDBC-ODBC, e já a embutiu no seu compilador Java gratuitamente), na figura (figura 7) abaixo mostramos uma aplicação JSP acessando uma base de dados.

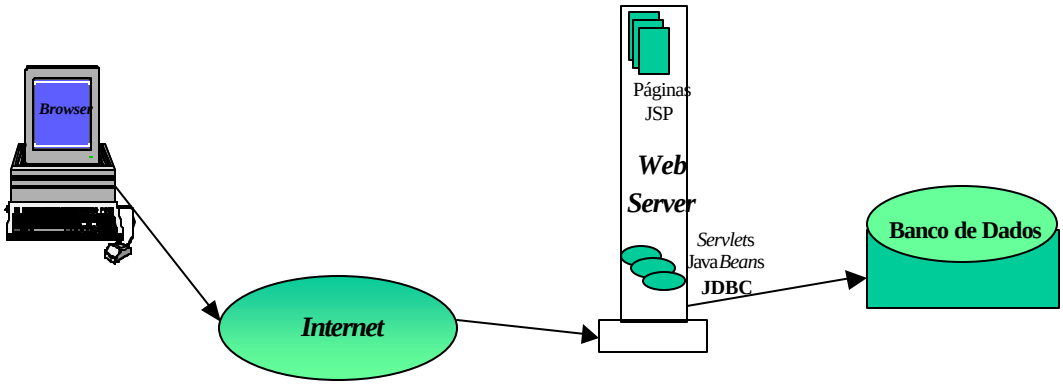

fig-7: JSP acessando BD

4.6 - Back-End

Quando se usa JSP, a separação do conteúdo estático do dinâmico é mantida pelo encapsulamento externo dentro dos componentes *JavaBeans*, que são criados e usados pelas páginas JSP através de *tags* especiais e *scriptlets*, desta forma, quando o *WebDesigner* faz qualquer alteração na apresentação, as páginas JSP serão automaticamente recompiladas e recarregadas no servidor *Web* pela máquina JSP(3).

O JSP Oferece uma oportunidade de desenvolvimento de aplicações em grande escala através das API's do Java, que recomendam usar JSP ao invés de *Servlets* para páginas com recursos de conteúdo dinâmico.

Um programador de *Servlet*, terá facilidade em aprender JSP, ou seja, quem já desenvolve *Servlet*, levará uma grande vantagem porque o JSP nada mais é que uma abstração em alto nível de *Servlet*.

4.7 - Front-End

As páginas JSP após serem escritas, podem ser levadas para qualquer plataforma compatível com JVM e sobre qualquer *WebServer* sem a necessidade de qualquer alteração.

Com JSP, os *WebDesigners* podem desenvolver suas páginas facilmente, já com *Servlets* é necessário um programador experiente de Java com experiência e sólidos conhecimentos da linguagem HTML.

Não existe nenhuma regra que diga que páginas JSP tenham que seguir um determinado formato, com isso ela poderá absorver diversos tipos de clientes como: *browsers* convencionais usando HTML/DHTML, dispositivos de Wireless (celular, PDA) usando XML, o que a torna uma ótima alternativa para aplicações de *E-commerce*.

Se existir em seu servidor o compilador dinâmico HotSpot, isto irá otimizar a performance, pois ele gera código nativo em tempo de execução, que permitirá que a execução da página seja tão rápido como aplicações nativas escritas em C (pelo menos essa é a promessa da *Sun*).

4.8 - Conclusão Parcial

JSP é uma ferramenta de desenvolvimento de paginas *Web* ativas, sendo uma tecnologia desenvolvida pela *Sun MicroSystems* para tentar convencer as pessoas a deixarem o ambiente ASP proprietário da *Microsoft* para um ambiente mais aberto baseado na tecnologia Java, no caso a JSP(4).

Apesar de JSP, usar como linguagem o Java, ter sua portabilidade em um alto nível, ser executada em vários servidores *Web* (IIS, necessita de um *plug-in* para executar páginas JSP), e utilizar como componentes os *JavaBeans*, não faz de JSP a melhor solução para os usuários da plataforma Java, e sim apenas mais uma solução Java que competirá com as diferentes soluções já existentes no mercado(2).

4.9 - Referências

1 - http://www.javasoft.com/products/jsp - 20/09/2000 (inglês)

2 - http://www.*Servlet*s.com/soapbox/problems-jsp.html - 21/09/2000 (inglês)

3 - http://www.asptoday.com/articles/19991022.htm - 21/09/2000 (inglês)

4 - www.uol.com.br/*Web*world/tecnologia/handerson/hander016.htm - 28/09/2000

A Tecnologia PHP (Personal Home Page Tools)

5.1 - Introdução

A linguagem para desenvolvimento de páginas *Web* dinâmicas PHP(1) foi criada em 1994 por Rasmus Lerdorf, e suas primeiras versões não foram disponibilizadas para a comunidade. A primeira versão disponível só surgiu em 1995, e ficou conhecida como "*Personal Home Page Tools*" (ferramentas para página pessoal), que era composta de:

- ✓ Um sistema bastante simples que interpretava algumas macros;
- ✓ Alguns utilitários que executavam no servidor *Web* (ex: um contador);

Após a primeira versão de 1995, surgiu um novo interpretador, que foi reescrito e ganhou o nome de PHP/FI, que era a combinação dos *scripts* PHP, do pacote FI (*Form Interpreter*) que interpretava dados de formulários HTML e de um suporte ao banco de dados mSQL.

Em pesquisas que foram realizadas na *Internet* mostraram que a disseminação da linguagem PHP/FI foi estupenda, já que em menos de um ano ela era usada por cerca de 15.000 páginas em todo mundo, e em dois anos esse número subia para mais de 50.000 páginas.

Após esta enorme aceitação por parte do mercado, houve uma mudança no seu desenvolvimento, o que antes era feito por Rasmus com contribuições de algumas pessoas espalhadas pelo mundo, agora seria desenvolvido por uma equipe de programadores bem mais organizada, que comandados por Zeev Suraski e Andi Gutmans reescreveram o interpretador que foi a base para a versão 3 (atualmente o PHP está na versão 4).

5.2 - Arquitetura

Por ser uma linguagem de *script* que é executada pelo *WebServer*, ela sempre ficará embutida no código HTML, portanto será necessária a instalação de um interpretador para esta linguagem no *WebServer* (o Apache já vem com este interpretador embutido), a cópia deste interpretador para sistemas UNIX está disponível gratuitamente para *download* na *Internet* (http://www.php.net), enquanto que para o sistema operacional *Windows* (NT 4 e 9x) será necessária a aquisição de uma licença. Na figura (figura 8) abaixo, descrevemos o funcionamento de uma arquitetura PHP.

