View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Repositório Institucional da UFSC

CLÁUDIO BARRADAS SEBASTIÃO

PROPOSTA DE UM MODELO CONCEITUAL DE FERRAMENTA PARA MONITORAMENTO DE DOCUMENTOS NA WEB

FLORIANÓPOLIS – SC 2003

UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CLÁUDIO BARRADAS SEBASTIÃO

PROPOSTA DE UM MODELO CONCEITUAL DE FERRAMENTA PARA MONITORAMENTO DE DOCUMENTO NA WEB

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Profº Dr Luiz Fernando Jacintho Maia

Professor Orientador

Florianópolis, agosto de 2003.

PROPOSTA DE UM MODELO CONCEITUAL DE FERRAMENTA PARA MONITORAMETO DE DOCUMENTO NA WEB

Cláudio Barradas Sebastião

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Área de Concentração de Sistema de Conhecimento e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____
Fernando Álvaro Ostuni Gauthier, Dr

Banca Examinadora

_____
Luiz Fernando Jacintho Maia, Dr. (Orientador)

_____
José Mazzucco Junior, Dr.

_____
João Bosco da Mota Alves, Dr.

Sumário

Lista de Figuras

Lista de Tabelas

RESUMO

A Web pode ser vista de duas formas: serviços e conteúdo. Conteúdo é o conjunto das informações eletrônicas que podem ser publicadas através do meio Web e por serviços designamos o conjunto de funcionalidades que possibilitam a extração, integração, publicação e visualização do conteúdo.

Com esta visão, este estudo comtempla uma grande estruturação de como desenvolver páginas Web e gerencia-las de uma forma prática, segura e responsável, utilizando-se de todas as opções que as inúmeras ferramentas de desenvolvimento Web nos proporcionam.

Este trabalho visa contribuir para um melhor gerenciamento e controle no desenvolvimento de documentos para Web, esplorando várias ferramentas e suas potencialidades para se fazer um controle e monitoramento destes documentos de uma maneira mais eficiente e eficaz.

Assim sendo, este estudo se desenvolve de uma maneira que proporcione que as principais ferramentas usadas no desenvolvimento Web, sejam usadas independente de plataforma de trabalho ou linguagem de programação.

Palavras Chave: Servidores Web, HTTP, ASP, PHP, Java, HTML, CGI, VRML, LOG, Métodos de Pedido, Autenticação, Linguagem de Programação.

ABSTRACT

The Web could be seen in two ways: services and content. Content is the set of electronic information that could be published through the Web medium, and through services we designate the set of functionalities that make possible the extraction, integration, publication and visualization of the content.

With this vision in mind, this study contemplates a large structuring of how to develop Web pages and manage them in a practical, safe and responsible way, utilizing all the options that the several Web development tools give us.

This work seeks to contribute to better management and control of Web documents development, exploring various tools and their potentiality in order to control and monitor these documents in a more efficient and effective way.

Thus, this study develops itself in a way that allows the main tools used for Web development to be used independent from the work platform or programming language.

Keywords: Web servers, HTTP, ASP, PHP, Java, HTML, CGI, VRML, LOG, Ordering Methods, Authentication, Programming Language.

1. INTRODUÇÃO

A era da informação foi precedida por métodos de comunicação novos e avançados. A prensa inventa por Gutenberg tirou os livros das bibliotecas eclesiásticas e os colocou à disposição de todos. Depois disso, o sistema telefônico surgiu para permitir que as pessoas esbalecessem uma comunicação instantânea. Agora a Internet une essas duas tecnologias, juntando pessoas e informações sem que haja um intermediário (editor), no caso dos livros, e sem as limitações das conversações telefônicas feitas com apenas dois interlocutores. Essa é uma nova dimensão – um mundo eletrônico e virtual em que tempo e espaço praticamente não tem significado. Pessoas que vivem em regiões cuja distância chega aos milhões de quilômetros se comunicam sem nunca terem se visto, e há informações disponíveis 24 horas por dia em centenas de lugares. As implicações dessa nova comunicação global e desse novo sistema de informações são assombrosas.

Há décadas a transmissão instantânea de informações vem sendo feita através da televisão. No entanto, muito do que assistimos é cuidadosamente selecionado e editado de acordo com os critérios e os interesses das principais redes e anunciantes. A nova era na televisão começou em 1991, quando o mundo testemunhou o ataque aéreo sobre a cidade de Bagdá através da CNN (Cable News Network), que cobria ao vivo e sem censura um evento militar histórico. Durante toda a guerra do Golfo, a CNN continuou a transmitir ao vivo da zona de batalha. Generais e chefes de estado recebiam informações ao mesmo tempo em que o resto do mundo. Lembre-se de que esse tipo de transmissão global e instantânea de informações já estava sendo feito na Internet (e em outras redes mundiais) há mais de uma década. Mesmo sem todo aquele aparato – as imagens e os heróicos repórteres – a Internet transmitiu boletins ao vivo durante a Guerra do Golfo, o confronto na Praça da Paz Celestial, a tentativa de golpe na União Soviética, a rebelião na Tailândia, os tumultos em Los Angeles e a guerra civil no que costumava ser a Ioguslávia.

Mas há uma diferença entre a televisão e a Internet. Durante a cobertura da Guerra do Golfo, nós éramos os espectadores que, para receber essas imagens, dependíamos de alguns homens e mulheres que tinham câmeras e da empresa que dispunha da tecnologia para apresentá-las. Na Internet, nós somos os repórteres, os espectadores e a equipe de produção, assim como as pessoas que utilizam a rede apenas para se comunicar com colegas e clientes e

para realizar seu trabalho. A expressão “democratização da informação” sempre surge em discussões sobre a Internet, que, sem dúvida, é um fórum democrático. Para a rede, não há menor diferença se você é o dono de uma das maiores empresas dos Estados Unidos, um atendente de loja, um fazendeiro ou um biólogo molecular. Não há distinção em relação a quem está se comunicando, e é o conteúdo das informações que determina o seu público – e não os títulos que você tem. Na maioria dos casos, você é livre para dizer o que quer. A Internet é um ambiente aberto, totalmente livre de censura – um tributo a suas origens nas comunidades acadêmicas e de pesquisa.

Isso tudo faz com que sejam registradas e armazenadas diariamente milhões de informações nos Logs de também inúmeros servidores Web. E em quase sua totalidade esses Logs estão de uma maneira bastante desestruturada e desorganizada, um retrato que nem sempre é fiel ao de uma cadeia de acessos às páginas, isso, porque, se torna virtualmente impossível descobrir com precisão quais foram as seqüências de páginas efetivamente visitadas.

Este trabalho, se propõe então, em fazer um estudo das várias ferramentas e suas funcionalidades, para em seu termino propor uma alternativa de monitoramento de Web sites que de uma maneira geral possa informar uma forma mais prática de desenvolve-los para que os usuários venham usufluir de toda sua potencialidade, ou de uma outra maneira, mostrar a complexidade que se tem para ajustar uma home page ao monitoramento confiável das informações imprecindiveis ao gerente de redes e outros.

2. ORIGEM DA INTERNET

Obviamente, quando surgiu a Internet não era o que é hoje – formada por centenas de conexões e redes mundiais. Ela teve um início humilde – mas muito interessante – como apenas uma rede, denominada ARPANET, que é considerada a “Mãe da Internet”. A ARPANET surgiu em 1969 como uma experiência do governo dos Estados Unidos em redes com comutação de pacotes. A ARPA, a agência de projetos de pesquisa avançada do departamento de defesa dos Estados Unidos (que posteriormente passou a se chamar DARPA), no início permitia que os pesquisadores acessassem centros de computação, permitindo que eles compartilhassem recursos de hardware e de software, como espaço em disco, bancos de dados e computadores. Outras redes experimentais que utilizavam ondas de

rádio e satélites foram conectadas à ARPANET através de uma tecnologia de interconexão criada pela DARPA. No início da década de 1980, a ARPANET original foi dividida em duas redes, a ARPANET e a Milnet (uma rede militar), mas a comunicação continuou sendo feita devido às conexões. Em princípio, essa interconexão de redes experimentais e comerciais foi denominada DARPA Internet, mas depois a forma resumida "Internet" passou a ser a denominação mais comum.

Nos primeiros anos, o acesso à ARPANET se limitava a empresas ligadas à defesa militar e a universidades que faziam pesquisas militares. As redes cooperativas e descentralizadas, como a UUCP, uma rede de comunicações UNIX, e a USENET (User's Network) surgiram no final dos anos 70, inicialmente servindo à comunidade universitária e depois a organizações comerciais. No final dos anos 80, mais redes coordenadas, como a CSNET (Computer Science Network) e a BITNET, começaram a oferecer conexões em âmbito mundial para as comunidades acadêmicas e de pesquisa. Essas redes não faziam parte da Internet, mas posteriormente foram criadas conexões especiais para permitir a troca de informações entre diversas comunidades.

O próximo grande momento da história da Internet foi a criação da NSFNET (National Science Foundation) em 1986, que ligava pesquisas feitas em todo o país a cinco centros de supercomputador. Logo ela se expandiu e passou a conectar redes acadêmicas federais e redes de nível intermediário que ligavam universidades e centros de pesquisa. O passo seguinte foi começar a substituir a ARPANET como rede de pesquisa. A ARPANET foi extinta (e desmantelada) em março de 1990. Depois disso, a CSNET percebeu que muitos de seus membros iniciais (departamentos de ciência da computação) estavam sendo conectados através da NSFNET, e deixou de existir em 1991.

2.1 O Crescimento Exponêncial da Internet

Na época em que a NSFNET foi criada, a Internet começou a crescer, mostrando seus ganhos exponenciais em números de redes, participantes e computadores. Redes internacionais semelhantes se espalharam rapidamente por todo o mundo e eram conectadas às redes americanas.

A Internet é a mais rápida dentre as redes globais. Com freqüência, a velocidade é tratada como **throughput** – a velocidade com que as informações podem ser propagadas através da rede.

Além do crescimento exponencial e da alta velocidade, outro fator que certamente contribuiu muito para a excelente reputação da Internet foi seu sucesso na obtenção de interoperacionalidade. Interoperacionalidade é a capacidade de muitos sistemas diferentes de funcionarem juntos de modo a permitir a comunicação. Essa capacidade só pode ser obtida se os computadores e o hardware da rede obedecerem a determinados padrões.

Apesar de não ser preciso se preocupar muito com eles, os padrões têm grande importância na sua vida diária. Os filmes e o papel que você compra sempre cabem na sua câmera e na bandeja da sua copiadora. As bibliotecas catalogam livros de acordo com um padrão. E quando passa a dominar esse padrão, se é capaz de percorrer qualquer biblioteca e localizar os livros que deseja. As coisas que não obedecem a padrões podem dificultar a sua vida. Os padrões têm a mesma importância no mundo dos computadores e das redes. Sem padrões, apenas computadores semelhantes poderiam se comunicar, criando uma Torre de Babel eletrônica. Os padrões, ou **protocolos**, que a Internet utiliza são considerados "abertos". Isso significa que eles estão publicamente disponíveis e permitem que computadores diversos de fornecedores diferentes se comuniquem. Os protocolos e as redes se combinam para fazer a Internet funcionar.

Em resumo, a Internet permite que se tenha acesso a um número maior de pessoas e a um volume maior de informações mais rapidamente do que se é capaz de imaginar.

2.2 Informações Básicas

O pai da Web, Tim Berners-Lee, desenvolveu o HTTP, junto com as especificações complementares para o HTML e o URI (Uniform Resource Identifier – Identificador de Recursos Uniforme), quando era membro do CERN (Laboratório Europeu de Física de Partículas, em Genebra, Suíça). Ao descobrir que o laboratório, assim como a maioria das organizações, tinha dificuldade para organizar as informações, Tim montou uma proposta para a estruturação das informações do laboratório, por meio de uma teia de documentos interconectados por ligações de hipertexto. A proposta foi um sucesso instantâneo no CERN.

Tim imaginou uma comunidade global de pessoas, que colaborariam para a criação de um repositório mundial de conhecimento, publicando seus próprios documentos interligados. Essa visão finalmente tornou-se realidade, quando publicou um trabalho científico pela Internet, em 1991, e anunciou um novo paradigma para a computação cliente/servidor via Internet.

Nesta época, o CERN tornou disponível seus protótipos de tecnologias cliente e servido para a Web, junto com as especificação preliminares para a definição da comunicação entre eles. O desenvolvimento do HTTP, da HTML e das especificações URI é das especificações URI é atualmente controlado por grupos de trabalhos da Internet Engineering Task force (IETF).

O protocolo HTTP original, Versão 0.9 (Berners-Lee, 1990), foi implementado no protótipo de software publicado originalmente pelo CERN. Essa versão é um subconjunto do protocolo completo, como é conhecido hoje em dia, e define um subgrupo restrito, com o qual todas as versões futuras apresentarão compatibilidade reversa.

A versão 1.0, definida no RFC 1945, é o protocolo utilizado atualmente. Essa versão, no entanto, apresenta problemas significativos relacionados com as possibilidades de expansão, e foi revisada atualmente na forma do HTTP 1.1 (RFC 2068). Uma nova geração de protocolos, HTTPng, é uma outra extensão proposta para o HTTP 1.0 e oferece um método alternativo para solucionar alguns problemas de desempenho do 1.0.

O HTML é apenas um dos formatos aceitos pelos clientes e servidores HTTP (outros formatos geralmente utilizados são texto puro, imagens, imagens, áudio, vídeo, arquivos PostScript e coisas do gênero). O HTML existe desde 1990, e uma de suas melhores descrições que se refere à versão 2.0 ocorreu no RFC 1866.

O conceito de um URI foi descrito pela primeira vez no RFC 1630, por Berners-Lee, em 1994. Desde então, também foram publicadas as especificações de suas formas derivadas, URL e URL.

2.2.1 HTTP 0.9

HTTP 0.9 é um protocolo simples baseado em mensagens, com o qual as versões mais novas do HTTP (1.0 e 1.1) apresentam compatibilidade reversa. O protocolo descreve um paradigma de pedido/resposta, onde um programa-cliente se conecta em um servidor, em um determinado endereço, e faz um pedido GET para transferir um objeto (geralmente um documento HTML), que se encontra no servidor; este responde com o envio do objeto (ou de uma mensagem de erro) para o programa-cliente, antes de encerrar a resposta com o fechamento da conexão.

Um pedido GET é apenas um entre vários tipos de comandos que podem ser emitidos por um cliente HTTP. oferece os comandos POST e PUT para o envio de dados para um

servidor, e o comando HEAD para obter informação sobre um objeto (em vez do próprio objeto).

2.2.2 HTTP 1.0

O HTTP 1.0 é uma atualização do HTTP 0.9, onde foram acrescentados os recursos necessários para oferecer suporte à transferência de diversos tipos de objetos por meio de conexões de rede mais complexas. Os recursos adicionais desta versão do protocolo são:

- Tipos adicionais de pedidos (por exemplo, HEAD e POST).
- Identificação da versão do protocolo em mensagens de pedido e resposta (por exemplo, a primeira linha de uma mensagem de resposta, se inicia por HTTP/1.0, indica que o servidor é compatível com a versão 1.0 do protocolo HTTP).
- Códigos de resposta do servidor indicando o sucesso ou o fracasso do pedido, e mensagens de resposta (por exemplo, a primeira linha de uma mensagem de resposta terminada com 200 OK indica que o pedido foi bem-sucedido).
- Formato de cabeçalho e corpo de mensagem baseados em MIME (Multipurpose Internet Mail Extensions), para a definição tanto do tipo de dado do objeto tranferido com de meta-informações adicionais (por exemplo, um cabeçalho MIME com Content-type: text/html indica que a mensagem de resposta contém um documento HTML).
- Um mecanismo básico de segurança, que utiliza um paradigma de desafio/resposta para a autenticação do acesso. Você já deve tê-lo visto em funcionamento ao acessar uma página HTML, e para tanto precisava fornecer seu nome de usuário e senha.

O MIME é uma extensão da especificação original para cabeçalhos de mensagens do correio eletrônico, encontrada no RFC 822. O MIME AMPLIA O RFC 822 para que as mensagens do correio eletrônico possam conter objetos mais complexos do que somente texto simples. Na verdade, o MIME permite que uma única mensagem contenha tipos de conteúdo deferentes. A especificação MIME foi atualizada nos RFCs 2045, 2046, 2049 e 2047. Os RFCs 2045, 2046 e 2049 substituem os 1521, 1522 e 1590, respectivamente.

No HTTP 0.9 a interação ocorre diretamente entre o cliente e o servidor. Essa visão limitada foi ampliada no HTTP 1.0, que permite conexões por meio de entidades intermediárias, tais como servidores proxy.

O uso do MIME pelo HTTP, para descrever o tipo de dado do objeto contido na mensagem, oferece uma maneira poderosa e que pode ser ampliada para tratar quase todos os tipos de dados. Esse método pode ser usado para tratar texto simples e HTML, assim como conteúdo de multimídia mais sofisticado, como áudio e vídeo.

2.2.3 HTTP 1.1

O HTTP 1.1 representa um salto significativo do HTTP 1.0 e procura corrigir muitas das suas falhas – particularmente nas áreas de desempenho, segurança, tratamento de tipos de dados e sistema de cachê. Sendo mais rigoroso em sua definição (por exemplo, na operação de cachê e de servidores proxy), o http 1.1 deixa muito menos espaço para a má interpretação. Isso deverá produzir implementações mais confiáveis do protocolo.
As melhorias principais do HTTP 1.1 foram:

- Melhor desempenho com a introdução das conexões persistentes como o modo padrão de conexão.
- Maior segurança na base na introdução do método de identificação condensado, que elimina a limitação do método básico de autenticação transmitindo informações de nome de usuário e senha **às claras**.
- Um mecanismo de negociação de conteúdo, que permite que cliente e servidor concordem quanto à melhor maneira de representar um objeto.

O HTTP 1.1 oferece um projeto de cachê de objetos no servidor e um protocolo cliente/servidor para o controle de operações de cachê, para melhorar ainda mais o desempenho do protocolo. O objetivo deste projeto é reduzir os tempos de pedido e resposta, e o consumo de banda disponível na rede, dando retorno somente quando necessário.

O HTTP 1.1 também define o relacionamento de um-para-um entre um servidor e um endereço IP do HTTP 1.0. No HTTP 1.1, um cabeçalho Host determina qual servidor deverá satisfazer ao pedido, definindo o nome do servidor.

2.2.3.1 Conexões Persistentes

O desempenho sempre foi um problema no HTTP, porque cada pedido exige sua própria conexão TCP/IP. Isso pode ser problemático quando o objeto HTML que está sendo

carregado contém referências a outros objetos incorporados- geralmente imagens – e todas devem ser transferidas para atenderem ao pedido original integralmente.

A conexão persistente do http 1.1 resolve em grande parte esse problema, pois a conexão entre cliente e servidor permanece ativa, até que um deles solicite seu fechamento. Isto é feito pela introdução formal do cabeçalho da conexão. Ao contrário do que ocorre no http 1.0, as conexões persistentes são o modo padrão das conexões.

Outro beneficio das conexões persistentes é que os pedidos e as respostas podem ser *canalizados;* por exemplo, um cliente pode enviar uma seqüência contínua de pedidos sem aguardar pelo resultado do pedido anterior. Alem disso, os custos de inicialização e encerramento dos pedidos reduzem-se aos de uma única conexão.

Algumas implementações do HTTP 1.0 (como o Netscape) oferecem suporte a um método experimental, em que o cliente pede uma conexão persistente em particular, usando o cabeçalho Connection: Keep-alive. Isto, no entanto, não é geralmente aceito e pode causar problemas, principalmente se o pedido passar por um servidor proxy que não ofereça suporte ao pedido Keep-alive, que, neste caso, encaminhará para um outro que o aceite.

2.2.3.2 Negociação de Conteúdo

O HTTP 1.1 introduz o método de negociação entre clientes e servidores, e permite que o cliente selecione a *melhor representação disponível* de um objeto solicitado, quando houver opção. O protocolo fornece três métodos de negociação: negociação comandada pelo servidor, negociação comandada pelo cliente e negociação transparente (uma combinação de métodos comandados pelo servidor e pelo cliente).

O método *comandado pelo servidor* é usado quando o servidor tem opções de representação e as regras para a escolha da forma apropriada não forem bem descritas para o cliente. A mensagem de pedido do cliente fornece ao servidor indicações para a decisão, como o tipo de dado aceito, a codificação e outras informações. Há duas grandes desvantagens nesse método: é difícil para o servidor decidir o que é melhor para o usuário, e a lista de alternativas definida pelo cliente poderia impor até 1 KB por pedido.

Na negociação *comandada pelo cliente*, o cliente é quem seleciona a representação apropriada a partir de formas alternativas definidas na resposta inicial. A desvantagem desse método é a necessidade de um segundo pedido, se o cliente optar por uma das representações alternativas.

O terceiro método, a *negociação transparente*, combina ambos os esquemas, comandados pelo servidor e pelo cliente, e pode ser aplicado aos ambientes de cache. Aqui, o cache é capaz de negociar no lugar do servidor de destino, escolhendo em uma lista de representações alternativas. Isso retira um pouco da carga de negociação do servidor e remove os custos de desempenho do segundo pedido do cliente, depois que a resposta correta tiver sido armazenada no cache.

2.2.4 HTTPng

O HTTP Next Generation descreve um protocolo completamente diferente do HTTP 1.X (especificamente 1.0 e 1.1) e não procura preservar nenhuma das características desses protocolos. Embora o HTTP 1.1 e o HTTPng incluam recursos parecidos (conexões persistentes e canalização (pipe-lining) de pedido/resposta), o HTTPng possui eficiência e desempenho melhores.

O HTTPng é mais eficiente em:

- Conexões persistentes, assim como o HTTP 1.1, onde uma única conexão entre cliente e servidor para processar os pedidos é estabelecida. Ao contrário do HTTP 1.1, cada conexão HTTPng é, na verdade, dividida em diversos canais diferentes (sessões virtuais) – um canal de controle para enviar e receber comandos, e um canal para cada objeto solicitado.
- Pedidos e respostas assíncronos permitem que um cliente inicie um novo pedido sem aguardar pela resposta do anterior e que o servidor envie respostas de volta na ordem que desejar, e a qualquer momento.
- São usados formatos simplificados de mensagens, no lugar de comandos e cabeçalhos que podem ser compreendidos por humanos. O HTTPng utiliza uma representação compacta e concisa, baseada na Abstract Syntax Notation (ASN.1 – Notação de Sintaxe Abstrata).
- A negociação de conteúdo pressupõe que seja usado somente um numero pequeno de tipos de objeto que poderão ser codificados em um formato compacto (bitmap – mapa de bits). Poderão ser propostos tipos mais complexos, usando o par nome/valor, e o valor poderá ser adicionado ao bitmap para negociações posteriores.

Além disso, o HTTPng introduz a segurança independente de política, estruturas de cobrança e um mecanismo para oferecer suporte à apresentação obrigatória de informações importantes sobre m objeto solicitado, como informações de autor e copyright. O mecanismo de cobrança

permite que um pagamento seja iniciado em resposta ao pedido de um cliente e que cliente e servidor negociem este processo.

2.3. URIs e Formas Relacionadas

Um URI é uma maneira genérica utilizada para identificar um recurso em particular, em qualquer lugar da rede. Um URI é subdividido em:

- O Uniform Resource Name (URN – nome de Recurso Uniforme) define um nome único par um recurso dentro de um conjunto global de nomes.
- A Universal Resource Characteristic (URC – propriedade de recurso universal) define um conjunto de características ou propriedades sobre o nome – sendo que um desses é um conjunto de localizações possíveis.
- O Universal Resource Locator (URL – Localizador Universal de Recursos) define a localização de um recurso.

Ao contrário do URL, o URN é invariável para a localização de um recurso. Seria possível para um recurso, identificador por um URN, ter várias localizações ou mesmo não existir. A independência de localização dos URNs oferece algum tipo de solução para o problema frustrante dos *vínculos partidos* dos URLs. Dos três, o URL é a única forma utilizada freqüentemente. Um trabalho para a definição de um método de URN está sendo realizado, mas ainda está longe de chegar a um consenso.

2.4 URLs

Um URL (de acordo com a definição no RFC 1738) é uma especificação genérica para a localização de um recurso na rede, nos informando como encontra-lo.

Por exemplo, o URL absoluto

http://www.optimation.co.nz/optimation/seref/feedback.htm

nos informa que o arquivo feeedback.htm pode ser acessado pelo protocolo http, no servidor **www.optimation.co.nz** e no subdiretório optimation/seref.

O componente http do exemplo era conhecido como métodos do URL. O método define o handler (manipulador) usado pelo cliente para acessar o recurso. Normalmente o

handler implementa um protocolo Internet, como sempre acontece. Por exemplo, geralmente o handler **mailto** permite que o usuário envie uma mensagem eletrônica para o endereço de destino especificado como parâmetro de **mailto**. A Tabela 1.1 apresenta alguns dos métodos mais comuns, mas existem muitos outros.

2.4.1 URLs HTTP

No método HTTP, o URL possui a seguinte sintaxe:

http://servidor[:porta]/[caminho-absoluto] [?parte-de-busca]

Os URLs descritos por esta sintaxe são conhecidos como *URLs absolutos* porque fornecem a descrição completa de como obter o recurso. A listagem a seguir descreve todas as partes da sintaxe:

- O componente do servidor especifica o endereço de rede do computador onde se encontra o recurso, que deverá ser um nome de domínio totalmente qualificado (por exemplo, **www.optimation.co.nz**) ou um endereço IP (por exemplo, 202.3.84.18).
- Porta é o numero da porta do servidor, responsável pelo atendimento ao pedido do recurso. Cada protocolo Internet recebe um número de porta exclusivo (uma *porta bem conhecida*) por onde o servidor, implementando o lado de servidos do protocolo, atende os pedidos que chegam. O componente de porta é opcional em URLs, e, quando omitido, a porta padrão para o método será utilizada (80 no caso HTTP). A Tabela 1.1 apresenta também os números de portas padrão para outros protocolos.
- O componente de caminho absoluto é opcional, e define o nome do caminho do recurso no servidor. Geralmente trata-se do nome do caminho de um arquivo HTML ou de um script CGI.
- O componente da parte de busca também é opcional, e permite a inclusão de uma string de consulta – mais comumente utilizado em pedidos GET gerador por formulários HTML (consulte a seção “Pedidos”, mais adiante neste capítulo).

Tabela 1.1: **métodos comuns de acesso ao URL.**

Método	**Descrição**	**Porta**
FTP	Protocolo de transferência de arquivos	21
File	Nomes de arquivos específicos do servidos	N/A
http	Protocolo de transferência de hipertexto	80
Gopher	O protocolo de Gopher	70
Irc	Internet Relay Chat	194
mailto	Endereço de correio eletrônico	N/a
News	Artigos da USENET	N/a
NNTP	Artigos da USENET com acesso NNTP	119
TELNET	Referência a sessões interativas	23
WAIS	Servidores de informações de área ampla	210

Fonte: Beveridge, Tony - 1998

Se ambos os componentes de caminho absoluto e string de consulta forem omitidos, a barra (/) anterior também poderá ser omitida.

2.4.2 URLs e Caminhos Absolutos

Um cliente pode enviar dois tipos de URLs, se o seu pedido estiver sendo feito por meio de um servidor proxy. Quando o destino é um servidor proxy, o cliente transmite um URL absoluto. Isso se faz necessário porque o servidor proxy precisa conhecer os nomes de todos os servidores para os quais está encaminhando pedidos, e de onde receberá as respostas. Se o servidor não for um proxy (o pedido é feito diretamente ao servidor de origem), o URL absoluto não será usado, e somente o componente de caminho absoluto do URL será transmitido.

Você poderá ver esse comportamento por si mesmo, analisando a conexão de rede entre seu navegador e seu servidor proxy interno (se tiver um) ao selecionar um URL que estiver relacionado com um recurso interno; por exemplo,/pessoal/home.htm. se você escolher a opção "**No proxies"** no seu navegador Web (Option Network Preferences: Proxies, no Netscape Navigator), o pedido gerado fará referência a um nome de caminho absoluto:

GET/pessoal/paulm/home.htm

Se a opção No proxies estiver desativada e um servidor proxy intermo chamado Morgan for especificado na porta 8080, com a opção Manual Proxy Configuration do Navigator, então o pedido gerado fará referência a um URL absoluto:
GET http://Morgan:8080/pessoal/paulm/home.htm

Durante todo este capítulo, usaremos o termo pedido de URL, significando tanto um URL absoluto como um nome de caminho absoluto, dependendo do fato de pedir ter sido feito a um servidor proxy ou ao servidor de origem.

2.4.3 URLs Relativos

Na criação de documento HTML, os URLs incorporados podem ser definidos de uma maneira absoluta ou relativa. Um URL relativo (como especificado no RFc 1808), às vezes chamado de URL parcial, assume o prefixo de servidor, porta, e o nome do caminho, do documento onde o URL aparece. Por exemplo:
<A HREF="resposta.html">Formulário de Resposta da SEREF</A>

Os URLs relativos são visto somente no contexto dos documentos HTML; não fazem parte do protocolo HTTP. O cliente HTTP construirá sempre o URL de pedido correto, colocando como prefixo do arquivo de recurso o método de acesso, o servidor, a porta e o caminho de diretório do documento, conforme necessário. Um navegdor Web (por meio de um proxy) acrescentaria ao anterior o seguinte:

http://www.optimation.co.nz/optimation/seref/resposta.html

Os URLs relativos são práticos na criações de grupos relacionados de documentos Web porque permitem a estruturação de uma hierarquia de documentos sem que seja preciso conhecer sua localização final. Dentro do URL relativo você poderá usa a anotação...para deslocar-se para cima e para baixo na árvore de documentos, da mesma maneira que faria dentro da hierarquia de diretórios do Unix. Por exemplo:
<HRE "F=../../home.html" >Volta à Home Page</A>

Os URLs relativos também funcionam corretamente quando o navegador Web acessa os arquivos diretamente, a partir do sistema de arquivos, sem conectar-se servidor algum.

2.4.4 Codificação de URL

Um URL de pedido é construído a partir de um subgrupo de caracteres, retirados do grupo US-ASCII. Esses caracteres consistem basicamente de letras, algarismos e um pequeno conjunto de *seguros* caracteres especiais. Um caractere é seguro quando não possui um significado especial para o servidor ou para o aplicativo que estiver processando o pedido. Cada caractere, por sua vez é basicamente uma entidade de 8 bits (um octeto). Todos os caracteres não-seguros devem ser codificados. A codificação consiste em substituir o caractere não-seguro por um sinal de porcentagem (%) seguido por dois dígitos hexadecimais que definam o valor do octeto.

Especificamente, as três situações em que um octeto/caractere de um URL deve ser codificado são:

- Se o octeto não estiver relacionado com um caractere US-ASCII que possa ser impresso; por exemplo, se o octeto tiver um valor igual a 00-1F e 7F (caracteres de controle) ou códigos 80-FF (não utilizados no US-ASCII).
- Se o caractere for não-seguro, tal como: SP,<,>,”,%,#.
- Se os caracteres forem reservados para a construção do próprio URL; por exemplo: ;,/,?,:,@,= e &.

Os demais caracteres considerados não-seguros, devido a possíveis interpretações especiais pelo servidor ou aplicativo que finalmente processar o pedido, são {,},|,\,^,~,[,] e ‘.

Os únicos caracteres que não precsiam ser codificados são alfanuméricos, os caracteres especiais $,+,!,*,’,(,) e aqueles especificamente reservados ao método do URL (no caso do HTTP esses caracteres são /,;,?).

2.4.5 Protocolo Básico

Basicamente, o HTTP implementa um paradigma de pedidos/respostas baseado em mensagens, que permite que programas-cliente, como por exemplo os navegadores Web, solicitem recursos do servidor, tais como documentos HTML. Geralmente esses recursos são documentos HTML.

No paradigma de pedido/resposta (ilustrado na figura 2.1), o cliente envia um único pedido ao servidor, e este retorna uma única resposta.

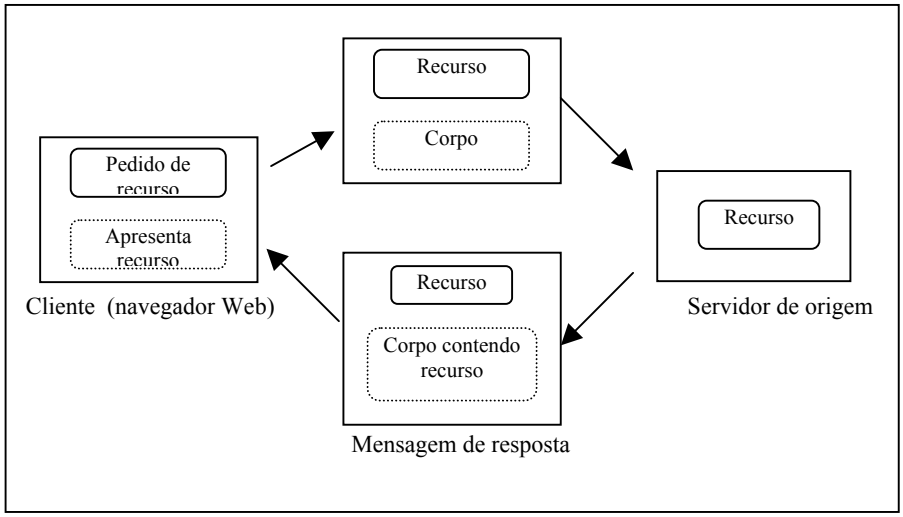

Figura 2.1: Paradigma de pedido/resposta. Fonte: Beveridge, Tony - 1998

O RFC 1945 introduz o termo *servidor de origem* para distinguir o servidor que finalmente atende o pedido de recurso. Entre o cliente e o servidor de origem, o pedido pode passar por uma corrente de entidades intermediárias, inclusive um ou mais proxies. Como mostrado na Figura 2.2, um servidor proxy encaminha um pedido integralmente para o servidor de origem e armazena em cache as respostas do servidor, de modo que os pedidos seguintes do cliente possam ser atendidos pelo proxy em seu lugar.

2.4.6 HTTP – Um Protocolo Sem Estado, em Nível de Aplicativo

O HTTP foi projetado inicialmente para ser leve, rápido e sem estado (embora as duas primeiras características tenham ficado comprometidas, até certo ponto,`a medida que o protocolo dói se tornando mais complexo). Com a expressão sem estado queremos dizer que cada pedido de HTTP é independente de qualquer pedido anterior, pois todas as informações necessárias para atender o pedido fazem parte da sua mensagem. Embora seja originalmente sem estado, há maneiras de se manter a informação de estado entre as transações, utilizando variáveis de ambiente, arquivos, campos HTML ocultos e cookies do HTTP.

O HTTP é parecido em estilo aos protocolos tradicionais de aplicativos internet, tais como SMTP e NNTP. No HTTP, tanto os clientes como os servidores trocam mensagens em ASCII. A conversação entre cliente e servidor se dá na forma de linhas de texto inteligíveis para humanos, terminadas por um retorno de carro (carriage return) e um quebra de linha (line feed) (CRLF). Isso torna o HTTP fácil de entender e depurar. Durante todo este

capítulo,utilizaremos esse recurso e apresentaremos exemplos do protocolo. Esses exemplos foram obtidos pela simulação de um cliente simples com Telnet, e à escuta oculta na conexão entre cliente e servidor, com base em utilitários de análise de rede, tais como o de domínio público tepdump ou o snoop da Solaris.

