MODELAGEM DE HIPERMÍDIA
Prof. Giangiacomo Ponzo Neto
Parte Teórica – I
Obs: o texto pode conter trechos da wikipedia

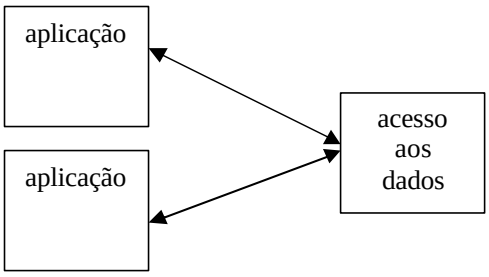
Aplicações em Camadas

A divisão de um programa em camadas se refere à organização de funcionalidades de forma distinta. Cada camada lida com tarefas específicas e oferece formas de conexão com as camadas adjacentes. Por exemplo: um modelo em 3 camadas pode se referir a separar a interface com o usuário, as regras de negócio e a conexão com o banco de dados. Uma coisa importante é não confundir as camadas com o meio físico onde se encontram. A divisão é lógica, ou seja, podemos ter *n* camadas rodando em um único computador (como fizemos até agora usando o NetBeans como ambiente de execução), embora a divisão em camadas permita a separação das tarefas entre vários computadores em uma rede distribuída, eventualmente melhorando o desempenho do sistema e reduzindo o eventual "custo" das mudanças em alguma camada (no banco de dados, por exemplo).

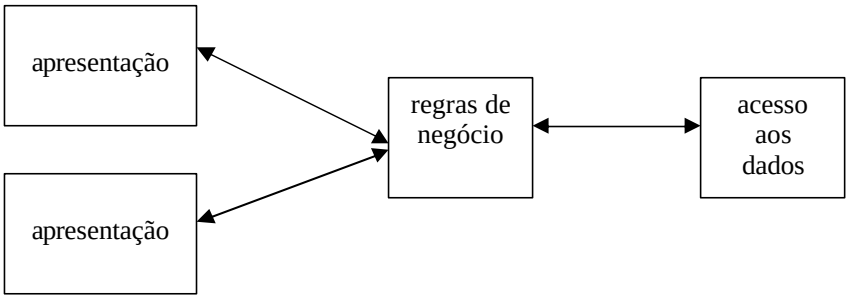
Algumas formas comuns de aplicação em camadas são as de 2 camadas em desktop no modelo cliente-servidor básico (interface e regras de negócio no cliente, banco de dados no servidor) e 3 camadas em aplicações *web* (interface com o usuário no *browser*, regras de negócio em um servidor, e banco de dados em outro servidor). Eventualmente, para que cada camada desta arquitetura se torne mais "desacoplada" das demais, algumas tecnologias de *middleware* podem ser utilizadas, por exemplo, CORBA, Web Services ou RMI e demais tecnologias conforme visto na disciplina de Sistemas Distribuídos.

Historicamente, até a década de 1980, tínhamos a arquitetura centralizada em *mainframes* ou aplicativos simples monousuário em micro-computadores onde a linguagem (Basic, Pascal, dBase, Clipper) concentrava toda a atividade em uma única camada.

Após o "*boom*" dos PCs, surgiu a idéia de dividir tarefas entre os micros e um computador servidor para aproveitar melhor o poder computacional de cada máquina. Estes primeiros sistemas cliente-servidor eram do tipo *desktop* de duas camadas (o cliente com a interface gráfica e as regras de negócio e o servidor com um banco de dados). Ferramentas como o Delphi, o PowerBuilder e o Visual Basic dominaram esse modelo de aplicação nos anos 90.



A arquitetura cliente-servidor de 2 camadas (com cliente "gordo" – assim chamado porque o programa a ser instalado deve se preocupar com a apresentação e a lógica, aumentando seu tamanho), embora muito melhor que as arquiteturas anteriores, sofre de problemas de manutenção, pois cada mudança na aplicação exige nova instalação. Foi quando surgiu a idéia de separar as aplicações em 3 camadas, que, a princípio, seriam: interface com o usuário (apenas a camada de apresentação, gerando um cliente "magro"), regras de negócio (lógica) e acesso aos dados (camada de dados).