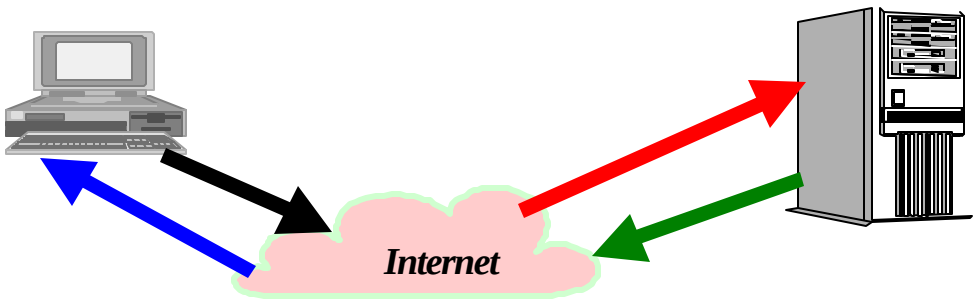

fig-8: Uma arquitetura PHP

① O *browser* acessa a página disponível e faz uma requisição.

② A requisição é passada através da *Internet* para o *WebServer* que possui o interpretador da linguagem PHP.

③ O servidor executa o código PHP e devolve em formato HTML para a página na *Internet*

④ A *Internet* devolve a resposta da requisição feita para o servidor *Web* através do *browser*

5.3 - Características

PHP é uma mistura das linguagens C, Perl e Java com HTML, porém um pouco mais limitado que elas, já que PHP é uma linguagem usada exclusivamente em aplicações baseadas na *Web* e as outras também podem ser usadas para aplicações *stand-alone*.

O que distingue PHP de algo como um *applet JavaScript*, é que o *applet* é executado no lado do cliente usando o *browser*, enquanto que *scripts* feitos em PHP são executados em qualquer servidor *Web*.

Por ser uma linguagem *Open Source*(4), ou seja, de código aberto, ela aceita inclusões de novas funcionalidades por parte dos usuários, isso a torna uma poderosa concorrente para as linguagens ASP e JSP, que atualmente dominam o mercado, pois já existem diversas funções desenvolvidas por usuários/empresas disponíveis na *Internet*, que irão facilitar e otimizar a criação de páginas *Web* dinâmicas.

Um *script* PHP que executará em um *WebServer*, fornecerá para o cliente apenas os resultados (em formato HTML) da sua execução, sem que ele jamais desconfie da existência do código PHP que está por trás de todo este processo.

Quando se utiliza PHP, deve-se ficar atento principalmente na hora de declarar variáveis e de escrever os comandos corretamente, pois PHP é *case sensitive*, ou seja, a variável "teste" é diferente da variável "TESTE".

PHP suporta todos os padrões mais usados na *Internet* como, IMAP (*Internet Message Access Protocol*), FTP (*File Transfer Protocol*), POP (*Post Officce Protocol*), XML (*eXtensible Markup Language*), WDDX (*Web Distributed Data eXchange*), LDAP (*Lightweight Directory Access Protocol*), NIS (*Network Information Service*), e SNMP (*Simple Network Manegement Protocol*), com grande facilidade, sem que para isso os programadores precisem fazer malabarismos ou usar ferramentas de terceiros.

Também utiliza checagem de tipos dinâmica, ou seja, uma variável pode conter valores de diferentes tipos em momentos distintos durante a execução do *script*. Por este motivo não é necessário declarar o tipo de uma variável para usá-la. O interpretador PHP decidirá qual o tipo daquela variável, verificando o conteúdo em tempo de execução.

Somente na versão 4 do PHP é fornecido um suporte para:

- ✓ Objetos Java, que irá executar em qualquer Sistema Operacional que possua uma JVM;
- ✓ Aplicações distribuídas em ambientes *Windows* que utilizam COM;

Estas aplicações poderão ter seus componentes reutilizados, fornecendo ao *WebDesigner* a possibilidade de separar a lógica do negócio dos componentes, e com isso usar o PHP somente para otimizar suas respostas.

Com o uso da ferramenta Zend Optimizer, que é um módulo adicional cedido gratuitamente pela Zend Technologies(5) a aplicação desenvolvida em PHP, executará cerca de 40% a 100% mais rápido que uma aplicação que não faça uso deste módulo.

Não podemos deixar de destacar outras características como:

- ✓ Suporte à orientação a objetos;
- ✓ Interação e persistência com bancos de dados;
- ✓ Criação de imagens GIF;
- ✓ Serviço de autenticação HTTP;
- ✓ Manipulação e personalização de erros;
- ✓ Manipulação de *cookies*;
- ✓ Suporte para *upload* de arquivos;
- ✓ Manipulação de arquivos remotos;

5.4 - PHP X CGI

Qualquer tarefa executada por um programa CGI, poderá ser feita utilizando PHP (ex: trabalhar com *forms*), a diferença está em escrever um *script* HTML com PHP embutido ao invés de escrever um programa com comandos de saída HTML.

Igualmente à outras linguagens de desenvolvimento de páginas *Web* dinâmicas, a PHP também oferece melhor performance que o CGI, pois além de ser executado no servidor ele não cria um processo para cada requisição do usuário.

5.5 - PHP & Banco de Dados

Com o uso da PHP, você possuirá acesso nativo a alguns bancos de dados existentes, além de suportar ODBC, o que dará a possibilidade de trabalhar praticamente com todos os grandes bancos de dados do mercado. Usando a versão 4 da PHP(1) poderá também fazer conexões com banco de dados através de *JavaBeans*. Na figura (figura 9) abaixo ilustramos uma aplicação PHP acessando uma base de dados.

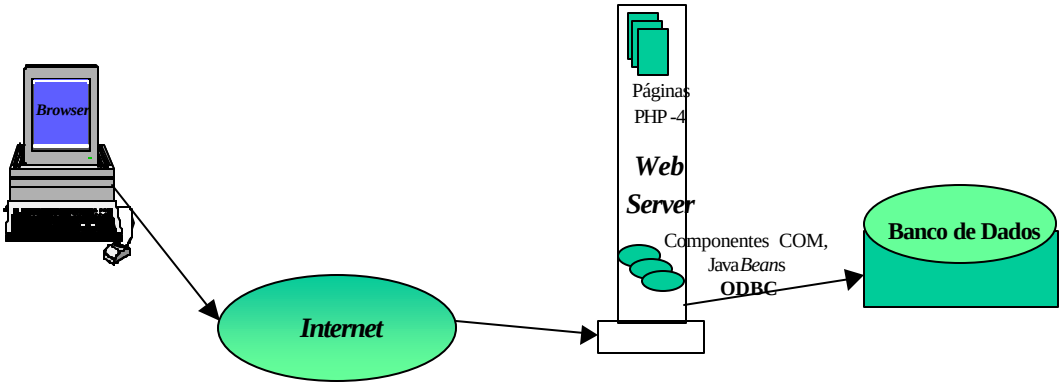

fig-9: PHP acessando BD

Agora iremos mostrar apenas os comandos básicos relativos ao banco mySQL(3), já que o PHP junto com o mySQL, está sendo muito utilizado por um grande número de *WebDesigners* que usam o ambiente Linux/Apache, vamos a eles:

- ✓ Fazer a conexão com o servidor de Banco de Dados – a conexão é feita usando a função **MYSQL_CONNECT(string1, string2, string3)** que retorna um valor int. A variável string1 é o endereço do servidor, a string2 é o login do usuário e a string3 a sua senha;
- ✓ Selecionar a base de dados a ser utilizada – após a conexão ser estabelecida, é preciso que seja informado o nome do banco de dados residente no servidor, que é feito usando a função **MYSQL_SELECT_DB(string, int)** que retorna um valor int. A variável string é o nome da base de dados, e o int é o número da conexão retornado da função anterior. Se este comando for omitido o PHP selecionará a base de dados usando o valor da última conexão estabelecida.
- ✓ Executar uma consulta SQL – depois de selecionar a base de dados, quase toda a interação com o servidor será feita usando cláusulas SQL que usam o comando **MYSQL_QUERY(string, int),** onde a variável string receberá a cláusula SQL completa (Ex; create table teste (cod int auto_increment primary key, desc char(50)), e o int é o número da conexão retornado da primeira função mostrada. Se a cláusula SQL for um select o valor retornado será um número que identifica o resultado, com isso para exibir as linhas desta consulta será usada a função **MYSQL_RESULT()**

5.6 - Back-End

Como ASP e JSP, ela também executa o código todo no servidor, sendo enviado para o cliente apenas HTML puro, na versão 4 ela trabalha com componentes COM e *JavaBeans*(2). Possui suporte a serviços como Gerência de Redes, Email e outros, através do uso de outros protocolos (IMAP, SNMP, NNTP (*Network News Transfer Protocol*), POP3, http (*Hiper Text Transfer Protocol*)), oferecendo também a possibilidade de abertura de *sockets* para interação com outros protocolos.

PHP possui uma grande capacidade de modificar as variáveis passadas através dos formulários HTML, com isso, será possível realizar tarefas como: envio de um e-mail baseado em informações de uma página, impressão de páginas personalizadas, passagem e armazenamento de informações em um banco de dados etc.

5.7 - Front-End

Configurando um *WebServer* para processar todos os arquivos HTML usando *scripts* PHP, não existirá nenhuma maneira dos usuários perceberem que existe código embutido na página HTML.

Desta forma é possível interagir com bancos de dados e aplicações existentes no servidor, com a vantagem de não expor o código fonte para o cliente. Isso é muito útil em *site*s de *E-commerce*, pois o programa estará sempre lidando com informações secretas (Ex: senhas, número do cartão de crédito).

O desenvolvimento de páginas *Web* dinâmicas usando PHP é muito rápido, pois elas adicionam os *scripts* PHP nos códigos HTML, *scripts* esses que ficam bem delimitados com o uso das *tags* "<?php" e "?>", que especificam o início e o fim de um bloco PHP, diferenciando-o do código HTML.

Suporta um grande número de bancos de dados existentes no mercado, entre eles estão o Adabas D, Dbase, mSQL, Sybase, MySQL, Oracle, Unix dbm, Informix, PostgreSQL, e vários outros, desta forma para construir uma página baseada em algum destes bancos de dados será uma tarefa extremamente simples e fácil.

As versões 3 e 4 do PHP oferecem a facilidade da criação de **templates** HTML, que são usados em um *site* que contenha muitas páginas (Ex: template do cabeçalho e dos frames). Para usarmos esses templates, teremos que adiciona-los em todas as páginas que serão afetadas, onde através da cláusula **include,** é que se permitirá a inclusão no documento HTML de um código que esteja em outro arquivo.

5.8 – Conclusão Parcial

PHP surge como mais uma alternativa para programadores que desenvolvem aplicações dinâmicas para a *Web*, com ou sem acesso à banco de dados, que preferem não utilizar a linguagem Java da *Sun MicroSystems* (JSP), e nem tão pouco ficar "dominado" pelo mundo das soluções proprietárias da *Microsoft* (ASP).

Oferecendo todas as funcionalidades de uma linguagem orientada a objetos(2), PHP possui algumas vantagens como ser multiplataforma, não possuir custo de aquisição, trabalhar com vários banco de dados, possuir uma sintaxe simples e fácil e principalmente já estar sendo usada a um tempo relativamente bom, oferecendo assim, uma grande segurança na hora da escolha de uma linguagem para desenvolvimento de páginas *Web*

A combinação de linguagens que executam no *WebServer*, como PHP, com linguagens que executam no cliente, como *JavaScript*, fornecem ao *WebDesigner* uma excelente forma de aproveitamento dos recursos disponíveis para criar páginas dinâmicas, bem como oferecer uma forma de personalização para cada usuário.

Considerando o grande aumento de servidores que usam PHP, levando em consideração o avanço dado da versão 3 para a 4, e vendo o amadurecimento constante desta linguagem, PHP se tornou uma ferramenta que precisa ser considerada na escolha de uma linguagem para criação de páginas *Web* dinâmicas, pois além de todas essas características citadas, ela oferece alto desempenho, baixo custo, escalabilidade, código aberto, segurança e principalmente portabilidade.

5.9 - Referências

1 - http://www.php.net - 15/09/2000 (inglês);
2 - http://www.phpbuilder.com - 17/09/2000 (inglês);
3 - http://www.vivas.com.br - 19/09/2000;
4 - http://wdvl.*Internet*.com/Authoring/DHTML/ - 01/10/2000 (inglês)
5 - http://www.zend.com - 07/10/2000 (inglês)
6 - www.wapmaster.com.br/default.asp?item=editorial_wml&tt=21 - 20/09/2000

A Tecnologia XML (eXtensible Markup Language)

6.1- Introdução

O XML (*eXtensible Markup Language*), e uma linguagem de marcação dados que obedece aos padrões estabelecidos pelo W3C(3) (*World Wide Web Consortium*) através da SGML (*Standard Generalized Markup Language*), e provê um formato para descrição de dados estruturados, facilitando declarações mais precisas do conteúdo, ganho nos resultados de busca através de múltiplas plataformas, isso permitiu o surgimento de uma nova geração de aplicações de manipulação e de visualização de dados via *Internet*.

Diferentemente do HTML, onde as *tags* são estáticas, estão padronizadas e só podem ser usadas para definir as formatações em geral (fonte, parágrafo, etc), quando usamos o XML podemos definir e criar um número infinito de *tags* para dados estruturados.

Um elemento XML pode ter dados declarados (Ex: preço, título de livro, marca de carro), ou qualquer outro tipo de elemento de dado. Como as *tags* XML são adotadas por *Intranet*s, e também via *Internet*, haverá uma grande habilidade em manipular e procurar por dados estruturados independentemente dos aplicativos utilizados e onde os quais são encontrados, desta forma, quando ele for localizado, pode ser disponibilizado para a rede, exibido em um *browser* (Ex: IExplorer 5), ou então transferido para outras aplicações.