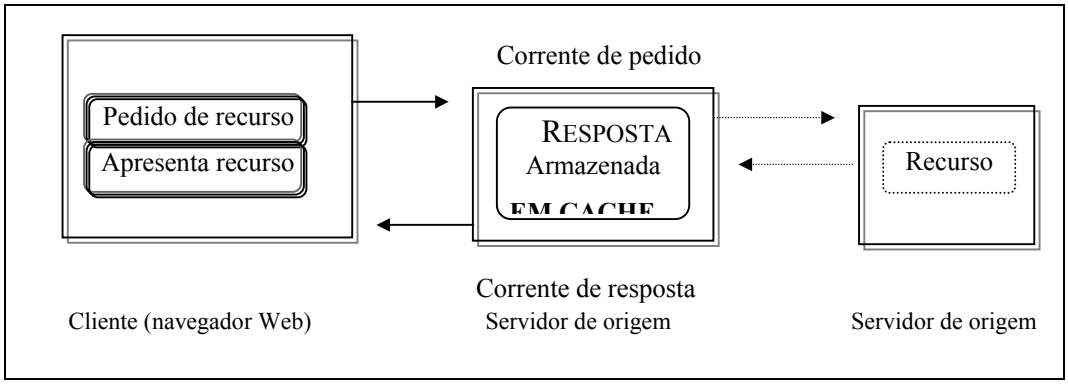

Figura 2.2: Intermediários entre cliente e servidor. Fonte: Beveridge, Tony – 1988.

2.4.6.1 O TCP Como Protocolo de Transporte

Embora o HTTP seja normalmente colocado uma camada acima do TCP/IP, não se trata de um pré-requisito, desde que o protocolo forneça um transporte confiável e livre de erros. Infelizmente, a disposição do HTTP pelo TCP pode prejudicar o desempenho do HTTP, devido à imperfeições do TCP.

Por exemplo, o handshake de três vias do TCP, no estabelecimento da conexão, exige que o cliente aguarde por uma confirmação, para que a informação possa ser enviada, resultando em um atraso no tempo de resposta.

Além disso, o desempenho inicial baixo do TCP significa que ele demora algum tempo até utilizar totalmente a banda disponível no canal. Essas ineficiências (exacerbadas pela conexão separada por transação do HTTP) são notadas particularmente em conexões discadas lentas, e começaram a ser resolvidas em implementações alternativas do TCP, como na Transaction TCP (T/TCP).

Observação: Para obter mais informações sobre TCP/IP, consulte os RFCs 1644 R 1323. Além disso, Richard Stevens mantém a home page T/TCP em www.noao.Edu/~rstevens/ttcp.html.

2.4.7 HTTP – Um Protocolo Independente de Mídia e Baseado em Objetos

O HTTP é capaz de trabalhar com vários tipos de informação, desde texto puro até objetos multimídia complexos. As mensagens HTTP especificam o tipo de dado do objeto transmitido, utilizando a noção de MIME de um tipo de mídia. O navegador Web utiliza o tipo de dado para chamar o visualizador mais apropriado apresentar o objeto.

O HTTP é conhecido informalmente como um protocolo baseado em objeto, pois fornece um mecanismo genérico para a chamada de uma ação (método), de um determinado objeto em um servidor, em um processo para a introdução de novos tipos e métodos tipo e métodos.

2.4.8 O Ciclo de Vida do Pedido

O HTTP é formado por fases distintas: conexão, pedido, resposta e fechamento da conexão, descritos na Tabela 1.2.

Podemos trabalhar facilmente em todas as fases do protocolo, se utilizarmos o Telnet como programa-cliente para estabelecer a conexão.

De acordo com o protocolo HTTP 0.9, a resposta ao pedido GET é o próprio documento HTML solicitado. Não há meta-informações adicionais enviadas na forma de códigos de status ou cabeçalhos.

2.4.9 Mensagens

O envio e o recebimento de mensagens são fundamentais para o HTTP. O HTTP define dois tipos distintos de mensagens – o pedido e a resposta. A regra de BNF a seguir mostra que, para cada tipo de mensagem, há formas simples e completas. A forma simples do HTTP 0.9 e a forma completa como definida no HTTP 1.0:

= Simples-Request
HTTP-message
| Simple-Response
| Full-Request
| Full-Response

Tabela 2.2: Elementos da sintaxe do protocolo.

Elemento	Significado
Regra1 | regre2	Indica duas regras altermativas.
(regra 1 regra2)	Indica um único elemento.
*regra	Indica repetição (por exemplo, 1* elemento significa pelo menos um elemento).
[regra]	Significa que a regra é opcional.
#regra	Indica uma lista de elementos separados por vírgulas (por exemplo, 1#elemento significa pelo menos um elemento na lista).
Token	Uma seqüência de uma ou mais caracteres, não incluindo caracteres de controle ou especiais.
Caracteres especiais	"(","")", "<", ">" ","@", ",", ".", "\", <">, "/" ,"[","]", "?", "=", "{", "}", SP,HT
CRLF	Um caractere de retorno de carro (CR), seguido por um caractere de quebra de linha (LF).
SP	Um caractere de espaço.
HT	Um caractere de tabulação horizontal.

Fonte: Beveridge, Tony – 1998

As mensagens inteiras são estruturadas da mesma maneira que as mensagens do correio da internet (conforme definido nos RFCs 822 e 1123) e são formadas por uma seqüência de campos de cabeçalho, seguidos por um corpo de mensagem. Os campos de cabeçalho apresentam uma descrição mais detalhada da mensagem e da entidade encapsulada pela mesma. O corpo da mensagem contém os próprios dados ou a própria entidade que estão sendo transmitidos, e consiste geralmente em um documento HTML (no caso de uma mensagem de resposta) ou no conteúdo de um formulário HTML (no caso de uma mensagem de pedido).

Tabela 2.3: Categorias de cabeçalho de mensagem.

Cabeçalho	Descrição
Genérico	**Os campos genéricos de cabeçalho se aplicam tanto a mensagens de pedido como de resposta. Descrevem aspectos da mensagem, e não do objeto nela contido.**
Pedido	**Os campos de cabeçalho de pedido fazem parte das mensagens de pedido, e fornecem informações sobre o pedido e o cliente que o envia.**
Resposta	**Os campos de cabeçalho de resposta são incluídos em mensagens de resposta, e oferecem uma maneira de o servidor enviar de volta informações adicionais.**
Entidade	**Os campos de cabeçalho de entidade permitem que as propriedades do próprio objeto sejam incluídas na mensagem.**

Fonte: Beveridge, Tony – 1998

2.4.9.1 Cabeçalhos de Mensagens

Há quatro tipos de campos de cabeçalho HTTP, apresentados na Tabela 2.3.

Embora a ordem em que os cabeçalhos aparecem não seja importante, geralmente os campos de cabeçalho Genéricos aparecem primeiro, seguidos pelos campos de Pedido, de Resposta e, em seguida, pelos campos de Entidade. Ao contrário dos nomes dos métodos, os nomes dos campos de cabeçalho não fazem distinção entre maiúsculas e minúsculas.

Os cabeçalhos das mensagens obedecem á seguintes sintaxe:

	= nome-campo “:” [valor-campos] CRLF
HTTP-header	
nome-campos	= token
valor-campo	= * (conteúdo-campos | espaço em branco)
conteúdo-campo	= <os bytes que compõem o campo-valor e que são formados por *TEXT ou combinação d tokens, caracteres especiais e strings entre aspas>
Espaço em branco	= [CRLF] | 1* (SP | HT)

2.4.9.1.1 Cabeçalhos Genéricos

Os cabeçalhos genéricos são aplicáveis tanto às mensagens de pedido como às de resposta. Eles contêm informações sobre a própria mensagem, e não sobre a entidade por ela encapsulada. Há atualmente somente dois cabeçalhos genéricos: Date e Pragma. Outros cabeçalhos genéricos poderão ser incluídos somente com a ampliação formal de protocolo.
Os cabeçalhos genéricos têm os seguinte formato:

General-Header = " Date" ":" HTTP-date
| " Pragma" " : " "no-cache" |
extension-pragma
extension-pragma = token ["=" word]

Observação: O HTTP 1.1 define cinco campos adicionais de cabeçalho genéricos – Cache-control, connection, Transfer-encoding, Upgrade e Via.

2.4.9.1.2 Cabeçalho Date

O cabeçalho Date define a data e a hora em que a mensagem foi criada no cliente ou no servidor. O servidor deve sempre incluir um campo de data nas respostas. O cliente deve incluir um campo de data somente se a mensagem de pedido contiver um corpo de entidade (por exemplo, o pedido POST).

Podem ser usados três formatos diferentes de data/hora dentro dos cabeçalhos HTTP, tais como Date (os valores de data/hora também são usados nos cabeçalhos de pedido If-Modified-Since, e nos cabeçalho de entidades Expires e Last-Modified). O primeiro e mais solicitado formato baseia-se no formato especificado nos RFCs 822 e 1123. O segundo formato usado tem sua origem no RFC 1036 da Usenet, e define um ano de somente dois dígitos, O terceiro formato usa asctime () do ANSI. Embora este formato deva ser aceito por clientes e servidores, não deve ser gerado por eles. Essas datas devem ser sempre representadas em GMT (Greenwich Mean Time ou Universal Time); se não houver um especificador GMT no formato asctime (), deverá estar implícito:

	RFC 822, atualizado pelo **RFC 1123**
Sun, 06 Nov 1994 08:49:37 GMT Sunday, 06-Nov-94 08:49:37 GMT Sun Nov 6 08:49:37 1994	RFC 1036 **Formato asctime ( )** C’ s ANSI

2.4.9.1.3 Cabeçalho Pragma

O cabeçalho Pragma é uma maneira de comunicar as diretivas de implementação, por meio da corrente de pedido, ao servidor de origem. A única diretiva definida formalmente é **no-cache**, que substitui as versões armazenadas em cachê do objeto, em qualquer versão intermediária, e faz com que o servidor de origem retorne uma resposta autorizada. Por exemplo:

Pragma : no-cache

2.4.10 Pedidos

Um pedido http pode ser um pedido simples (http 0.9) ou completo. O formato do pedido simples é um comando GET de apenas uma linha com um argumento URL. No pedido completo, cada mensagem de pedido é formada por uma linha de pedido, seguida por zero ou mais cabeçalhos de pedido, uma linha em branco para indicar o final da informação de cabeçalho e um corpo de entidade opcional (somente operações POST). A sintaxe de uma mensagem de pedido é definida da seguinte maneira:

Request	=	Simple-Request | Full-Request
Simple-Request	=	“GET” SP Request-URL CRLF
Full-Request	=	Request –Line ▪ (General-Header | Request—Header | Entity-Header) CRLF [Entity-Body]
Request-Line	=	Method SP Request-URL SP HTTP - Version CRLF

Method = “GET” | “HEAD” | “POST” | Extension-method

Extension-method = token

A linha de pedido define o método de pedido (como GET e POST), o URL do pedido e uma string definido a versão http que está sendo usada. Podem-ser introduzir facilmente novos métodos, sem a necessidade de alteração formais no protocolo. Se um servidor não oferecer suporte a um método, então o cliente será informado pelo código apropriado de status de resposta (501 Not Implemented)

Um campo http-Version é formado por um número da versão mais atual, seguido pelo numero da versão mais antiga, como mostra o formato:
http-Version = “http” “/” 1*DIGIT “.” 1*DIGIT

2.4.10.1 Cabeçalhos de Pedido

O cabeçalho de pedido permite que o cliente passe as informações sobre a mensagem ou sobre o próprio servidor. Cada cabeçalho modifica o significado do pedido de alguma maneira. O http 1.0 define cinco tipos de cabeçalhos de pedido:

Request-Header = “Authorization” “:” Credentials
| “From” “:” mailbox
| “If-Modified-Since” “:” http-date
| “referrer” “:” (absolute-URL | relative-URL)
| “User-Agent” “:” 1* (product | comment)

Observação: O http 1.1 combina 12 tipos adicionais: Accept, Accept-Charset, Accept-Accept-encoding, Accept-Language, Host, If-Match, Ifnono-Match, If-Range, If-Unmodified-Since, Max-Forwards, Proxt-Authorization, Range.

2.4.10.2 Cabeçalhos Authorization

O campo de cabeçalho Authorization é usado para autenticar o pedido do cliente em um servidor. O campo de cabeçalho especifica as credenciais de autorização necessárias para

obter o acesso ao reino da segurança de recurso. Os detalhes completos do mecanismo de autenticação http são dados na seção “Autenticação de acesso”.

2.4.10.3 Cabeçalho From

O cabeçalho From define o endereço eletrônico do usuário responsável pala geração da mensagem. Por exemplo:

From: paulm@optimation.co.nz

2.4.10.4 Cabeçalho If-Modified-Since

O cabeçalho If-Modified-Since é usado na construção de um pedido GET condicional, onde o objeto somente é transmitido se tiver sido modificado após a data especificada.

Por exemplo:

If-Modified-Since: Thursdey, 08-aug-96 04:01:54 GMT

2.4.10.5 Cabeçalho Referer

O cabeçalho Referer permite que o cliente especifique o endereço do objeto de onde o URL do pedido foi obtido. Isto geralmente é útil para fins de conexão, para detectar a fonte de vínculo antigos, ou para retornar a uma página anterior. Por exemplo, um script CGI pode inspecionar o campo Referer, e utiliza-lo como um mecanismo genérico para a geração de um vínculo que leve de volta a uma página de índice. Esse endereço pode ser um URL absoluto ou relativo. Os URLs relativos são interpretados como sendo relativos ao URL de pedido. Por exemplo:

Referer: http://www.optimation.co.nz/ optimaton/index/htm

2.4.10.6 Cabeçalho User-agent

O cabeçalho User-Agent contém informações especificas ao produto para o cliente que faz a requisição. De forma semelhante ao cabeçalho Server Response, esta informação assume o formato Produto/Versão. Por exemplo, o Netscape Navigator Gold versão 3.0 para Windows 95 emite:

User-Agent: Mozilla/3.0 Gold (Win95; I)

Os programas CGI que determinam dinamicamente o tipo e o conteúdo exibido pelo navegador aproveitam os recursos especiais do navegador que é usado freqüentemente neste campo.

Exemplo de sessão, mostrando um pedido GET gerado pelo Navigator.

GET http://ow:8080/ose/staff/web/paulm/paulm.htm HTTP/1.0
If-modified-Since:Thursday, 08-Aug-96 04:01:54 GMT; length = 7305
Proxy-connection: Keep-Alive
User-Agent: Mozilla/3. 0Gold (win95; I)
Host: ow : 8080
Accept : image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,*/*

A análise resultante mostra a inclusão de cabaeçalhos de pedido padrão (conforme definidos acima) e também de alguns que, formalmente, não fazem parte do HTTP 1.0 (a partir do http 1.1). Por exemplo, o campo de cabeçalho Accept define uma lista de tipos de mídia, que são respostas aceitáveis ao pedido. O asterisco é usado para agrupar tipo e subtipos de mídia dentro de faixas – assim,*/* significa todos os tipos e subtipos conhecidos, e image/* significa todos os subtipos do tipo imagem.

A partir do HTTP 1.1, o cabeçalho Host está também incluindo e, se aceito pelo servidor, seria capaz de determinar o servidor utilizado para atender o pedido, caso não houvesse um URL absoluto (contendo o endereço do servidor).

O Navigator informa que oferece suporte à forma Keep-Alive de conexão persistente, que costuma ser utilizada no HTTP 1.0, usando o cabeçalho Proxy-Connection. Embora o cabeçalho Proxy-Connection não esteja definido no HTTP 1.0 ou 1.1, presume-se que seja específico dos servidores Netscape.

2.4.11 Respostas

O servidor gera mensagem de resposta com resultado de mensagens de pedido enviadas por um mesmo cliente. As respostas simples são somente geradas em resposta aos pedidos HTTP 0.9 (ou se o cliente oferecer suporte apenas a HTTP 0.9). Os pedidos HTTP 0.9 podem ser identificados pela ausência do campo de versão do protocolo. Uma resposta

simples é apenas a própria entidade, geralmente codificada como HTML, e sem qualquer informação adicional sobre status ou cabeçalho.

Como não aparece mensagem de status alguma no HTTP 0.9, todos os erros do servidor devem ser retornados dentro do corpo da mensagem HTML, portanto o cliente deverá analisá-la para descobrir o erro.

A regra a seguir descreve a sintaxe de resposta completa. Uma resposta completa é parecida em sintaxe com a do pedido completo, e é formada por uma linha de status, seguida por zero ou mais linhas de cabeçalho. Uma linha em branco indica o final dos cabeçalhos, e um corpo de entidade opcional.

Response	=	simple-Response | Full-Response
Simple-Response	=	[Entity-Body]
Full-Response	=	Status-Life (General-Header | Response-Header | Entity-header)
Status-line	=	CRLF [Entity-Body] HTTP-Version SP Status-Code Sp Reason-Phrase CRLF

A linha de status é sempre a primeira linha de um cabeçalho de resposta, e contém a string da versão do protocolo, um código de status de três dígitos, e uma frase com o motivo, legível por humanos, explicando o código de status em linguagem clara (inglês) – todos separados por um espaço. O código de status informa ao cliente se o pedido foi ou não bem sucedido.

A string de versão e o código de status, juntos, geralmente são suficiente para diferenciar uma resposta simples HTTP 0.9 de uma resposta completa.

2.4.12 Códigos de Status

Embora os códigos de status sejam definidos formalmente, as frases de motivo correspondentes podem ser escritas para atender às condições locais. Não é necessário um aplicativo para entender todos os códigos de status, desde que se entenda, analisando o primeiro dígito do código, pelo menos as cinco classes de códigos apresentadas na Tabela 2.4.

Um aplicativo também pode, arbitrariamente, ampliar a faixa dos códigos de status, introduzindo novas classes ou ampliando a faixa das classes existentes. Os verdadeiros valores de código para cada categoria estão definidos na Tabela 2.5.

Tabela 2.4 : Categorias dos códigos de status de resposta.

Código	Tipo de código de status	Significado
1xx	Informational	Usado somente para fins experimentais
2xx	Success	**Indica que o pedido foi recebido, entendido e aceito pelo servidor.**
3xx	REDIRECTION	Indica que o cliente deve realizar mais algum trabalho para atender ao pedido. No caso de um pedido GET, isto geralmente significa uma repetição do pedido, contra um dos URLs opcionais definidos na resposta.
4xx	Client Error	INDICA QUE O CLIENTE GEROU UM ERRO NO SERVIDO. O CLIENTE DEVE PARAR DE ENVIAR DADOS, SE RECEBER ESSA CLASSE DE ERRO.
5xx	Server Error	Indica que o próprio servidor gerou um erro ou não pôde satisfazer um pedido. O cliente deve para de enviar dados, se receber essa classe de erro.

Fonte: Beveridge, Tony – 1998

Tabela 2.5: Valores dos códigos de status de resposta .

Código	Significado
200	O pedido foi bem-sucedido e foi enviada a resposta correta.
201	Um novo recurso foi criado pelo servidor (POST).
202	O pedido foi aceito, mas o processamento não está completo – por exemplo, uma operação longa em banco de dados.
203	O pedido foi bem-sucedido, mas não há conteúdo para enviar.
301	O recurso solicitado foi movido para um novo URL permanente.
302	O recurso solicitado foi movido para um novo URL temporário.
304	O recurso solicitado não foi modificado (GET condicional).
401	Um pedido não autorizado – necessita de autenticação.
403	O pedido foi entendido, mas o servidor recusou o acesso ao recurso – geralmente o servidor ou o cliente não têm permissão de acesso ao arquivo.
404	O recurso requisitado não pôde ser encontrado.
500	Ocorreu um erro interno no servidor.
501	O servidor não implementou o método pedido.
502	Gateway inválido (resposta inválida do gateway ou do servidor superior).

503	O servidor está ocupado demais par atender ao pedido.

Fonte: Beveridge, Tony – 1988

2.5 Métodos de Pedido

2.5.1 GET

O método GET transmite o objeto definido pelo URL de pedido. Quando o URL define um recurso estático (geralmente um documento HTML), o conteúdo deste recurso é desenvolvido como o corpo da entidade. No caso do recurso dinâmico, como um script CGI, a saída gerada pelo recurso é retornada.

O método GET também pode ser usado para enviar dados, digitados em um formulário HTML, a um servidor. Nesse caso, os dados são anexados ao URL de pedido como uma seqüência de pares nomes/ valor (consulte o exemplo na Sessão "POST", mais adiante nesta seção). Esta técnica deve ser usada somente com pequenas quantidades de dados.

Quando o pedido GET contém um campo de um cabeçalho If-Modified-Since, o GET passa a ser condicional e o objeto somente será retomado se tiver sido modificado após a data e a hora deferidas no cabeçalho.

2.5.2 HEAD

HEAD é idêntico ao método GET, com a diferença de que não retorna um corpo de entidade na resposta. Todas as outras informações, inclusive os campos de cabeçalho da entidade, são retornadas. Esse método é usado com freqüência para confirmar a existência de um recurso sem precisar transferi-lo. Não existe tal coisa como um HEAD condicional. Se houver um campo If-Modified-Since no pedido, este será ignorado.

2.5.3 POST

O método POST permite o encapsulamento de uma pedido e sua transmissão para o servidor para ações posteriores. Geralmente, os dados transmitidos são o conteúdos são o conteúdo de um formulário digitado pelo usuário. A ação realizada pelo servidor é definida pelo URL do pedido, que no caso de um formulário trata-se normalmente de um script CGT que aceita os campos de formulário como entrada, e gera um documento HTML como saída.

O método POST deverá ser usado quando houver uma grande quantidade de dados, que podem ter sido fornecidos por um formulário extenso, a serem enviados para o servidor. Talvez não seja possível usar um pedido GET para transferir uma grande quantidade de dados, pois a maioria dos navegadores e servidores restringem o comprimento de um URL a 1024 bytes.

2.5.4 PUT

O método PUT é bastante parecido com o POST. A diferença entre eles é a maneira como o URL de pedido é tratado. O URL de um pedido POST define um recurso que consome a entidade incorporada, que pode ser um script CGI que aceita o corpo da entidade como entrada, e o processa. Por outro lado, o URL especificado em um pedido PUT identifica um recurso que deverá ser criado (se o recurso ainda não existir) ou atualizado (se já existir). A entidade incorporada é composta pelos dados a serem dados a serem colocados sob o recurso.

2.6 A Relação Entre HTTP e MIME

O HTTP baseia-se fortemente no MIME para estruturar os seus cabeçalhos e codificar o conteúdo das mensagens. Contudo, como MIME está voltado especificamente para as necessidades do correio Internet, o HTTP introduziu algumas diferenças, relacionadas a seguir:

- O MIME exige que todas as entidades sejam convertidas para um formato padrão (formato canônico), específico do tipo de mídia da entidade. O HTTP segue essa exigência, mas regras diferentes para o tratamento do tipo de mídia de texto, particularmente para a utilização de CR e LF. No HTTP, CRLF, CR ou LF podem ser usados como caracteres de quebra de linha dentro do texto, enquanto no MIME somente o CRLF pode ser usado.
- No MIME, o cabeçalho Content-Length aplica-se somente ao tipo de mídia message/external-body e é opcional; no http, ele deve ser apresentado sempre que possível, para calcular o tamanho do corpo da entidade.
- O HTTP restringe a ampla variedade de formatos de datas aceitas no MIME a apenas três formatos, para simplificar o tratamento delas.

- O MIME não possui equivalente ao campo de cabeçalho Content-Enconding utilizado em HTTP.
- O HTTP não usa o cabeçalho Content-Transfer-Encoding do MIME. Em vez disso, ele utiliza o cabeçalho Transfer-Encoding para introduzir uma transformação em uma entidade. Esse cabeçalho é uma propriedade da mensagem, e não da entidade que está dentro da mesma. Enquanto o MIME aplica Content-Transfer-Encoding a uma entidade, para garantir que poderá ser passado com segurança pelo ambientes de 7 bits, o HTTP assume um ambiente limpo de 8 bitse, em vez disso, define um mecanismo para o envio confiável de grandes quantidades de dados, dividindo a mensagem em trechos bem definidos.
- O HTTP permite que as partes do corpo multiparte contenham as informações de cabeçalho relevantes para a parte, enquanto o MIME restringe esse cabeçalho aos pré-fixados com Cotent-.

As mensagens do HTTP podem ainda usar o cabeçalho MIME-Version, para indicar que a mensagem é compatível com MIME:

MIME-Version: 1.0

Atualmete a utilização desse cabeçalho é opcional no HTTP, por não haver critérios que restrinjam sua atualização. Contudo, de acordo com o RFC 2049, que define o critério de compatibilidade do protocolo MIME, a inclusão do cabeçalho MIME-Version é, na verdade, obrigatória.

2.6.1 Autenticação de Acesso

O HTTP oferece suporte a um mecanismo simples de identificação de resposta, para que o servidor autentique os pedidos dos clientes. O diálogo entre o cliente e o servidor está descrito na figura 2.3.

Quando um pedido é feito inicialmente para um recurso protegido, o servidor responde com uma mensagem de resposta 401 (Unauthrized Access – acesso não – autoriza), que contém um cabeçalho de resposta www-authenticate. O cabeçalho de resposta contém um pedido de identificação que especifica o método a ser usado para identificar o pedido, a área de proteção à qual pertence o recurso solicitado e uma lista opcional de pares nome=valor. A

área é identificada completamente pela concatenação do valor da área definido na resposta com o URL-raiz do servidor. Cada área mantida pelo servidor pode ter seu próprio método de autenticação e seu próprio banco de dados de acesso associado.

www-authernticate	=	"www-authenticate" ":" challenge
Challenge	=	auth-scheme 1*SP realm * ("," auth-param)
auth-scheme	=	Token
auth-param	=	token "=" quoted-string
Realm	=	"realm" "=" realm-value
realm-value	=	quoted-string

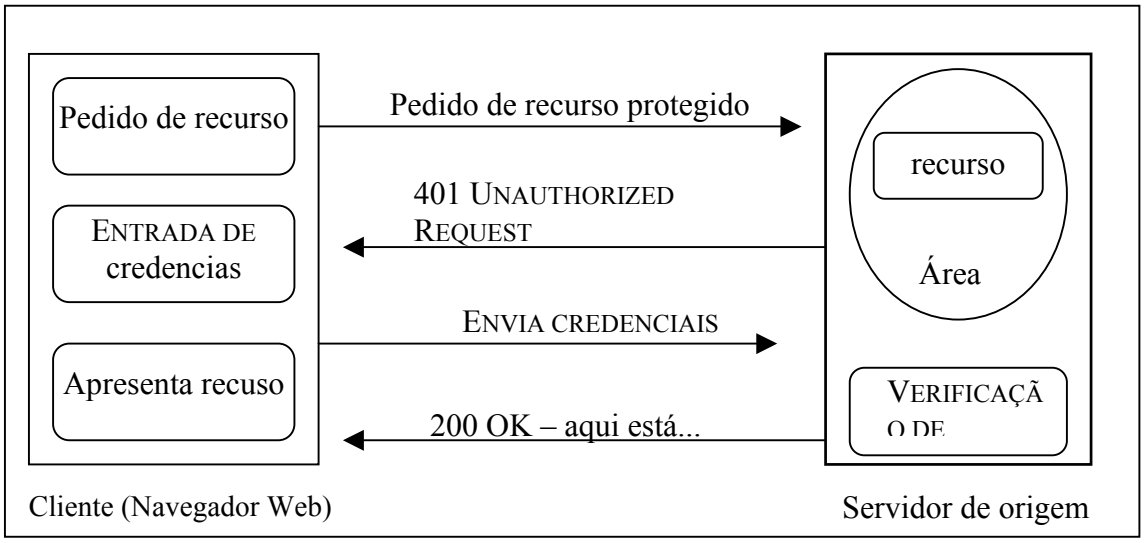

Figura 2.3: Diálogo de identificação de resposta. Fonte: Beveridge, Tony – 1998

Depois que o cliente tiver pedido a identificação para a resposta 401 do servidor, o cliente poderá responder enviando um novo pedido pelo recurso que contenha Authorization para definição das credenciais necessárias à autenticação do pedido para a área que contém o recurso.

Authorization	=	"Authorization" ":"credentials
Credentials	=	Basic-credentials
	|	(auth-scheme #auth-param)

Uma vez autenticadas, as mesmas credenciais podem ser usadas para os novos pedidos. O tempo de vida dessas credenciais depende do método. O servidor poderá negar um pedido de autorização com uma resposta 403 (Forbiddem – proibido).

Além do mecanismo simples de resposta de identificação de pedido do HTTP, outros mecanismos de autenticação e segurança também podem ser realizados. Entre eles estão

criptografia e camada de transporte, utilizando SSL (Secure Sockets Layer – Camada de Soquetes de Segurança) ou métodos baseados em mensagens, como S-HTTP [Rescola, Schiffman].

2.6.2 Método Básico de Autenticação

O HTTP define somente um método básico de autenticação. Outros métodos podem ser introduzidos, bastando para tal definir o nome do esquema e os parâmetros opcionais da mensagem 401 de pedido de identificação. Qualquer método novo, claro, teria de ser entendido por ambos os clientes e servidores que estivessem participando da sua utilização. O método básico exige que as credenciais de clientes de cada área esteja na forma de um nome de usuário e uma senha. A sintaxe das credencias é:

Credenciais-básicas	= “Basic” SP cookie-básico
cookie-básico	= <base64 [5] codificação de userid-senha, exceto os não limitados a 76 caracteres/linha>
Userid-senha	= [token] “:” *TEXT

No exemplo a seguir, vemos um pedido de cliente com uma solicitação de recurso protegido, o paulm.html, e a mensagem resultante com pedido de identificação 401, emitida pelo servidor. O pedido de autenticação contém o cabeçalho WWW-Authenticate:
www-authenticate: basic realm=“Páginas da Equipe da Optimation”
Isso indica que o URL pedido foi autenticado pelo método básico e protegido pela área chamada Páginas da Equipe da Optimation. Em resposta a esse pedido de autenticação, o navegador Web pede ao usuário um nome e uma senha para a área definida. Neste exemplo foram fornecidos o nome do administrador e a senha let me in (deixe-me entrar).

2.6.3 Codificação Base64

O BASE64 (conforme descrito no RFC 2045) é uma maneira simples para converter uma seqüência arbitrária de octetos (entidades de 8 bits) em uma string equivalente de caracteres de 6 bits. Esse tipo de codificação é usado com freqüência para a codificação MIME de anexos de correio eletrônico, de modo que os anexos binários possam ser transmitidos sem que seus conteúdos sejam interpretados erroneamente pelos sistemas que tratam a correspondência.

O HTTP utiliza a codificação BASE64 para permitir a codificação segura dos nomes de usuários e senhas sujeitos a terem caracteres que não possam ser impressos. Essa codificação também funciona como um mecanismo de criptografia extremamente fraco.

2.6.4 Digest Authentication

O HTTP 1.1 define o método Digest Authentication (RFC 2069) que substitui as limitações do método Basic do HTTP 1.0, garantindo que somente uma forma criptografada da senha será transmitida. Nesse método, tanto cliente como servidor já devem ter entrado em acordo quanto a nomes de usuário e senhas, utilizando o mecanismo apropriado.

Assim como o método Basic, o método Digest baseia-se no paradigma pedido de identificação/resposta. Quando o cliente tenta acessar um recurso protegido, o servidor responde com um pedido de identificação que contém uma string criptografada (um valor nonce), construído pelo servidor. A string faz parte da informação que apresenta uma identificação exclusiva do cliente, e que pode conter o endereço IP do cliente e um selo de tempo, que determina o fim da validade. O servidor pode recalcular o valor de tempo se o cliente tentar fazer novamente um pedido para verificar se sua origem é a correta. O valor de tempo é incompreensível para o cliente. Em outras palavras, o cliente não o interpreta, mas somente o envia de volta como parte da informação de autorização.

3. A TECNOLOGIA DOS SERVIDORES WEB

A sofisticação da tecnologia dos servidores Web cresceu rapidamente para oferecer suporte a uma gama muito mais ampla de aplicativos, do que apenas a transferência de documentos simples. Agora as arquiteturas de aplicativos multinível baseadas na web estão substituindo o paradigma tradicional cliente/servidor, que começou a crescer no final dos anos 80 e início dos 90.

3.1 Arquitetura Típica de Servidores Web

A arquitetura típica de servidores Web oferece suporte ao paradigma de pedido/resposta, que é formado pelas fases a seguir:

- Aceita mensagens de pedido HTTP de um programa-cliente (navegador Web).
- Localizar o recurso definido pelo URL no cabeçalho do pedido da mensagem.
- Acessa o recurso utilizando o método de pedido especificado.
- Prepara uma mensagem de resposta HTTP, contendo informações de status e o próprio recurso (o corpo da mensagem).
- Envio da resposta.
- Encerramento da conexão com o cliente.

Geralmente, o recurso pedido é um arquivo HTML, uma imagem GIF ou algum objeto multimídia (áudio, vídeo e coisas do gênero). O recurso também poderá ser um objeto de produção de dados, como um programa de gateway que utiliza dados da mensagem de pedido para solicitar mais informações de um banco de dados (ou alguma outra fonte) antes de retornar como corpo de uma mensagem de resposta.

Os programas de gateway (que servem como ligação) são geralmente implementados como scripts Perl ou programas C, e se comunicam pelo servidor utilizado a CGI. Quando um servidor recebe um pedido em CGI, ele chama um programa para gerar a mensagem de

resposta. Embora esse método seja simples e fácil de entender, torna-se ineficiente quando o servidor precisa responder a um grande número de pedidos simultâneos.

3.2 Operação de Servidor

A Figura 3.1 mostra que o funcionamento de um servidor Web pode ser dividido em nove etapas distintas. Primeiro, o cliente estabelece uma conexão como servidor e envia uma mensagem de pedido HTTP. Quando o pedido é recebido, o servidor interpreta todos os cabeçalhos na mensagem dele. Com recursos protegidos, o servidor autenticará o cliente com um método de autorização (geralmente o método básico de HTTP) e usará as informações de cabeçalho para ter acesso ao recurso (se necessário, convertendo primeiramente o URL, utilizando as regras de conversão do servidor).

O método definido no cabeçalho de pedido da mensagem define como o recurso será acessado. Os métodos pelo HTTP 1.0 comumente utilizados são: GET, HEAD e POST. O URL de pedido informa o servidor sobre a localização de recursos. E poderá também conter pares nome=valor codificados, se o método GET tiver sido usado para enviar o conteúdo de um formulário HTML ao servidor. Opcionalmente, se for um pedido POST, os pares nome=valor de formulário serão então enviados no corpo de mensagem de pedido.