A principal vantagem dessa arquitetura é que os problemas de manutenção são reduzidos, pois mudanças nas camadas de aplicação e de dados não necessitam de novas instalações nos clientes. Ainda assim, como essa arquitetura nasceu no mundo *desktop*, a instalação inicial dos programas continua sendo necessária, bem como a manutenção quando ocorrem mudanças na camada de apresentação.

A próxima arquitetura que surge (via *web*), resolve todos os problemas anteriores, pois usa o *browser* como camada de apresentação, o que dispensa qualquer instalação nova nas máquinas clientes. Eventualmente, a camada intermediária (aplicação) passou a ser dividida, mas a idéia continua a mesma e a maior vantagem é justamente o cliente "magro". O modelo *web* pode ter 4 camadas, sendo o *browser* responsável pela apresentação, o servidor *web* com as páginas, o servidor de aplicações com regras de negócio/classes/persistência e o banco de dados no fim da linha.

As soluções mais utilizadas hoje no desenvolvimento de sistemas (ASP.NET, PHP ou JSP no modelo de *n* camadas, com o usuário tendo apenas um navegador instalado na sua estação de trabalho) têm alguns custos, no entanto: os programas *desktop* têm, em geral, mais recursos de apresentação do que os *browsers*. Isso tem melhorado ao longo do tempo, mas ainda hoje persiste. Além disso, o desenvolvimento *desktop* conta, em muitos casos, com ferramentas mais poderosas para sua criação; muitas vezes o desenvolvimento *web* é mais "manual" do que estamos acostumados no mundo *desktop*. Outro ponto é a maior lentidão em usar a *web* comparativamente a um sistema *desktop* instalado em sua máquina, pois partes do programa que estariam disponíveis no seu HD tem que ser recebidas via *web*. Para uma discussão interessante (e longa) sobre este tema (em inglês) acesse a página "Are Web Interfaces Good Enough?" em http://www.codinghorror.com/blog/archives/000815.html .

As camadas mais comumente encontradas em um sistema de várias camadas são as seguintes:

- Apresentação ou Interface com o Usuário: camada que é executada no computador cliente e que contém as telas de acesso (no *browser*, se for um sistema *web*).
- Lógica ou Regras de Negócio: São as regras que determinam de que maneira os dados serão validados ou utilizados. Qualquer cálculo ou preparação de dados para escrita ou leitura pode estar nesta camada. Pode estar fisicamente em um servidor de aplicações (uma máquina intermediária), de forma que, quando uma regra do negócio for alterada, basta atualizá-la nesta máquina.
- Persistência: como vimos, é responsável por persistir os objetos no formato relacional do banco de dados.
- Dados: no servidor de Banco de dados, onde residem os dados do sistema.

Discussão em sala

- Sistemas *web* x *desktop*.

Hipertexto e Hipermídia

Optamos por começar a disciplina pela parte prática por questões técnicas e didáticas, principalmente para que não houvesse um longo tempo sem programação, evitando que muitos alunos perdessem o "fio da meada" entre LP6 e MH. Agora estamos no caminho teórico da disciplina. Faremos o possível para que continue interessante e, sempre que der, citaremos aplicações práticas do que estivermos vendo.

Hipertexto

Hipertexto pode ser entendido como um texto de suporte acoplado a outros textos, cujo acesso se dá através dos *links* que têm a função de conectar estes textos, dando-lhes um sentido comum, estendendo-os ou complementando-os.

Juntamente com o termo hipermídia, hipertexto foi cunhado, em 1965, pelo filósofo e sociólogo norte-americano Ted Nelson, pioneiro da tecnologia da informação.

Em computação, hipertexto é um sistema para a visualização de informação cujos documentos contêm referências internas para outros documentos (chamadas de *hypertext links* ou *hyperlinks* ou, simplesmente, *links*), e para a fácil publicação, atualização e pesquisa de informação. O sistema de hipertexto mais conhecido atualmente é a *web*, no entanto a Internet não é o único suporte onde este modelo de organização da informação e produção textual se manifesta.