6.2 - Arquitetura

Diferentemente do HTML, que só precisava da criação de um arquivo texto, usando XML para a criar aplicações, será necessária a confecção de alguns novos documentos e também o cumprimento de algumas fases neste processo, são elas:

- ✓ É fundamental a criação ou escolha de um DTD (*Document Type Definition*), pois neste documento será definido todas as propriedades das *tags* como: sua definição, quais delas podem conter outras, seu número, sua seqüência, seus atributos e seus valores;
- ✓ Após a especificação do DTD, vem criação dos documentos XML, que será feita através do recebimento de um formulário preenchido segundo o DTD, os dados contidos neste formulário podem vir de uma consulta a um banco de dados, de uma busca em documentos, ou de uma pesquisa em um catálogo *online*, assim quando o formulário estiver preenchido, ele será enviado a quem o solicitou;
- ✓ Para a interpretação dos documentos XML criados anteriormente, são normalmente usada uma das duas APIs, a SAX (*Simple API for XML*), que é bem mais rápida, ou a DOM (Document Object Model), que cria uma visão árvore, e é o padrão da W3C(1).
- ✓ A exibição dos documentos XML, fica a critério da aplicação, pois existem várias maneiras de se formatar os documentos XML conforme o equipamento (PDA, Micro, Celular, etc) que necessitará da visualização, assim se for um *browser* que entende XML, o documento pode ser enviado diretamente para ele, caso contrário pode-se usar o XSL para transformar o arquivo XML em algo que o *browser* entenda (HTML).
- ✓ O XML separa a apresentação dos dados estruturados, desta forma é o HTML quem especifica como o documento deve ser apresentado na tela por um navegador, deixando para o XML a tarefa de definir o conteúdo do documento, assim o HTML utiliza *tags* para definir formatação (Ex: tamanho e cor de fonte, etc), e o XML(4) utiliza para descrever os dados (Ex:autor, data, etc).

Agora exibiremos uma arquitetura genérica de um modelo XML(7):

- ✓ O *Document Integrator*, é quem decide como as entradas de documentos XML serão processadas;
- ✓ O processo está baseado em *scripts* de alto nível que, serão usados pelo *Integrator* para processar um documento de XML passando-o por diferentes Módulos de Transformação (*Transformation Modules*), onde cada módulo é projetado para dirigir um tipo especial de tarefa.
- ✓ O Módulo de Transformação processa o XML passado pelo *Integrator*, e então retorna um documento de resultado, no formato XML.
- ✓ O *Integrator* então processa o documento resultante, e invoca outros Módulos se necessários, então ele devolve o resultado final ao visitante no formato XML.
- ✓ A capacidade de processamento do sistema XML pode ser acrescida de novos Módulos de Transformação, já que os existentes não necessitam modificações, desta forma o projeto fica flexível e extensível.
- ✓ A arquitetura do nosso Sistema Processador de Documento é mostrada na figura (figura 10) abaixo, que contém um *Document Integrator* e também Módulos de Transformação TM.

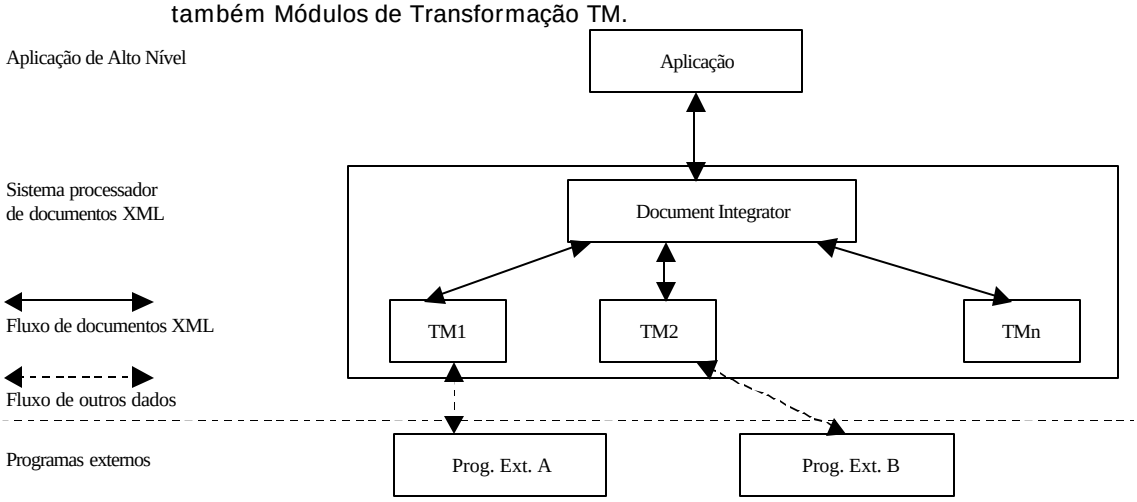

fig-10: Arquitetura XML genérica

- ✓ Como vemos o *Document Integrator* é o núcleo desta arquitetura, sendo responsável pelo recebimento das entradas de dados XML do programa de aplicação.
- ✓ Ao receber um documento de XML, `vindo de um disco ou de uma conexão de rede, o DI processa o documento de entrada de acordo com os *scripts*, e de acordo com a lógica descrita neles, o DI comunica-se com vários TMs, passando-lhes os documentos de XML apropriados
- ✓ Os documentos devolvidos pelo TM, também estão no formato XML, podem ser armazenados temporariamente pelo DI, que pode processa-los, ou passa-los para outro TM se necessário.
- ✓ Depois de colecionar todos os resultados do TM, o DI combina os resultados usando XSL-T, e então retorna este resultado ao programa de aplicação.
- ✓ O papel do DI é agir como uma ponte entre o programa de aplicação e o TM, podendo passar dados entre eles.
- ✓ O processamento atual de documentos XML é delegado para vários TMs, porém, o DI tem que ser capaz de, passar os dados em XML entre os TMs, isto significa a transformação constante de documentos XML.
- ✓ Claro que, tal tarefa de transformação poderia ser delegada para um TM apropriado, porém, para maior eficiência, nós decidimos adicionar esta capacidade para o DI (isto é realizado incluindo XSL-T no DI)

6.3 - Características

A facilidade da implementação e do desenvolvimento para representação estruturada dos dados, mostrou ser um dos pontos fortes do XML(6), bem como a força industrial do seu formato de documentos estruturados em árvore (DOM).

O XML provê um padrão de codificação, do conteúdo, das semânticas, e dos esquemas que será disponível para uma grande variedade de aplicações (desde as mais simples até as muito complexas), dentre elas:

- ✓ Um simples documento;
- ✓ Um registro estruturado (Ex: ordem de compra de produtos);
- ✓ Um objeto com métodos e dados (*JavaBeans* ou componentes ActiveX)(6);
- ✓ Um registro de dados (Ex: o resultado de uma consulta a um BD);
- ✓ Apresentação gráfica, como interface de aplicações de usuário;
- ✓ Todos os links entre informações e pessoas na *Web*.

Após o cliente receber o dado, ele pode ser manipulado, editado e visualizado sem a necessidade de extraí-lo novamente do servidor, com isso, os servidores ficam menos sobrecarregados, reduzindo as necessidades de computação e requisição de banda passante para as comunicações entre cliente e servidor.