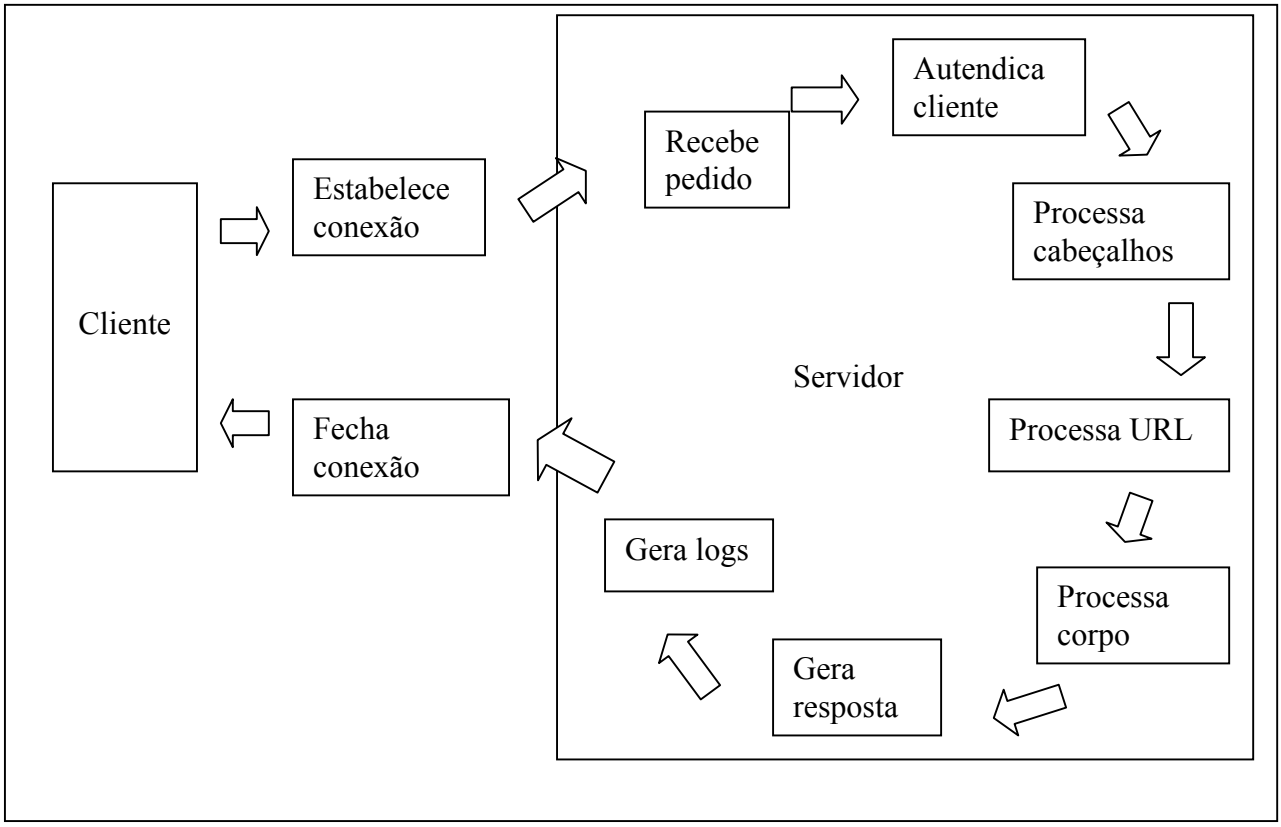

Figura 3.1: O ciclo de vida do pedido HTTP. Fonte: Beveridge, Tony – 1998

O servidor então identificará o tipo de recurso, analisando a forma do URL de pedido. No caso de recursos e baseados em arquivos, isso significa analisar a extensão do nome do arquivo e relaciona-la com algum tipo de mídia utilizada pelo servidor para preencher o cabeçalho de responsta Content-Type. Esse é um dos cabeçalhos mais importantes, pois define como o programa-cliente interpretará a mensagem. Para um navegador, ele indicará qual visualizador ao aplicativo auxiliar deverá ser chamado.

No caso de URLs que fazem referência a um programa CGI (seja pela extensão de arquivo. CGI ou pela referencia ao subdiretório cgi-bin do URL), o servidor chamara o programa CGI. Nesse caso, esse programa é o responsável pela geração de mensagem de resposta.

O servidor ou o programa CGI prepara então uma mensagem de resposta contendo:

- Uma linha de status indicando o sucesso ou outro estado do pedido. Por exemplo, uma resposta bem-sucedida começaria com a linha de status parecida com: 'HTTP/1.0 200 OK'.
- Os cabeçalhos genéricos que descrevem as propriedades da mensagem. Por exemplo, um Date.
- Os cabeçalhos de resposta contendo informações sobre a resposta. Por exemplo, um servidor ou cabeçalho www-Authenticate.
- Cabeçalhos de entidade descrevendo as propriedades de recurso. Um cabeçalho Content-Type é um exemplo.

Finalmente , o próprio recurso (no caso de recursos estáticos), ou saída do programa CGI, será transmitido como o corpo da mensagem, antes que a conexão do cliente seja fechada.

Veremos posteriormente neste capítulo como é possível alterar o comportamento de um servidor em vários pontos de ciclo de vida, ignorando ou ampliando o comportamento padrão do servidor.

3.3 Common Gateway Interface (CGI)

A CGI é a maneira tradicional usada para ampliar a funcionalidade de um servidor. Nesse método, o servidor chama um programa de gateway (geralmente um script em Perl ou um programa em C) e passa a ele os dados de entrada da mensagem de pedido.

O servidor faz isso criando um processo CGI (geralmente fazendo uma copia de si mesmo) e canais de entrada e saída entre ele próprio e o processo CGI (figura 2.4). As informações de estado sobre o servidor, o cliente e o pedido são herdadas pelo processo CGI, a partir de um grupo de variáveis de ambiente definidas pelo servidor da CGI.

Uma vez estabelecido, o processo CGI executa o programa CGI definido no URL de pedido. O programa aceita entradas de servidor por meio de um grupo de variáveis de ambiente, ou pela entrada padrão, analisa e processa os dados de entrada, e gera uma mensagem de resposta em sua saída padrão. Essa saída deve ser interpretada pelo servidor antes de ser enviada ao cliente.

Mais detalhes sobre a especificação CGI 1.1, no site do HTTPd Development Team do NCSA (national Center for Supercomputing Applications – Centro Nacional para Aplicativos de Supercomputação), em hoohoo.ncsa.uiuc.Edu/cgi/.

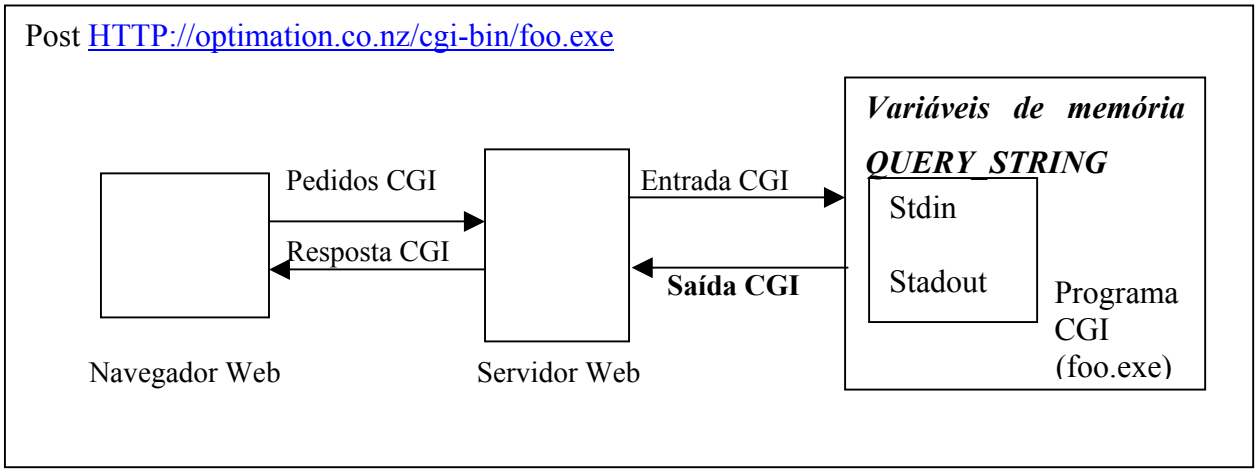

Figura 3.2: Servidor Web e porcesso CGI. Fonte Beveridge, Tony – 1998

3.3.1 Variáveis de ambiente da CGI

O processo CGI herda um conjunto de variáveis de ambiente do servidor. Essas variáveis de ambiente descrevem o estado do servidor, do cliente, do pedido e, possivelmente, dos dados de entrada. Veja a lista completa na tabela 3.1.

As variáveis de ambiente são agrupadas em cinco categorias:

- As variáveis de informação geral permitem que um programa CGI determine como entrar com dados a partir do servidor. SERVER_PROTOCOL contém a versão do protocolo HTTP implementada pelo servidor, sendo assim uma variável de informações gerais.
- As variáveis de informação de entrada contêm os dados de entrada fornecidos pelo servidor. CONTENT-TYPE contem o tipo de mídia, e com base nela define-se como os dados de entrada serão codificados, sendo assim uma variável de entrada de informações.
- As variáveis de informações do cliente descrevem o programa-cliente 9navegador Web). REMOTE_HOST contém o nome de host (um nome de domínio totalmente qualificado) do computador-cliente, sendo assim variável de informação do cliente.
- As variáveis de informações do servidor descrevem o servidor SERVER_PORT contém o numero de porta usada pelo servidor para receber os pedidos de conexão, sendo assim uma variável de informações do servidor.
- As variáveis de informação de HTTP descrevem as propriedades adicionais do cliente. Essas propriedades são obtidas a partir dos cabeçalhos das mensagens de pedido e das informações armazenadas pelo próprio servidor. HTTP_USER_AGENT contem o nome do prouto e a versão do programa cliente, e HTTP_ACCPT a lista de tipos de mídia (tipos MINE) aceitos pelo programa-cliente, sendo assim variáveis de informações HTTP.

Tabela 3.1 Variáveis de ambiente da CGI.

Variável de ambiente	Descrição
AUTH_TYPE	Se a autenticação estiver sendo usada, esta variável será definida com o método usado geralmente o básico.
CONTENT_LENGTH	O tamanho dos dados de entrada, quando enviados em um pedido POST. Essa variável não deve ser utilizada em pedidos GET ou quando não há dados de entrada.
CONTENT_TYPE	Um tipo de mídia que define a maneira como os dados foram codificados. Em formulários, esse tipo é definido como application/www-form-urlencoded.
DOCUMENT_ROOT	O topo da árvore de documentos do servidor.
GATEWAY_INTERFACE	A versão do protocolo CGI aceita pelo servidor. Definida sempre com CGI/1.1
PATH_INFO	Caminho após o nome do programa CGI no URL de pedido.
PATH_TRANSLATED	Caminho em PATH_INFO convertido para um caminho físico no servidor.

QUERY_STRING	A entrada, quando enviada em um pedido GET.
REQUEST_METHOD	GET ou POST, dependendo do método usado para passar dos dados ao programa CGI.
REMOTE_ADDR	Endereço IP do computador-cliente.
REMOTE_GROUP	Grupo ao qual pertence o usuário, obtido durante o processo de autenticação.
REMOTE_HOST	Nome de host do computador-cliente
REMOTE_IDENT	Nome do usuário do computador-cliente (o computador-cliente deverá estar executado um agente compatível com o RFC 931, como por exemplo, identd)
REMOTE_USER	O nome do usuário, obtido durante a autenticação com o método básico.
REQUEST_METHOD	GET, HEAD, POST ou um método de extensão definido pelo cliente.
SCRIPT_NAME	Um URL que identifica o scrip CGI a ser chamado.
SERVER_ADMIN	Endereço de correio eletrônico do administrador do servidor.
SERVER_NAME	Nome do host do computador que esta executando o servidor.
SERVER_PORT	Porta TCP onde o pedido foi recebido - normalmente 80.
SERVER_PROTOCOL	Versão do protocolo HTTP que esta sendo usado; por exemplo, HTTP/1.0 ou HTTP/1.1
SERVER_SOFTWARE	Nome e versão do servidor.
Variáveis de cabeçalho	Um subgrupo das variáveis de cabeçalho que podem ser definidas. São preparadas diretamente a partir dos cabeçalhos fornecidos na mensagem de pedido do cliente.
Variável de Cabeçalho	Descrição
http_ACCEPT	Lista dos tipos de mídia que podem ser tratados pelo srvidor.
HTTP_ACCEPT_LANGUAGE	Lista de linguagens aceitas pelo cliente.
http_COOKIE	Lista de cookie de HTTP armazenados pelo cliente.
HTTP_USER_AGENT	Informação de produto e versão sobre o programa-cliente.
http_REFERER	URL do documento que fez referencia ao URL atual.

Fonte: Beveridge, Tony – 1998

3.3.2 URLs de CGI

Um servidor é capaz de distinguir entre um recurso estático e um programa CGI baseado em dois métodos. O primeiro baseia-se na análise de extensão do arquivo do recurso solicitado. Veja a seguir um Url de um script CGI:

HTTP://www.optimation.co.nz/optimation/feedback.cgi

Este método basei-se em cgi registrado no servidor como um tipo de mídia e permite que o programa CGI sejam gravados juntos com outros arquivos na árvore de documentos do servidor.

O registro dos diretórios dedicados aos programas CGI no servidor consiste em um método mais comum. Qualquer URL que fizer referência a um arquivo localizado em um desses diretórios será tratado como uma chamada a um programa CGI. Os programas CGI armazenados dessa maneira são muito mais fáceis de proteger e manter. Esses diretórios são freqüentemente identificados côo dgi-bin, como no exemplo a seguir:

HTTP://www.optimation.co.nz/optimation/cgi-bin/feeblack.bat

Um URL que fizer referencia a um programa CGI será interpretado de maneira diferente de um URL que fizer referência a um recurso estático, como um arquivo HTML. Um URL CGI divide-se em três partes:

[Caminho virtual] [Informação adicional de caminho] ? [string de consulta]

O URL a seguir foi gerado por um formulário que usa um método GET para enviar seus dados para o programa CGI feedback.bat:

HTTP://www.optimation.co.nz/optimation/cgi-bin/feedback.bat/
feedback?MessageType=Praise&
Comments=Great+framework%0D%oAWhat+comes+next%3F&
Username=Paul+McGlasban&companyName=Optimation+NZ+Ltd&

UserEmail=paulm@optimation.co.nz&userPhone=&userFAX=&
SubmitButton=submit+Feedback

O caminho virtual é o prefixo do URL, incluindo o nome do programa CGI, como no exemplo a seguir:

HTTP://www.optimation.co.nz/optimation/cgi-bin/feedback.bat

As informações adicionais de URL sobre o caminho estão entre o caminho virtual e o ponto de interrogação (?). Por exemplo:/feedback.

A variável do ambiente, PATH_INFO, que contém informações adicionais sobre o caminho, fornece CGI. Normalmente, é usada em conjunto com o recurso de conversão de caminho virtual para físico do servidor, que relaciona a informação adicional do caminho com um nome de caminho físico, em relação à raiz da árvore de documentação do servidor.

Neste exemplo, a raiz da árvore de documento é /web/optimation; portanto, /feedback é convertido automaticamente para /webs/optimation/feedback e passado para o programa CGI pela variável de ambiente PATH_TRANSLATED.

O componente de string de consulta pode ser fornecido de três maneiras:

- Explicitamente, com uma âncora HTML.
- O resultado dos dados digitados pelo usuário em uma consulta ISINDEX.
- A saída de um formulário HTML (este é o método mais comum).

3.3.3 Dados de Entrada na CGI

Normalmente, os dados de entrada de um programa CGI vêm do conteúdo dos formulários HTML. Esses dados são formatados como uma string de pares nome=valor, onde *nome* é o nome de um campo do formulário e o *valor* é o valor do campo, digitado pelo usuário. Cada par é separado por & e codificado, de modo que todos os espaços sejam substituídos pelo sinal de soma (+), e outros caracteres não–seguros são substituídos pelo anotação %## (onde ## são dois dígitos hexadecimais, que representam o valor ASCII do caractere substituído; a interrogação (?) seria convertida para %3F).

Os programas CGI recebem os dados de entrada do formulário de acordo com o método de pedido usado. Se o método de pedido GET for usado, o servidor copiara os pares

nome=valor codificados, que serão anexados pelo cliente ao URL do pedido, para a variável do ambiente QUERY_STRING. O programa analisará então a variável QUERY_STRING como conteúdo de entrada. Uma interrogação (?) separa os dados do formulário do URL de pedido. Contudo, se o método PUSH for usado, a string transmitida como corpo de mensagem de pedido será colocada nos dados de entrada padrão do programa CGI, e a variável de ambiente CONTENT_LENGTH será definida como o numero de bytes do corpo da mensagem.

Depois que o programa CGI tiver analisado o fluxo de entrada (ele deverá decodificar a string de consulta codificada), ele executará sua função; talvez unindo-se a uma fonte remota de informações ou a um banco de dados, para executar uma consulta antes de gerar a resposta apropriada.

Além disso, também é possível passar os dados de entrada para o programa CGI por meio da linha de comando. Isso ocorre quando o tag ISINDEXTHML é encontrada pelo navegador-cliente (observe que a utilização de ISINDEX foi substituída pelos formulários HTML).

Os navegadores Web interpretam o tag ISINDEX apresentado um campo de busca, que solicita ao usuário, que digite um seqüência de palavras-chave separadas por espaços. A string de consulta digitada pelo usuário é codificada, e todos os espaços são substituídos pelo sinal de soma(+). O cliente anexa então uma interrogação (?) e a string codificada ao URL definida no tag ISINDEX.

Quando um servidor encontra esse tipo de pedido, além da string de consulta que está sendo copiada para a variável de ambiente QUERY_STRING, todas as palavras-chaves são extraídas da string de consulta pelo servidor e associadas a um argumento correspondente de linha de comando de programa CGI. O servidor distingue entre esse tipo de pedido e aquele gerado por um formulário HTML, procurando pelo sinal de igualdade (=) na string de consulta codificada.

3.3.4 Dados de Saída da CGI

O programa CGI é responsável pela formação de todos os dados de resultado, em uma mensagem de resposta HTTP, como todos os cabeçalhos e o corpo de entidade que os dados contêm.

Os cabeçalhos mais importantes são a linha de status e o cabeçalho. Content-Type. Se tudo tiver corrido bem, a linha de status deverá indicar uma operação bem-sucedida, por meio de um código de status 200 OK, seguido pelo cabeçalho Content-Type HTTP (definido o tipo de mídia dos dados).

Outros cabeçalhos que também podem estar presentes:

- Location – Uma nova localização para o recurso a ser retornado.
- Content-=Length – O número de bytes dos dados a serem enviados na saída.
- Expires – A data após a qual os dados não estarão mais disponíveis.
- Content-Enconding – O método de decodificação que o cliente deverá usar antes de prosseguir com a apresentação ou o processamento dos dados.

Uma vez gerados os cabeçalhos, o programa CGI gera uma linha em branco, que indica o final dos cabeçalhos, antes de enviar os dados. O programa CGI deve garantir que os dados sejam formatados corretamente para o tipo de mídia definido no cabeçalho Content-Type, em qualquer codificação posterior aplicada de acordo com a especificação passada pelo cabeçalho Content-Encoding.

Quando um programa CGI envia dados de volta ao servidor, este interpreta o fluxo de saída e realiza operações adicionais, se necessário. Por exemplo, se o programa CGI não enviar um cabeçalho de estado, o servidor criará um: 200OK por padrão. O servidor também interpretará cabeçalhos como Location, a partir do qual localizará um nome de caminho definido no cabeçalho Location e, de uma maneira transparente para o cliente, era gerar uma resposta novamente, contendo o novo recurso com todos os cabeçalhos apropriados.

Se um URL absoluto for definido no lugar de um nome de caminho em um cabeçalho Location, o servidor irá gerar uma resposta com um estado 302 (Temporarily Moved – Movido temporariamente), indicando que o recurso solicitado foi movido para outro local. Nesse caso, o cliente será responsável pela navegação até o novo URL.

Por razões de eficiência, o servidor armazena em buffer todos os dados de saída do programa CGI. Desta forma, os dados são enviados de volta ao cliente como blocos de dados, e não como um fluxo simples de bytes individuais.

Há situações em que o programa CGI pode querer ignorar completamente o servidor, para que possa enviar os dados diretamente ao cliente; aplicativos push do lado do cliente que não utilizam buffers funcionam dessa maneira. Isso é possível pela utilização de recursos de cabeçalhos não-compartilhados do servidor. Quando este recurso é utilizado,a saída padrão do

cliente é enviada diretamente a uma cópia conexão de soquete de volta ao cliente. Neste caso, o programa CGI deve formatar corretamente toda a mensagem de resposta, inclusive todos os cabeçalhos. Os programas CGI que usam essa interface tem um prefixo rph= em seus nomes.

3.3.5 CGI V*ersus* Extensões In-process

A CGI é o método mais utilizado para a ampliação de servidores Web. Isso se deve principalmente às suas vantagens inerentes, como descritas a seguir:

- Simplicidade – O mecanismo da chamada de um programa é simples de entender e de implementar. O mecanismo de chama da é análogo à chamada remota de procedimento, com parâmetros para o programa que esta sendo passado por um conjunto simples de variáveis de sistema, ou através de uma linha de comando.
- Segurança – Os programas CGI funcionam como processos independentes, completamente isolados do servidor Web. Isso torna mais difícil um programa problemático comprometer a integridade interna do servidor. Os programas CGI também são mais fáceis de depurar, por ser possível aconpanha a execução do processo CGI, sem a interferência de outros aplicativos (como um servidor Web).
- Proteção – O servidor pode controlar estritamente o acesso aos programas CGI.
- Portabilidade – Os aplicativos compatíveis com a especificação CGI 1.1 são portáteis para praticamente qualquer servidor Web. Além disso, como muitos programas são escritos em linguagens de script portáteis como Perl, são portáteis para uma ampla gama de plataforma de computadores.

As desvantagens da CGI são:

- Desempenho – A CGI não apresenta um bom desempenho em ambiente onde deve-se trabalhar com um grande números de pedidos. Para cada pedido CGI, o servidor deve criar um novo processo. A sobrecarga gerada pela criação de cada processo e os recursos consumidos por um numero potencialmente grande de programas CGI concomitantes podem reduzir muito a velocidade de um servidor.
- Estado – Não existe uma maneira natural de faze com que programas CGI diferentes compartilhem dados de trabalho. Como cada programa é um processo

independente, não é possível usar variáveis de memória ou estruturas de dados para passar informações de estado de um programa para outro. Em vez disso, você deverá basear-se na utilização de técnicas de armazenamento externo. Há técnicas no lado de cliente, tais como cookies de cliente e campos HTML ocultos; técnicas no lado do servidor, tais como arquivos temporários; e técnicas de codificação da Url, nas quais as informações de estado são passados pelo próprio URL.

3.4. Extensões ISA

No ambiente do Windows, o Aplicativo de Serviço da Internet é implementado como uma DLL. Este método proporciona uma maneira mais eficiente de implementar programas CGI, pois as DLLs são carregadas diretamente no espaço de endereço do sistema, e não fora desse, como um processo independente. Isso não apenas diminui a sobrecarga no carregamento de uma DLL, principalmente se comparado com a criação de um processo CGI, mas por ser uma parte do espaço de endereço do servidor também oferece ao ISA o acesso completo às estruturas internas de dados do servidor.

Um pedido ISA é especificado da mesma forma que um pedido CGI. A única diferença é a de que o nome do arquivo no URL de pedido possui a extensão DLL, em vez de CGI, BAT, EXE ou PL. esse padrão de nomeação ajuda no processo de conversão de programas CGI para ISA.

Com a introdução do ISAPI versão 2.0, os ISAs são identificados pela extensão ISA (DLL ainda é aceito).

Cada ISA deve ser registrado no servidor pelo registro do Windows. Geralmente, as DLLs Isa são carregadas sob solicitação do servidor quando este recebe um pedido e permanece na memória enquanto estiver sendo utilizada. O tempo que uma DLL não utilizada permanece na memória pode ser configurado no servidor. Quando esse tempo chegar ao final, o servidor poderá descarregar a DLL da memória, liberando lugar para outras.

Embora o tempo necessário para o inicio de um ISA seja curto, se comparado com a CGI, ainda há um atraso de inicio. Esse atraso pode ser reduzido a zero pelo carregamento prévio das DLLs, durante a inicialização do servidor.

Os processos CGI comunicam-se como o servidor por meio de um grupo de variáveis de ambiente (Tabela 3.1). Por outro lado, um ISA utiliza uma estrutura de dados na memória, chamada Extension Control Block (ECB – Bloco de Controle de Extensão). Que oferece

maneiras equivalentes de fazer referência às mesmas informações contidas nas variáveis de ambiente do servidor.

3.4.1 Filtros ISAPI

A arquitetura do servidor da Microsoft baseia-se em um modelo controlado por evento. Á medida que o servidor entra em cada fase do ciclo de vida do pedido, ele gera um evento correspondente à fase. Capturando esses eventos, um filtro é capaz de acessar um pedido antes que o servidor o faça. Isso oferece uma maneira poderosa de ampliar os recursos de servidor, tais como criptografia a conexão configuradas.

Cada filtro deve registrar-se para os eventos em que estiver interessado. As DLLs de filtro são registradas no servidor pelo registro do Windows. Quando o servidor carrega a DLL, esta indica sobre quais eventos possui interesse, junto com a prioridade. Assim, o filtro é chamado sempre que o servidor gera um evento de interesse.

Ter mais de um registro de filtro para o mesmo evento pode formar seqüências de filtros. A ordem em que os filtros são chamados é determinada pela prioridade do filtro, que é anunciada pelo mesmo quando esta sendo carregado. Especificando cuidadosamente a prioridade, é possível para o filtro ampliar a ação padrão do servidor. Cada filtro é capas de determinar se deve ou não passar o controle para o filtro seguinte na seqüência.

Os filtros podem registrar-se com a prioridade recomendada padrão ou com low, médium ou high. Nos casos em que dois filtros tiverem a mesma prioridade, a ordem de chamada será determinada pela ordem em que estiverem no registro.

Tabela 3.2 Eventos aos quais um filtro ISAPI deve responder.

Evento	*Descrição*
Conexão com uma porta segura	O filtro é notificado quando um pedido de conexão de cliente é feito em uma porta segura, como quando o cliente quer estabelecer uma conexão Secure Socketes Layer (SSL - Camada de Soquete de Segurança).
Conexão com porta não-segura	O filtro é notificado quando um pedido de conexão de cliente é feito em uma porta não-segura, normalmente a porta 80.

Leitura de dados puros	O filtro é notificado antes de o servidor ler os dados da conexão, dando ao filtro o acesso a toda a mensagem de pedido, seus cabeçalhos e a qualquer dado.
Processamento prévio de cabeçalhos	O filtro é notificado antes de o servidor ter feito o processamento prévio dos cabeçalhos da mensagem de pedido.
Conversão de URL	O servidor esta prestes a associar o URL virtual a um URL físico.
Autenticação de transação	O servidor esta prestes a autenticar o cliente.
Envio de dados puros	O servidor esta prestes a retornar a mensagem de resposta ao cliente.
Atualização de log	O servidor está prestes a gravar informações nos arquivos de registro (logs) do sistema.
Encerramento de conexão	A conexão com o cliente está prestes a ser fehada.

Fonte: Beveridge, Tony - 1998

O filtro deverá registrar-se somente para um evento com o qual fará alguma coisa. Registrar-se para eventos e depois não fazer algo com eles pode reduzir significativamente o desempenho do servidor, particularmente quando vários filtros estiverem sendo chamados por evento. Da mesma maneira, um filtro deverá atribuir somente a sim mesmo uma alta prioridade se for realmente processar a maioria dos eventos.

3.4.2 Como Funciona um ISA

No Windows, um Isa é implementado como uma DLL. Quando um cliente faz um pedido ISA, por exemplo para http://ntserver/Scripts/Samples/isasample0.dll, o servidor IIS carrega a DLL em seu espaço de endereço.

Depois que a DLL for carregada, o controle será passado para a mesma pelos pontos de entrada predefinidos, expostos pela DLL. Há dois pontos de entrada obrigatória. GetExtensionVersion e HttpExtensionProc (e um ponto de entrada opcional TerminateExtension).

Pode-se usar o utilitário dumpbin.exe do Windows NT para apresentar os pontos de entrada de uma DLL. Por exemplo, no prompt do MS-DOS, digite: dumppbin olamundo.dll/esports.

A figura 3.1 mostra o fluxo de controle entre o servidor e o ISA.

Há duas maneiras de chamar uma função em uma DLL.

- Ligação dinâmica durante o carregamento – O aplicativo chama uma função direramente na DLL. Isso exige que o aplicativo esteja previamente ligado estaticamente com a biblioteca de importação da DLL, para definir o endereço da função.
- Ligação dinâmica durante a execução – Uma DLL é carregada pelo aplicativo durante a execução, utilizando a função LoadLibrary do Windows, e o endereço da função, obtido com a função complementar GetProcAddress.

O IIS utiliza a ligação dinâmica durante a execução para acessar funções dentro de uma DLL ISA.

O ponto de entrada DIIMain é executado automaticamente quando a DLL é carregada pela primeira vez, e novamente ao ser descarregada. DIIMain é opcional – se você não fornecer essa função, sistema definirá uma pra você. DIIMain é um local onde poder ser executadas as funções de inicialização e encerramento do ISA. Opcionalmente, você pode ainda usar GetExtensionVersion e TerminateExpresion.

O controle é passado para GetExtensionVersion todas as vezes que sua DLL de extensão é carregada e após a execução de DIIMain. GetExtensionVersion é usada para definir a versão e a descrição do filtro,e e possivelmente para alocar recursos globais para a extensão.

Cada vez que o ISA é chamado (incluindo a primeira vez que a DLL for carregada), o controle é passado para HttpExtensionProc, que fica então responsável pela implementação completa de extensão – você pode considerar essa função como a função main() do ISA.

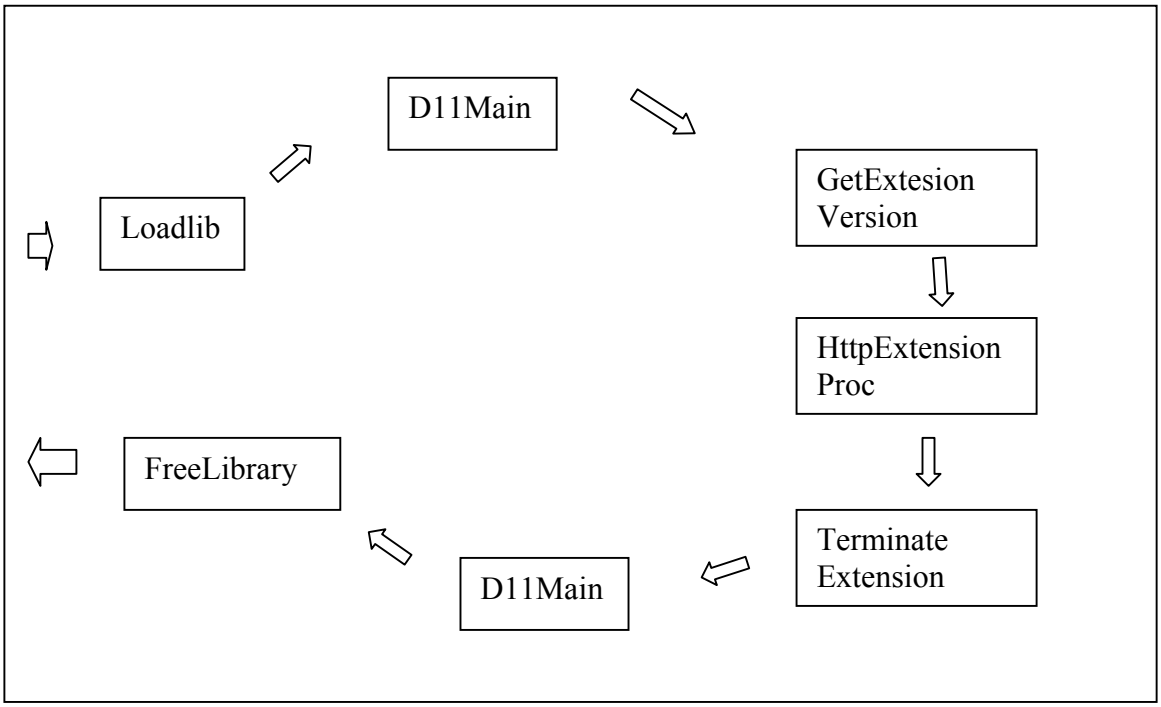

Figura 3.3 Fluxo de controle entre IIS e ISA. Fonte: Deveridge, Tony – 1998

Quando o servidor encerra a execução ou você pede que a extensão DLL seja descarregada após pedido, o IIS chama explicitamente TerminateExpension logo antes de descarregar a biblioteca, utilizando FreeLibrary. FreeLibrary dispara a execução de DIIMain mais uma vez. Você pode usar TerminateExpresion ou DIIMaim para limpar os recursos globais usados em sua extensão.

3.4.3 ISA Como Uma Alternativa aos Aplicativos CGI

Os ISAs são uma alternativa de alto desempenho para os aplicativos CGI. Já vimos que ISA é uma DLL que é carregada automaticamente pelo IIS em resposta a um pedido ISA. O pedido ISA foi projetado para fazer exatamente a mesma coisa que o pedido CGI. Isso facilita muito a conversão do método CGI para o ISA. A única diferença sintática visível é que, geralmente, a extensão do arquivo é DLL, em vez de EXE (aplicativo executável) ou PL (script Perl). Por exemplo, http://ntserver/samples/isasample1/isasample1.dll.

Tabela 3.3: Membros do ECB comumente utilizados e seus correspondentes CGI.

Variável membro ECB	Variável CGI equivalente	Significado
IpszMethod	REQUEST_METHOD	Método usado no pedido, por exemplo, GET ou POST

IpszQueryString	QUERY_STRING	Dados da string de consulta, logo apões o'?' do pedido GET http.
IpszPathinfo	PATH_TRANSLATED	Informação extra de caminho do URL do pedido.
IpszPath Translated	PATH_TRANSLATED	Versão traduzida de PATH_INFO, após a execução de qualquer mapeamento de ficico-para-virtual.
CbTotalBytes	CONTENT_LENGTH	Comprimento do corpo da mensagem de pedido, em bytes.
IpsxContentType	CONTENT_TYPE	Tipo de mídia do pedido.

Fonte: Beveridge, Tony – 1998

A vantagem óbvia de um ISA sobre um programa CGI é o desempenho. Como a DLL é carregada diretamente no espaço de endereço do IIS, a sobrecarga do carregamento de um ISA, se comparada à do programa CGI, é muito menor. Com o CGI, o servidor deve criar um processo independente, onde irá executar o aplicativo CGI. A sobrecarga da criação do contexto do processo e, em seguida, o início do aplicativo, pode ser significativa em um site muito concorrido, principalmente se o aplicativo tiver sido escrito em uma linguagem interpretada, como a Perl, que exige que o interpretador Perl (ele próprio um programa grande) seja carregado, para possibilitar a execução do script.

Depois que a DLL ISA tiver sido carregada no servidor, ela normalmente fica´ra lá até que o servidor encerre a conexão, ou descarregue explicitamente a DLL. Isto significa que, normalmente, a sobrecarga de carregamento da DLL ocorre somente uma vez. Os pedidos seguintes são relacionados diretamente com a cópia em memória da DLL.

Esse comportamento contrasta com o CGI, onde o aplicativo é carregado para caa pedido. Além disso, as DLLs de múltiplas extensões podem ser carregadas no servidor a qualquer momento.

3.4.4 Configuração ISA

Há duas propriedades de servidor que afetam diretamente a execução de um ISA. Veremos a seguir.

3.4.4.1 Registro de Diretório

O diretório que contém o ISA deve ser registrado com inetmgr, para que o servidor possa executar um programa ISA. Você poderá então usar a pasta Directories, de Service Properties, para que se servidor adicione o caminho físico do diretório e um nome de diretório virtual correspondente. Uma vez incluída, a propriedade de acesso como Read-only, o servidor abrirá seu ISA e o enviará para você como um arquivo binário, em vez de executá-lo.

3.4.4.2 Acesso de Registro

Quando um ISA é executado, ele o faz no contexto do usuário anônimo IUSR_nomeComputador, ou do nome usuário fornecido como parte do processo de autenticação do cliente, descrito anteriormente. Contudo, o perfil de usuário e os parâmetros de registro tornados disponíveis para o ISA pelo IIS, são os do usuário padrão. Você deverá ter cuidado para que seu ISA não se baseie em parâmetros de string para um determinado usuário, pois esses não estarão disponíveis.

3.4.5 Filtros ISAPI

Um filtro ISAPI é um mecanismo poderoso de ampliação ou substituição do comportamento padrão do IIS. Com os filtros, você poderá implementar método personalizados de conexão, autenticação e criptografia, implementar relacionamentos URL sofisticados, e muito mais.

Um filtro é algo diferente de um ISA. Um ISA é escrito para tratar um determinado pedido, e é referido diretamente no URL de pedido; em filtro, uma vez instalado no IIS, responde a todos os pedidos do cliente. Sem olhar explicitamente o URL do pedido (supondo-se que esteja nesse ponto do ciclo do pedido), o filtro não sabe de onde vem um pedido, ou como ele é feito (por exemplo, CGI, ISA, HTML estática, entre outros) – o filtro é simplesmente um dente da engrenagem do IIS.

4. HTTP

O HTTP (Hypertext Transfer Protocol – protocolo de transferência de hipertexto) é um protocolo padrão proposta. Seu status é eletivo. Esta descrita na Rfc 2068. O HTT 1.0 mais antigo é um protocolo informativo, como descrito na RFC 1945.