História

A idéia de hipertexto não nasce com a Internet, nem com a *web*. De acordo com Burke (2004) e Chartier (2002), as primeiras manifestações hipertextuais ocorrem nos séculos XVI e XVII através de manuscritos e *marginalia*. Os primeiros sofriam alterações quando eram transcritos pelos copistas e assim caracterizavam uma espécie de escrita coletiva. Os segundos eram anotações realizadas pelos leitores nas margens das páginas dos livros antigos, permitindo assim uma leitura não-linear do texto. Essas *marginalia* eram posteriormente transferidas para cadernos de lugares-comuns para que pudessem ser consultadas por outros leitores.

Provavelmente, a primeira descrição formal da idéia apareceu em 1945, quando Vannevar Bush publicou na *The Atlantic Monthly*, "As We May Think", um ensaio no qual descrevia o dispositivo "Memex". Neste artigo, a principal crítica de Bush era aos sistemas de armazenamento de informações da época, que funcionavam através de ordenações lineares, hierárquicas, fazendo com que o indivíduo que quisesse recuperar informações tivesse que percorrer catálogos ordenados alfabética ou numericamente ou então através de classes e subclasses (não, não era OO ainda...). De acordo com Bush, o pensamento humano não funciona de maneira linear, mas sim através de associações e era assim que ele propunha o funcionamento do Memex. O dispositivo nunca chegou a ser construído, mas hoje é tido como um dos precursores da atual *web*. A tecnologia usada seria uma combinação de controles eletromecânicos e câmeras e leitores de microfilme, todos integrados em uma grande mesa. A maior parte da biblioteca de microfilme estaria contida na própria mesa com a opção de adicionar ou remover rolos de microfilme à vontade. A mesa poderia também ser usada sem a criação de referências, apenas para gerar informação em microfilme, filmando documentos em papel ou com o uso de uma tela translúcida sensível ao toque. De certa forma, o Memex era mais do que uma máquina hipertexto. Era precursor do moderno computador pessoal embora baseado em microfilme. O artigo de novembro de 1945 da revista Life que mostrava as primeiras ilustrações de como a mesa do Memex podia ser, mostrava também ilustrações de uma câmera montada na cabeça, que o cientista podia usar enquanto fazia experiências, e de uma máquina de escrever capaz de reconhecimento de voz e de leitura de texto por síntese de voz. Juntas, essas máquinas formariam o Memex, provavelmente, a descrição prática mais antiga do que é chamado hoje o "Escritório do Futuro".

O trabalho de Ted Nelson, junto a outros sistemas pioneiros de hipertexto como o "NLS", de Douglas Engelbart (NLS ou oN-Line System, um sistema de computação inventado nos anos 60 e suportado pela ARPA (é... a mesma da ARPANET – precursora da Internet) a NASA e a USAF, foi o primeiro a fazer uso prático do hipertexto e a usar o mouse, entre outras novidades), e o HyperCard, incluído no Apple Macintosh (o HyperCard foi uma aplicação da Apple entre as primeiras a ter sucesso com sistemas hipermídia antes da *web*, combinando capacidades de banco de dados com recursos gráficos e acoplamento a linguagens RAD), foram rapidamente suplantados em popularidade pela *World Wide Web* de Tim Berners-Lee, embora faltasse à mesma muitas das características desses sistemas mais antigos como *links* tipados, transclusão (inclusão de trechos de outros documentos) e controle de versão.

Principais Características do Hipertexto

- Intertextualidade
- Velocidade
- Precisão
- Dinamismo
- Interatividade
- Acessibilidade
- Estrutura em rede
- Transitoriedade
- Organização multilinear

Hipertexto e Internet

Uma das maiores controvérsias a respeito deste conceito é sobre sua vinculação obrigatória ou não com a Internet e outros meios digitais. Alguns autores defendem que o hipertexto acontece apenas nos ambientes digitais, pois estes permitem acesso imediato a qualquer informação. A *web* seria o meio hipertextual por excelência, uma vez que toda sua lógica de funcionamento está baseada nos *links*.