Recursos como folhas de estilo usando o XSL (*eXtensible Stylesheet Language*) e o CSS (*Cascading Style Sheet*) também são utilizados para apresentação dos dados em um *browser*, pois o XML separa os dados da apresentação de forma que eles poderão ser visualizados em diversas aplicações com diferentes folhas de estilo.

Por promover uma separação entre a apresentação e os dados, o XML permite que haja a integração de diversas fontes. No exemplo abaixo podemos dizer que: informações de consumidores, compras, resultados de buscas, pagamentos, catálogos, e outros, podem ser convertidos para XML no *middle-tier(1)* (espécie de servidor), permitindo que os dados sejam trocados *online*, tão facilmente quanto a exibição dos dados nas páginas HTML, assim todos os dados em XML podem ser distribuídos através da rede para os clientes que desejarem, como vemos na figura 11.

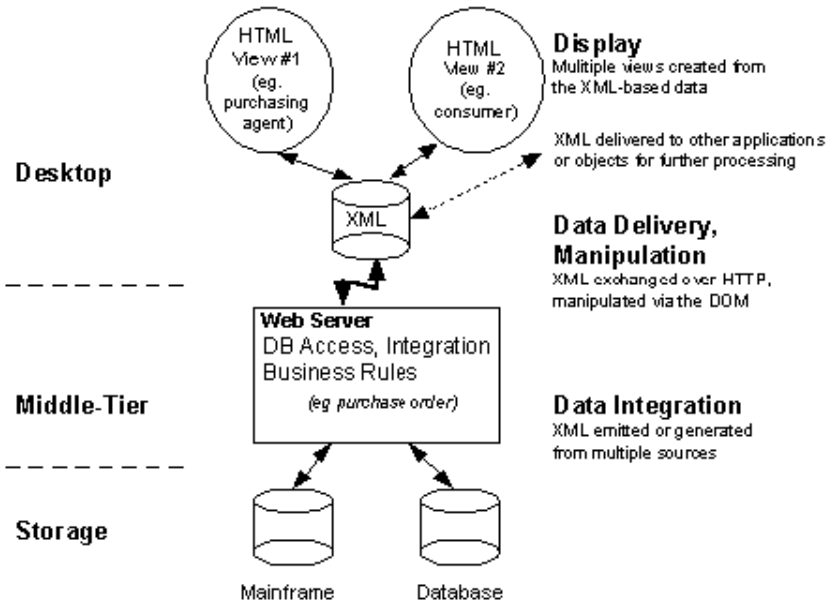

fig-11: Aplicação *Web* em três níveis, por ser flexível permite a troca de dados entre *mainframes* e clientes (*desktops*).

6.4 - Vantagens

O principal objetivo do XML, é trazer flexibilidade e poder às aplicações *Web*, com isso ela resultará em alguns benefícios como:

- ✓ Os dados em XML devem possuir *tags* únicas, o que permitirá por exemplo, buscar livros pelo nome do autor, já que se colocarmos o nome de um autor em qualquer mecanismo de busca hoje, a resposta será todos os *site*s que possuam uma referência a este nome, não importando se ele é ou não o autor de um livro. Com o XML as buscas na *Web* serão facilitadas, pois será possível encontrar livros pelo nome do autor, bem como por título, editora, etc;
- ✓ Permite múltiplas formas de visualização, ou seja um único documento pode ser apresentado de diversas formas, de acordo com o gosto do usuário ou de acordo com a aplicação usada, esta visão será processada localmente, no cliente;
- ✓ Facilita o desenvolvimento de aplicações mais flexíveis, ou seja, em n-camadas, assim os dados podem ser distribuídos para as aplicações, objetos, *desktops* ou servidores intermediários para processamento;
- ✓ Com o aumento da procura em múltiplos e incompatíveis bancos de dados é praticamente impossível, a condensação desses dados, mas usando o XML esses dados podem ser facilmente combinados, necessitando apenas da presença de pelo menos um servidor intermediário que, consolidaria via software as informações contidas nos bancos de dados das extremidades da rede;
- ✓ O XML permite atualizações granulares, evitando que uma simples modificação (inclusão, exclusão, atualização), em um documento, resulte na necessidade de atualização completa do mesmo;
- ✓ Possui fácil distribuição na *Web*, pois como o HTML, XML, por ser um formato baseado em texto aberto, pode ser distribuída via HTTP sem necessidade de modificações nas redes existentes;
- ✓ Como os documentos XML separam completamente os dados da apresentação, o usuário terá liberdade para escolher sua forma de visualização preferida;
- ✓ Devido à repetição das *tags* usadas para definir a estrutura dos dados, a compressão de documentos XML é feita usando os padrões de compressão do HTTP 1.1.

6.5 - XML X HTML

As principais diferenças e semelhanças entre estas duas linguagens vindas da SGML são as seguintes(2):

	HTML	XML
Função	Descreve o formato da apresentação	Define o conteúdo, ou seja os dados
Tags	Limitadas e estáticas	Ilimitadas e definidas pelo usuário
Aplicação	Inadequada para gerenciar um grande volume de dados	Formatar os dados e permitir sua apresentação com folhas de estilo XSL
Funcionalidade para *E-commerce*	Não oferece, sendo necessários o uso das linguagens de *scripts* (ASP, JSP, PHP), ou de programas CGI.	Excelente, pois separa a apresentação, do processamento dos dados

6.6 - Conclusão Parcial

Quando ouvirmos falar de XML, o primeiro pensamento que deve vir em nossa mente é que, ele é simplesmente um documento que segue uma notação de texto hierarquicamente estruturado armazenado em um arquivo texto.

O XML é a mais nova coqueluche relativa ao desenvolvimento de páginas *Web* ativas, e atualmente já se especula que daqui a pouco tempo será este o padrão do mercado para apresentação de informação da *Internet*.

Por ser independente de plataforma, o XML, é apoiado por muitas ferramentas de todos os grandes fabricantes de software como *Microsoft*, IBM, *Sun* e outros, que também estão investindo "pesado" nesta nova tecnologia *Web*.

6.7 - Referências

1 - http://www.gta.ufrj.br - 18/09/2000

2 - http://www.ccuec.unicamp.br/ - 18/09/2000

3 - http://www.w3.org/ - 19/09/2000 (inglês)

4 - http://www.upf.tche.br/computacao - 20/09/2000

5 - www.wapmaster.com.br/default.asp?item=editorial_wml&tt=21 - 20/09/2000

6 - http://lci.upf.tche.br/~27902/xml/ - 20/09/2000

7 – Distributed and Scalable XML Document Processing Architecture for E-Commerce Systems – E-Business Technology Institute – The University of Hong Kong - 2000

Avaliação Comparativa

7.1 – Introdução

A partir do conhecimento prévio de um problema, é que poderemos levar em consideração quais fatores serão mais críticos, durante o processo de personalização de *WebSites*, já que para algumas aplicações o fator que mais pesa é o custo, para outras pode ser o desempenho, escalabilidade ou até mesmo a portabilidade.

Outros fatores que sempre serão considerados em uma comparação de tecnologias para personalização de *WebSites* serão abordados na seção 7.2 deste trabalho.