O protocolo de transferência de hipertexto é um protocolo projeto para permitir a transferência de documentos HTML. HTML é usada criar documentos de hipertexto. Estes incluem links para outros documentos que contem informações adicionais sobre a expressão ou assunto assinalado. Tais documentos podem conter outro elemento além de texto, como figuras clipes de áudio, applets Java e mesmo mundos de realidade virtual (que são descritos em VRML, uma linguagem de elaboração de roteiros para esse tipo de elemento).

4.1 Visão Geral do HTTP

O HTTP se baseia em uma atividade de requisição-REPOSTA. Um cliente, executando um aplicativo chamado de navegador estabelece ama conexão com um servidor uma requisição ao servidor na forma de um método de requisição. O Servidor responde com uma linha de status, incluído a versão do protocolo da mensagem e ama código de sucesso ou erro, seguida por uma mensagem contendo informações do servidor, informações sobre a entidade e um possível conteúdo.

Uma transação HTTP divide-se em quatro etapas:

1. O navegador abre uma conexão.
2. O navegador envia uma requisição ao servidor.
3. O servidor envia uma resposta ao navegador.
4. A conexão é fechada.

Na Internet, a comunicação HTTP geralmente ocorre em conexões TCP 80. A porta, mas outras portas também podem ser usadas. Isso não impede que o HTTP seja implementado sobre qualquer outro protocolo Internet, ou outras redes. O HTTP somente oferece um meio de transporte confiável; qualquer protocolo que garanta isso pode ser usado.

Exceto em aplicativos experimentais, a pratica atual requer que a conexão seja estabelecida pelo cliente antes de cada requisição e fechado pelo servidor depôs dede enviar a

resposta. Ambos, clientes e servidores devem estar cientes de que qualquer lado pode fechar a conexão prematuramente, devido a ação do usuário, temporizarão automática ou falha de programa, tendo que lidar com tais fechamentos de uma maneira previsível. Em qualquer caso, o fechamento de uma conexão por um lado ou ambos, sempre encerra a requisição atual, não importa seu status.

Em termos simples, o HTTP é um protocolo sem estado, porque ele não guarda informações sobe as conexões. Para carregar uma pagina incluindo duas figuras, por exemplo, um navegador que possa exibir figura irá abrir três conexões TCP, uma para a página e duas para as figuras. Contudo a maioria dos navegadores é capaz de lidar com varia dessas conexões simultaneamente.

Esse comportamento pode utilizar recursos de forma mais que intensa se uma pagina consistir de uma grande quantidade de elemento com um bom numero de paginas da Web, hoje em dia o faz. O HTTP1. 1 conforme definido na RFC 2068 alivia esse problema ao permitir que uma conexão TCP seja estabelecida para cada tipo de elemento de uma pagina, e que todos os elemento desse tipo serão transferido pela mesma conexão respectivamente.

Entretanto se uma requisição depender de informações trocadas durante uma conexão interior, então essa informação tem que ser mantida fora do protocolo. Uma maneira de armazenar e controlar essas informações persistentes é usar os chamados “COOKIES”.Um “cookie” é um conjunto de dados entre um navegador da Web e um servidor Web durante uma transação HTTP. O tamanho Maximo de um “cookie” é de 4KB. Todas essas parcelas de informações cookies são então armazenas em um único arquivo e colocadas no diretório do navegar da Web. Se os Cookies foram desativados, esse arquivo será automaticamente excluído um cookie pode ser lido e examinado pelo servidor em condições subseqüentes. Como os Cookies são tidos como uma exposição em potencial de privacidade, um navegar da Web deve permitir ao usuário decidir se aceita ou não os cookies e de quais servidor deve aceitá-los. Enquanto os cookies serve meramente manter algum tipo de estado nas conexões HTTP autenticações de clientes e servidores protegidas é propiciada pela camada SSL-SecureSocketsLayer.

4.2 Operação do HTTP

Na maioria dos casos a comunicação HTTP é iniciada pela agente do usuário requisitando um recurso no servidor de origem. No caso mais simples a conexão é estabelecida através de uma única conexão entre o agente do usuário e o servidor de origem.

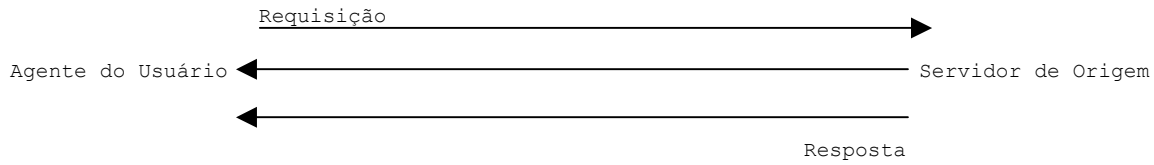

Figura 4.1. – Conexão Única Cliente/Servidor. Fonte: Martin, Murhammer – 2000

Em alguns casos, não há uma conexão direta entre o agente o usuário e o servidor de origem. Existe um ou mais intermediário entre o agente e o usuário e o servidor de origem como um Proxy, gateway ou túnel requisições e repostas são avaliadas pelos intermediários e redirecionadas para o destino ou outro intermediário na seqüência de requisição e resposta como mostrada na figura 4.2

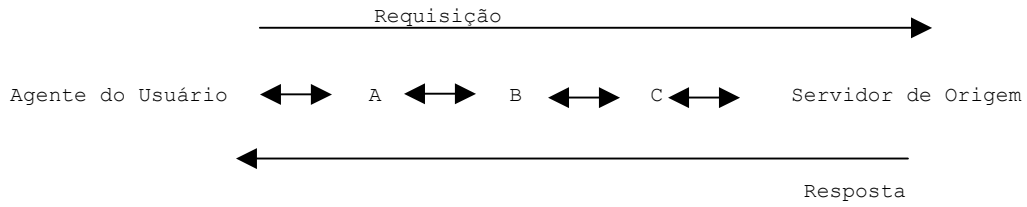

Figura 4.2. HTTP-Conexao Cliente/Servidor com Intermediários. Fonte: Martin, Murhammer – 2000

Um Proxy pode lidar com o conteúdo dos dados e assim modificar os dados de acordo. Quando uma requisição chega em um Proxy, ele reescreve toda ou parte de mensagem e a direciona para o próximo destino. Um gateway recebe a mensagem e a envia para os protocolos subjacentes em um formato apropriado. Um túnel não lida com o conteúdo da mensagem, apenas direcionando-a assim como é.

Os proxies e as Gateways de um modo geral podem gerenciar o armazenamento (caching) de mensagens HTTP, o que permite reduzir dramaticamente o tempo de resposta e o

trafego IP na rede. Como os túneis não entendem o conteúdo da mensagem, eles não podem armazenar em cachê dados de mensagens HTTP. Na figura anterior, se um dos intermediários (A,B, e C) adotasse um cachê interno parta mensagens HTTP, o agente do usuário poderia obter a resposta do intermediário se esta tivesse sido antes armazenada a partir do servidor de origem na seqüência de resposta. A figura a seguir ilustra que A tem um copia de uma resposta anterior oriunda do servidor de origem na seqüência de resposta. Assim, se a resposta do servidor para a requisição não estiver presente no cachê interno do agente do usuário, ela poderá ser obtida diretamente de A.

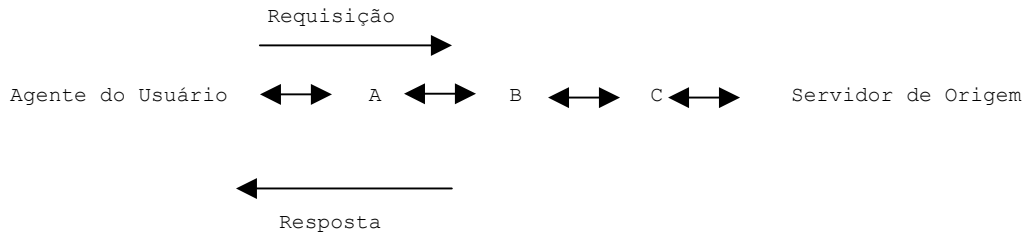

Figura 4.3. HTTP-Resposta de Servidor Usando Cachê. Fonte: Martin, Murhammer – 2000

Um sistema de cachê não se aplica a todas as respostas de um servidor. O comportamento de um sistema de cachê pode ser modificado por requisições especiais a fim de determinar que respostas do servidor podem ou não ser armazenadas. Com essa finalidade, as respostas do servidor podem ser marcadas como não armazenáveis em cachê, publicas ou particulares (não podem ser armazenadas em um sistema de cachê publico).

4.3 Parâmetros do Protocolo

Alguns dos parâmetros do protocolo HTTP são mostrados a seguir. Consulte a RFC 2068 para obter uma lista completa com todos os detalhes:

4.3.1 HTTP - Version

O protocolo HTTP usa esquema de numeração <maior> <menor> para indicar as versões do protocolo. A conexão mais distante será realizada de acordo com as regras da versão do protocolo.

O numero < maior> é incrementado quando há alterações significativas no protocolo, tal como a mudança no formato das mensagens. O numero < menor> é incrementado quando a alteração não afeta o formato das mensagens.

A versão das mensagens HTTP é enviada por um Campo HTTP – Version na primeira linha da mensagem. O formato do campo HTTP-Version é seguinte (consultar a RFC 822 sobre a Forma Ampliada de Backus-Naur):

HTTP-Version = "HTTP" "/" 1*DIGITO "." 1* DIGITO

4.3.2 Identificadores de Recursos Uniformes (URI- Uniform Resource Identifiers)

Os URIs são tidos geralmente como endereços WWW e combinações de URLs (Uniform Resource Locations – localizadores de recursos uniformes) e URNs (Uniform Resource Names- nomes de recursos uniformes). De fato, os URLs são strings (cadeias de caracteres) que indicam a localização e o nome da origem no servidor. Consulte as RFCs 2086 e 2396 para saber mais detalhes sobre a sintaxe de URIs e URLs.

4.3.3 Sistema de Cachê do HTTP

Um dos recursos mais importantes do protocolo HTTP é sua capacidade de cachê. Como o HTTP é um protocolo baseado em informações distribuído, o armazenamento dessas informações em um sistema de cachê pode aumentar muito o desempenho. Há inúmeras funções no protocolo HTTP 1..1 que usam cache de maneira eficiente e adequada.

Na maioria dos casos, as requisições dos clientes e as respostas dos servidores podem ser armazenadas em um cache durante um período de tempo razoável, a fim de processar requisições futuras correspondentes. Se a resposta estiver no cache e for correta, não haverá necessidade de requisitar outra resposta do servidor. Essa metodologia não apenas reduz a largura de banda exigida na rede, mas também aumenta a velocidade. Há um mecanismo mediante o qual o servidor estima um tempo mínimo durante o qual a mensagem de resposta será valida. Isto significa que é determinado um tempo de *validade* pelo servidor para uma mensagem de resposta particular. Assim, durante esse período, a mensagem poderá ser usada sem consultar o servidor.

Considere agora que esse tempo foi esgote é necessária à mensagem de resposta. Os dados dentro da mensagem podem ter sido modificados ou não após a data a limite. Para pode

ter certeza se os dados foram alterados ou não, um mecanismo de *validação* é definição da seguinte forma:

4.3.3.1 Mecanismo de Expiração

A fim de decidir se os dados são novos ou não, precisa determinado um período de validade. Na maioria do casos, o servidor de origem define explicitamente o tempo de validade para uma mensagem de resposta particular dentro dessa mensagem. Se esse for o caso, os dados armazenados no cache podem ser enviados nos requisições subseqüências dentro do período de validade.

Se o servidor de origem não definiu prazo, há alguns métodos para se estimar/calcular um tempo de validade razoável (como a hora da ultima modificação). Como estes não são originados no servidor, devem ser usados com precaução.

4.3.3.2 Mecanismo de Validação

Quando o tempo de validade é esgotado, existe a possibilidade de que os dados não estejam atualizados. A fim de garantir a validação da mensagem de resposta, o cache precisa verificar no servidor de origem (ou, possivelmente, em um cache intermediário com uma resposta mais recente), se a mensagem de resposta ainda é utilizável. O HTTP 1.1 oferece métodos condicionais com esse propósito.

Quando um servidor de origem envia uma resposta completa, ele acrescenta algum tipo de validação á mensagem. Isso será então usado como *mecanismo* de *validação de cache* pelo agente do usuário ou cache do Proxy. O cliente (agente do usuário ou cache de Proxy) gera uma requisição condicional com um mecanismo de validação de cache anexado. O servidor então avalia a mensagem e responde com um código especial (geralmente, 304 – Não Modificado) e sem corpo de entidade. De outro modo, o servidor envia a resposta completa (incluindo o corpo da entidade). Esse método evita uma ida e volta extras caso a validação coincida. Esse método evita uma ida e volta extras caso o mecanismo de validação não coincida e também evita enviar a resposta completa caso a validação coincida.

4.4 Linguagem HTML

A linguagem HTML (Hypertext Markup Language – Linguagem de marcação de hipertexto) é uma da maiores atrações da Web. Ela tem um conjunto arquitetado de instruções que devem ser entendidas por todos os navegadores e servidores Web, embora á medida que novos recursos são adicionados á HTML, eles não sejam suportados pelos navegadores da Web mais antigos. Essas marcas (tags) independem do dispositivo. O mesmo documento pode ser enviado de um PC, uma maquina AIX ou UNIX, OU de um computador de grande porte, e o navegador Web em qualquer maquinas de cliente poderá entender as marcas HTML, refazendo e exibindo os dados no dispositivo de destino. As marcas HTML descrevem elementos básicos de um documento Web, como cabeçalho, parágrafos, estilos de texto e listas. Há também marcas sofisticadas para criar tabelas e incluir elementos interativos, como formulário, scripts ou applets Java.

4.5 A XML

A linguagem XML (Extensible Markup Language – linguagem de marcação extensível) descreve uma classe de objetos de dados chamados documento XML que são armazenados em computadores e que descrevem parcialmente o com portamento de programa que processam esses objetos. A XML é um perfil de aplicativo ou forma restrita da SGML. O objetivo da XML é um perfil de aplicativo ou forma restrita da SGML. O objetivo da XML é possibilitar que a SGML genérica possa ser servida, recebida e processada na web da forma que atualmente é possível com a HTML. A XML foi projetada visando a facilidade de implementação e interoperacao com SGML e HTML.

4.6 JAVA

Java é uma nova e importante tecnologia no mundo da Internet. Em resuma, Java é um ambiente de programação dinâmico de proposição de propósito, geral, simples robusto, orientado a objetos, independente de plataforma e de multiprocessamento para criação de aplicativos para Internet e intranet.

4.6.1. Visão Geral dos Componentes Java

Java, em si, é uma linguagem de programação desenvolvida pela Sum Microsystems . Ela é orientada a objetos e foi projetada para ser um projetada para ser um primo próximo do C E C ++, porém fácil de programa, mais portável e mais rosulta. Os autores fizeram um trabalho excelente removendo da recursos do CC++ que costumam deixar os programadores noites sem dormir, embora retendo a sua boa funcionalidade.

Os recursos do Java que são especialmente importantes no ambientes Web são os seguintes:

4.6.1.1 Portabilidade

Java é uma linguagem compilada, mas que diferente de outras linguagens, não é compilada em instruções de maquina, mas em bytecodes (códigos de bytes) independentes de arquiteturas. Isso significa que o mesmo programa irá executar em qualquer sistema que tenha o ambiente de tempo de execução Java, não importa o sistema operacional ou tipo de hardware.

4.6.1.2 Suporte ás Linhas de Execução (threads)

Java não tem suporte e linhas de execução (threads) incorpocorado, tornando- a muito eficiente em aplicativos dinâmico e multimídia.

4.6.1.3 Gerenciamento de Memória

Java não requer que o programador faca qualquer alocação de memória: ele faz isso automaticamente. Ela também não disponibiliza nenhum tipo de ponteiro, de modo que é impossível para um programa tentar ler ou gravar em uma memória não reservada. Essas são, provavelmente, as duas causas mais comuns de erros de programas escritos em linguagem convencionais. Fora o fato que isso torna a linguagem mais robusta e confiável, também remove uma falha de segurança em potencial. Uma técnica favorita de ataque em linguagens convencionais é encontrar erros de códigos que permitam que dados sejam sobrescritos.

4.6.1.4 Verificação de Código

Em uma linguagem de programação mais convencional, é possível para o programada alterar os ponteiros de execução e de endereços de dados em tempo de execução. Em Java isso não é permitido; todas as referencias são feitas por nome. Isso significa que é possível verificar, antes, se um programa está fazendo algo quer que ele faca.

Um sistema Java inclui os componentes examinados nas seções a seguir.

4.6.2 Linguagem Java

Java é uma linguagem de programação desenvolvida pela Sun Microsytems, orientada a objeto, distribuída interpretada, independente de arquitetura de arquitetura e portável. Java pode ser usada para criar fragmentos de programa descarregáveis, chamados apples, que aumentam a funcionalidade de um navegador capaz de Java como o HotJava ou Nestcape Navigator. Para desenvolver aplicativos em Java, é necessários um Kit de desenvolvimento Java (JKD, Java Development Kit). Para executar programas Java, é necessário ou um navegador da Web que suporte Java ou um JRE (Java Run – Time Environment - ambiente de execução em tempo real Java), que é o menor subconjunto do Kit de desenvolvimento Java.

4.6.3 Máquinas Virtual Java

A JVM (JavaVirtual Machine – Máquina Virtual Java) – é um computador abstrato que executa programas compilados Java (ou, mais precisamente, interpreta os bytecodes produzidos por um compilador Java) A JVM é virtual porque ela é geralmente implementada em softwares no topo de uma plataforma. Todos os programas Java devem ser compilitados para executar em uma JVM.

O diagrama a seguir descrever em termos simples como Java é implementado:

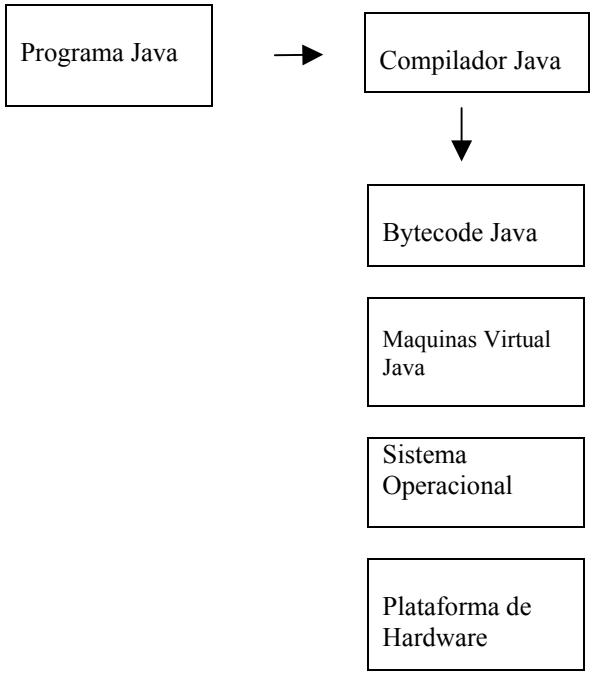

Figura 4.4 Java – Implementação de Java. Fonte: Martin, Murhammer – 2000

4.6.4 Programas, Applets e Servlets

Quando um programa Java é Iniciado a partir de uma pagina Web HTML, ele é chamado de applet Java em oposição a um programa Java é executado a partir da linha de comando ou de algum outro modo no sistema local. A applets são recebidas através do nagevador Web vinda de um servidor e, por definição, são um tanto quanto limitadas na maneira com podem usar recurso do sistema local.

Originalmente, nenhuma applet Java poderia tocar em tocar em algo fora de sua localização JVM e só poderia se comunicar com o servidor do qual foi transmitido. A partir da Java 1.1, as applets podem ser assinadas com chaves de proteção e certificados, podendo assim ser autenticadas. Com isso uma applet pode ser autorizada a acessar recurso locais, como sistemas de arquivos, e se comunicar com outros sistemas.

A fim de distribuir os recursos dos clientes e das redes, applets Java podem ser executadas no servidor em vez de descarregadas e executadas no cliente. Tais programas são então denominados servlets. Embora tal método exija um servidor bem mais bem mais poderoso, ele é altamente apropriado em ambiente com sistemas com poucos meios de comunicação, tal como computadores em rede.

4.6.5 HotJava

HotJava é o navegador da Web capaz de interpretar Java desenvolvido pela Sun Microsystems. Ele vem com suporte complete a Java, o que permite executar applets Java.

4.6.6 JavaOS

JavaOS é um sistema operacional altamente compacto, desenvolvido pela JavaSoft, projetado para executar aplicativos Java diretamente em microprocessadores desde computadores pessoais a "pagers". O JavaOS irá executar igualmente bem em um computador de rede, um PDA, uma impressora, uma maquina de jogos ou inúmeros outros dispositivos que exijam um sistema operacional muito impacto e a habilidade de executar aplicativos Java.

4.6.7 Java Beans

Uma iniciativa chamada JavaBeans oferece um conjunto semelhante de APIs que torna simples a criação de aplicativos Java a partir de componentes reutilizáveis. JavaBeans poderão ser usados em inúmeros aplicativos, desde simples widgets (Windows god gets, elementos de tela) de larga escala, até aplicativos de missão critica. Vários fabricantes de software, incluindo a IBM, tem divulgado seus planos de oferecer suporte a JavaBeans.

4.6.8 Java Script

JavaScript é uma linguagem de programação e extensão de HTML desenvolvida pela Netscape, sendo uma linguagem simples baseada em objetos compatível com Java. Programas JavaScript são embutidos como fonte diretamente em um documento HTML. Eles podem controlar o comportamento de formulários, botões e elementos de texto. JavaScript é usado para comportamentos dinâmicos em elementos de paginas Web. Além disso, ele pode ser usado para criar formulários cujos compôs tem Rotinas de verificação de erros incorporada.

4.7 O Que é VRML?

VRML é sigla para *Virtual Rality Modeling Language,* uma linguagem para descrever cenas interativas multiparticipativas tridimensionais em tempo real, independentemente da plataforma, através da Internet.

4.7.1 VRML Versus HTML

VRML está para o mundo tridimensional assim como HTML esta para o mundo bidimensional. Enquanto a linguagem HTML especifica como documento bidimensionais (2D) são representados, VRML é um formato que descreve como ambientes tridimensionais (3D) podem ser explorados e criado na WWW.

No entanto, VRML permite uma interação como o usuário muito maior do que HTML. O usuário, em paginas 2D, tem as opções limitadas de pular de pagina a pagina e receber textos, imagens e outras mídias, como áudio, vídeo e imagens dinâmicas. Ao passo que, nos mundo VRML, o usuário pode escolher livremente a perspectiva de visualização e movimentar-se de acordo com a sua vontade.

Os espaços virtuais possuem também algumas vantagens suas limitações são determinadas apenas pelos recursos de hardware e software disponíveis na maquina e existe uma melhor assimilação das informações pelo usuário, a partir da percepção de ambientes tridimensionais, tal qual ocorre no mundo real.

4.7.2 Mundos Virtuais

A linguagem VRML acrescenta percepção á navegação na Web, através de discrições completas da cena em que o usuário se encontra, montando verdadeiros “mundos virtuais”(razão pela qual os arquivos de VRML apresentam a extensão “.wrl”- de world).

Da mesma forma que no mundo real, o usuário mantém uma perpctiva individual perante a cena, controla completamente quão perto quer chegar de um objeto e a partir de que perspectiva visualizá-lo.

Além disso, os objetos presentes na cena podem apresentar referencias. Se o usuário clicá-los, será levado para outro mundo VRML ou outro tipo de mídia como sons, imagens ou

páginas 2D, semelhantemente à navegação em um hipertexto HTML, em que se pode pular de um documento a outro.

A navegação pelos mundos virtuais é semelhante a andar em uma cidade o visitante olha em volta e cria um mapa mental do local. À medida que passeia pela cidade, o visitante entra em ruas e faz curvas, traçando o caminho que o levará aonde deseja chegar. Existe um senso de continuidade, embora algumas vezes se possa ficar perdido. De qualquer forma, é bastante diferente de escolher uma URL e esperar um arquivo ser carregado na tela.

Os mundos virtuais utilizam uma filosofia de gráficos 3D. Eles se constituem entre outros aspectos, de geometria, transformações, iluminação e textura.

5. A INTERNET ESTÁTICA

Nos primórdios da Word Wide Web, todas, as informações servidas ao navegador do cliente eram estatísticas. Em outras palavras, o conteúdo de páginas. A que servia ao cliente 1 erra exatamente o mesmo que o conteúdo da página a servido ao cliente 2. O servidor da Web não gerava dinamicamente nenhuma parte do conteúdo do site, mas simples, e te atendia a solicitações de páginas de HTML, estatísticas, que eram carregadas a partir do sistema de arquivo do servidor Web e enviadas ao cliente solicitante. Não havia nenhuma interatividade entre o usuário o servidor. O navegador solicitava informações e o servidor as enviava.

Embora a Internet estática se desenvolvesse rapidamente para incluir imagens gráficas e sons, a Web ainda era estática, com pouca interatividade e muito pouca funcionalidade além daquela fornecida pelos o hiperlinks simples.

5.1 Internet Dinâmica, Parte I: Aplicativos CGI

Uma das primeiras extensões da Internet estática foi a criação do Common Gateway Interface. O CommonGateway Interface, ou CGI fornece um mecanismo pelo qual um navegador da Web pode comunicar uma solicitação de execução de um aplicativo no servidor da Web. O resultado desse aplicativo é convertido/formatado em uma forma legível pelo navegador (HTML) e enviado para o navegador solicitante.

Os aplicativos CGI elevaram o nível no que era esperado de um site da Web fizeram a transição da Word Wide Web entre aquilo que era apenas uma maneira fácil de compartilhar informações e ama plataforma viável para processamento de informações. A resposta para essa evolução da Web foi o crescimento acelerado e o inicio do interesse do mundo dos negócios pela Internet.

Parte desse crescimento foi a criação de várias soluções de escript do lado do cliente que permitiriam a maquina do cliente assumir parte das tarefas de processamento. Entre essas soluções do lado do cliente as mais notáveis são o JavaScript do Netscape e o VBScript da Microsoft.

Durante esse enorme crescimento de tecnologia baseadas na Internet, a Microsoft lançou o seu Internet Information Server (IIS). Elogiado por ser de fácil utilização, escalável, portável, seguro e extensível, esse softwer também é gratuito e minuciosamente integrado

com sistemas operacionais Windows NT e Windows 2000 da Microsoft. Ele rapidamente Tornou-se muito popular.

5.2 Internet Dinâmica, Parte II: ISAPI

Além de suportar a especificação de CGI, a Microsoft introduziu uma alternativa ao CGI, o Internet Server Application Programming Interface (ou ISAP). O ISAPI resolve um dos recursos mais limitados dos CGI.

Toda vez que um cliente solicita a execução de um aplicativo CGI, o servidor da Web executa uma instancia separada do aplicativo, envia informações da solicitação do usuário do e fornece os resultados do aplicativo CGI precessando para o cliente. O problema dessa abordagem é que um aplicativo CGI separado é carregado para cada solicitação. Isso pode drenar significativamente os recursos do servidor se houver muitas solicitações para o aplicativo CGI.

O ISAPI alivia esse problema contando com bibliotecas de link dinâmico(dynamic link libraries – DLLs). Cada aplicativo de ISAPI está na forma de uma única DLL que é carregada no mesmo espaço de memória que o servidor da Web na primeira solicitação ao aplicativo. Uma vez na memória, a Adll permanece lá, respondendo a solicitações do usuário até ser explicitamente liberada da memória. Esse aumento da eficiência na utilização da memória tem um custo. Todas as DLLs da ISAPI devem ser thead-seguras (*thread-safe*) de modo que os diversos threads possam ser instanciados na DLL sem causar problemas com a função do aplicativo.

Os aplicativos de ISAPI normalmente são mais rápidos que os aplicativos de CGI equivalentes, uma vez que o servidor da Web não precisa instanciar um novo aplicativo toda vez que uma solicitação é feita. Uma vez que a DLL de aplicativo de ISAPI é carregada na memória, ela permanecerá lá. O servidor da Web não precisa carregá-lo novamente.

Além dos aplicativos de ISAPI, o ISAPI permite o desenvolvimento de filtros de ISAPI. Um filtro de ISAPI é uma DLL PERSONALIZADA que esta no mesmo espaço de memória que o servidor da Web e é chamada pelo servidor da Web em resposta a cada solicitação de http. Dessa maneira, o filtro de ISAPI altera a própria maneira como o servidor da Web se comporta. O filtro de ISAPI então instrui o servidor da Web sobre como tratar a solicitação. Os filtros de ISAPI permitirem assim personalizar a resposta de seu servidor da Web para tipos específicos de solicitações de usuário. Para tornar mais clara a diferença entre filtros de

ISAPI e aplicativos de ISAPI (e aplicativos CGI), os filtros de ISAPI oferecem três tipos de funcionalidade que os distinguem dos aplicativos ISAPI(ou CGI):

- Um filtro de ISAPI permite fornecer uma forma de segurança no nível de página ou site da web mediante sua inserção como uma camada entre o cliente e o servidor da Web.
- um filtro da ISAPPI permite monitorar mais informações sobre as solicitações ao servidor da Web e sobre o conteúdo fornecido ao requisitante de quem um servidor da web de http padrão por si só. Essas informações podem ser armazenadas em um formato separado das funções de registro em log do servidor da web.
- um filtro de ISAPI pode fornecer informações aos clientes de ama maneira diferente que o servidor da Web pode fazer pó si só.

Alguns exemplos de possíveis filtros de ISAPI:

- Uma camada de segurança entre o cliente e o servidor da web. Essa camada de segurança poderia oferecer uma triagem mais completa das solicitações dos clientes do que aquela oferecida pela simples autenticação de nome de usuário e senha
- um filtro personalizado poderia interpretar o fluxo das informações do servidor e, com base nessa interpretação, apresentar o fluxo em um formato diferente do apresentado pelo servidor origina da web. O ASP.DLL é um exemplo desse tipo de filtro de ISAPI. ele interpreta o código de servido em um script solicitado pelo cliente e dependendo de sua interpretação, fornecedor o conteúdo personalizado de cliente conforme a solicitação do cliente
- um filtro personalizado poderia mapear uma solicitação do cliente para uma localização física diferente no servidor. Isso poderia ser utilizado em sites de alto volume onde você talvez queira mover o cliente para um servidor diferente.

5.3 O Modelo de Objeto do ASP

O ASP abrange as propriedades e métodos das sete objetos predefinidos a seguir:

- Application
- ASPEerror
- ObjectContext

- Request
- Response
- Server
- Session

Esses objetos são da ASP.DLL e estão sempre disponíveis para os seus aplicativos ASP.

O objeto *Application* representa o próprio aplicativo ASP. Esse objeto é universal para todos os usuários associados a um aplicativo e há um objeto application para todos os usuários. Esse objeto tem dois eventos, Application_OnStart e Application_OnEnd, que são acionados, respectivamente, quando o primeiro usuário solicita uma pagina de seu aplicativo e quando o administrador descarrega explicitamente o aplicativo utilizando o Microsoft Mangement Console. O evento OnStart ode ser utilizado para inicializar informações necessária para cada aspecto do aplicativo. O evento OnEnd pode ser utilizado para fazer qualquer tipo de trabalho de limpeza personalizada depois do seu aplicativo. E possível armazenar qualquer tipo de variável com escopo de nível de aplicativo. Essas variáveis armazenam o mesmo valor para cada usuário site.

Adicionando ao Active Server Pages 3.0, o objeto ASPError permite que os desenvolvedores acessem propriedade que caracterizam o último erro ocorrido no script atualmente em execução. Ele somente é acessivo pelo método de GetLastError do objeto Server e todas as suas propriedades são apenas de leitura. A adição ASPError é um aperfeiçoamento no Active Server Pages 3.0 e aprimora significativamente tratamento dos erros nos scripts de Active Sever Pages.

O objeto *ObjectContext* é realmente parte do Microsoft Transaction Server e sua interface só é obtida através do ASP. O ObjectContext permite criar Active Server Pages para realizações de transações as funções dessas paginas que suportam transações ou serão bem-sucedidas como um todo ou falharam completamente. Se seu aplicativo exige o uso de funções que originalmente não suportam transações (notavelmente acesso de arquivo), você deve escrever um código personalizado para lidar com o sucesso ou fracasso dessas funções.

O Objeto *Request* representa a maneira como você interage com a solicitação HTTP do cliente. Esse é um dos objetos mais importante no modelo de objeto do ASP. É pelo uso do objeto Request que você acessa parâmetros e dados de HTML baseados em formulários, que são enviados pela linha de endereço. Além disse é possível utilizar o objeto Request para receber informações de cookies HTTP e informação de certificado de cliente de seus usuários.

Por fim a coleção ServerVariables do objeto request fornece acesso a todas as informações no cabeçalho de solicitação de HTTP. Essas informações contem (alem das informações de cookies) outros dados relevantes que descrevem a maquina do cliente sua conexão e suas solicitações propriamente ditas. A coleção ServerVariables é equivalente as variáveis nos aplicativos de CGI tradicionais.

O objeto *Response* representa seu controle/acesso sobre a resposta de HTTP enviada de volta para o usuário. Através do objeto Response você pode enviar cookies para o cliente e configurar seu conteúdo deve espirar e quando isso deve acontecer ale disso o objeto response é uma forma de controlar completamente a maneira como os dados são enviados para o cliente. Eles são armazenados no buffer? Eles são enviados como foram construídos? Por fim, o objeto Response permite redirecionar o usuário transparentemente de um URL para outro URL.

O objeto *Server* fornece aceso ao próprio servidor da Web. Esse objeto contém muitos recursos de utilitário que você utiliza em quase todos aplicativos. Através do objeto Server, é possível configurar a variável de tempo e limite para seus spripts (por quando tempo o servidor da Web tentara servir um script antes de fornecer uma mensagem de erro em seu lugar). Você também pode utilizar o objeto Server para mapear um caminho virtual para um caminho físico ou codificar informações para enviar sobre a linha de endereço. Entretanto, o método mais importante do objeto Serve é seu método CreatObject, que permite criar instancias do componentes do lado do servidor você utilizará esse método todas vez que precisar de funcionalidade além daquela fornecida pelos objetos predefinidos. O acesso de bancos de dados, por exemplo, é tratado por vários Active X Data Objects que devem ser instanciados no servidor antes de serem utilizados.

Por fim, o objeto *Session* armazena informações que são únicas para uma seção de andamento, de um usuário especifico no servidor da Web. Cada sessão de usuário é identificada pelo uso de um cookie que é enviado para o usuário toda vez que ele faz uma solicitação. O servidor da Web inicia uma sessão para cada usuário novo que solicita uma pagina de seu aplicativo para Web. Essa seção permanece ativa por padrão por 20 minutos desde a ultima solicitação do usuário ou até que a sessão seja explicitamente abandonada por meio de código.

5.4 Script do Lado do Cliente

A Hypertext Markup Language, ou HTML oferece recurso de formatação muito detalhados para conteúdo textual estático. Esse conteúdo pode conter imagens, tabelas, texto e hiperlinks formatados cuidadosamente, o que a torna uma mídia muito poderosa para apresentação de informação. Entretando, independentemente da interatividade de nível muito baixo dos hyperlinks e sua capacidade de mover o usuário de uma pagina para outra, a HTML por si mesma não permite nenhuma interatividade verdadeira. A HTML não permite que a página da Web reaja à entrada do usuário de nenhuma maneira além de navegar para outra página. A HTML é uma excelente maneira de permitir a apresentação de informações, mas não permite a interatividade requerida para transformar paginas as Web a partir de uma mídia de informações para uma solução dinâmica de aplicação para a Web.

A Netscape Communications, juntos com a Sun Microsystems, criaram uma solução chamada LiveScript que permite a inclusão de instruções de programação limitada que residem nas paginas da Web visualizadas utilizando o navegador de Netscape Navigator na maquina cliente. Essa linguagem de programação era limitada em sua capacidade de interagir com a maquina do usuário fora do navegador e, ao longo de desenvolvimento, se tornou segura e protegida. Você não poderia utilizar instruções de programa LiveScript na maquina do cliente para subverter a segurança inata do navegador Netscape Navigator. O LiveScript, de acordo com o furor de marketing que cercava o Java, foi rapidamente renomeado para JavaScript. Infelizmente, essa reatribuicao de nome induziu, erroneamente, muitas pessoas a acreditarem que ele era um subconjunto da poderosa linguagem Java, embora apenas sua sintaxe fosse semelhante á da Java.

A HTML ganhou vida. Utilizando o JavaScript, você poderia construir formulários e calculadoras de hipoteca e topo tipo de paginas da Web interativas. A única desvantagem era que seu navegador tinha de ser um host de script para essa linguagem de criação de scripts. Mas, dito isso, o conteúdo da Web rapidamente transformou-se estático e amplamente simples em interativo e vivo.

Antes de JavaScript, toda interação com o usuário e todas as reações da parte do servidor da Web requereram os usos de aplicativos sofisticados de servidores da Web e maquinas mais sofisticadas de servidores da Web. Com o Advento do JavaScript, a maquina do usuário era agora adicionada á equação , tornando possível fazer o offload de alguma dessa

potencia computacional sobre o cliente, ao passo que antes ele unicamente permanecia no servidor.

Para não ficar para trás, a Microsoft Corporation rapidamente criou uma linguagem de criação de Scripts própria:Visual Basic, Scripting Edition ou VBScript, para abreviar. VBScript é um subconjunto da linguagem Visual Basic for Applications e como JavaScript, permite a criação de pagina da Web interativas, diferentemente do JavaScript cuja a síntese era semelhante a essa do Java (e assim semelhante áquela do C+ +), a sintaxe de VBScript era exatamente essa do Visual Basic. Se você soubesse (e muitas, muitas pessoas sabem), já teriam uma boa compreensão do VBScript. Além disso, a Microsoft também criou sua própria versão do Java Script chamada JScript que era semelhante mais não idêntica ao seu predecessor.

Hoje (apenas alguns anos mais tarde), o JavaScript sofreu uma transformação em uma nova linguagem construída utilizando contribuição tanto da Netscape como da Microsoft. Essa nova linguagem é chamado ECMAScript (da European Computer Manufactures Association).De acordo com David Flanagan em *JavaScript: The Definitive Guide,* esse nome foi escolhido especificamente porque ele não tinha nenhuma relação com a empresa de origem e não tinha nenhuma ostentação da marketng do Java artificialmente a ele associado a ele. Tanto a Netscape como a Microsoft continuaram a ajudar o ECMAScript (ainda chamado JavaScrept por todo mundo exceto pelos membros da European Computer Manufacturers Association) a desenvolver-se. Para mais detalhes sobre as implementações de navegadores diferentes, Flanagam fornece uma excelente cobertura em seu livro.

Embora a discussão precedente sugira que somente o JavaScript e VBScript existem, o navegador da Web realmente permite ama variedade de alternativas de linguagem de criação de scripts. Você mesmo poderia construir a sua própria. Algumas outras linguagens incluem PerlScrpt, Python e Awk, com PerScript sendo a mais popular depois do JavaScript e do VBScript.

Entretanto duas coisas que todas as linguagens de criação de scripts tem em comum são as maneira como elas são incluídas em uma pagina da Web como o navegador às reconhece como escript e não como HTTP. Todo o script é cercado pelos tags pareados <script></scriopt>, como ilustram uma das três exemplos de escript do lado do cliente no exemplo a seguir. Cada uma das três rotinas realiza exatamente a mesma ação uma exibe uma caixa de mensagem na tela contendo a frase “Hello world”.

Criação de scripts do lado do cliente utilizado três linguagens de scripts

```
<script languare = “javascript” >
<!-
Function arlerjs( )
{
alert(“Hello world.”)
}
->
</SCRIPT>
<script languare = “vbscript”>
<!-
Sub AlertVBS ( )
MsgBox “Hello World.”
End Sub
->
</SCRIPT>
<SCRIPT language=”PERLsCRIPT”>
<!-
Sub AlertPS( )
{
$window->alert(hello word.”);
}
->
</SCRIPT
```

Ha dois recursos do exemplo a observar. O primeiro é como o código real é cercado pelos símbolos de comentário de HTML:

```
<!-
Código aqui
->
```

Isso permite que a pagina seja mostrada em navegadores que não suportam o tag <SCRIPT> sem causar problemas ou exibir o script na pagina.

O segundo recurso é atribuído LANGUAGE do tag, <script>, que você sem duvida adivinhou, indica qual linguagem de criação de scripts o navegador deve utilizar para executar o código incluído. Essa pode ser qualquer linguagem que ser navegador suporta. JavaScript é provavelmente a aposta mais segura para o script do lado do cliente, visto que o VBScript é suportado somente com o uso de plug-ins em navegadores não-Microsoft e que outras linguagens de criação de scripts não são comumente instaladas em maquinas de usuário. Para

mais informações sobre o JavaScript, veja o excelente livro de David Flanagan, *JavaScript: the definitive guide,* terceira edição. Para mais informações sobre VBScript, consulte o VBScript in a Nutchell, de Matt Chids, Paul Lomax e Ron Petrusha. Ambos foram publicados pela O"Reilly& Associates. Vamos revisitar a questão da criação de linguagem de scripts no final deste capitulo".

5.5 Script do Lado do Servidor

Quando o navegador faz uma solicitação de um arquivo terminando com a extensão de arquivo.ASP, o IIS sabe colocar ASP.DLL e ação para interpretar o código ASP NO ARQUIVO. Uma vez interpretado, os resultados desse código são colocados no documento, que é um documento de HTML simples antes de ele ser enviado para o usuário.

Como o ASP.DLL sabe que código interpretar? a resposta a essa pergunta é a chave para executar código no servidor. O ASP.DLL interpreta todo o código em um arquivo(com a extensão de arquivo ASP) que é delimitado com <%...%> como sendo o código ASP. (Há outra maneira de delinear o código do lado do servidor que será abrangido daqui a pouco.) O exemplo a seguir uma página de servidor ativa chamada *ExempleChap2*.ASP, com o código de VBScript que será interpretado pelo ASP.DLL em negrito.

ExampleChamp2.ASP

```
<HTML>
<HEAD><TITLE>Example</TITLE></HEAD>
<BODY>
<%
• constrói uma string boas-vindas com uma saudação e a
• hora atual no servidor (recuperada a partir da função).
• Time( ) e depois exibe isso na HTML enviada para o
• cliente
 strGreetingMsg = "Hello. It is now" & Time ( ) &-"on the server."
Response. Write strGreetingMsg
%>
/BODY>
```

Quando um usuário solicita o ExempleChap2.ASP, o IIS puxa o arquivo do item de arquivos em sua memória. Reconhecendo a extensão. AXP a partir das configurações no

Mangement Console, ele utiliza ASP.DLL para ler e interpretar o arquivo. uma vez interpretado, o IIS envia o resultado final para o navegador do cliente solicitante.

O IIS trata todo o tráfego de http, O ASP.DLL somente interpreta código do lado do servidor, puxando a DLL do mecanismo de script apropria do quando necessário. Vamos assumir que a hora é 10:43: 43. O arquivo ASP anterior, uma vez interpretado, resultaria na próxima página HTTML dinamicamente criada que por sua vez será enviada para o cliente pelo IIS:

```
<HTML>
<HEAD><TITLE.EXEMPLE,?TITLE><HAED>
<BODY>
HELLO. It is now 10:42:43 on the server.
</BODY>
</HTML>
```

A chamada de método Response. Write é ama maneira de inserir código no fluxo de HTML que é enviado de volta para o cliente, mas há um atalho para essa chamada de método: delimitadores<%=...%>. Note a inclusão do sinal de igual(=). O sinal de igual é o que diferencia isso como uma chamada de atalho para o método Response. Write e não simplesmente mais código ASP para interpretar.

Os delimitadores <%=...%> oferecem alguns efeitos sutis que permitem que você produza algumas poderosas combinações de HTML no lado do servidor/cliente. eis o exemplo reescrito utilizado os delimitadores **<%=...%>**

```
<HTML>
<HEAD><TITLE.EXEMPLE,?TITLE><HAED>
<BODY>
HELLO. It is now 10:42:43 on the server.
</BODY>
</HTML>
```

Utilizar os delimitadores <%=...%> é o mesmo que utilizar o método Write do objeto Response. Simplesmente ínsita no fluxo qualquer coisa que esteja entre o tag de fechamento <%= e o tag de fechamento %>. Se o conteúdo entre os delimitadores representa uma variável, esse valor da variável é inserido no fluxo de HTML.

Com a utilização cuidadosa desses delimitadores, você pode construir dinamicamente não apenas o comentário de HTML, mas também o código script do lado do cliente, como demonstrado no próximo exemplo. O script é chamado *DynamicForm*.ASP e aceita um único parâmetro, *button-count*. Baseado no valor de *button-count DynamicForm.*ASP construirá dinamicamente entre um e dez botões Submete de HTML e também gera dinamicamente script para os eventos de onClick para cada um deles.

Exemplo DynamicForm.ASP

```
<HTML>
HEAD><TITLE.DynamicForm.asp, /TITLR></HEAD>
<BODY>
welcome to the dynamic form!
<%
“Recupera o número de botões que o usuário deseja criar”.
intCmdcount = Request.queystring(“button-count”)
“Assegura que o parâmetro enviado esteja dentro dos limites aceitáveis”.

If intCmdCount < 1 Then
  intCmdCount = 1
End if

If intCmdCount < 10 Then
  intCmdCount = 10
End if
 “Cria botoes.
for intCounter = 1to intCmdCount
%>
<INPUT TYPE= button VALUE = button<%=intcounter%>
Onclick = Button<%=intCounter%> click( )”>
<%Next
%>
<script Language = VBScript”>“.
<%
Cria os scripts para cada um dos botões  criados.
for inforCunter = 1 to intCmdCount
%>Sub Button<%=intCounter%>-click( )
MsgBox “you just cliked button <%=intCounter%>.”
End sub
```

```
Next
%>
 </SCRIPT>
</BODY>
</HTML>
Exemplo Codigo-fonte de HTML produzido pelo DynamicForm.ASP
<HTML>
HEAD><TITLE.DynamicForm.asp,/TITLR></HEAD>
<BODY>
welcome to the dynamic form!
<INPUT TYPE = button VALUE = button1
onclick = “Button1-click ( )”>
<INPUT TYPE = button VALUE = button2
onclick = “Button2-click ( )”>
<INPUT TYPE = button VALUE = button3
onclick = “Button3-click ( )”>
<SCRIPT LANGUAGE = “VBScript”>

Sub button1 Click( )
  MsgBox “Youjust Clicked Button 1 .”
End Sub

Sub button2 Click( )
  MsgBox “Youjust Clicked Button 2 .”
End Sub

Sub button3 Click( )
  MsgBox “Youjust Clicked Button3 .”
End Sub

</SCRIPT>
</BODY>
</HTML>
```

O parâmetro button count=3 se traduziu na construção de três elementos de botão de HTML e o código correspondente que os acompanha. Note os nomes e os nomes de procedure de evento onClick para cada um desses botões (em negrito no seguinte código):

```
<INPUT TYPE = button VALUE = button1
```

```
onclick = "Button1-click ( )">

<INPUT TYPE = button VALUE = button2
onclick = "Button2-click ( )">

<INPUT TYPE = button VALUE = button3
onclick = "Button3-click ( )"
```

Cada um desses nomes de elementos de botão e títulos de procedure de evento vieram da seguinte linha de código:

```
<INPUT TYPE = button VALUE = button<%=intCounter%>
onclick = "Button<%=intConter%>_click ( )"
```

Observe que o resultado de <%=intCounter%> é inserido no fluxo de texto de HTML. Utilizando ASP, fomos capazes de gerar nomes dinamicamente para cada um de nossos botões acrescentados o valor de uma variável contadora no final da palavra "Button"no fluxo de HTML.

Esse é um ponto sutil. um dos erros mais comuns no desenvolvimento do ASP é tratar o resultado de <%=...%> como um nome variável. Por exemplo, a próxima linha de código do lado do servidor não resulta na saudação"Oi Margaret", embora alguns desenvolvedores equivocadamente acreditem nisso:

```
<%
CODIGO INCORRETO
strUserName = "Margaret"
%>
MsgBox "Hello"& <%strUserName%>
```

Quando o precedente é enviado para o cliente., ele aparecerá assim:

```
MsgBox "Hello" & Margaret
```

A Linha de código correta para produzir o resultado desejado é a seguinte:

```
MsgBox  "Hello" & "<%strUserName%>"
```

O ponto aqui é que aquilo que esta entre os delimitadores<%+...%> entra n fluxo de HTML *como esta mesmo dentro de uma string.* Qualquer que seja o valor do contudo ele será inserido no de HTML. Não trate <%...%> como uma variável.

5.6 Linguagens de Criação de Scripts

Não se precisa utilizar uma única linguagem para i aplicativo do ASP inteiro. Não há nenhum problema com a mistura e combinação pó conveniência. em Geral , utiliza-se VBScript no código do lado do servidor e o JavaScript no lado do cliente, mas é obrigado a utilizar a especifica linguagem em qualquer situação. Entretando, pó forçar o ASPa padronizar um script especifico utilizando a diretiva @language de pré-processador do ASP. Saiba que pode utilizar a seguinte linha de código como a primeira linha em seu script-padrão ao interpretar seu código:

<%@ language = JScript%>

Se se colocar essa linha em qualquer lugar exceto a primeira linha, você recebera um erro , Alem disso note que o VBScript é o padrão para todos os scripts do lado do servidor. Entretanto, pode-se alterar isso mas na opção de aplicativo no diretório virtual do seu aplicativo de ASP.

5.7 Estendendo ao ASP

Estender os aplicativos Asp normalmente assume a forma de instanciar objetos do lado do servidor expõem métodos e propriedade que você pode acessar por meio do seu lado do servidor. a Microsoft inclui muitos outros desses componentes de servidor Active com o IIS 5.0. por exemplo, um dos componentes de servidor incluído no IIS é o componente Browser Capabilities. Uma vez instanciado, um objeto Browser Capabilities permite discerni detalhes sobre o navegado da Web do usuário: que script ele suporta, em que plataforma ele esta executando e assim por diante. Esse componente permite alterar seu site dinamicamente em resposta ä presença ou ausência de certos navegadores

Para criar um objeto MyInfo em sua Active Server Pages, você poderá utilizar um código semelhante ao seguinte:

```
<%
‘ Declara variáveis locais.
```

```
Dim objMyInfo

‘  Instancia um objeto MyInfo.
Set objMyInfo = Server.CreateObject(“MSWC.MyInfo”)

“Agora pode inicializar os valores”.
objMyInfo.PersonalName = ª Keyton Weissinger”
...[código adicional]
%>
```

Como se pode ver nesse exemplo, é simples instanciar esses componentes de servidor. Uma vez Instanciado, Você pode utilizar qualquer um dos métodos ou propriedade expostos do objeto para estender aplicativo para a Web.

Embora o IIS venha com vários componentes de servidor, você também pode escrever o seu próprio componente em qualquer linguagem de desenvolvimento que possa criar objetos COM, como o Visual Basic, O Visual C+ +, o Visual J + +, todos da Microsoft, ou o Delphi, da Omprise. Os detalhes de escrever componentes de servidor estão do escopo deste livro, portanto recomendo a leitura de *developing* ASP *components*, de Shelley Powers, publicado pela O’Reilly & Associates

Os componentes de servidor são descritos na tabela 5.1

Tabela 5.1 componentes de servidor

componente de sevidor	**Descrição**
ADO	adiciona acesso de banco de dados aspo aplicativo feitos em active Server Pages por meio de sua interface COM para provedores de dados de OLE DB , você é capaz de acessar qualquer origem de dados compatível com OLE DB ou ODBC
Ad Rotator component	Escolhe aleatoriamente um de uma variedade de anúncios para uma pagina da web baseada em um sistema de pesos.
Broser Capabilities	determina facilmente a funcionalidade suportada por seu navegador da Web do usuário.

Collaboration Data Objects for NT	Adiciona a funcionalidade de um sistema de mensagem para aplicativos Web. Estilizado os objetos que compõem o CDONTS, é possível Criar aplicativos robustos de gropware (grupo de trabalho) capacitados para trabalhar com correio eletrônico utilizando ASP.
Content Linking	Mantém uma lista vinculada de arquivos de conteúdo estatístico. A partir de dentro desses arquivos estatísticos, o componente Content Linking permite configurar a navegação fácil de usar de uma pagina para a próxima pagina (ou a anterior).
Content Rotator	Cria um arquivo de agenda contendo várias partes de HTML, que são posicionadas alternadamente em seu site da Web. Esse componente é semelhante ao componente Ad rotator mas funciona com conteúdo HTML simples em vez de funcionar com anúncios (imagens).
Counters	matem uma coleção de contadores, dentro do escopo de um aplicativo ASP inteiro, o que pode ser incrementado ou decrementado a partir de qualquer lugar em seu site da Web.
FileAccess componets	Permite acessar seu sistema de arquivo local e seu sistema de arquivos de rede. ë parte da biblioteca de tempo de execução de script é instalada e registrada por padrão quando você ainda instala o IIS.
Logging Utility Componet	Fornece acesso programático ä web do IIS ou log do servidor de FTP.
MyInfo	Mantém informações acessadas comumentemente, como o nome do webmaster, endereço, empresa etc., a partir de seus aplicativos Wes.
Page Couter	Cria um contador de qualquer pagina em qualquer pagina em seu site da Web. A contagem de pagina é salva regularmente em um arquivo de texto Isso permite manter informações de contagem de Páginas mesmo se o servidor da Web for reiniciado.
Permission Checker	Verifica as permissões em um determinado recurso na máquina loca ou na rede . isso permite determinar instantaneamente (*on the fly*) se o usuário atua tem ou não permissão para ver um arquivo.
Tools Components	Fornece uma variedade de funções de utilitários úteis.

Fonte: Weissinger, Keyton – 2000

5.8 O Objeto Application

No contesto de Active Server Pages, um aplicativo é a soma de todos os arquivos que podem ser acessados por um determinado diretório virtual e seus subdiretórios.Esse contexto de aplicativo de ASP é o mesmo para todos os clientes utilizando o aplicativo . Por exemplo, um cliente da Tailândia que solicita páginas de seu diretório virtual / *searchApp* está acessando o mesmo "aplicativo"que um segundo cliente da Suécia que esta solicitando paginas do mesmo diretório virtual independentemente de qual página da Web especifica dentro do diretório virtual cada um esta solicitando .

Do mesmo modo que os aplicativos independentes (standalone) permitem que você compartilhe informações por todo o aplicativo, os aplicativos ASP também permitem , você pode compartilhar informações entro todos os clientes de um determinado aplicativo do ASP utilizando o objeto *application* . Esse objeto predefinido representa o aplicativo do próprio ASP e é mesmo independentemente qual parte do aplicativo esse cliente esteja solicitando.

O objeto Application é inicializado pelo IIS no momento que primeiro cliente solista qualquer arquivo por diretório virtual dado. Ele permanece na memória de servidor até que o Serviço da Web seja parado ou aplicativo seja explicitamente descarregado do servidor da Web utilizando o Microsoft Management Console

O IIS permite instanciar variáveis e objetos com escopo de nível de aplicativo. isso significa que uma determinada variável contém o mesmo valor para todos os clientes de seu aplicativo. ele também pode instanciar objetos do lado de servidor com escopo de nível de aplicativo que da mesma forma contem os mesmo valores para todos os clientes.Essas variáveis de nível de aplicativo e objetos podem ser acessadas a partir do contexto de qual quer sessão do usuário e de qualquer arquivo dentro do aplicativo atual

5.9 O objeto ASPError

Introduzido com ASDP 3.0, o objeto ASPError predefinido permite visualizar informações detalhadas sobre o último erro ocorreu no script de uma página na ASP na sessão atual. Esse objeto, por meio de suas propriedade de leitura, fornece mais detalhadas sobre o tipo e a fonte de erro do que o objeto Err fornecido VBScript.

Para utilizar o objeto ASPError e sua propriedades, você deve chamar um novo método do objeto Server GetLastError, que retorna uma ASPerror Objects com seus valores de propriedades configurados para refletir sobre o último que ocorreu no script:

```
Dim objMyASPError
Set objMyASPError = Server.GelastError
```

Quando se instala o IIS5.0, por padrão, todo o pré-precessamento, script e erros de tempo de execução no IIS fazem, com que o código do ASP pare de processar o script atual e redirecione a execução de script para uma página de erro personalizada chamada 500-100.ASP. Esse redirecionamento ocorre por meio de uma interna para o método Server. Transfer que, alterna a execução de um script para outro enquanto protegendo todos as informações de estado no primeiro script.

O script 500-100.ASP (localizado por padrão no diretório C:\WINNT\help\iisHelp\common\) contém uma chamada ao método de Server.GetlastError. Quando esse script é executado, ele formata e exibe os valores de propriedade atuais do objeto ASPError retornado por esse chamada GetLasError.

Pode-se utilizar a página-padrão de erro , 500-100.ASP como ela é: edite-a para refletir a aparência e comportamento do seu site; ou utilize o snap-in de Internet Services Manager para redirecionar o IIS para uma página de sua escolha.

As propriedades, coleções métodos e eventos ASPError são delineados na próxima Tabela.

Tabela 5.2 Resumo do Objeto ASPError

Propriedades
ASPECode
ASPEDescription
Category
Column
Description
File
Line
Number
Source
Coleçoes
Nenhum
Metodos
Nenhum
Eventos
Nenhum

Fonte: Weissinger, Keyton – 2000

5.10 O objectContext

Como para a versão 2.0, um recurso importante das Active Server Pages é a capacidade de criar um script transacional: um script cujos segmentos de código constituintes são completamente bem-sucedidas ou falham como um todo. por exemplo, utilizando esse script, uma seção de código poderia remover um registro de uma tabela estoque e uma segunda se não poderia adicionar um registro a uma tabela de log de Vensad. Entretanto, somente se as seções de código forem inventario ou a adição de vendas falhar, o script falhará. Os dois processos são revertidos (*rollec back)*: o registro excluído, se foi removido, é adicionado de volta no banco de dados e o registro de vendas, se for adicionado, é removido a tabela de log de vendas. Essa capacidade de empacotar várias funções em uma unidade e transacional que é bem-sucedida ou falha como um todo é um aprimoramento significativo na capacidade dos aplicativos AS. Anteriormente, todas as transações contavam com suporte de transação de banco de dados.

As transações do aplicativo ASP são controladas pelo Windows 2000 COM= Component Services ou pelo Microsoft Transaction Server do Windows NT (MTS). Essa parte do conjunto BackOffice permite controle sobre todas a ações de banco de dados codificadas para utilizá-lo. O suporte para scripts configuração especial. Sem o COM+ component Sevices ou, ASP2.0, suporte transacional MST, seus aplicativos teriam de monitorar todas as alterações de banco de dados manualmente e reverter todas as ações de banco de dados manualmente, monitorando questões sde múltiplos usuários , concorrências etc. o MTS ou o COM + ou o component Services esse suporte ao seu aplicativo com muito pouca codificação extra-contanto que o banco de dados esteja conectado ao Microsoft SQL Server ou suporte o protocolo XA do consórcioX/Open. Observe que isso significa que as ações de arquivo ainda não são suportadas ou, pelo menos , não automaticamente.

O suporte das transações do ASPE codificado pelo uso do objeto ObjectContext, que representa o objeto ObjectContext real do próprio COM+ Componet Services. chamado métodos do objetos ObjectContext e codificado seus eventos, você pode criar um script transacional com apenas algumas linhas de códigos a mais.

Para declarar todo o script em uma determinada pagina como sendo transacional simplesmente adicione a seguinte linha de código como a primeira linha em seu script:

<%@ TRANSACTION = Requires %>

É importante somente que essa linha seja a primeira em seu script: a inclusão dessa linha alerta o servidor da Web para utilizar Component Services a fim de assegurar que o script seja bem sucedido ou falhe como um todo.

Para salvar (commit) a transação ou abordá-la, você simplesmente chama os métodos SetComplete ou SetAbort ou o objeto ObjectContext, respectivamente, Se você estiver lidando com uma transação complexa contendo segmentos de código que não são suportados pelos Component se você (notavelmente ações de arquivo), você pode codificar especialmente essas ações nos eventos ObjectContext On TranssactionCommit e On TransactionAbort.

Tabela 5.3. Resumo do Objeto ObjectContext

Resumo do objeto objectContext
Propriedades
nenhuma
C*oleções*
nenhuma
Métodos
setAbort
SetComplete
Eventos
OnTrasactionAbort
OnTrasactionCommit

Fonte: Weissinger, Keyton – 2000

5.11 O objeto Request

O objeto Request fornece acesso ao cabeçalho e corpo de solicitação de http do usuário. Ele é argumentavelmente o abjeto predefinido mais importante do ASP a entender, visto que é através desse objeto que voce será capaz de reagir ás decisões feitas pelo usuário. Utilizando o objetivo Request, você dinamicamente cria páginas da Web e realiza ações do lado do serviço mais significativas (como atualizar um bando de dado) baseadas na entrada do usuário.

5.11.1 Como o HTTP Funciona

É importante entender os princípios básicos do protocolo HTTP. Com essa introdução, a utilização do objeto Request é traduzida de uma forma fácil de compreender.

5.11.2 HTTP: Um Exemplo Simples

O HTTP é um protocolo de estilo "transação". O navegador (o cliente) envia uma solicitação para o servidor. O servidor obedece à solicitação se ele puder e envia uma resposta de volta para o cliente. O servidor então esquece completamente a transação. O navegador pode ou não esquecê-la.

1. quando o usuário termina inserido o URL PARA HELLO. HTM, o Navigator envia (send) o seguinte fluxo para o servidor.

```
[73: send: (179)] GET/ HELLO.HTM/1.0
conection; keep-alive
UserAgent: Mozila/3.0 (wun95; I)
host: pc229. wet.ora.com
accept:image/gif/gif, image/xbitmap, image/jpeg,
    image/pjpeg, */*
```

Esse é um cabeçalho de solicitação , O navegador indica que quer que o servidor obtenha o documento ?HELLO. HTM. Get é mais que uma descrição genética do que o servidor deve fazer; ele indica o tipo de solicitação de http O navegador tambem indica que esta utilizando a versão 1.0 hypertxt Transfer protocol.

2. o servidor recebe os cabeçalhos enviados pelo navegador, como mostra os na seguinte produzida por nosso programa espião e processa a solicitação;

```
[21: recv: completed (179)Get /hello.htm http/1.0
Conection: Keep-alive
User-agent Mozilla/3.0 (win95; I)
Host: pc299.west.ora.com
accept: image/gif, image/x-xbitmap, iamge/jpeg, iamge/pjpeg, */*
```

3. o servidor envia o documento HOLLO.HTM para o navegador

```
[21:send☹535)http/1.0 200PK
Date :MONDAY, 30 SEP-98 23:3300gmt
Server: website/1.1
Allow-ranges:bytes
```

```
accept-ranges:bytes
connection:keep-Alive
Content-type: text/html
Las-modified: Monday, 30-sep98 23:30:38 GMT
Content-length:297

<HTML>
<HEAD><TITLE.Hello,  Word!</TITLE></HEAD>
<BODY>
<FORM ACTION=”/cgi-wn/hello.exe”METHOD=”POST”>
What is your name?<IMPUT TYPE=”text”NAME” SIZE=60><BR>
<IMPUT TUPE=”submet” VALUE=”submit the fotm>
<IMPUT TYPE=”reser”VALUE=clar all fields
</Fform>
</body></html>
```

Aqui, o site da Web envia um total de 535 bytes para o navegador. Isso consiste em um cação de resposta, seguido por uma linha em branco, seguida pelo próprio documento de HTML. Os campos de cabeçalho indicam, entre outras coisas, o numero de bytes o cabeçalho content-(length) e o formato (o cabeçalho content-type) dos dados transmitidos.”200ÖK” é um código de status indicado que a solicitação do navegador foi cumprida. O servidor também indica que, como o navegador, esta utilizando a versão 1.0 de HTTP.

4. O navegador lê os cabeçalhos e os dados enviado pelo servidor:

```
[23; recv: posted]
[73recv:completed (260) http/1.0 200OK
Date :MONDAY, 30 SEP-98 23:3300gmt
Server: website/1.1
Allow-ranges:bytes
accept-ranges:bytes
connection:keep-Alive
Content-type: text/html
Las-modified: Monday, 30-sep98 23:30:38 GMT
Content-length:297
<HTML>
<HEAD><TITLE.H
[73:recv:poted]
[73:recv: completed (275)]ello, World!<TITLE></HEAD>
<BODY>
<FORM ACTION=”/cgi-wn/hello.exe”METHOD=”POST”>
What is your name?<IMPUT TYPE=”text”NAME” SIZE=60><BR>
<IMPUT TUPE=”submet” VALUE=”submit the fotm>
<IMPUT TYPE=”reser”VALUE=clar all fields
</Fform>
</body></html>
```

Embora as duas operações *recv* sejam requeridas para recuperar os registros de cabeçalho junto com o documento, o numero total de bytes lidos nessas duas operações é igual ao numero total de bytes envidados pelo servidor.

5. o navegador exibe o formulário pedindo o nome de usuário e, quando o usuário o preenche e clica no botão submit, envia o seguinte para o servidor.

```
[70:send☹232)]POST /cgi-win/hello.exe http/1.0
Referer; http://pc229.west.ora.com/hello.htm
Content-type: text/html
User-Agent: Mozilla/3.0 (win95; I)
Host: pc229.west.ora.com
Accept: image/gif, iamge/x-xbitmap, iamge'jpeg, iamge/pjpeg, */ *
[70;send☹69)]Content-type: application/x-www-Form-urLencoded
Content-Length: 14
[70:send:2)]
[70:send:16)]name=jayne+Doe
```

Como o navegador está transmitindo dados de formulários, o tipo de solicitação de HTTP é "POST", como o primeiro registro de cabeçalho indica de maneira semelhante, os registros de content-leght e content-type indicam que o navegador esta transmitindo 14 bytes de dados de x-www-form-urlencoded no corpo da solicitação . isso consiste na entrada de informações pelo usuário no único campo de dados de formulários, a caixa de texto name.

6. o servidor recebe registros de cabeçalho e só dados de formulários transmitidos elo navegador no passo anterior . URL (*cgi-win/hello.exe)* faz com que o servidor carregue o aplicativo CGI HELLO.EXE e transmita os dados do formulário para ele. O aplicativo de CDI pode fazer algum processamento de back-end, e então constrói um documento de HTML instantaneamente (on the fly) e retorna para o servidor.

7. o servidor retorna o documento de HTML para o navegador junto com os registros de cabeçalho necessário, como a seguinte saída a partir do Wsck32 Spv mostra

```
[18:send)422)http/1.0 200OK
Date :MONDAY, 30 SEP-98 23:33:10gmt
Server: website/1.1
Allow-ranges:bytes
accept-ranges:bytes
connection:keep-Alive
Content-type: text/html
Content-length:231

<HTML><HEAD>
<TITLE.WLCOME TO THIS Wb PAGES!,/TITLE><HEAD.

<BODY><H1.Welcome to our Web Server!h1><p><p.
HELLO, jayne Doe! We' re glad thar you took
the time ou of your busu day to visit us!
<HR><PRE></BODY><?HTML>
```

Observe que o servidor indica para o navegador que esta enviada 231 bytes de um documento HTMS.

8. o navegador recebe o fluxo de dado enviado pelo servidor e utiliza para gerar a página HTML esperamos que isso forneça uma idéia relativamente boa do que está envolvido na troca entre navegador e servidor. Mas é importante examinar mais profundamente alguns pontos que só abordamos superficialmente, bem como abranger alguns recursos adicionais que não são incluídos neste exemplo simples.

5.12 Tipo de Solicitação de HTTP

O tipo de solicitação pelo cliente para servidor a fim de indicar o que o servidor deve fazer com o URL que também pelo navegador.

Embora a especificação de HTTP detalhe um numero de tipos de solicitação como PUT e DELETE, somente dois são suportados por todos os servidores e são comumente utilizas; GET e POST. A solicitação GET solicita para "obter"uma parte das informações em geral um documento, e retorná-lo para o cliente. Se a solicitação incluir quaisquer informações adicionais, estas são acrescentadas como argumento são URL. Uma solicitação POST por outro lado, fornece informações ao servidor para serem "postadas"para o URL: em geral essas informações são utilizada para enviar o conteúdo de um formulário de HTML para o servidor ou fornecer informações ao servido que são necessárias ara o processamento de Back-end. As informações em si estão contidas no copo de solicitação.

A maioria dos servidores não pode tratar dados recebido dos métodos POST ou GET internamente. Normalmente, as solicitações POST, bem como as solicitações GET que também enviam dados para o servidor , são tratadas por programas acessórios ou DLLs (aplicativos CGI e ISAPI e filtros de ISAPI). Ambas as solicitações POST e GET podem retornar qualquer tipo d dados de qualquer tamanho.

Enquanto possa parecer para um servidor da Web que GET e POST são semelhantes, ao transmitir dado uma regra é muito rígida: *uma solução get nunca deve mudar qualquer coisa*. Não escreva um script do ASP que faz alterações para um banco de dados, por exemplo, em resposta a uma solicitação GET.

5.13 Envio de Formulário

Um usuário insere entrada nos campo de um formulário. Quando o formulário é enviado , os dados contidos em cada campo do formulário são transferido para o servidor, que então os passa para ASP. Esses dados são enviado nos formato nome valor onde nome valor é o nome atribuído ao campo pelo atributo name= do tag <INPUT>, e o valor inserido nesse primeiro exemplo, se o usuário insere"Arcchie"em um campo solicitando seu primeiro nome o navegador enviar a string first-name=Archie

Se o formulário é escrito para utilizar o METHOD=GET , os dados do formulário são acrescentados ao URL como uma string de argumentos. Se o formulário contiver muito campos ou se os campos contiverem longas strings de texto, o URL completo pode tornar-se muito grande e difícil de lidar. Alem disso o limite do numero de caracteres enviado em um GET - em geral aproximadamente 2000 – e muito mais baixo que um POST

Se o formulário utiliza o METHOD<POST, os pares name=valor são como o corpo da solicitação enviada como o corpo da solicitação em vez de serem acrescentados ao URL. Alem da maior facilidade de tratamento ao extrair do corpo uma solicitação que a partir de uma URL no cabeçalho da solicitação.

Sempre utilize o método post com formulários que alterem algo ou causam qualquer ação irrevogável (a maioria faz isso). POST é seguro e mais eficiente: Get nunca deve ser utilizado para alterar nada. No desenvolvimento de seu scripts do ASP, você pode decidir se que r ou não suportar dados passados para seu programa utilizando o método GET.

5.14 Solicitação e Resposta de HTTP

Os cabeçalhos são parte do HTTP mais mal compreendida, muito embora entende seu papel possa tornar o entendimento das propriedade e métodos dos objetos ASP Request e Reponser muito mais fácil.

De uma olhada em qualquer mensagem de correio eletrônico de Internet ela consiste em duas partes, o cabeçalho e o corpo. o cabeçalho consiste em varias linha que descrevem o corpo da mensagem e talvez a maneira como a mensagem foi tratada e rodeada pra voce. O cabeçalho e o corpo separados por uma linha em branco . Para mais informação sobre a cinese de cabeçalho , consisti RFC-822.)

Uma mensagem de HTTP (ou uma solicitação ou uma resposta)é estruturada da mesma maneira. A primeira linha é especial, mas o restante das linha até a primeira em

branco são os cabeçalhos exatamente como uma mensagem correio. o cabeçalho descreve a solicitação e seu conteúdo, se houver algum, ou resposta a seu conteúdo.