7.2 – Tabela Comparativa

Recurso	ASP - *Active Server Pages*	JSP - Java *Server* Pages	PHP - Personal Home Page Tools	XML – eXtensible Markup Language
Integração com fontes de dados	Trabalha com qualquer banco de dados compatível com ODBC	Trabalha com qualquer banco de dados compatível com ODBC e JDBC	Possui acesso nativo a alguns bancos de dados e pode utilizar ODBC	Possui um servidor intermediário para consolidar os dados de fontes heterogêneas.
Compatibilidade com BD.	Sim. (usando COM)	Sim. (usando a API JDBC™)	Sim. (acesso nativo ou ODBC)	X-X-X
Componentes	COM	*Beans* ou *Tags*	Na versão 4 usa *Beans* e/ou COM	X-X-X
Componentes reutilizáveis multiplataforma	Não.	Sim. arquitetura de componentes Java *Beans*™	Sim, na versão 4 utiliza uma arquitetura de componentes Java *Beans*™	X-X-X
DHTML	Sim.	Sim.	Sim.	Sim.
Editor de texto	Todos	Todos	Todos	Todos
Escalabilidade	Sim.	Sim.	Sim.	Sim
Extensão	.asp	.jsp	.php	.xml
Geração do conteúdo separado da apresentação.	Sim. Usando Objetos COM	Sim. Usando *Beans* (*JavaBeans*)	Sim, a versão 4 já está usando *Beans* (*JavaBeans*)	Sim, através do XSL-T
Integração automática dos arquivos .	Sim.	Sim.	Sim.	Sim.
Linguagens suportadas	*VBScript*, *Jscript*	Java™, *JavaScript*™ (intenção de adicionar mais)	PHP	XML
Manutenção de estado	Sim.	Sim.	Sim.	X-X-X
Multiplataformas	Sim, através do uso de ferramentas desenvolvidas por terceiros	Sim.(*Windows* NT e 9x, Linux, Mac OS, etc.)	Sim. (*Windows* NT, Mac OS, Linux, etc.)	Sim.
Múltiplos servidores *Web*	Sim, através de componentes de terceiros (Chili Soft). (83% do mercado)	Sim. (Apache, Netscape, IIS) (mais de 85% do mercado)	Sim. (cerca de 87% do mercado)	Não, atualmente só o Apache, Netscape e IIS versao 4 ou superior
Orientação a Objetos	Sim (usando *Jscript*)	Sim	Sim	Sim. Quando usado com Java
Padrão aberto para a indústria	Não.	Sim. (Parceiros incluem: Oracle, Netscape, IBM & *Web*Logic)	Sim.	Sim.
Preço	Free na *Web*	Free na *Web*	Free na *Web*	Free na *Web (hoje)*
Sessões	Sim (objeto Session)	Sim (HTTPSession)	Sim	X-X-X
Cookies	Sim.	Sim	Sim	X-X-X
Suporte a XML	Sim	Sim	Sim	Sim
Tags customizáveis	Não.	Sim.	Não	Sim
WML	Sim	Sim	Sim	Sim

Conclusão

Após o estudo das novas tecnologias disponíveis no mercado para o desenvolvimento de soluções para personalização de *E-commerce*, pude observar que, nenhuma delas é infinitamente superior à outra, e que em um contexto global todas se equivalem.

Por isso, o que realmente decidirá qual vai ser a tecnologia que será utilizada na sua aplicação de comércio eletrônico, será outros fatores como:

- ✓ Valor do investimento;
- ✓ Profissionais de tecnologia;
- ✓ Plataforma de Hardware existente;
- ✓ Plataforma de Software existente;
- ✓ Controle sobre a aplicação (onde as páginas ficarão hospedadas);
- ✓ Suporte;
- ✓ Relação Custo/Benefício de cada Aplicação.

Abaixo descreveremos 03 (três) cenários onde cada uma das tecnologias seria mais adequada, vamos a eles:

- **Cenário 1:** Vamos começar uma aplicação de *E-commerce*, ou seja, não existe nada (hardware ,software, pessoal), e temos que nos preocupar muito com os custos, ou seja, **custo zero**. Então a partir deste parâmetro cheguei a conclusão através deste estudo que a melhor maneira seria com o uso da seguinte arquitetura:
  - ✓ Sistema Operacional de rede – Linux (de graça);
  - ✓ Banco de Dados – mySQL (de graça);
  - ✓ Servidor *Web* – Apache (de graça);
  - ✓ Linguagem de *Script* – PHP versão 4 (de graça);
  - ✓ Servidor *Proxy* – *Squid* (de graça);
  - ✓ Protocolo de Segurança – SSL (de graça);

  Com esta arquitetura acima citada, que já está em uso na *Internet* em diversas aplicações de todos os portes, com excelentes resultados de performance, segurança e confiabilidade, podemos começar e expandir durante algum tempo nosso *site* de comércio eletrônico via *Web* (www.mysql.com).
- **Cenário 2:** Empresa do ramo de tecelagem, que já possui uma equipe de desenvolvedores treinada na ferramenta *Visual Basic* da *Microsoft,* e também uma equipe de suporte para *Windows NT Server,* está agora querendo implantar um sistema de parceria com seus fornecedores B2B (Business to Business), **utilizando para isso toda a sua infra-estrutura já existente**. A empresa também possui um contrato de suporte e manutenção de sistemas com uma empresa *Microsoft Solution Provider,* à partir dessas características citadas, optamos por usar o:
  - ✓ Sistema Operacional de rede – *Windows NT Server* (já existente);
  - ✓ Banco de Dados – SQL *Server* (já existente);
  - ✓ Servidor *Web* – *Internet Information Server* 4 (de graça);
  - ✓ Linguagem de *Script* – ASP (de graça);
  - ✓ Servidor *Proxy* – *MS Proxy Server 2* (já existente);
  - ✓ Protocolo de Segurança – SSL (de graça);

  De acordo com os dados apresentados, nós optamos pela plataforma ASP, pois a empresa já possuía uma equipe de desenvolvedores na linguagem *Visual Basic,* e como já apresentamos o ASP possui suporte à *VBScript*, ou seja, o custo com treinamento de pessoal seria zero, além de não precisar ficar trabalhando com plataformas diferentes (www.microsoft.com), ela também não precisará mudar a cultura de softwares já adquirida por seuss usuários.

- **Cenário 3:** Uma *softhouse* que desenvolve todos os seus sistemas corporativos utilizando a linguagem Java e técnicas de orientação a objetos, agora está entrando no mercado de desenvolvimento para *Web.* **Toda sua equipe de desenvolvedores já domina as tecnologias *Servlet, JavaBeans e Enterprise JavaBeans*.** Como a empresa resumirá seu trabalho em desenvolver o *software* e implanta-lo, não podemos amarra-lo à uma só tecnologia, por isso decidimos pela arquitetura que segue mostra abaixo:
  - ✓ Sistema Operacional de rede – o existente no cliente;
  - ✓ Banco de Dados – o existente no cliente;
  - ✓ Servidor *Web* – de preferência o Apache (de graça);
  - ✓ Linguagem de *Script* – JSP (de graça);
  - ✓ Servidor *Proxy* – o existente no cliente;
  - ✓ Protocolo de Segurança – SSL (de graça);

  Por ser uma empresa de desenvolvimento, a questão tempo é crucial, pois, sua sobrevivência estará associada à rapidez de desenvolvimento de cada *software.* Então se ela já possui todo seu pessoal de desenvolvimento familiarizado com as ferramentas do Java, nada mais natural que, optarmos pelo JSP, desta forma o ganho na produtividade aparecerá muito mais rápido que com o uso de outras soluções (www.javasoft.com).

Links Interessantes

http://www.revista.unicamp.br/ - 11/09/2000;
http://www.widesoft.com.br/*site*/ - 11/09/2000;
http://www.informationweek.com.br/ - 12/09/2000;
http://www.chapmanrg.com/IMR/IMRECOMM.HTM - 17/09/2000; (inglês)
www.umich.edu/~cisdept/mba/CIS518/assignments.html - 19/09/2000; (inglês)
http://news.cnet.com/ 20/09/2000; (inglês)
http://www2.uol.com.br/info/index.shl - 22/09/2000;
http://www.yahoo.com - 24/09/2000; (inglês)
http://www.advancedsolutions.com.br/cgi-bin/*Web*developer - 26/09/2000; (inglês)

Anexos

10.1 – ASP acessando Banco de Dados usando ODBC

```
<html>
<head><title>dbfull1.asp</title></head>
<body bgcolor="#FFFFFF">
<% ' Conexão com o banco
       set conntemp=Server.createobject("adodb.connection")
       'Aqui abrimos um banco com DSN "estudante", usuário "estudante" e uma senha "magic".
       conntemp.open "Estudante","Estudante","magic"
       'Aqui selecionamos todos os registros da tabela autores
       set rstemp=conntemp.execute("select * from autores")
       qtde_campos=rstemp.fields.count -1
%>
<table border="1">
<tr>
<td valign="top">---</td>
<% 'Preenche a primeira linha com o nome dos campos
       for i=0 to qtde_campos
%>
<td><b>
<%
       =rstemp(i).name
%>
</b></td>
<%
       next
%>
</tr>
<% ' Preenche a tabela com os registros do banco
       do while not rstemp.eof
%>
<tr>
<td valign="top"><a HREF="dbfull2.asp?str_id=
<%
       =rstemp("AU_ID")
%>
">Editar</a></td>
<%
       for i = 0 to qtde_campos
%>
<td valign="top">
<%
       = rstemp.fields(i).value
%>
</td>
<%
       next
%>
</tr>
<%
       rstemp.movenext
       loop
       conntemp.close
%>
</table>
</body>
</html>
```

10.2 – JSP acessando Banco de Dados usando JDBC

```
public class FAQ  {
         // Variáveis Globais
         // Declara o DSN que aponta para o arquivo faq.mdb
         private String DBLoc = "jdbc:odbc:FAQ";
         // Declara o driver
         private String DBDriver = "Sun.jdbc.odbc.JdbcOdbcDriver";
         // Declara o recordset
         private ResultSet RS = null;
         //Declara a conexão
         private Connection conn = null;

         //-------------------------------------------------------
         // Método executeQuery – para executar consultas.
         // Retorna um Resultset como no ADO
         //-------------------------------------------------------
         public ResultSet executeQuery(String stmt)  {
                  // Testa para ver se a conexão foi bem sucedida
                  if(conn == null)
                   RS = null;
                Else  {  // Tenta executar a consulta
                    try {
                     Statement  s = conn.createStatement();
                      RS = s.executeQuery(stmt);
                     }
                   catch (SQLException e) {} // Lança um erro
        }
         return(RS); // Armazena o valor da consulta na variável RS
}
```

Este JavaBean de exemplo conecta com um BD Access chamado faq.mdb através da classe FAQ. Uma página JSP, poderá instanciar este *Bean*, navegar através do ResultSet e imprimir os resultados na página FAQ.jsp que mostraremos abaixo:

```
<H3>Perguntas mais frequentes sobre Gatos</H3>
  <b>Raça: <%= raça%><b><p>
    <% out.println(b_FAQ.DBConnect());
       RS = FAQ.executeQuery("SELECT pergunta, resposta” +
                             “FROM gatos WHERE raça = '" + raça + "'");
       if(RS == null)
           out.println("<P>Não existe perguntas para a raça ”+raça+</P>");
       else  {
          count=1;
          while (RS.next()) {
           out.println("<TABLE BORDER=0><TR BGCOLOR=#D9D9F3><TD><B><OL>");
           out.println("<LI><A Nome=\"#"+count+"\"></A>");
           out.println(RS.getString("Pergunta”));
           out.println("</TD></TR></TABLE></OL></B>");
           out.println(RS.getString("Resposta") + "<p>");
           count++;
         }
       }
    %>
  </BODY></HTML>
```

10.3 – PHP acessando Banco de Dados usando conexão nativa

Agora veremos como ficará o código para executar uma query SQL numa base de dados mySQL, com um exemplo que cria uma tabela chamada exemplo e adiciona alguns dados:

```
<html><head><title>PHP & BD mySQL</title></head>
<body>
  <?php
    ‘Declara a variável $conexão
    $conexao = mysql_connect(“localhost”, “root”, “phppwd”);
    mysql_select_db(“ged”, $conexao);
    ‘Cria a variável de criação da tabela exemplo
    $cria = “CREATE TABLE exemplo (codigo INT AUTO_INCREMENT PRIMARY KEY,
nome CHAR(40), email CHAR(50))”;
    ‘Cria as variáveis de inserção na tabela exemplo
    $insere1 = “INSERT INTO exemplo (nome,email) VALUES (“Carlin”,
”caomf@uol.com.br”);
    $insere2 = “INSERT INTO exemplo (nome,email) VALUES (“Fabio”,
”pilatt@uol.com.br”);
    $insere3 = “INSERT INTO exemplo (nome,email) VALUES (“Carlos”,
”carlin@ivia.com.br”);
    ‘Faz a criação da tabela exemplo no banco de dados
    mysql_query($cria, $conexao);
    ‘Insere valores na tabela exemplo
    mysql_query($insere1, $conexao);
    mysql_query($insere2, $conexao);
    mysql_query($insere3, $conexao);
    ‘Criando a variável $consulta
    $consulta = “SELECT nome, email FROM exemplo WHERE email LIKE ‘cao’”;
    ‘Executa a consulta e armazena o resultado na variável $resultado
    $resultado = mysql_query($consulta, $conexao);
    ‘Imprime o nome da primeira coluna de $resultado
    printf("Nome: ", mysql_result($resultado,0,"nome"), “<br>\n”);
    ‘Imprime o email da primeira coluna de $resultado
    printf("e-mail: ", mysql_result($resultado,0,"email"),“<br>”);
  ?>
</body></html>
```