5.14.1 A solicitação de HTTP e o Request do ASP

O objeto Request do ASP permite acessar o cabeçalho e o corpo da solicitação de HTTP enviada para o servidor da Web cabeçalho e o corpo da solicitação do cliente. o método de recuperar informação da solicitação de HTTP é basicamente o mesmo para um script em ASP e para um aplicativo em CGI. As exceções não vem dos mecanismos reais s de solicitação, mas da maneira como cada tipo d aplicativo é carregado no servidor da Web (CGI versus um filtro de ISAPI)

Assim como com aplicativo CGI, o navegador do cliente pode enviar informações para um script do ASP de duas maneiras diferentes. Primeiro, ele pode enviar informações por meio de um formulário de HTML utilizando método *GET*.

```
<HTML>
<HEAD><TITLER>Welcometo Corp, </TITLE></HEAD>
<BODY>
,FORM ACTION=”http://mycop.com/secure.asp”  METHOD=”GET”>
First name: <IMPUT type=”text ” NAME=”first name” SIZE=60><BR>
Last name: <IMPUT TYPE=”text”NAME=”” last_name”SIZE=60><BR>
<INPUT TYPE=”submitT” VALUE=”submit the form”>
<INPU TYPE=”resetT” VALUE= “Clear all fields”>
</FORM>
</BODY> </HTML>
```

Quando o cliente envia uma solicitação GET, as informações sobre a solicitação são acrescentadas ao final da solicitação URL como pares e nome/valor separados pelo e comercial e precedidos por um ponto de interrogação. Cada nome corresponde a um elemento no formulário. Pó exemplo suponha que o usuário inseriu Horatia e Thompson nos dois campo do ultimo exemplo e clicou no botão Submit. o envio de formulário precedente é, no que diz respeito ao servidor , idêntico ao seguinte:

http://mycop.com/secure.asp?first_name=horatia&last_name_thopson

Isso é um ponto importante. Seguindo esse exemplo, considere a seguinte linha código:

http://mycop.com/secure.asp?first_name=horatia&last_name_thopson

Se o usuário fosse digitar isso na linha de endereço ou clicar em um link contendo o anterior como um URL, o servidor da Web trataria essa solicitação de HTTP resultante exatamente como se as informações tivessem sido enviadas como parte de um formulário utilizado a solicitação GET. Por dentro do seu aplicativo do ASP, é possível acessar essas informações pela coleção QueryString do objeto Request. Por exemplo,

```
<%
strFirstName = Request.Querysting(“first_name”)
%>
```

Inicializada a variável strFirstName para o valor no parâmetro strFirstName.

Assim como com aplicativo CGI, você também pode enviar informações para um script do ASP utilizando o método PROST. nesse caso em vez de ser do cabeçalho de solicitação de HTTP, as informações estariam no corpo do objeto Request:

```
<HTML>
<HEAD><TITLER>Welcometo Corp, </TITLE></HEAD>
<BODY>
,FORM ACTION=”http://mycop.com/secure.asp” METHOD=”GET”>
First name: <IMPUT type=”text ” NAME=”first name” SIZE=60><BR>
Last name: <IMPUT TYPE=”text”NAME=”” last_name”SIZE=60><BR>
<INPUT TYPE=”submitT” VALUE=”submit the form”>
<INPU TYPE=”resetT” VALUE= “Clear all fields”>
</FORM>
</BODY> </HTML>
```

O envio desse formulário resultaria em Uma solicitação de HTTP semelhante a seguinte:

```
POST / Asecure. asp http/1.0
Accept: iamge/gif, iamge/jpeg,*/ *
user-Agent: Mozilla/2.ON(Windows; I ; 32bit)
Content-type:application/x-wwwform-urlenconded
Content-legth:35
[mandatory blank line]
first_nome=horatio&last_name=aubrey
```

Para seu aplicativo manipular as informações enviadas nessa solicitação de HTTP, você teria de utilizar a coleção Form do objeto Request:

```
<%
strFirstName = .form(“first_name”)
```

```
%>
```

Isso inicializará variável strFirstName para o valor enviado parâmetro first_name.

5.14.2 O Objeto Request do ASP

As propriedades, coleções, métodos e eventos do Objeto Request do ASP são mostradas na tabela a seguir:

Tabela 5.4 Resumo do Objeto Request

Propriedade
TodalByetes
Coleções
ClienteCertificante
Cookies
Form
QueyVariables
Métodos
Binary Read
Eventos
Nenhum

Fonte: Weissinger, Keyton – 2000

5.15 O Objeto Response

Assim como o objeto Request permite recuperar e manipular informações enviadas pelo navegador do cliente na solicitação de HTTP, o objeto Response fornece muito controle sobre a resposta de HTTP ao cliente. Esse controle vem três categorias amplas:

- Controle sobre que dados e tipos de dados são enviado para o cliente no cabeçalhos de resposta de HTTP
- controle sobre que dados e tipo de dados enviados para o cliente no corpo da resposta de HTTP
- Controle sobre quando e como esses dados são enviados

O controle sobre os cabeçalhos Response de HTTP inclui a configuração de cookies na maquina do cliente, configuração de vários valores de expiração de cabeçalho de HTTP

(como o tipo de conteúdo e informações de expiração para uma determinada pagina) e, por fim , a edição de seus próprios cabeçalhos personalizados á reposta de HTTP.

Você controla o corpo de resposta de HTTP diretamente pelos métodos Write e binary Wryte. como você talvez infira a partir dos nomes , esses métodos do objeto Response permitem escrever informações diretamente no corpo da resposta que será recebida pelo cliente exatamente como quaisquer outras informações recebidas em uma resposta de solicitação de HTML.

POR FIM O OBJETO Response permite controlar como e quando a reposta é enviada para o cliente. Por exemplo, utilizando as propriedades e método envolvido no armazenamento em buffer da resposta, você pode determinar se envia a resposta de HTTP como uma única unidade para o cliente ou envia os resultados da solicitação em partes Você pode determinar dinamicamente se o cliente ainda está ou não conectado a seu site d a Web. Pode-se redirecionar a solicitação dela como se solicitasse algo mais. Por fim, pode-se utilizar o objeto Response para escrever as entradas no log de servidor d a Web

Tabela 5.5 Resumo do Objeto Response

Propriedades
Buffer
CacheControl
Charset
ContentType
Experies Experes
EspresAbsolute
IsClientConnnect
PICS
Status
Coleções
Cookies
Métodos
Addheader
AppendToLog
BinaryWrite
Clear
End
Flush
Redirect
Write
Eventos
nenhum

Fonte: Weissinger, Keyton – 2000

5.16 O Objeto Server

O objeto Server fornece várias funções que você pode utilizar em seus aplicativos ASP. Embora a maioria de seus métodos seja esotérico e raramente utilizado, três métodos , CreateObject, Execute e Transfer, e a única propriedade do objeto Server, ScriptTimeout, são inestimáveis.

O objeto Server, como seu nome implica, representa o próprio servidor da Web e grande funcionalidade que ele fornece é simplesmente a funcionalidade que o próprio servidor da Web utiliza no processamento normal solicitações de cliente e respostas de servidor.

Tabela 5.6 Resumo do Objeto Server

Propriedades
ScriptTimeout
Coleções
Nenhuma
Métodos
CreateObject
Execute
GetLastError
MapPath
Trasfer
HTMLEncode
URLEncode
Eventos
Nenhum

Fonte: Weissinger, Keyton – 2000

5.17 O Objeto Session

Um dos maiores desafios com que você se depara ao construir um aplicativo completo com todos os recursos para a Web é monitorar informações especificas do usuário enquanto um usuário navega pelo sei site sem solicitar para identificar-se em cada solicitação ao servidor. Entre outras informações que você precisa manter, estão a identificação do usuário, o nível de acesso do usuário dentro do esquema de segurança, se aplicável, e, em aplicativos preferenciais do usuário que permitem personalizar a aparência e o comportamento de seu site Web em resposta a seleções feita pelo usuário. o problema principal com a manutenção especifica do usuário limitações no protocolo de HTTP 1.0

Embora o http 1.0 forneça um mecanismo para conexões persistentes que permitem manter a identificação do usuário e doados específicos do usuário sua utilidade e limita. Sem entrar nos detalhes técnico, o Hypertext Transfer protocolo 1.0 permite que os navegadores clientes enviem mensagem Keep-Alive(“mantenha ativo”) para servidores Proxy. Essas mensagens basicamente informam o servidor Proxy para manter uma conexão aberta com o cliente solicitante. Entretanto, essas solicitações freqüentemente não são re conhecidas pelo servidor Proxy. Esse problema no servidor Proxy resulta em uma conexão”pendurada”entre o servidor Proxy e o servidor da entre o servidor Proxy e o servidor da Web solicitando . Em resumo, a manutenção de conexões com servidores da Web é propensa a erro, portanto é incerta no HTTP 1.0, que ainda é mais comumente utilizado pelos navegadores clientes.

O objeto Session representa a sessão do usuário atual no servidor da Web. O objeto Session é o usuário especifico, e suas propriedades e métodos permitem manipular informações no servidor que são especificas desse usuário até o fim da conexão desse usuário. Essa duração é definida como o período da primeira solicitação do cliente de uma pagina dentro de seu aplicativo para a Web até 20 minutos (20 minutos é um valor padrão que pode ser mudado) depois da ultima solicitação do usuário ao servidor da Web.

Uma sessão de usuário pode ser iniciada de uma de três maneiras:

- Um usuário ainda não conectado ao servidor solicita uma Active Server Page que reside em um aplicativo contendo um arquivo GLOBAL.ASA com condigo para o evento Session_OnStart.
- Um usuário solicita um Active Server Page cujo script armazena informações de qualquer variável de escopo de sessão.
- Um usuário solicita uma Active Server Page em um aplicativo cujo arquivo GLOBAL.ASA instancia um objeto utilizando o tag <OBJECT> com o parâmetro SCOPE configurado como Session.

Uma sessão de usuário é especifica quando a um determinado aplicativo em seu site da Web. De fato é possível manter informações de sessão para mais de um aplicativo por vez se um aplicativo estiver enraizando em um diretório virtual que reside sob o diretório virtual que designa outro aplicativo.

O servidor da web identifica cada usuário com um único valor de SessionID. Essa variável SessionID é atribuída á cada usuário com um ínicio de sua sessão no servidor na Web

e é armazenada na mesma do memória do servidor da web. O SessionID é armazenado no cliente gravado um cookie contendo o SessionID da máquina do usuário. esse cookie é enviado para o servidor toda vez o usuário faz uma solicitação. Para identificar o usuário, o servidor recupera o cookie e o compara com um SessionID armazenado na memória

Alem da variável SessionID, é possível armazenar outra informações especificas a usuários individuais. Você pode inicializar (ou alterar) qualquer variável de nível de sessão em qualquer lugar, em qualquer script Active Server Page. para assegurar que uma variável de nível de sessão seja inicializada para um valor especifico, você pode codificar um script nas procedure de evento Session_onStart do arquivo GOLOBAL.ASA. essa procedure de evento é desencadeada quando a sessão de usuário inicia , O arquivo GLOBAL.ASA (consulte o Capítulo11) é um arquivo especial que você pode codificar de forma especifica para cada aplicativo ASP. esse código de arquivo é processado quando ao sessão de usuários inicia.

Como discutido anteriormente o objeto Session é muito importante na manutenção de informações sobre usuários individuais. Você também pode utilizar o objeto Session para tratar algumas questões especiais que são especificas de clientes que não falam a língua inglesa solicitando informações do seu site da Web.

Tabela 5.7 Resumo do Objeto Session

Propriedades
CodePage
LCID
SssionID
Timeout
Coleções
Contents
StaticObjects
Métodos
Abandon
ContensRemove
ContentsRemove AII
Eventos
Session_OnEnd
Session_OnStar

Fonte: Weissinger, Keyton – 2000

6. O MUNDO DA WEB DINÂMICA

A WEB de hoje é muito distinta da WEB de poucos anos atrás, quando somente tínhamos acesso à página estáticas que sempre mostravam a mesma "cara" para todos os internautas. Bem, isto mudou para melhor, posso garantir, pois hoje em dia você pode acessar sites na WEB que fazem de tudo um ouço, desde pesquisa em outros sites até compra de ações, carros, barcos, casas, etc. enfim, tudo que o mundo real pode ter, provavelmente, você encontrará na WEB.

Mas, para que tudo isto aconteça, é necessário que milhares de pessoas trabalhem nos bastidores do mundo virtual, para prover os sites de vida própria. São essas pessoas que transformam páginas estáticas sem graça em verdadeiras fontes de informação e negócios. E quais são as ferramentas que ajudam essas pessoas a atingirem seus resultados? São as ferramentas de programação para Internet, programas que fazem com que o HTML, ser duro e inflexível, transforma-se em algo mutante e totalmente dinâmico. Podemos utilizar as mais variadas ferramentas para isso: Java, ASP, Cold Fusion, Perl, DGI, Javascript e PHP. E é sobre isto que vamos falar neste livro; como alcançar resultados melhores utilizando HTML, PHP, JavaScript e Banco de dados para WEB.

6.1 Uma Breve História Do PHP

Em 1994, um grande homem de negócios chamado Rasmus Lerdorf desenvolveu uma série de ferramentas que são utilizadas como mecanismos colaboradores para interpretar alguns macros. Elas não eram extravagantes: um livro de visitas, um contador e alguns outros elementos de "home page" que eram interessantes quando a Web estava na sua infância (FI – Form Interpretation) que escreveu, adicionou alguns suportes a Banco de Dados e lançou o que ficou conhecido como PHP/FI.

Então, no espírito de desenvolvimento de software com código abertos, desenvolvedores de todo o mundo iniciaram suas constibuições ao PHP/FI. Em 1997, mais de 50.000 sites da Web estavam utilizando o PHP/FI para desempenhar diversas tarefas – conexão a um bando de dados, exibição de resultados dinâmicos, entre outras.

Naquele ponto, o processo de desenvolvimento com certeza começou a tornar-se um esforço de equipe. Com a assistência fundamental dos desenvolvedores Zeev Suraski e Andi

Gutmans, a versão 3.0 foi então criada. O lançamento da versão final do PHP3 ocorreu em Junho de 1998, quando foi atualizada para incluir suporte para múltiplas plataformas (o PHP não é mais destinado apenas ao Linux!) e servidores da Web, diversos bancos de dados e SNMP (Simple Netword Management Protocol) e IMAP (Internet Message Access Protocol).

PHP é uma combinação de linguagem de programação e servidor de aplicações. Você pode programar em PHP como em qualquer outra linguagem, definindo variáveis, criando funções, realizando loops, enfim tudo que é necessário no mundo da programação. Mas o que realmente difere PHP das outras linguagens de programação é a sua capacidade de interagir com o mundo WEB, transformando páginas estatísticas em verdadeiras fontes de informação.

O PHP foi criado originalmente por Rasmus Lerdorf. Escrito em Perl, o PHP foi utilizado a princípio para atender às necessidades pessoais de Rasmus, sendo reescrito depois em C, quando foram incluídas funções para acesso a bancos de dados. Com a crescente solicitação de cópias do PHP, Ramos providenciou uma documentação para a ferramenta e disponibilizou o PHP v 1.0. Com isto mais e mais pessoas em todo o mundo começaram a utilizar o PHP, clamando cada vez mais por novas implementações, sendo as maiores colaboradores Zeeev Suraski e Andy Gutmans, para tornar o PHP a ferramenta que temos hoje (assim nasceu o PHP v 3.0).

Atualmente, o PHP está sendo utilizado por aproximadamente 1 milhão de sites no mundo inteiro, e o trabalho atual da equipe de desenvolvimento é disponibilizar a versão 4 do PHP que utiliza a poderosa tecnologia Zenda, a qual tornará o PHP extremamente potente, permitindo que ele rode de maneira mais eficiente em Servidores WEB, que não o Apache.

6.1.1 Quais São as Opções?

Existe uma série de opções de pré-processadores hipertexto no mundo da WEB. Os principais existentes hoje são: PHP, ASP, ColdFusion e Java, todos têm suas vantagens e desvantagens. Uns rodam apenas em um ambiente, outros dependem de tecnologia proprietária para rodar e outros são pagos. Eu gosto do PHP porque, além de ser gratuito, rodar em diversas plataformas, ter documentação ampla e abrangente na rede e ser muito robusto, é ainda muito fácil de aprender, possuir acesso direto a uma gama muito grande de bancos de dados relacionais (Oracle, Sybase, Postgres, mSQL e tantos outros), ainda suporta acesso via ODBC aos bancos de dados que não possuem acesso direto. O PHP também está

em contínuo desenvolvimento, agregado novas funções, corrigindo os erros reportados, enfim PHP torna-se a cada versão um software cada vez mais robusto.

O PHP armazena as informações enviadas em arrays, sendo um para os dados enviados pelo método GET e outros para o método POST. Os arrays têm os seguintes nomes: $HTTP_GET_VARS e $HTTP_POST_VARS, sendo que os campos são armazenados na seqüência em que se encontram no formulário, tendo os nomes dos campos como chaves associativas. Uma exceção ocorre com os campos do tipo file. Nesses casos, o PHP armazena algumas informações no array $HTTP_POST_VARS (nome do arquivo do usuário, tamanho do arquivo,....), sendo o conteúdo (os dados enviados pelo usuário) armazenado em um subdiretório temporário (geralmente /tmp). O programa recebedor dos dados deve então tratar esse arquivo (copiá-lo uma área definitiva, por exemplo).

Além dos arrays $HTTP_GET _VARS e $HTTP_POST_VARS, podemos ainda referenciar as variáveis pelo nome dado no formulário, ou seja, podemos referenciar o conteúdo do campo nome, por exemplo, por meio de $HTTP_POST_VARS [“nome”], ou diretamente por $nome. A segunda opção, deixa os programas mais limpos e fáceis de manter, e é necessária apenas uma referência ao nome do campo criado no formulário. Mas esta opção tem uma desvantagem: se por descuido referenciarmos uma variável com o mesmo nome do campo, inserindo um valor qualquer nesta variável, o conteúdo enviado pelo formulário será perdido, ou seja, devemos ser extremamente cautelosos quanto formos utilizar a recuperação direta de um campo de formulário.

6.2 Autenticação de Usuários de Servidores WEB

Autenticação de usuários é um recurso provido pelo Apache e outros servidores WEB para limitar o acesso a determinadas áreas somente a usuários autorizados. O conceito inicial de autenticação diz respeito à proteção de diretórios e subdiretórios, nos quais deve existir um arquivo de configuração de acesso (geralmente **.**htaccess) que controla quem pode acessar determinado diretório e quais são seus privilégios, sendo que este procedimento utiliza um arquivo de usuários (geralmente armazenado no diretório raiz do site, cujo nome você pode definir), que define os usuários e suas senhas. O arquivo.htaccess tem as seguintes aparência:

.htaccess

AuthType basic

```
AuthName        IonPlus Design
AuthUserFile    /home/ionplus/users

<Limit GET POST>
require valide-user
</Limit>
```

Este arquivo contém basicametne parâmetros de autenticação (AuthType, AuthName e AuthUserFile) e as regras de validação para acesso ao diretório (require valide-user). Os parâmetros de autenticação tem as seguintes funções:

AuthType Define o tipo de autenticação que o servidor irá realizar. A maioria dos servidores reconhece apenas o tipo Basic para nomes de usuários e senhas de acesso.

AuthName Especifica o nome do campo de autenticação do diretório. Este nome é utilizado pelo navegador (browser) para armazenar nomes e senhas de usuários (caso o usuário deseje). Ele também é mostrado na tela de autenticação do navegador.

AuthUserFile Define o caminho e nome do arquivo de senhas. O caminho deve ser absoluto, ou seja, você não pode especificá-lo em relação a seu site (algo do tipo public_html/senhas não irá funcionar). Você precisa especificar o nome completo (/home/ionplus/public_html/senhas, por exemplo).

No bloco LIMIT, podemos inserir as seguintes diretivas:

Order Determina a ordem de avaliação de usuários com acesso permitido e bloqueado. Podemos utilizar as especificações allow (autorizado), deny (bloqueado) e mutualfailure. As combinações de especificações aceitas são:

Allow, deny Verifica primeiramente se o host é utilizado, sendo que o usuário estiver presente nas duas especificações (autorizado e bloqueado), ele será recusado.

	Deny, allow	Verifica primeiro se o host está bloqueado. Usuários presentes nas duas especificações são liberados.
	Mutual-failure	Separa as duas listas, ou seja, hosts na lista de autorizados são permitidos, os presentes na lista de bloqueados e todos os outros são recusados.
Allow from	Especifica quais hosts têm permissão de acesso. Podemos utilizar nomes de domínios, nomes de hosts, endereços IP dos hosts, endereços parciais (indicando subnets) ou a palavra "all" a qual libera todos os usuários.	
Deny from	Especifica quais hosts têm o acesso negado. A Forma de especificação é a mesma utilizada em Allow.	
Require	determina a forma como serão liberados os acessos. As opções válidas são:	
	User	Somente os usuários especificados têm acesso autorizado. Esses usuários devem estar presentes no arquivo de usuários válidos (parâmetro AuthUserFile).
	Group	Libera o acesso aos usuários de um dos grupos especificados.
	Valid-user	Qualquer usuário válido tem acesso.

Se quisermos autorizar, por exemplo, o acesso a um dos diretórios do site (ou ao site todo se for o caso) aos usuários walace e mara, teríamos:

```
<Limit GET POST>
require user walace mara
</Limit>
```

Se for necessário um controle mais rigoroso, em que somente será permitido o acesso aos usuários walece e mara, porém de uma sub-rede específica (por exemplo, 200.210.107), teríamos:

```
<Limit GET POST>
order deny, allow
deny from all
allow from 200.210.107
require user walece mara
```

</Limit>

Para liberar o acesso a qualquer usuário válido:

<Limit GET POST>
require valid-user
</Limit>

O arquivo de usuários, cujo nome é definido pelo administrador do site (você ou o Webmaster), deve ser criado pelo utilitário htpasswd que criará o arquivo de usuários e senhas especificas de cada um, sendo que a senha será criptografada para garantir a segurança dos dados. Existem dois formatos para a utilização desse utilitário. O primeiro formato é:

Htpasswd –c <nome_arquivo> usuário

Neste formato, o utilitário criara o arquivo <nome_arquivo) e incluirá o usuário solicitado, requisitando antes que seja informada a senha para esse usuário (solicitará ainda a confirmação da senha). Devemos utilizar este formato somente quando quisermos criar um novo arquivo ou recriar um arquivo existente (caso o arquivo informado exista, irá apaga-lo antes de inserir o usuário. O segundo formato é:

Htpasswd <nome_arquivo> usuário

O arquivo <nome_arquivo> deverá esistir par aque este formato possa ser utilizado. Por meio dele podemos incluir novos usuários no arquivo de senhas. A tela exibida para ambos os formatos é:

Adding password for usuário
New passoword:
Re-Type new passoword:

6.2.1 Utilizando o Método de Autenticação do PHP

A forma de autenticação mostrada no tópico anterior pode ser utilizada em um programa PHP com uma finalidade diferente daquela. Em vez de permitir o acesso de um usuário autorizado a uma área do site, podemos utilizar a autenticação para validar um usuário cadastrado em nosso bando de dados, por exemplo. Para realizar esta tarefa, o PHP disponibilizara a instrução Header, a qual tem a função de enviar cabeçalhos HTTP ao browser. Podemos enviar qualquer comando de cabeçalho que o browser aceite, entre eles data de expiração da página, se o browser pode utilizar cachê para a página, ect. A sintaxe da instrução Header é:

Header (<string_comandos>)

A instrução Header deve ser enviada antes de qualquer outro comando de saída PHP, ou de qualquer tag HTML, tais como <html> ou <head>.

6.3 Cookies

Cookie é um mecanismo utilizado pelo servidor para armazenar ou recuperar dados no computador do usuário. Ou seja, quando o servidor envia um cabeçalho HTTP para o cliente (browser do usuário), pode também enviar um dado para ser armazenado (=cookie). Com isto, qualquer requisição HTTP enviada pelo cliente incluirá os valores dos cookies armazenados, desde que certos parâmetros sejam coincidentes (veja abaixo a relação de parâmetros). Este mecanismo simples proporciona aos scripts (PHP, por exemplo) um modo prático e poderoso de armazenar informações a respeito de seus visitantes. Podemos armazenar os produtos já selecionados por clientes (carrinho de compras), guardar o login dos usuários, fazenod com que eles não se autentiquem sempre que entrarem no site, armazenar preferência do visitantes (cor do fundo do browser, tipo de fonte e tamanho, etc), enfim uma série de informações úteis para a melhor interação entre site e visitantes.

Considerados pequenos fragmentos de texto enviados ao browser do usuário durante o período de conexão, enquanto ele verifica o conteúdo de um site. Eles podem ajudar você a criar cestas de compras, comunidades de usuários e personalização de sites. Digamos que você tenha planejado determinar uma variável de identificação para o usuário para que possa controlar quando ele voltar ao seu site. Primeiro, o usuário estabelece login e depois envia-se

um cookie com as variáveis determinadas para dizer: “Este é o usuário X, o usuário X está autorizado a estar aqui”. Enquanto o usuário X navega pelo seu site, você pode dizer “Olá, X” em cada uma das páginas. Se X clicar em seu catalogo e escolher 14 itens diferentes a comprar, você pode armazena-los e exibi-los todos em um grupo quando X clicar em Checkout. Mas o que acontece quando o usuário não aceita cookies? Você tem algum plano paralelo ou tudo será zerado? X vai deixar de comprar estes 14 itens?

Estes tipos de identificação de cookies são muito utilizados em sites de e-commerce, quando se esta criando um sistema de compras que permita ao usuário ir cumulando os itens selecionados até que deseja pagá-los. Pode-se utilizar os cookies para tudo o que se possa imaginar.

Um cookie é enviado ao computador cliente pela instrução Set-cookie enviada num cabeçalho HTTP. A sintaxe desta instrução é:

Set-Cookei: Name=VALOR; expires=DATA; path=Caminho;
Domain=Nome_dominio; secure

Em que,

Name = VALOR	É a única informação requerida. Nela informamos o nome do cookie seu valor (nome não pode conter os caracteres ponto-e-vírgula, vírgula e espaço em branco).
Expires = DATA	O atributo expires especifica a data de validade do cookie, ou seja, uma vez que a data especificada seja alcançada, o cookie é excluído do computador cliente. Este atributo é opcional, sendo que, quando não informado, o cookie será excluído no término da sessão do usuário (geralmente quando o usuário fecha o browser).
Path = Caminho	Este atributo especifica o diretório (ou o nome inicial dos diretórios) em que o cookie é válido. Desta forma podemos especificar que um cookie deve ser enviado ao servidor somente para as páginas de um diretório específico. Quando não especificado o atributo path, o cookie será enviado a qualquer página do site.

Domain = Nome_domínio	Especifica para qual domínio o cookie será enviado. Geralmente o cookie é enviado somente para o computador que o gerou (condição padrão), mas podemos determinar um domínio para o qual o cookie será enviado. Desta forma, podemos determinar que o cookie seja enviado para o domínio site.com.br. Com isto qualquer host com esta terminação recebe o cookie (site1.site.com.br, cookie.site.com.br, etc.).
Secure	Os cookies marcados com este atributo somente serão enviados ao servidor se a comunicação entre o servidor e cliente for uma conexão segura. Atualmente isto significa que o cookie será enviada apenas para servidores HTTPS (HTTP sobre SSL).

6.3.1 Como Definir Cookies

Antes de você começar a definir cookies, determine como irá usa-los e em que ponto irá "defini-los". Você deseja chegar um cookie em uma página qualquer em seu site e descobrir que ele não existe? Como você utilizará os cookies de dados? Sempre que optar pelo uso dos cookies, lembre-se de que precisa cria-los antes de enviar qualquer valor a eles através do browser. Se você lembrar disso, não gastará horas com mensagens como esta: "Não é possível enviar informações adicionais, o cabeçalho já enviou".

Aqui está a sintaxe da função setcookie ( ) no PHP:

Setcookie("name", "value", "expiration", path", "domain", "security");

- Name. O parametro name controla o nome da variável que será armazenada globalmente em $HTTP_COOKIE_VARS, e será acessível nos scripta subseqüentes.
- Value. O parâmetro value é o valor da variável passada por parâmetro nome.
- Espiration. O parâmetro expiration determina um tempo durante o qual o cookie estará acessível. Os cookies sem um tempo de experição determinado expiram quando o usuário encerra a sessão (quando o browser é fechado).

- Path. O parâmetro path determina para quais diretórios o cookie é válido. Se uma barra simples for colocada no parâmetro path, o cookie será valido para todos os arquivos e diretórios do servidor Web. Se um diretório especifico for determinado, este cookie estará disponível para as paginas dentro deste diretório.
- Domain. Os cookies são válidos somente para o host e domínio que os definiram, e se nenhum domínio for especificado, então o valor default será o nome do host do servidor que gerou o cookie. O parâmetro domain precisa ter mais de 2 períodos na string para que seja valido.
- Security. Se o parâmetro security é 1, então o cookie será transmitido via HTTPS.

6.3.2 Autenticando Usuários por meio de Cookies

A utilização de cookies para autenticação de usuários é uma das melhores forma de garantirmos o acesso dos usuários do nosso site, de modo seguro e organizado. Entre as vantagens em relação ao método anterior, podemos citar:

- O método de autenticação de cookies é aceito pela maioria dos servidores WEB.
- Garante uma autenticação segura e confiável.
- Podemos manter a autenticação por um prazo indeterminado.
- Além dos dados de autenticação, podemos armazenar outras informações que facilitem a navegação do usuário pelo site (preferências do usuário, por exemplo).
- O método de “logout” é o mais fácil e prático.

A autenticação por meio de cookies envolve os seguintes passos:

- Construção de uma página de “login” para autenticação dos usuários.
- Construção da rotina de validação do usuário e criação dos cookies.
- Alteração das páginas privativas de usuários para verificação dos cookies.
- Construção de uma página de “logout” para exclusão dos cookies, se assim o usuário deseja.

6.4 Trabalhando com Sessões

A versão 4 do PHP disponibiliza um método muito prático para gerenciamento de usuário, no qual são abertas sessões por usuário. Podemos registrar a quantidade de variáveis

que desejarmos nesta sessão, permitindo que a visita do usuário ao site seja controlada e sejam criadas páginas totalmente customizadas.

Cada visitante do site recebe um número único de sessão (session id). Este método pode utilizar tantos cookies, quanto variáveis propagadas na URL para enviar informações sessão. No caso de variáveis propagadas na URL, o PHP suporta o método de envio de identificação da sessão (session id) transparentemente. Para isto, basta habilitar o parâmetro –with-trans-sid durante a compilação do PHP. Com esse parâmetro ativado, o PHP inserirá a identificação da sessão no corpo de mensagem. Alternativamente podemos propagar a identificação da sessão por meio da constante SID, criada automaticamente pelo PHP para conter a identificação da sessão criada (este método deve ser utilizado para cliente que não aceitam cookies). Se não existir nenhuma sessão criada, essa constante retorna uma string vazia.

6.4.1 Tudo sobre variáveis

Para falar de maneira bem simples, variáveis representam dados. Se você desejar que um script guarde uma informação específica, primeiro crie uma variável e então determine um valor literal para a mesma, usando o sinal de igualdade (=).
Por exemplo, a variável “username” guarda o valor literal “joe” quando colocada em seu script assim:

```
$username = “joe”;
```

Os nomes das variáveis iniciam com o sinal de cifrão ($) e são seguidas de um nome conciso e significativo. Um nome de variável não pode iniciar com um caractere numérico, mas pode conter números e sublinhados (_). Adicionalmente, os nomes de vaiáveis são sensíveis ao tipo da letra, considerando que $YOURVOER e $yourvar são duas variáveis diferentes.

Criar nomes de variáveis significativos é outra forma de evitar dores de cabeça na hora de escrever um código. Se seu script lida com valores de nome e senha, não crie uma variável chamada $n para name (nome) e outra chama $p para password (senha) – este não são nomes significativos. Se você usar este script semanas mais tarde, deve achar que $n é a variável para “numero” em vez de para “nome” e que $p se refere a “página” e não “senha”.

Esta seção descreve vários tipos de variáveis. Alguns trocam quando o script é executado e outras preservam seus valores além do script PHP – como os formulários HTML.

6.4.1.2 Variáveis PHP

Você irá criar dois tipos principais de variáveis em sue código PHP: escalar e vetores. As variáveis escalares contêm somente um valor por vez, enquanto os vetores contem uma lista de valores ou outros vetores (produzindo um vetor multidimensional). Dentro das variáveis, existem valores que foram associados, que podem sr de tipo diferentes, como a seguir:

- Inteiros. Números inteiros (números sem decimais). Alguns exemplos são 1,345 e 9922786. Você só poderá usar notações octal e hexadecimal: o octal “0123” é o decimal 83 e o hexadecimal “0x12” é o decimal 18.
- Números de ponto flutuante (“flutuantes” ou “duplos”). Números com casas decimais. Exemplos são 1.5, 87.3446 e 0.88889992.
- Strings. Informações de texto e/ou numéricas, especificadas quando colocadas entres aspas duplas (“ “) ou aspas simples (‘ ‘).

Quando for iniciar um script PHP, planeje suas variáveis, atribua nomes adequados e sue comentários em seus código para relembrar as associações.

6.4.1.3 Variáveis Formulário HTML

Os formulários HTML possuem sempre três elementos: um método (method), uma ação (action) e um botão de envio (submit). Ao clicar em um botão de envio de um formulário HTML, as variáveis são passadas para o script especificado no “action” do formulário, como método (method) específico. Neste exemplo, ao clicar no botão “calcular”, o formulário passa as variáveis para o script chamado docalc.php utilizado o método POST:

```
<FORM method=”POST” acition=”docalc.php”>
    <INPUT type=”submit”value=”calcular”>
</FORM>
```

Dependendo do método de seu formulário html! (GET ou POST), as variáveis serão parte do array associativo global HTTP_POST_VARS ou $HTTP_GET)_VARS. O nome do

campo de entrada era tornar-se o nome da variável. Por exemplo, o campo de entrada a seguir gera a variável $first_name:

<input type = “text” name =”first_name” size=”20”>

Se o método utilizado neste formulário foi POST, esta variável pode ser referida também como #HTTP_POST_VARS[“first_name”]. Se foi utilizado o método GET, você também pode usar $HTTP_GET_VARS [“first_name”].

As variáveis passadas do formulário para o script PHP são colocadas em um vetor global associativo $HTTP_POST_VARS (dependendo do método do formulário) e ficam automaticamente disponíveis para o script.

6.4.1.4 Variáveis de Cookie

Como variáveis de formulários, variáveis de cookie são guardadas em um array associativo global chamado $HTTP_COOKIE_VARS. Se você configura um cookie chamado “user” com o valor de “joe Smith”, como:

SetCookie (“user”, “Joe Smith”, time ( ) + 3600);

Uma variável chamada $user será colocada em $HTTP_COOKIE_VARS, com o valor de “Joe Smith”. Você pode se referi a $user ou $HTTP_COOKIE_VARS[“user”] para obter o valor.

6.4.1.5 Variáveis de Ambiente HTTP

Quando um browser faz uma solicitação a um servidor da Web, ele envia juntamente com a solicitação uma lista de variáveis extras. Elas são chamadas variáveis de ambiente e são muito utilizadas para exibir conteúdo dinâmico ou autorização de usuários.

Adicionalmente, a função phpinfo ( ) exibe muitas informações sobre a versão do PHP que você está utilizando e o software utilizado como servidor da Web, em complemente a um projeto básico de HTTP. Crie um arquivo chamado phpinfo.php, contendo somente as seguintes linhas:

```
<?php
        phpinfo ();
?>
```

Por default, as variáveis de ambiente estão disponíveis para PHP como $VAR_NAME. Por exemplo, a variável de ambiente REMOTE_ADDR já existe como $REMOTE_ADDR. Portanto, para ter certeza absoluta de estar lendo o valor correto, utilize a função getenv () para determinar o valor para a variável que você escolher. Por exemplo, o seguinte código coloca explicitamente o valor da variável de ambiente REMOTE_ADDR em uma variável chamada $remore_address:

```
$remote_address = getenv (“REMOTE_ADDR”);
```

6.4.1.5.1 REMOTE_ADDR

A variável de ambiente REMOTE_ADDR contém o endereço IP da máquina que esta fazendo a solicitação. Crie um script chamado remote_address.php, contendo o seguinte código:

```
<?php
        $remote_address = getnv (“REMOTE_ADDR”);
        echo “You IP address is $remote_address.”);
?>
```

Salve este arquivo e coloque-o em seu servidor da Web, então acesse-o com o browser em sua URL: HTTP://www.seuservidor.com/remote_address.php.

Você deve ver o “Seu endereço IP é [algum número]” na tela. Por exemplo, eu vejo “Seu endereço IP é 209.244.209.209” este endereço IP é atualmente o valor determinado ao meu computador pelo meu provedor da Internet.

6.4.1.6 Variáveis de Projeto

Quando um browser de Web faz uma requisição ao servidor da Web, ele envia juntamente com a requisição uma lista de variáveis extras chamadas variáveis de projeto. Elas podem ser muito úteis para mostrar conteúdo dinâmico e/ou autorizar usuários.

Por default, as variáveis de projeto estão disponíveis para o php como $VAR_NAME. Entretanto, para ter absoluta certeza de que você está vendo o valor correto, você pode usar a função getnv ( ) para determinar uma valor para a variável de sua escolha. A seguir estão algumas variáveis comuns de projeto:

REMOTE_ADDR obtém o endereço IP da máquina fazendo a requisição. Por exemplo:

```
<?php
        $remote_address = getnv (“REMOTE_ADDR”);
        echo “seu endereço de IP é $remote_address.”;
?>
```

HTTP_USER_AGENT obtém o tipo de browser, versão, linguagem de decodificação e plataforma. Por exemplo:

```
<?php
        $brwser_type = getnv (“HTTP_USER_AGENT”);
        echo “Você está usando $browser_tupe.”;
?>
```

Para uma lista das variáveis do projeto HTTP e suas descrições, visite HTTP://hoohoo.ncsa.uiuc.edu/cgi/env.html.

6.4.1.6.1 Operadores Lógicos

Os operadores lógicos, assim como os operadores de comparação, são geralmente encontrado dentro das declarações de controle if...else e while. Estes operadores permitem que seu script determine o status da condição e, no contexto das declarações if...else ou while, execute determinado código baseado em quais condições são verdadeiras e quais são falsas.

Um operador lógico comum é o | |, conhecido como OR. O exemplo a seguir mosta a comparação de duas variáveis e o resultado da declaração. Neste exemplo, eu ralmente desejo tomar café. Eu tenho duas opções, $drinki2 e $drink2. se a minha opção for “café”, eu ficarei muito feliz. Caso contrário, ainda estarei precisando de cafeína.

```
$drink1 = “café”;
$drink2 = “leite”;
        if (($drink1 = “café”) | | ($drink = café)) {
        echo “Eu estou feliz!”;
}else {
        echo “Eu ainda preciso de cafeína.”;
}
```

Neste exemplo, em virtude do valor da variável $drink1 ser “café”, a comparação OR de $drink1 e $drink2 é verdadeira (TRUE), e o script retorna “Eu estou feliz!”.
Outros operadores lógicos incluem ANH (&&) e NOT (!).

6.4.1.7 Operadores

Um operador aritimético é um símbolo que representa uma ação especificada. Por exemplo, o operador aritmético + adiciona dois valores, e o operador de atribuição = atribui um valor para a variável.

6.4.1.7.1 Operadores Aritméticos

Operadores aritméticos agem com a incrível semelhança de operadores simples, como mostrado na Tabela 6.1. No exemplo, $a = 5 e $b = 4.

Tabela 6.1 Operadores aritméticos

Operador	Nome	Exemplo	
+	Adição	$c=$a+$b	//$c=9
-	Subtração	$c=$a-$b	//$c=1
*	Multiplicação	$c=$a*$b	//$c=20
/	Divisão	$c=$a/$b	//$c=1.25
%	Módulo, ou "resto"	$c=$a%$b	//$c=1

Fonte: Meloni, Julie – 2000

6.4.1.7.2 Operadores de Atribuição

0 = é o operador básico de atribuição:

$a = 124; // o valor de $a é 124

Outros operadores de atribuição incluiem +=,-= e .=.

$ex +=1; //Atribui o valor de ($ex + 1) para $ex.
// Se $ex = 2, então o valor de ($ex += 1) é 3.

6.4.1.7.2.1 Sybase_query ( )

Esta função emite o comando SQL. Requer uma conexão fechada ao bando de dados.
$sql_result = sybase_query (SELECT * FROM SOMETABLE", $connection);

6.4.1.7.2.2 Sybase_fetch_array ( )

Esta função coloca automaticamente os resultados do comando SQL em um array.
$row = sybase_fetch_array ($sql_result);

6.4.1.7.2.3 Sybase_free_result ( )

Esta função libera os recursos de memória utilizados por uma consulta a um banco de dados.
Sybase_free_result ($sql_result);

6.4.1.7.2.4 Sybase_close ( )

Esta função fecha explicitamente uma conexão a um banco de dados.
Sybase_close ($connection);

6.4.1.8 Funções de data e hora

As funções básicas de data e hora do PHP permitem que você formate facilmente horários para uso em consultar de banco de dados e funções de calendário, bem como uma exibição simples de data em um formulário de pedidos recebido.

6.4.1.8.1 Date ( )

A função date ( ) retorna o intervalo de tempo atual do servidor, formatado de acordo com um grupo de parâmetros fornecidos. Aqui esta a sintaxe da função date ( ):

Date (format, [timestamp]);

Se o parâmetro timestamp não for fornecido, o formato atual será assumido. A tabela 6.2 mostra os formatos disponíveis:

Tabela 6.2 Formatos Data e Hora

Caractere	Significado
A	Exibe “am” ou “pm”
A	Exibe “AM” ou “PM”
H	Hora, formato 15-houras (01 a 12)
H	Hora, formato 24-houras (00 a12)
G	Hora, formato 12-horas sem zero (1 a 12)
G	Hora, formato 24-horas sem zero (0 a 23)
I	Minutos (00 a 59)
S	Segundo (00 a 59)
Z	Zona de tempo em segundos (-43200 a 43200)
U	Segundos (janeiro 1, 1970 00:00:00 GMT)
D	Dia do mês, dois dígitos (01 a 31)
J	Dia do mês, dois dígitos sem zero (1 a 31)
D	Dia da semana, texto (seg a dom)
L	Dia da semana, texto por extenso (segunda a domingo)
W	Dia da semana, numérico, Domingo a Sábado (0 a 6)
F	Mês, texto por extenso (Janeiro a Dezembro)
M	Mês, dois dígitos (01 a 12)
N	Mês, dois dígitos sem zero (01 a 12)
M	Mês, texto de três letras (Jan a Dez)
Y	Ano, quatro dígitos (2000)
Y	Ano, dois digitos (00)
Z	Dia do ano (0 a 365)
T	Número de dias do mês atua (28 a 31)
S	Sufixo ordinal em inglês (th, ndm st) (primeiro, segundo, terceiro)

Fonte: Meloni, Julie – 2000

Por exemplo, o exemplo a seguir era exibir a data atual neste formato: Janiero 10 th 2000, 08:08 AM.
Echo date (F jS Y, h:iA,");

6.4.1.8.2 Symlink ( )

A função symlink ( ) cira um link simbólico de um arquivo ou diretório existente no sistema de arquivo a um link especifico. Sua sintaxe é

Symlink ("targetname", "linkname");

Por exemplo, para criar um link simbólico chamado index.html para um arquivo chamado index.html, use

Symlink("index.html", "index.phtml");

6.4.1.8.3 Unlink ( )

A função unlink ( ) deleta um arquivo do sistema de arquivos. Sua sintaxe é

Unlink ("filename");

Por exemplo, deletar um arquivo chamado index.html no seu diretório local, use:

Unlink ("/home/username/index_html");

O usuário PHP precisa ter permissão de gravação para este arquivo.

6.4.1.9 Funções HTTP

As funções embutidas para enviar cabeçalhos HTTP específicos e dados de cookie são aspectos cruciais no desenvolvimento de aplicações para Web com o PHP. Felizmente, a sintaxe para essas funções é muito fácil de entender e implementar.

6.4.1.9.1 Header ( )

A função header ( ) envia a string de cabeçalho THHP, como um redirecionamento de localização. Esta saída precisa ocorrer antes que qualquer outro seja enviado ao browser, inclusive tags HTML.

Por exemplo, para usar a função header ( ) para redirecionar o usuário para uma nova localização, use este código:

```
Header ("location:HTTP://www.newlocation.com");
Exit;   // segue um comando header (cabeçalho) com o comando exit.
        // assegura que o código não continuara a ser executado.
```

6.4.1.9.2 Setcookie ( )

A função setcookie ( ) envia um cookie ao usuário. Os cookies precisam ser enviados atnes que qualquer outra informação de cabeçalho seja enviada ao browser. Sua sintaxe é

```
Setcookie ("name", "value", "expire", "path", "domain", "secure");
```

Por exemplo, voce usaria o seguinte código para enviar um cookie chamado username com um valor de joe válido por uma hora em todos os diretórios do domínio testcompany.com:

```
Setcoolie ("username", "joe", time( ) + 3600, "/". Testcompany.com");
```

6.4.1.10 Função mail

A função PHP mail faz a interface entre os formulários HTML e o seu programa servidor de email.

6.4.1.10.1 Mail ( )

Se seu servidor tem acesso a sendmail ou um gateway SNMP, a função mail ( ) envia mail para um destinatário especifico. Sua sintaxe é

Mais (“recipient”, “subject”, “message”, “mail headers”);

Por exemplo, o codigo seguinte envia mail para Julie@thicbook.com, sob o assunto “Eu estou enviando um email!” e uma mensagem dizendo “O PHP é interessante!”. A linha “From:” (remetente) é parte do cabeçalho adicional do email.

Mail (julie@thickbook.com, “eu estou enviando um email!”, “O PHP é interessante”, “From: youremail@yourdomain.com\n”);

6.4.1.11 Funções matemáticas

Já que possuo muitas poucas habilidades matemáticas, encontrei nas funções matemáticas embutidas no PHP algo de muita importância! Em complemento a todas as funções, o valor de PI (3.14159265358979323846) já é definido como uma constante no PHP (M_PI).

6.4.2 Como Exibir Conteúdo Dinâmico

A Web é um ambiente dinâmico, está sempre mudando e evolui constantemente, então por que não utilizar seus conhecimentos de programação para exibir conteúdo dinâmico? Você pode criar um projeto customizado para que o usuário receba informações específicas, de acordo com os valores com os valores através de formulários HTML, o tipo de browser utilizado as variáveis armazenadas em cookies.

6.4.3 Como Redirecionar Para um Novo Endereço

Redirecionar automaticamente um usuário para uma nova URL significa que seu script tem que enviar um cabeçalho HTTP ao browser antes de qualquer informação, indicando um novo endereço. Existem muito tipos de cabeçalhos HTTP, a partir dos quais você ode indicar os códigos de caracteres utilizados e a data de expiração do documento para qqueles que enviam a mensagem de erro 404 – File Not Found (arquivo não encontrado). Se você deseja aprender mais sobre o que pode ser feito com os cabeçalhos, leias as especificações atuais do HTTP 1.1 em HTTP://w3.org/Protocols/rfc2068/rfc2068.
O formato para enviar um cabeçalho HTTP a partir de um script PHP é?

Header (string);

Onde string é o texto do cabeçalho, entre aspas. Por exemplo use este código para exibier um cabeçalho redirecionado que encaminhará o brouser ao site PHP:

```
<?php
        header (“Location:HTTP://www.php.net/”);
        exit;
?>
```

Utilize a declaração exit para determinar que o script não continuará sendo executado. Leve este script de redirecionamento um pouco mais adiante e adicione um formulári HTML com um caixa de lista drop-down como front end para a rotina de redirecionamento. Dependendo dos valores selecionados na lista, o script redireciona o browser para qualquer uma URL.

Para começar, abra seu editor de textos favoritos, crie um arquivo chamado show_menu.html e defina um “Shell” HTML.

Agora, crie o código do formulário, assumindo que seu script PHP será chamado do_redirect.php e seu formulário utilizará o método POST?

```
<FORM method=”POST” action=”do_redirect.php”>
```

Agora crie uma caixa de lista drop-down contendo as opções de menu?

```
<P> Eu desejo ir para:
<SELECT name=”location” size=”1”>
        <OPTTION value=HTTP://www.prima-tech/>Prima-Tech</OPTION>
        <OPTTION value=HTTP://www.php.net/”PHP.net<OPTION>
        <OPTTION value=HTTP://www.slashdor.org/”>Slashdot</OPTION>
        <OPTTION value=HTTP://www.linuxchix.org/”>Linuxchix</OPTION>
</SELECT>
```

E, finalmente, adicione um botão Submit:

```
<INPUT type=”submit” value=”IR!>
```

Não esqueça de fechar o tag </FORM>!

6.4.4 Como Ler e Gravar Arquivos de Dados

Além de enviar um e-mail repleto de dados, você pode criar script simples para criar, abrir e gravar arquivos em seu servidor da Web utilizando a função open ( ).

6.4.4.1 Como Gravar Arquivos de Dados

A função open ( ) recebe dois argumentos, nome de arquivo e modo, e retorna um ponteiro para o arquivo. O ponteiro fornece informações sobre o uso do arquivo e é utilizado como referência. Um nome de arquivo é o caminha inteiro para o arquivo que você quer criar ou abrir, e o modo pode ser um dos seguintes:

- r Abre o arquivo existente para seja possível ler os dados ali contidos. O ponteiro é colocado no início do arquivo, antes de qualquer dado.
- r + Abre o arquivo existente para leitura ou gravação. O ponteiro é colocado no início do arquivo, antes de qualquer dado.

- w Abre um arquivo somente para gravação. Se o arquivo com este nome não existir, então tenta criar um novo arquivo. Se o arquivo já existir, deleta todo o conteúdo e coloca o ponteiro no início do arquivo.
- w + Abre um arquivo para leitura e gravação. Se um arquivo com este nome não existir, então tenta criar um novo arquivo. Se o arquivo já existir, deleta todo o conteúdo e coloca o ponteiro no início do arquivo.
- a Abre um arquivo somente para gravação. Se um arquivo com este nome não existir, então tenta um novo arquivo. Se o arquivo já existir, coloca o ponteiro no final do arquivo, depois de todos os dados.
- a + Abre um arquivo para leitura e gravação. Se um arquivo com este nome não existir, então tenta criar um novo arquivo. Se o arquivo já existir, coloca o ponteiro no final do arquivo, depois de todos os dados.

Então, para criar um novo arquivo no root de um documento de seu servidor da Web (por exemplo, /usr/local/apacha/htdocs/) chamado mydata.txt, para ser utilizado para leitura e gravação de dados, utilize este código:

```
<?php
$newfile = fopen (“usr/local/apache/htdocs/mydata.txt”, “a+”);
?>
```

Neste exemplo, $newfile é um ponteiro de arquivo. Você pode se referir a este ponteiro de arquivo quando for ler, fechar ou executar outras funções com o arquivo.

Depois que você abrir o arquivo, certifique-se de fecha-lo usando a função fclose ( ):

```
Fclose ($newfile);
```

Mas abrir um arquivo apenas e ter que fecha-lo depois pode ser chato, então use as funções fwrite ( ) ou fputs ( ) para colocar dados no arquivo aberto:

```
Fwrite ([file], [data]);
```

Crie um script chamado write_data.php, contendo o seguinte código:

```
<?php
$newfile = fopen (“/usr/local/apache/htdocs/mydata.txt”, “a+”);
fwrite($newfile, “Este é um novo arquivo.”);
```

```
fclose ($newfile);
echo “Tudo concluído!”;
?>
```

Certifique-se de modificar o caminho para o arquivo de forma a combinar com seu próprio ambiente. Adicionar uma declaração echo depois que o arquivo for aberto, gravado e fechado fará com que a mensagem seja exibida depois que as ações forem concluídas. Se algum erro surgir, como problemas com permissões de arquivo, você os verá em sua tela.

6.4.4.2 Como Ler Arquivos de Dados

Você também pode usar o PHP para verificar se foi feita alguma gravação no arquivo, utilizando a função fread ( ) para reunir os dados em uma variável. A função fread ( ) usa dois argumentos

```
Fread ([filename], [length]);
```

Para ler completamente o arquivo, o tamanho pode ser encontrado usando a função filesize ( ):

```
Fread ([filename], filesize ([filename]));
```

A autenticação de usuários é um esquema que verifica se o usuário tem permissão para acessar determinados conteúdos de seu site da Web. Quando você iniciar o desenvolvimento do site, pode desejar restringir o acesso somente a determinados membros da sua equipe de desenvolvimento. Ou, se o site corporativo de sua empresa contém dados financeiros delicados, você pode desejar que o acesso a estes dados fiquem restritos a uma lista particular de investidores.

Os desenvolvedores da Web geralmente utilizam um dos seguintes tipos de autenticação de usuários para razões que variam desde a facilidade na instalação e manutenção até a forma como o esquema de autenticação funciona dentro do desenvolvimento da aplicação como um todo:

- Autenticação básica HTTP básica – este é o tipo mais popular de autenticação. Utiliza funções embutidas do servidor da Web para limitar o acesso a documentos e diretórios inteiros. A popularidade deste esquema deve-se ao fato de que qualquer desenvolver de site, tendo a controle de seu próprio servidor ou mantendo seu site com um Provedor de Serviços na Internet, tem a habilidade de usar autenticação HTTP Básica dentro dos diretórios de seu documento.
- Autenticação direcionada por bando de dados – os nomes de usuários e senhas são mantidos em uma tabela de banco de dados e acessadas via script.
- Restrição por endereço IP – Acesso limitado a um IP específico ou faixa de IP.

A seguir, serão mostradas varias formas de autenticação de usuário em seus sites da Web com permissão para PHP.

6.5 Autenticação HTTP Básica

A autenticação HTTP usa em esquema de pergunta/resposta para autenticar usuários. O processo tem início quando o usuário solicita um arquivo de um servidor de Web. Se o arquivo estiver dentro de uma área protegida, o servidor responde com o erro 401 (usuário não autorizado) e o browser exibe uma caixa de diálogo familiar tipo username/password (nome do usuário/senha). O usuário então entra um nome e senha e clica em OK, enviando as informações ao servidor para autenticação. Se o nome do usuário e a senha forem válidos o servidor exibe o arquivo solicitado. Caso contrário, a caixa de diálogo reaparece para que o usuário tente novamente.

Para utilizar a autenticação básica HTTP, você precisa ter disponíveis dos elementos: um servidor habilitado a autenticação e uma lista de usuários e senhas. No entanto, a configuração da autenticação varia de acordo com o servidor da Web. Embora o Netscape e a família Microsoft utilizem uma interface gráfica de usuário de usuário para criação e administração de nomes de usuários e lista de senhas, o servidor Apache usa o método da "velha escola" que requer em usuário modifique manualmente a configuração do servidor e nos arquivos de senha.

6.5.1 Como Trabalhar Com Variáveis de Autenticação no PHP

Um script personalizado PHP pode simular uma autenticação HTTP do tipo pergunta/resposta determinada os cabeçalho HTTP que causam a exibição automática da caixa de diálogo username/password. O PHP armazena estas informações digitadas na caixa de diálogo em três variáveis ($PPHP_AUTH_USER, $PHP_AUTH_PW e $PHP_AUTH_TYPE) que podem ser utilizadas para validação da entrada.

As variáveis de autenticação HTTP no PHP estão disponíveis somente quando o PHP é instalado como um módulo. Se a versão CGI do PHP estiver instalada, você estará limitando à autenticação normal baseada em .htaccess ou autenticação direcionada por banco de dados utilizando formulários HTML para receber o nome e a senha do usuário.

6.5.2 Autenticação Direcionada Por Banco de Dados

A validação feita por um banco de dados alivia a necessidade de uma configuração adicional no seu servidor da Web e aumenta o tempo de resposta do servidor. Ao utilizar variáveis de autenticação PHP, você ainda pode ser a caixa diálogo username/password ou criar um formulário de login curto em HTML que exija um nome de usuário e senha. Seja qual for método que você escolher, você precisará ter acesso a uma tabela de usuários que guarde as informações de nome de usuário e senha.

6.5.3 Restrições Por Endereço IP

Outro método de proteger dados importantes é limitar o que será exibido para um IP especifico ou faixa de IP. Embora este seja um processo muito simples, não é o mais eficiente. Muitos usuários não possuem IP estático, e aqueles que possuem normalmente estão por trás da parede protetora e acessam a World Wide Web via proxy. Quando você acessa via proxy, um tipo de gateway que filtra o tráfego para a Internet, o endereço remoto é sempre o mesmo valor, porque pertence ao proxy e não à tentativa específica do usuário em acessar um site da Web.

Portanto, se você está criando um site em um ambiente fechado, adicionar estas poucas linhas no inicio de seu script PHP irá determinar o endereço IP remoto e limitar o acesso baseado nos resultados encontrados.

```
<?php // limitbyIP.php
        #userIP = getenv (“REMOTE_ADDR”);
        if ($userIP ! = “127.0.0.1”) {
        echo “Não é um acesso local…”);
        } else {
        echo “Usuário atutorizado!”);
        }
?>
```

Você pode usar expressões regulares para combinar um bloco de endereços IP, neste caso qualquer endereço IP que inicie com 208.56.5.

```
<? // limitbyIP_range.php
        $userIP = getenv (“REMOTE_ADDR”);
        if (preg_match (“/208.56.8./”, “$userIP)) {
        echo “Voce não esta dentro da faixa especificada…”;
        }else {
        echo “Usuário autorizado!”;
        }
?>
```

O REMOTE_ADDR como já descrito no item 6.4.1.5.1, é uma variável de ambiente HTTP padrão que é sempre enviada pela máquina fazendo a solicitação. O script limitbyIP.php usa 127.0.0.1 ou o valor default para o host local, como o único endereço IP autorizado. Neste script, se o endereço remoto não for 127.0.0.1 ou o host local, não é mostrado ao usuário qualquer conteúdo, e em vez disso a mensagem “Não é local...” aparece no browser.

Embora este seja um bom truque, e muitos rápido já que o script não se conecta a um banco de dados para validar usuários específicos, perde seu valor em um ambiente de produção dada a prevalência de Ips não estáticos e servidores proxy.

6.6 JavaScript e PHP

Por sua característica de processamento do lado servidor, o PHP não consegue interagir com os usuários depois que as páginas são montadas e enviadas ao brouwser. Mas na grande maioria das situações, nós precisamos realizar determinadas tarefas na própria página. Como exemplo posso citar: validação dos campos digitados, carregamento dinâmico de listas conforme parâmetros informados na página, abertura de novas janelas, redirecionamento entre o site e seus visitantes.

O javascript é uma ferramenta suportada pela maioria dos browsers atuais (pelo menos os mais utilizados, como IE e Nestcape) e possui uma grande biblioteca espalhada em vários sites especializados em JavaScript, sendo que as fontes geralmente são grátis e extremamente úteis para desenvolvimento de páginas profissionais na WEB.

JavaScrit é uma linguagem de programação inicialmente conhecida como LiveScript, desenvolvida pela Netscape, para tornar mais poderoso o seu browser, proporcionando maior interatividade com os usuários. Atualmente, a maioria dos browsers suportam JavaScript (ao menos os mais utilizados têm suporte a JavaScript, apesar de existirem diferenças entre os diversos browsers para utilização de funções específicas). Os programas JavaScript permitem que manipulemos praticamente tudo no browser do usuário, desde validação de formulários, apresentação de novas janelas, manipulação de imagens, criação de camadas, cálculos complexos, e muitas outras ações que podem tornar nossas páginas extremamente interativas.

Podemos inserir os códigos JavaScript em qualquer lugar de nossa página, mas como forma de organizar as minhas páginas, procuro inserir todas as funções que vou utilizar em um único lugar, geralmente antes da tag <body>, o que permite uma melhor manutenção de código nos prováveis erros que venham a ocorrer. Os programas JavaScript são colocados diretamente nas página HTML e são delimitados pelas tags <Script> e </Script>. Podemos inserir todo o código necessário entre estas tags, ou chamar uma biblioteca JavaScript. Sua sintaxe é:

```
<Script Language = “JavaScript” src=”<nome_biblioteca>” ></Script>
ou
<Script Language = “JavaScript”>
<!—Aqui vai todo o código JavaScript  necessário - ->
</Script>
```

Em que <nome_biblioteca> pode ser um nome de arquivo em seu site ou em outro site qualquer (por exemplo, HTTP://livrophp.com.br/biblioteca/exemplo.js). Caso você queira prevenir erros em browsers que não reconhecem JavaScript, basta inserir o código em uma tag de comentários do HTML, ou seja:

```
<Scrip Language=”JavaScript”>
<!—Evita problemas com browser que não reconheçam JavaScript
alert (“Se você esta vendo esta mensagem é porque seu browser aceita
JavaScript”);
// - ->
</Script>
```

6.6.1 Variáveis no JavaScript

As variáveis devem ser explicitamente definidas. Para isto utilizando a declaração var seguida do nome da variável. Como no PHP, os nomes das variáveis no JavaScript são case sensitive, ou seja, o JavaScript). Você não precisa definir o tipo de variável. O JavaScript distingue entre maiúsculas e minúsculas (Teste, teste, tEste, TESTE são variáveis diferentes). Você não precisa definir o tipo de variável. O JavaScript é que os determirá o seu tipo conforme o contexto do programa. Outro detalhe importante é que os nomes das variáveis devem começar com uma letra, ou opcionalmente com o símbolo _ (sublinhado), e diferentemente do PHP não é necessário o caractere $ para identificar uma variável.

```
Var teste;
Var nome, estado, pais;
Var erro=true;
Var mensagem=”esta é uma mensagem de teste”;
```

As variáveis definidas no Java Script podem ter seu valor inicial atribuído na sua criação, sendo que variáveis definidas dentro de funções têm seu escopo restrito à existência da função, ou seja, ao término da função, todas as variáveis definidas internamente deixam de existir.

6.7 Alterando Propriedades de Elementos da Página

Além de responder a eventos gerados, por meio do JavaScript podemos alterar as propriedades dos elementos de uam página. Podemos, por exemplo, alterar a propriedade ACTIOM da tag <form>, ou inserir um valor em campo do tipo texto, ou até mesmo acrescenta mais um elemento a um campo do tipo select, ou mesmo criar um SELECT dinâmico, que seria carregado conforme os cliques do usuário página (por exemplo, podemos tem um SELECT com os grupos de produção, e ao escolher um grupo, por exemplo, bebidas, montar um outro SELECT com a relação de bebidas disponíveis). As propriedades podem ser acessadas por meio do nome do campo, precedido pelo nome do formulário (este precedido por document) maio o nome da propriedade que desejamos.

7. CRIPTOGRAFIA DE DADOS E ARQUIVOS DE LOG

Com o propósito de que se entenda como criptografar e descriptografar dados (arquivos, e-mail etc), falar-se-a aqui sobre chave pública de criptografia. Para mais informações o site da Web sobre o crypto ensina sobre outros criptosistemas e determina vários outros métodos que podem ser utilizados. Um bom lugar para iniciar é o FAQ do RAS Lab Crypto, encontrado em HTTP://www.rsasecurity.com/rsslabs/fag/.

Quando utilizamos chaves públicas de criptografia, o usuário tem um par de chaves: uma pública, outra privada. Com nomes sugestivos, a chave privada é mantida em segredo e a chave publica é distribuída para outros usuários. As chaves públicas e privada de um usuário são linkadas por uma complexa estrutura matemática que vai além da minha compreensão, mas que é fundamental para estabelecer os relacionamentos de chave. A chave pública é usada como base para criptografar uma mensagem e a chave privada é necessária para que o destinatário descriptografe a mensagem criptografada.

Por exemplo, suponha que o usuário Y possua um par de chaves. O usuário X também possui um par de chaves. Y e X desejam enviar uma mensagem criptografada um para o outro. Então eles trocarão chaves publicas. As chaves são armazenadas em grupos – um para as chaves privadas e um para chaves públicas – que são essencialmente arquivos que contêm informações de chave. Apenas possuindo o conjunto de chaves corretos você poderá abrir seu carro, casa e outros. No grupo possuindo o conjunto de chaves públicas de Y, ele possui a chave pública de X. No grupo de chaves públicas de X, ela possui a chave pública de Y. Ambos possuem também um grupo de chaves privadas que controlam somente suas próprias chaves privadas.

Quando Y deseja enviar uma mensagem criptografada para X, ele utiliza um software de criptografia para estruturar a mensagem baseado nas chaves públicas de X. X recebe a mensagem e usa seu software de criptografia para descriptografar a mensagem com seu grupo de chaves decodificadores – ou seja, suas chaves privadas. Somente X pode descriptografar a mensagem criptografada recebida, usando suas chaves públicas.

No início de 1990, Phil Zimmerman desenvolvedor o PGP, ou Pretty Good Privacy, que rapidamente tornou-se um software muito popular para critografia de email e arquivos.

Entretanto, em virtude dos regulamentos de exportação dos Estados Unidos e os regulamentos de importação de outros países que estavam de olho nos algoritmos de

criptografia, o padrão OpenPGP foi desenvolvido, e o software GNUPG foi construído com base neste. Diferentemente do software GNUPG não utiliza algoritmos de criptografia patenteados ou restritos, então tornou-se a alternativa popular para o PGP.

Embora as leis de exportação dos EUA tenham sido modificadas tanto o PGP como o GNUPG continuam coexistindo na comunidade de desenvolvedores. Após determinar qual software de criptografia você deseja utilizar, siga os passos mostrados nas próximas seções para aprender como criptografar e enviar para destinatários específicos.

7.1 Como Adicionar Uma Chave Pública ao Conjunto de Chaves

Após o PGP ter sido instalado em seu servidor e na máquina do destinatário, e o par de chaves necessários para o destinatário tiver sido criada, a chave pública do destinatário precisa ser adicionada ao conjunto de chaves publicas de usuários PHP. Se o PHP roda como um usuário "www" em seu servidor Web, este usuário precisa ter sua chave contida no grupo de chaves publicas. Os passos a seguir adicionam uma chave ao grupo de chaves do usuário:

1. Exporte uma versão ASCII da chave pública do usuário, seguindo os passos na documentação do PGP.
2. Atualize o arquivo de chaves publicas no diretório de usuários do PHP no servidor (/home/www/, por exemplo).
3. Efetue login em seu servidor via telnet ou SSH, ou navegue por ele e digite usando o teclado, se você tiver esta sorte.
4. Torne-se usuário PHP. Isto envolve o uso do comando su, como su www.
5. Adicione a chave ao grupo de chaves: pgpk –a /caminho/para/o_arquivo. Você verá a confirmação de que a chave foi adicionada ao seu grupo de chaves. Neste momento, pode deletar o arquivo texto contendo a chave pública.
6. Determine o nível de confiança para a chave: pgpk –e [keyname]
7. Selecione always trust (confiar sempre).
8. Teste este processo criando um arquivo de entrada contendo algo sem importância, como a linha "Eu desejo testar esta seqüência de criptografia".
9. Emita manualmente o comando para criptografar o arquivo de texte: pgp –r [keyname] –o [arquivo de saída] –a [arquivo de entrada]
10. Quando fosse for orientado a responder sobre a confiança de arquivo de chave, responda y.

11. O arquivo de saída conterá uma versão criptografada do arquivo de texto de entrada.

Se você segui estes passos, poderá ler os dados criptografados com PGP, usando seus scripts PHP para emitir os comandos.

7.2 Estatisticas de Utilização e Manutenção da HTML

Quando o volume de informações armazenadas no servidor se torna muito grande, é cada vez mais difícil verificar todas as informações, conferir se os hipervinculos operam de maneira correta e se todas os arquivos necessários foram vinculados ao seu servidor. À medida que cresce a quantidade de documento inter-relacionados, a única maneira pratica de realizar essa tarefa é usar programas automatizados que façam a verificação de todos os seus documento.

Itens importantes para uma ferramenta de análise de Logs:

- Como extrair e interpretar as informações dos logs e de erro do servidor.
- Quais são as ferramentas disponíveis para analise e representação gráfica das estatísticas de utilização.
- Quais são as ferramentas disponíveis para assegurar a integridade dos arquivos de HTML.
- Como encontrar automaticamente os arquivos novos e alterados no seu servidor

7.2.1 Os Logs de Utilização

Quando seu servidor da Web está em funcionamento, cada solicitação de documentos ou arquivo é registrada sob a forma de um item separado no arquivo de log do servidor. Como padrão, esse arquivo denomina-se “logs/Access.log”e se encontra sob o diretório principal do servidor, conforme está definido no arquivo de configuração do HTTPD. Os erros são registrados separadamente, no arquivo “logs/erro.log”. Os acesso e de erro são muito semelhante, mas iremos discuti-los em separado, por questão de clareza.

7.2.2. Os Formatos dos Logs

Todos os servidores importantes da Web, inclusive o do NCSA e o do CERN, produzem arquivos de log em um formato comum, para que você possa empregar os utilitário destinado a realizar a analise desse logs em qualquer tipo de servidor. O formato inclui quase as informações imagináveis a respeito de cada solicitação de documento, exeto o tempo de duração da transferência.

7.2.2.1 O Log de Acesso

A maior parte dos programas servidores possui um diretório padrão para armazenamento de arquivos de log, ou então lhe permite configurar o programa servidor com o objetivo de indicar onde os logs devem ser mantidos. No httpd Windows, os arquivos "ACCESSE.LOG"e "ERROR.LOG"ficam armazenados em um subdiretório /HTTPD, chamado"/LOGS". As informações contidas no log de acesso incluem:

- O endereço da cliente que solicitou o documento.
- A data e a hora precisa em que ocorreu a transferências.
- O método de HTTP e o protocolo usado na transferência.
- O caminho virtual até o documento transferido.
- O status da transferência.
- Quando bytes foram transferidos.

O fragmento a seguir foi extraído de um log de acesso gerado pelo httpd Windows:

```
s115.indonet.net - - [20/0ct/1994:20:53:17 – 0500]
->"GET / http/1.0"200 418
s115.indonet.net - - [20/0ct/1994:20:53:37 – 0500]
->"GET / httpdoc/ overview.html/1.0"200 3572
s115.indonet.net - - [20/0ct/1994:20:54:00 – 0500]
->"GET / httpdoc/setup/admin/Overview.html/1.0"200 1165
s115.indonet.net - - [20/0ct/1994:20:54:17 – 0500]
->"GET / httpdoc/setup/Configure.html HTTP/1.0"200 2500
s115.indonet.net - - [20/0ct/1994:20:54:27 – 0500]
->"GET / httpdoc/setup/Overview.html HTTP/1.0"200 1121
s115.indonet.net - - [20/0ct/1994:20:54:43 – 0500]
->"GET / httpdoc/setup/srm/Overview.html HTTP/1.0"200 1334
s115.indonet.net - - [20/0ct/1994:20:54:53 – 0500]
->"GET / httpdoc/setup/srm/Alias.htm HTTP/1.0"200 1191
s115.indonet.net - - [20/0ct/1994:20:55:20 – 0500]
->"GET / httpdoc/setup/access/Overview.html HTTP/1.0"200 3544
s115.indonet.net - - [20/0ct/1994:20:55:40 – 0500]
->"GET / httpdoc/setup/http/AccessConfig.html HTTP/1.0"200 1308
```

```
s115.indonet.net - - [20/0ct/1994:20:56:26 – 0500]
->”GET / httpdoc/setup/access/AuthType.html HTTP/1.0”200 1132
s115.indonet.net - - [20/0ct/1994:20:56:49 – 0500]
->”GET / httpdoc/setup/access/AddEncoding.html HTTP/1.0”200 721
s115.indonet.net - - [20/0ct/1994:20:56:57 – 0500]
->”GET / httpdoc/setup/srm/AddEncod.htm HTTP/1.0”200 1058
s115.indonet.net - - [20/0ct/1994:20:57:55 – 0500]
->”GET / httpdoc/setup/access/AddDescruption.html HTTP/1.0”200 709
s115.indonet.net - - [20/0ct/1994:20:58:07 – 0500]
->”GET / httpdoc/setup/srm/AdddDescr.htm HTTP/1.0”200 1270
s115.indonet.net - - [20/0ct/1994:20:58:39 – 0500]
->”GET / httpdoc/setup/Configure.html HTTP/1.0”200 2500
s115.indonet.net - - [20/0ct/1994:20:58:43 – 0500]
->”GET / httpdoc/setup/admin/Overview.html HTTP/1.0”200 1165
s115.indonet.net - - [20/0ct/1994:20:02:20 – 0500]
->”GET / httpdoc/setup/admin/AcessinggFiles.html HTTP/1.0”403 190
```

O primeiro item em cada entrada de log é endereço que solicitou o documento, seguindo pela data e hora, pelo método de HTTP (no exemplo, GET), pelo caminha virtual até o arquivo, pelo nível do protocolo HTTP (1.0, nesse exemplo), pelas informações sobre o status (200significa OK e pelo número de bytes transferido.)