10.4 – Glossário

Termo	Significado
ADO (ActiveX Data Objects)	Ferramenta da Microsoft que é uma coleção de objetos utilizados para qualquer ação sobre os registros em bancos de dados ODBC
Apache	Servidor de páginas HTML
API (Aplication Program Interface)	Descrição do conjunto de interfaces para a interação com programas externos
Aplicações stand-alone	Programas que executam em uma única máquina
Applets	Programa que possui pedaços de código HTML embutidos, que executa no cliente.
ASP (Active Server Pages)	Linguagem de script para desenvolvimento de páginas dinâmicas da Microsoft
B2B	Negócio feito entre duas empresas utilizando à Internet como meio eletrônico
B2C	Negócio feito entre uma empresa e o consumidor final, utilizando à Internet como meio eletrônico
Back-End	Estrutura tecnológica montada para dar suporte às aplicações Internet (ex: Servidores, roteadores, softwares).
C	Linguagem de programação de baixo nível
C++	Linguagem de programação orientada à objetos da Microsoft
CGI (Common Gateway Interface)	Programas feitos para que usuários através da Internet consigam executar tarefas não oferecidas pelo HTML (Ex: acesso à Banco de Dados)
COM (Component Object Model)	Componentes desenvolvidos externamente que são adicionados no IIS, para expandir sua capacidade
Cookies	Arquivostexto que armazenam no computador do cliente informações sobre a sua navegação na Web.
DHTML (Dynamic HTML)	A tecnologia que fornece uma imensa riqueza de efeitos, que servirão para aumentar a interatividade com o usuário
DOM (Document Object Model)	API que cria uma visão em árvore dos documentos XML sendo responsável pela sua interpretação
Download	Cópia através da Internet de qualquer arquivo que esteja contido nela
DTD (Document Type Definition)	Documento que define todas as propriedades das Tags
E-Commerce	Comércio através de meios eletrônicos (no nosso caso Internet)
Email	Mensagem eletrônica enviada através da Internet
Escalabilidade	Propriedade que habilita sua utilização desde pequenos problemas até os mais complexos
Front-End	Parte da aplicação que é apresentada para o usuário
FTP (FileTransfer Protocol) GUI (Graphic User Interface)	Protocolo para transferência de arquivos Interface gráfica de apresentação para o usuário. (ex: Windows é uma GUI)
HTML (Hiper Text Markup Language)	Linguagem padrão para desenvolvimento de páginas Web
IIS (Internet Information Server)	Servidor de páginas HTML da Microsoft
Internet Explorer	Browser da Microsoft
JavaBeans Javascript	Componentes desenvolvidos em Java que esperam por um comando do programa principal para executarem Linguagem de script baseada em comandos do Java
JDBC (Java DataBase Connectivity)	Driver padrão de comunicação com Banco de Dados da linguagem Java
JDK (Java Development Kit)	Compilador da linguagem Java oferecido ela Sun MicroSystems
Jscript (Java Script)	Linguagem de script da Microsoft baseada em comandos do Java
JSP (Java Server Pages)	Linguagem de script para desenvolvimento de páginas dinâmicas da Sun MicroSystems
JVM (Java Virtual Machine)	Ferramenta da Microsoft responsável pela execução de programas feitos em Java
Linguagem Case sensitive	Linguagem que faz distinção entre letras maiúsculas e minúsculas
Linguagem open source	Linguagem de código aberto
Linux	Sistema Operacional de rede que possui código fonte aberto que é distribuído gratuitamente através da Internet
Login	Identificação necessária para executar alguma tarefa, geralmente é acompanhado por uma senha

Microsoft Solution Providers	Empresas habilitadas pela Microsoft, que estão distribuídas pelo mundo, que possuem profissionais com alto conhecimento nos seus produtos
Middle-Tier	Espécie de servidor intermediário que consolida os dados, que são muito usados em aplicações XML que recupera informações de BD heterogêneos
MTS (Microsoft Transaction Server)	Ferramenta da Microsoft responsável pelas transações ocorridas com o Banco de Dados
Netscape Navigator	Browser da Netscape
ODBC (Open DataBase Connectivity)	Driver padrão de comunicação com Banco de Dados do Windows
On-the-fly	Expressão Americana que pode se traduzir como "no mesmo instante"
PERL (*Practical Extraction and Report Language)*	Linguagem prática para extração de relatório, sendo de alto nível, ela é utilizada para produzir e processar formulários, muito utilizada na produção de CGIs.
PHP (Personal Home Page Tools)	Linguagem de script para desenvolvimento de páginas dinâmicas
Plug-in	Programa para executar uma tarefa específica (ex: Animação, tradução)
POP (Point of Presença)	Endereço válido para a Internet
Python	Linguagem de script
REXX	Linguagem de script
SAX (Simple API for XML)	A mais simples e mais rápida API, que é responsável pela interpretação dos documentos XML
Scriptlets	Pedaços de códigos escritos em linguagem Java que podem estar incluídos em ferramentas de editoração de JSP
Scripts	Pedaços de código que são embutidos no HTML para executar uma operação e podem ser escritos em diversas linguagens
Servlet	Programa escrito em Java que possui pedaços de código HTML embutidos, que executa no servidor.
Sessões	Locais do site onde o usuário fornece alguma informação que poderá ser útil durante a navegação dentro deste site
SGML (Standard Generalized Markup Language)	Padrão de linguagem de marcação desenvolvido pelo W3 Consortiun
Sites	Locais na Internet que são formados por páginas HTML, onde se divulgam informações sobre determinado tema.
SMNP (Single Network Management Protocol)	Protocolo para gerenciamento de redes e aplicações
SSL (Secure Socket Layer)	Protocolo de comunicação segura via Internet
Tags	Delimitadores usados em linguagens de marcação, o mais utilizado é "<" e o ">"
Templates	Modelos previamente criados, que poderão ser reutilizados no futuro
Threads	São tarefas para serem executadas
Upload	Cópia para a Internet de qualquer arquivo que esteja em seu computador pessoal
VBScript (Visual Basic Script)	Linguagem de script da Microsoft baseada em comandos do Visual Basic
Visual Basic	Linguagem de programação estruturada da Microsoft
W3C (World Wide Web Consortium)	Entidade não governamental formada por empresas e órgãos do governo dos EUA que, é responsável por todas as padronizações da Internet
WAP (Wireless Aplication Protocol)	Protocolo de aplicação usado para comunicação com dispositivos sem fio (celular, PDA)
WebDesigners	Pessoas que desenvolvem aplicações para a Internet
Wireless	Redes sem fio
XML (eXtensible Markup Language)	Linguagem de marcação para manipulação de dados
XSL (Extensible Stylesheet Language)	Linguagem atualmente composta de duas outras linguagens o XSL-T e o XSL-FO
XSL-FO (XSL Formating Objects)	Um dos componentes da linguagem XSL, que é responsável pela formatação dos objetos
XSL-T (XSL Transformation)	Um dos componentes da linguagem XSL, que é responsável pela transformação de documentos XML em qu7alquer forma de apresentação (WAP, HTML)