A partir desses fragmentos de informações, é possível reunir uma grande variedade de estatísticas sobre a utilização do seu servidor, inclusive:

- Quais foram os documentos procurados com maior freqüência.
- Quais foram os períodos em que o servidor esteve mais ocupado (horas do dia, dias do mês, etc).
- O volume total de trafego em bytes (e a porcentagem da largura de banda da sua conexão) correspondente a qualquer período de tempo determinado.

Tendo em vista que todos os acessos aos documentos são registrados, os arquivos de log podem crescer com grande rapidez. Também contribui para esse crescimento o fato de que os arquivos GIF em linha constituem solicitações isoladas; assim, a solicitação de um documento com três GIFs em linha aparece, na realidade, como quatro solicitações – uma para o documento e três para os GIFs. Em servidor com uma ocupação moderada, o log de acesso pode chegar a mais de 10M em cada mês. Se você quiser conservar os dados Históricos de logs, é uma boa idéia compactar periodicamente o arquivo de log em uso e transferi-lo para um arquivo de armazenamento. Isso poderia ser feito no inicio de cada mês.

7.2.2.2 O log de Erro

O formato do log de erros é muito semelhante ao do log de acesso. Porem, em vez de informar o numero de bytes transferido, o log de erros informa o motivo o erro. O trecho a seguir foi obtido em um log de erros gerado pelo httpd Windows:

```
[20/Oct/1994:21:02:20  - 0500] httpd:Access  to
->c:/httpd/htdocs/setup/admin/AcessingFiles.htmal  failed
->for  s115. infonet. net.  reason: client denied by server
->configuration
[20/Oct/1994:21:07:53  - 0500]
->http: access to c:/httpd/htdocs/docs failed for
->s115. infonet. net. reason: file does not exist
[20/Oct/1994:21:08:13  - 0500]
->http: access to c:/httpd/htdocs/ failed for s115. infonet. net.
->reason: client denied by server cofiguration
[20/Oct/1994:21:10:01  - 0500]
->http: access to c:/httpd/htdocs/docs failed for
->s115. infonet. net. reason: file does not exist
[20/Oct/1994:21:10:25  - 0500]
->http: access to c:/httpd/htdocs/. index. html failed for
->s115. infonet. net. reason: file does not exist
[20/Oct/1994:21:10:35  - 0500]
->http: access to c:/httpd/htdocs/. index. html failed for
->s115. infonet. net. reason: client denied by server configuiration
[20/Oct/1994:21:11:13  - 0500]
->http: access to c:/httpd/htdocs/. index. html failed for
->s115. infonet. net. reason: client denied by server configuiration
[20/Oct/1994:21:11:44  - 0500]
->http: access to c:/httpd/htdocs/docs failed for
->s115. infonet. net. reason: file does not exist
[20/Oct/1994:21:15:42  - 0500]
->http: access to c:/httpd/htdocs/docs failed for
->s115. infonet. net. reason: file does not exist
[20/Oct/1994:21:17:01  - 0500]
->http: access to c:/httpd/htdocs/ failed for s115. infonet. net.
->reason:client denied by server configuration
[20/Oct/1994:21:18:31  - 0500]
->http: access to c:/httpd/htdocs/demo failed for
->s115. infonet. net. reason: client denied by server configuration
[20/Oct/1994:21:18:36  - 0500]
->http: access to c:/httpd/htdocs/demo/ index.htm failed for
->s115. infonet. net. reason: client denied by server configuration
```

O formato do arquivo é bastante claro. A primeira da linha indica a data e a hora do erro. A se segunda parte do log indica que o cliente estava tentando obter acesso quando ocorreu o erro. A terceira parte da entrada do log explica o motivo do erro.

Os logs de erros são valiosos por mostrarem tentativas de acesso a documentos controlados feitas por usuários não autorizados, por relatarem problemas do servidor. Se

forem acompanhados com freqüências, os logs de erros podem ser o seu primeiro indicio de que um hipervinculos foi “quebrado” por falar um documento, ou então por que o documento foi movido para outra localização. Se encontrar varias tentativas malsucedidas de conexão relacionadas com o mesmo documento não existir, você pode localizar o hipervinculos rompido (ou um vinculo omitido) pelo exame do mesmo período no log de acesso a fim de observar o vinculo de onde o cliente tentava alcançar o arquivo.

Se tudo funcionara relativamente bem, a velocidade de crescimento do seu log de erro não chegara nem perto da rapidez com que o log de acesso irá aumentar; dessa forma, não é tão importante compactá-la em um arquivo de armazenamento para conversar o espaço na unidade de disco.Contudo, se houver documento protegido no sue servidor, talvez não seja má idéia guardar o log de erros, pois ele poderá ser necessário para ajudar a determinar a origem de problemas de segurança que possam ser descobertos mais tarde.

7.3 O Exame Minucioso dos Dados Utilização

O arquivo de log de acesso registro da atividade do seu servidor, mas é bastante complicado de significativo das dados brutos. No entanto, usando algumas pesquisas simples, você pode encontrar vários itens de que precisa, sem ter de escrever sequer uma linha de código. Existem diversas ferramentas básicas de pesquisas destinadas aos iniciantes, tanto no UNIX quanto no DOS.

7.3.1 A Pesquisa no UNIX

Em uma estação de trabalho UNIX, a maneira mais simples de realizar a pesquisa em um arquivo de texto é usar o consagrado utilitário grep.Por exemplo, para encontrar no log de acesso todas as entradas que contem grep apresenta muitos recursos poderosos, que vão além da realização de pesquisas básicas. Como exemplo você poderia procurar todas as linhas, exceto as que contivessem becky, usando a opção – v:

```
grep  - v ‘bectky’  Access.log
```

Outras opções úteis de grep incluem:

- -i Desativa a distinção entre letras maiúsculas e minúsculas durante a execução da pesquisa.

- -c Retorna apenas uma contagem de todas as linhas que coincidem com o argumento da pesquisa.
- -n Exibe o numero de cada linha que satisfaz a pesquisa.

Também pode-se empregar varias opções para ajudar a limitar sua pesquisa, bastando juntar diversos flags de opções depois dos traços. Por exemplo, se quisesse procurar todas as linhas que não contem 'becky'e preferisse desativar a distinção entre maiúsculas e minúsculas durante a pesquisa de 'becky', você digitaria

```
grep  - vi 'becky' accesse.log
```

7.3.2 A Pesquisa no DOS

O Comando FIND do DOS tem praticamente a mesma função do comando GREP do UNIX. S e quiser todas as ocorrências de NASA.GOV no log de acesso digite:

```
FIND "NASA.GOV" Access.LOG
```

8. Evolução das Ferrametas de Monitoramento em Relação aos Dados Históricos de Performance

Em relação aos seus recursos de persistência, podemos dividir a evolução das ferramentas de monitoramento em cinco fases.

Na "pré-história" das ferramentas de monitoramento de performance, não havia persistência. O máximo que um administrador de sistemas podia contar era com uma ferramenta para exibir na tela algumas informações sobre a performance atual. A ferramenta *top*, utilizada amplamente no Unix, é dessa fase.

Nessa fase, o ajuste de desempenho é um conjunto de técnicas empíricas, embasadas apenas nas práticas bem sucedidas de administradores experientes e nas instruções dos fabricantes de software e hardware que, por conhecerem internamente os seus sistemas, dão por vezes informações valiosas para a otimização. Nessa fase, a Performance and Tuning não é uma atividade constante, mas uma atividade que só é realizada no inicio do funcionamento do sistema ou quando alguma situação excepcional ocorre.

Na segunda fase o histórico é armazenado em um arquivo. Na falta de soluções típicas de bancos de dados como índices e linguagens de consulta, a única vantagem da persistência é permitir que os dados possam ser exibidos na ferramenta de monitoramento muito tempo depois da sua obtenção. A ferramenta "Performance Monitor", parte integrante dos sistemas operacionais de rede Windows, é um exemplo de uma ferrameta dessa fase.

A terceira fase é a fase em que as ferramentas implementam bancos de dados proprietários com funcionalidade reduzida. A característica principal desses bancos de dados é que eles estão amarrados às ferramentas do sistema de monitoramento. Não é possível utilizar uma ferramenta genérica de consulta ou de geração de relatório ou mesmo, uma ferramenta OLAP para fazer a exploração dos dados. Também é muito difícil de integrar as informações de performance com outras bases de dados. Exemplos de ferrametna dessa fase incluem Perfman Analyst e Sysload.

A quarta fase, representada pela ferramenta Webmatcher, utiliza um SGBD de uso geral para guardar dados coletados de vários sistemas.

E a quinta fase é a fase do Data Warehouse genérico. Em toda bibliografia consultada não conseguimos descobrir se existe ainda uma ferrametna de monitoramento de desempenho que possua um Data Warehouse com as características desejadas.

Tabela 8.1 Fases da evolução de ferramentas de monitoramento

FASES	FERRAMENTAS
1ª – Nenhuma persistência	Top
2ª – Persistência em arquivo	Performance Monitor
3ª – Persistência em um SGBD especializado	Sysload, Perfman Analyst, Perfstat e Perfalert
4ª – SGBD genérico	Webwatcher
5ª – O DW Performance	Ainda não há

Fonte: Webmatcher home page; Sysload home page; Perfman Analyst; Windows NT server resource Kit.

8.1 IMPLEMENTAÇÃO E ESTRATÉGIAS PEDAGÓGICAS UTILIZADAS PARA CONFECÇÃO DE UM MODELO DE FERRAMENTA PARA MONITORAMENTO DE DOCUMENTOS WEB

A concepção Construtivista é usada neste momento, pois estimula a participação dos aprendizes em todo ciclo de vida do programa. Assim os participantes (alunos e professores) podem não somente explorar o ambiente, navegador, efetuando praticas e experimentação, mas também opinar sobre conteúdo, refinar o programa, escrever códigos de simulação, testar e efetuar a manutenção das partes desenvolvidas ou em desenvolvimento, no local do ambiente em aprendizagem colaborativa. Todavia, isto é altamente dependente da parte presencial, ou seja, dependerá da dinâmica do processo de Ensino.

Ao entrar em um ambiente, o aprendiz que já domina bem o assunto, poderá ser capaz de, em cerca de 20 a 30 minutos, percorrer todo trajeto (que inicia no nó unidade 1 e termina no nó unidades 8), tipo “espinha dorsal”. Neste tempo ele terá uma visão geral do conteúdo abordado não entrando em detalhes e nem fazendo experimentações. Para percorre-lo completamente atingindo todos seus estados, calcula-se que ele deverá dispor de cerca de 10 horas em média, caso o assunto seja completamente explorado.

O “controle pedagógico” é efetuado por uma seqüência de nós. As funções de sistema resultante não são pré-fixadas e podem ser facilmente adaptadas de acordo com as

necessidades de autoria ou de situações de aprendizagem necessárias. As funções de sistema resultantes não são pré-fixadas e podem ser facilmente adaptadas de acordo com as necessidades de autoria ou de situações de aprendizagem necessárias. As informações de alto nível tais como perguntas, manipulações de erros, comportamento de aprendiz, são ainda objeto de estudo.

A título de simplificação da Figura 8.1 as linhas pontilhadas mostram os arcos que ligam-se a outros nós (mostrados na ponta das setas). Por exemplo, os nós 2.2 e 4 foram omitidos porque são funcionalmente iguais ao nó 2. A descrição do grafo é sumarizada a seguir:

- nó apresentação – o nó 1 apresenta o ambiente e oferece informações gerais sobre a autoria, objetivos, navegação, recursos, endereço. Está ligado ao nó que possui nó de detalhamento e retorno, a nó de recursos locais Lc e ao nó de recursos externos Ne. O estado inicial $X_0 = 1$. A entrada u = "acionar o nó 1" conduz como saída y_1 que é a apresentação de uma página introdutória sobre o ambiente. E assim as transições de estado vão ocorrendo de acordo com a manuseio do ambiente pelo aprendiz.

  A definição dos objetivos e das utilidades são sustentados pelas teorias de aprendizagem de Thorndike, Guthrie apud Hergenhan e Skinner. Na aprendizagem via rede, se o aprendiz não for capaz de identificar claramente os objetivos do conteúdo abordado pode transferir seus interesses para outro local.

- Mp – nó mapa – oferece um mapa completo de ambiente. Através do nó mapa todos os estados são atingíveis. Este nó é muito importante para evitar que o aprendiz se confunda ou se perca no ambiente. Funcionalmente, acessar um nó mapa pode ser comparada ao fato de "folhear um livro rapidamente para ver seu conteúdo do principio ao fim".

- 1, ... 8 – nós unidades – Nestes nós são apresentadas as unidades com o assunto diversos sobre as RNA. No ambiente desenvolvido, tais unidades possuem ligações a nós de detalhamento, recursos internos e externos, entre outros. Em qualquer destes nós o aprendiz pode atingir nó de unidades e nós

subtópicos através do nó mapa. Os nós de detalhamento não foram colocados no nó mapa para reduzir os recursos exigidos a oferecer um certo controle na navegação.

Skinner desenvolveu o aprendizado programado. Estes conceitos foram aplicados durante muito tempo em sistemas de instrução programada. Muitos pesquisadores em Sistemas de Ensino à Distância Computadorizados afirmam que as redes de computadores são ambientes abertos e assíncronos que permitem aos estudantes aprenderem em seu próprio ritmo. Um certo cuidado deve ser tomado para que o aprendiz não tenha um ritmo lento, ou muito acelerado, em relação ao seu grupo de interação. Isto é essencial, pois o ambiente aqui considerado não é um sistema de ensino totalmente à distância.

- 2.1, ..., 7.8 – nós tópicos e subtópicos. Os nós tópicos são uma extensão dos nós unidades. Os nós subtópicos contêm exercícios e ligações para os nós de simulação.

- A, ..., P – nós de detalhamento de tópicos e subtópicos. Caso o aprendiz deseje saber um pouco mais sobre determinado tópico, poderá acessar tais nós que contêm informações adicionais ou mesmo ligações para nós externos na rede com endereços de assunto relacionados na Web. Por exemplo, uma linha de tempo, no nó 2.1, possui ligações que levam aos nós A, ..., P contendo informações sobre pioneiros no estudo das RNA. Por exemplo, o nó K oferece detalhes sobre a pessoa de Marvin Minsky na história da IA. Alguns destes nós podem se ligar a nós externos, como no caso de Minsky, para um no contendo seus trabalhos no MIT.

A aplicação da teoria de Thorndike na educação pressupõe que o aprendizado procede do simples ao complexo. Tratando-se de um ambiente que irá apoiar o Ensino RNA em cursos, constituídos de partes presenciais e não presencias, é importante a definição do público – alvo, além da apresentação de conceitos gerais aos mais complexos que poderão ser incorporados com o tempo e a utilização do ambiente.

- Dt – nó de detalhamento de unidade, O Dt oferece um detalhamento do nó 3. Possui ligação a nós externos da Web. Se o aprendiz quizer saber mais informações sobre as características gerais que não são relevantes ao assunto principal pode acionar o “mouse” para atingir tal nó e retornar através do nó de retorno Re.

  Os excessos devem ser evitados. A teoria de Hull apud Hergenhahn observa que a sobrecarga e a fadiga cognitiva podem romper totalmente o processo de ensino-aprendizagem. Problemas de ordem tecnológicas tais como, dificuldades de conexão, complexidade de recursos computacionais, mudanças tecnológicas regulares, entre outros, também podem contribuir para desestimular o aprendiz. Caso ele não encontre um nó externo pode ficar frustrado. Caso hajam muitos nós de detalhamento, ele pode se cansar rapidamente.

- Bb – nó de bibliografia. Os nós bibliográficos oferecem a base bibliográfica utilizada nos assuntos abordados. Nota-se no grafo que estes nós estão ligados a todos os nós unidades 2 a 8.

- Gi – nó glossário. Como em um livro, se o aprendiz quiser saber o significado de alguma palavra considerada incomum ou própria da linguagem elaborada, poderá acessar o nó GI e retornar aos nós 2 a 8.

- Lc – nó local. Este nó contém recursos da rede interna do INE, isto é, no servidor, onde podem estar arquivos de dados, listas, acesso ao instrutor, horários de aulas, trabalhos, programas, paginas locais, entre outros.

  Engajar os aprendizes em grupos de discussões para que possam desenvolver estratégias de resolução de problemas tem suas raízes na Teoria de Tolman, cuja principais diretrizes podem ser encontradas em Hergenhahn. A composição de grupos e listas de discuções nas redes facilitam esta estratégia.

- Rc – nó recursos. Este nó pode ser ligado a diversos nós, tais como arquivos disponíveis (interna ou externamente), nós para listas de discussões, correio eletrônico, painel de horários, informações do instrutor, que deverão ser incorporados de acordo com as necessidades observadas. No momento, o nó recurso liga-se ao nó mapa, ao nó recursos locais e nós externos.

- Mm – nó memória. O objetivo do nó memória é oferecer um mapeamento de nós visados anteriormente. Uma opção de utilização deste nó pode ser a colocação de marcas ("bookmarks") semelhantes aos livros, no qual o aprendiz pode deixar o sistema e quando voltar verificar onde estava. Este tipo de nó,assim como o nó mapa, é interessante como guia de navegação.

- Ne – nó externo para a Web. Permite ao aprendiz obter recursos disponíveis na Web com ligações para mecanismos de busca, bibliotecas virtuais, repositórios de artigos, programas acadêmicos e de domínio público (conectores, utilitários, etc.), simuladores, endereços de empresas de programas comercias e tópicos relacionados.

  Deve-se evitar a colocação excessiva deste tipo de nó por dois motivos. Primeiro, o aprendiz pode dispersar-se para outros endereços da Web ou perder-se no espaço de informações. Segundo, alguns locais de rede podem ser ou tornar-se (devido a constantes atualizações) incompatíveis com o navegador utilizado, então os erros causados podem interferir na navegação do aprendiz.

- Pr – nó prática. O aprendiz pode praticar e testar seus conhecimentos através de exercícios (que contém nó erro). A prática envolve também a simulação. As práticas mais simples englobam opções a serem selecionadas por questões de simplicidade. Por exemplo, uma opção também é colocar no nó 2 ou nó Rc arquivos com questões para serem respondidas e enviadas ao instrutor individualmente ou através de um nó Fórum que poderá ser incorporado.

Em teorias como de Guthrie, Bandura, Thorndike apud Hjergenhahn as praticas causam estímulos que provocam o comportamento desejado. Em modelagens comportamentais de instruções programada com fundamentos "Skinnerianos" a prática é utilizada ao longo dos anos. Na caso deste trabalho, observa-se que a abordagem mais adequada é a que permite a construção conhecimento através de experiências.

- Er – nó erro – Serve para verificação de resposta em exercício. Ao estudar conceitos no nó 5, se o aprendiz não acionar um nó opção de resposta definido no sistema, convencionado como resposta correta será retornado ao nó 5.

Para muitas aplicações de teorias de Thorndike e Bandura apud Hergenhahn as avaliações regulares e auto-avaliação podem ser meios de verificar se o aprendiz observou o conteúdo. Alguns programas de instrução programada baseados em teorias comportamentais enfatizam a observação do comportamento do aprendiz para avalia-lo, não considerando assim os aspectos pscologigos envolvidos no processo. As avaliações em um ambiente via rede de computadores devem ser diferentes das praticas convencionais. As práticas avaliativas, neste caso, devem enfatizar mais a participação e a integração do aprendiz ao ambiente na rede.

Os estímulos por recompensa ou motivação foram considerados essenciais ao processo de aprendizagem. Teóricos com Bandura apud Hergenhahn, Skinner e Hebb consideram estes estímulos sob diferentes pontos de vista. Hebb sustenta que as característica físicas do ambiente de aprendizado são muitos importantes. Para qualquer tarefa dada e para qualquer estudante existe um nível ótimo de interesse que permitirá um aprendizado mais eficiente. Desde que o nível de interesse é controlado primeiramente por um estimulo externo, o nível de excitação no ambiente de aprendizado determinará como este se efetuará. Se existe bastante estimulação o aprendizado será dificultado. E se o contrario, existir pouca estimulação o aprendizado também será dificultado.

Um ponto interessante na teoria de Bandura é a ênfase dada aos processos motores, processos de atenção, retenção e de motivação dos estudantes.

Assim, observa-se que, de acordo com esta teoria, filmes, televisão, leitura, transparências e outras mídia podem ser usada numa ampla variedade de experiências educacionais. Maiores detalhes sobre a Teoria de Bandura podem ser encontrados em Hergenhahn.

A utilização do método Lúdico pode ser uma estratégia alternativa, oferecendo estímulos por motivação similares e alguns jogos de computador. Sabe-se que gráficos, animações, sons, filmes e outras mídias são formas de apresentação do conteúdo estimulantes. Todavia, sabe-se que outras formas de estímulos devem ser considerados ao longo do processo, principalmente a cooperação entre os atores envolvidos no ensino-aprendizagem.

- Re – nó retorno simples. O aprendiz retorna ao nó anterior. Não possui memória. São geralmente feitos com ligações do botão de "voltar" ou com o próprio navegador. Sendo que este último parece ser mais interessante pois reduz os recursos computacionais oferecidos pela "cachê" do navegador.

- Rm – nó retorno com memória. Armazena os caminhos percorridos e as ações executadas pelo aprendiz. Este tipo de nó oferece uma vantagem em relação ao nó de retorno simples pelo fato de que pode agir como um "marcador de passos". Pode ser implementado com rotinas em Java mas em alguns sistemas é possível haver problemas com memória e o programa trave a máquina. Isto é muito comum em sistemas denominados "webmails"

- Sm – nó simulações. O aprendiz desenvolve simulações, após atingir o nó Pr ou ir diretamente no nó 6.1, que aborda os principais algoritmos de aprendizado das RNA. As simulações disponíveis no momento são executadas por programas obtidos em nós Ne, em código já testados em diversas máquinas.

As experimentações remotas, desenvolvidas com facilidade em linguagens de roteiros que utilizem a metáfora cliente – servidor, oferecem amplas possibilidades na criação de ambiente exploratórios com simulações quantitativas e qualitativas. As simulações podem ser feitas localmente ou remotamente. O aprendiz pode acessar o nó Rc e obter simuladores que

podem ser rodados localmente em horário que lhe convier, ou então pode executa-las “on-line”.

Um exemplo de nó simulação conduz o aprendiz a desenvolver simulações em Java com Rede de Kohonem. Neste caso, uma Rede de Kohonem executa o mapeamento de determinados pontos (valores de entrada de rede) no espaço tridimensional no menos percurso sem cruzar duas vezes o mesmo ponto. Este problema é similar ao clássico problema de Caixeiro viajante (no qual deve ser encontrado o menor percurso entre um certo número de cidades, sem que se passe duas vezes pela mesma cidade. As classes utilizadas são: número de neurônios (diferentes camadas de neurônios que podem ser conectadas) e matriz de pesos entre duas camadas de neurônios.

O aprendiz executa esta experimentação em cerca de dois minutos. Uma primeira análise sobre este sistema demonstrou ser satisfatória para os propósitos didáticos.

Outras implementações estão sendo elaboradas para serem colocadas no ambiente considerando fatores como simplicidade e o tempo de execução. Uma simulação que seja lenta pode desestimular os estudantes e ser economicamente inviável.

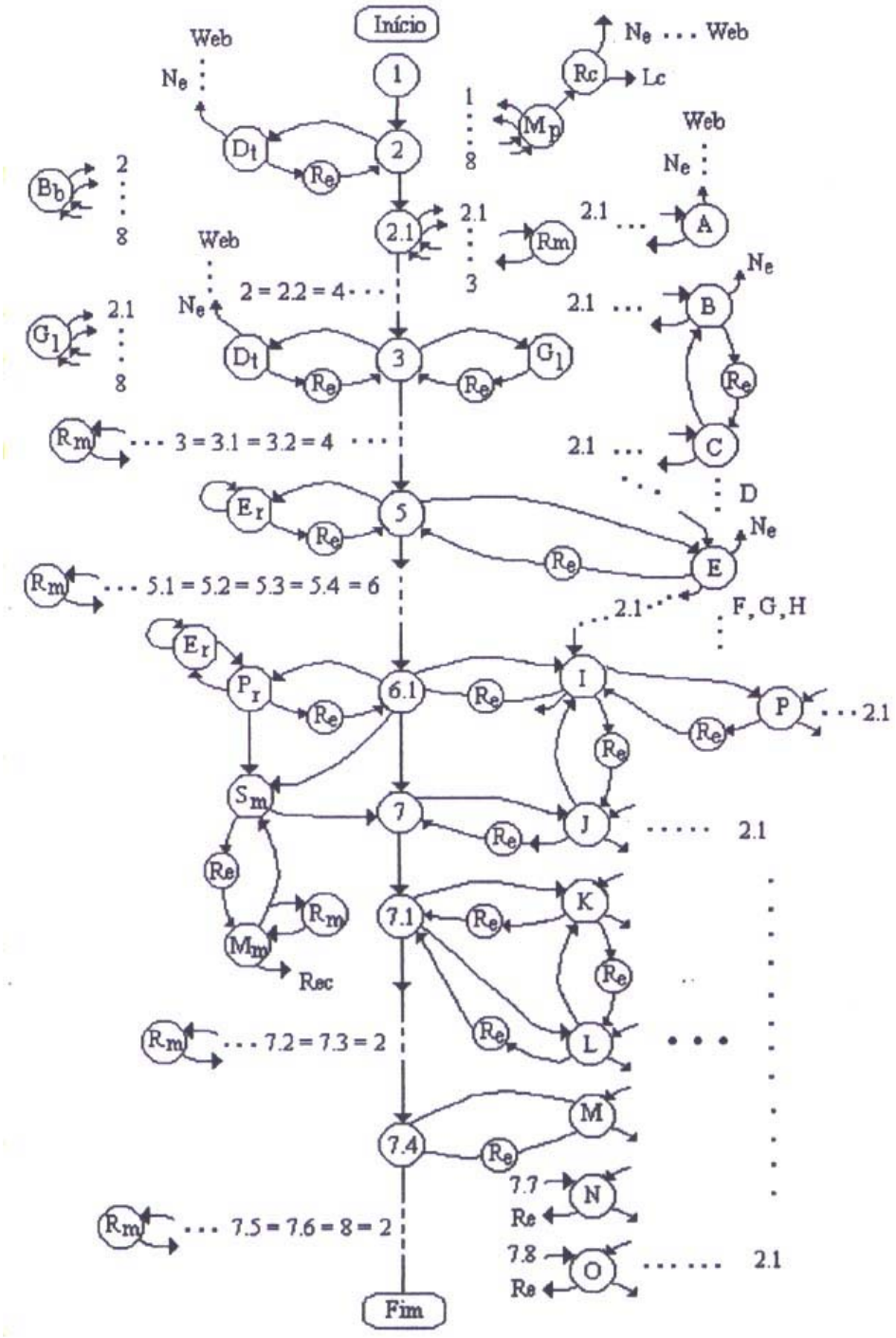

Figura 8.1 Grafo do hipertexto – Fonte: Barreto, J. M. – 2001

9. Conclusões

Vê-se neste trabalho que a administração de serviços Web é quase toda baseada em critérios empíricos. Os dados históricos sobre o funcionamento desses serviços raramente são utilizados ou, quando o são, quase nunca de maneira mais efetiva. Contribui para isso a ausência de tecnologias para guardar o histórico do funcionamento de um serviço Web e permitir a posterior análise desses dados.

A necessidade de se ter uma visão clara não somente dos objetivos que se deseja atingir, mas também dos perfis dos usuários que irão acessar um determinado site e a maneira pela qual eles estarão navegando nas diversas páginas que formam o site, deve ser o objetivo que o projetista deste site deve atingir.

Então, deve-se ter uma visão e uma análise de como o site é percorrido pelos visitantes os quais podem fornecer pistas inestimáveis de como ele esta atendendo os requisitos a que se propõe.

Como vimos pelos estudos realizados neste trabalho, as várias ferramentas aqui abordadas e suas inúmeras funcionalidades, se propõe a dar referências e subsídios que permitam a projetistas de sites Web avaliarem como serão construídas as páginas e como referência-las de uma maneira que as decisões possam ser apropriadas em cima de uma estrutura ou topologia que se queira utilizar (SPILIOPOULOU, 2000).

Percebeu-se que pela grande variedade de Logs existente, os quais não são padronizados, uma das melhores maneiras de se fazer um monitoramento bastante eficiente é usar padrões que estão melhor definidos pela Inteligência Artificial, como por exemplo a mineração de dados que pode fazer uma análise de todos os dados brutos que são gerados nos Logs dos servidores Web. Percebe-se ainda, que esses Logs possuem uma série de limitações, decorrentes, entre outras coisas, do próprio fato de ser o HTTP um protocolo sem estado, não guardando informações sobre as sessões.

Para contornar-mos estes problemas, deve ser feito um pré-processamento dos dados brutos, que permita a limpeza dos registros indesejados e a identificação dos usuários e sessões, com uma eventual agregação dos dados em unidades mais semânticas, como as transações. No entanto, ainda assim, alguns questionam a validade dos Logs como uma maneira confiável de se analizar a utilização de um site.

Referências Bibliográficas

ALCÂNTARA, Andréia Almeida de. **Home Pages: Recursos e Técnicas Para Criação de Páginas na WWW**/Andréia Almeida de Alcântara, Carlos Santos de Figueira Filho, Cibele César do Amaral Brasil. Rio de Janeiro: Campus, 1997

BARRETO, J. M. **Inteligência Artificial no Limiar do Século XXI**, 3ª ed. Duplic, Florianópolis, SC, 2001.

BEVERIDGE, Tony. **Programação de Alto Desempenho na Web com ISAPI/NSAPI**/Tony Beveridge, Paul McGlashan. Tradução: Marcos J. Pinto. São Paulo: ed Berkeley, 1998.

CHANDLER, David M. **Como Montar o seu site na World Wide Web**/David M Chandler; Tradução de Publicare Consultoria e Serviços, Rio de Janeiro: Campus, 1996

HELD, Gilbert. Comunicação de Dados. Tradução da 6ª ed. Original de Vandenberg Dantas de Souza. Rio de Janeiro: Campus, 1999.

LAQUEY, Tracy; RYER, Jeanne C. **O Manual da Internet**; Tradução Insight Serviços de Informática. Rio de Janeiro: Campus, 1994.

LOSHIN, Pete. **Big Book of World Wide Web RFCs**; ed Morgan Kaufmann, 2000.

MARTIN W. Murhammer; ATAKAN O.; BRETZ S.; PUGH L. R.; SUSUKI K.; WOOD David H. **TCP/IP – Tutorial e Técnico**. Tradução: Jussara Licinia Souza Gaertner, Lavio Pareski; revisão técnica: Álvaro Antunes. São Paulo: Makron Books, 2000

MELONI, Julie C. **Fundamentos de PHP**. Rio de Janeiro: Editora Ciência Moderna Ltda, 2000.

PERFMAN Analyst. Disponivel em http://www.infosysman.com (15/06/2003).

RFC 1630 Requests For Comments: Universal Resource Identifiers in WWW. Disponivel em http://www.ietf.org/rfc (16/07/2003).

RFC 1736 Functional Recommendations For Internet Resource Locators. Disponivel em http://www.ietf.org/rfc (22/07/2003).

RFC 1738 Requests For Comments: Uniform Resource Locators (URL). Disponivel em http://www.ietf.org/rfc (14/06/2003).

RFC 1808 Requests For Comments: Relative Uniform Resource Locators. Disponivel em http://www.ietf.org/rfc (28/07/2003).

RFC 1945 Requests For Comments: Hypertext Transfer Protocol -- HTTP/1.0. Disponivel em http://www.ietf.org/rfc (11/05/2003).

RFC 2191 Requests For Comments: HTTP State Management Mechanism. Disponivel em http://www.ietf.org/rfc (13/07/2003).

RFC 2141 Requests For Comments: URN Syntax. Disponivel em http://www.ietf.org/rfc (15/04/2003).

RFC 2145 Requests For Comments: Use And Interpretation Of HTTP Version Numbers. Disponivel em http://www.ietf.org/rfc (21/05/2003).

RFC 2227 Requests For Comments: Simple Hit-Metering And Usage-Limiting For HTTP. Disponivel em http://www.ietf.org/rfc (27/05/2003).

RFC 2276 Requests For Comments: Architectural Principles Of Uniform Resource Name Resolution. Disponivel em http://www.ietf.org/rfc (22/07/2003).

RFC 2396 Requests For Comments: Uniform Resource Identifiers (URI): Generic Syntax. Disponivel em http://www.ietf.org/rfc (07/08/2003).

RFC 2483 Requests For Comments: URI Resolution Services Necessary For URN Resolution. Disponivel em http://www.ietf.org/rfc (19/06/2003).

RFC 2518 Requests For Comments: HTTP Extensions For Distributed Authoring – Webdav. Disponivel em http://www.ietf.org/rfc (12/07/2003).

RFC 2594 Requests For Comments: Definitions Of Managed Objects For WWW Services. Disponivel em http://www.ietf.org/rfc (18/06/2003).

RFC 2609 Requests For Comments: Services Templates And Service: Schemes. Disponivel em http://www.ietf.org/rfc (23/07/2003).

RFC 2611 Requests For Comments: URN Namespace Definition Mechanisms. Disponivel em http://www.ietf.org/rfc (21/07/2003).

RFC 2616 Requests For Comments: Hypertext Transfer Protocol – HTTP/1.1. Disponivel em http://www.ietf.org/rfc (17/05/2003).

RFC 2617 Requests For Comments: HTTP Authentication: Basic and Digest Access Authentication. Disponivel em http://www.ietf.org/rfc (09/07/2003).

RFC 2717 Requests For Comments: Registration Procedures For URL Scheme Names. Disponivel em http://www.ietf.org/rfc (10/06/2003).

RFC 2718 Requests For Comments: Guidelines For New URL Schemes. Disponivel em http://www.ietf.org/rfc (16/05/2003).

SOARES, Walace. **Programando em PHP: Conceitos e Aplicações**. São Paulo: Érica 2000.

SPILIOPOULOU, M., 2000, **Web Usage Mining for Web Site Evalution**, Comunication Of The ACM, v.43, m. 8 (Aug).

STOUT, Rick. Dominando a Word Wide Web. Tradução João Eduardo Nobrega Tortello. São Paulo: Makron Books, 1997.

SYSLOAD home page. Disponivel em http://www.sysload.com (28/05/2003)

WEBWATCHER home page. Disponivel em http://www.landmark.com (17/06/2003)

WEISSINGER, A. Keyton. ASP: O Guia Essencial. Tradução da 2ª ed. Original de Edson Furmankiewicz, Joana Figueiredo. Rio de Janeiro: Campus, 2000.

WINDOWS NT Server 4.0 Resource Kit, Microsoft Press, 1996.