Outros pesquisadores acreditam que a representação hipertextual da informação independe do meio. Pode acontecer no papel, por exemplo, desde que as possibilidades de leitura superem o modelo tradicional contido das narrativas contínuas (com início, meio e fim). Uma enciclopédia é um clássico exemplo de hipertexto baseado no papel, pois permite acesso não-linear aos verbetes contidos em diferentes volumes. Um exemplo de hipertexto tradicional são as anotações de Leonardo da Vinci e também a Bíblia com referências cruzadas, devido a sua forma não linear de leitura.

Hipermídia

Como vimos acima, foi Ted Nelson, um dos pioneiros da Tecnologia da Informação, quem criou o termo hipermídia.

Segundo Laufer & Scavetta, hipermídia é a reunião de várias mídias num suporte computacional, sobre sistemas eletrônicos de comunicação. De acordo com Vicente Gosciola, hipermídia é "o conjunto de meios que permite acesso simultâneo a textos, imagens e sons de modo interativo e não linear, possibilitando fazer *links* entre elementos de mídia, controlar a própria navegação e, até, extrair textos, imagens e sons cuja seqüência constituirá uma versão pessoal desenvolvida pelo usuário". Ele afirma que a hipermídia é o meio e a linguagem das "novas mídias", às quais pertencem a Internet, os jogos de computador, o cinema interativo, o vídeo interativo, a TV interativa, as instalações informatizadas interativas e os sistemas de comunicação funcionais, entre outros e suas respectivas interfaces.

De um modo mais simplista, poderíamos dizer que Hipermídia é Hipertexto+Multimídia e não estaríamos muito longe da realidade (pelo menos, do ponto de vista prático), portanto, as características vistas para o Hipertexto valem aqui também, porém acrescidas das imensas possibilidades dos conteúdos multimídia.

O texto escrito para o congresso da ACM (*Association for Computer Machinery*, a mesma que vemos hoje ligada à maratona de programação) por Tim Berners-Lee (o "pai" da *web*) de certa forma fez com que o campo da hipermídia se expandisse. Ao propor a *WWW* como conceito e ferramenta de interligação de computadores ao redor do mundo, Berners-Lee acabou por realizar, ainda que de forma limitada, o sonho de Ted Nelson com seu projeto Xanadu (o primeiro projeto de hipertexto), ou seja, de interligar todos os documentos textuais e visuais em sistemas informacionais.

Hipermídia une os conceitos de hipertexto, interface e multimídia numa só linguagem. Não se configura só como meio de transmissão de mensagens, e sim como uma linguagem com características próprias, com sua própria gramática. Hipermídia, diferentemente de multimídia, não é a mera reunião dos meios existentes, e sim a fusão desses meios a partir de elementos não-lineares.

O hipertexto seria então apenas uma forma simples de hipermídia, no qual a informação é apresentada ao usuário sob a forma de texto, através de uma tela do computador. O usuário pode iniciar uma leitura de forma não linear, ou seja, escolhe entre o início, meio ou fim de um texto. A hipermídia pode ser considerada uma extensão do

hipertexto, incluindo além de textos comuns, sons, animações e vídeos, e de uma forma interativa, com apenas um clicar de botão, o computador responde ao caminho desejado.

Conferências Acadêmicas

Uma das principais conferências sobre novas pesquisas na área é a *Conference on Hypertext and Hypermedia*, realizada anualmente pela ACM.

Leitura Complementar

BUGAY, Edson Luiz; ULBRICHT, Vania Ribas. Hipermídia. Florianópolis: Visual Books, 2000.

Conclusão

O fato curioso acerca do tema "hipermídia" é a quase ausência de literatura específica. O livro acima citado está esgotado e não deve ser reeditado. Se procurarmos em sites de livros brasileiros, quase não achamos nada. O pouco que há é mais voltado às áreas de educação, jornalismo e comunicação do que informática. Por quê? Talvez por ser algo genérico demais ou por ter sido exageradamente valorizado há alguns anos, justamente quando foi colocado como o termo principal no nome desta disciplina. Como podemos, ainda assim, aproveitar o tema, atualizá-lo e torná-lo útil para o aprendizado? A proposta, que já vem sendo implementada, é dar ênfase ao que é importante no tema para a realidade atual (principalmente no que se refere à Internet) e acrescentar conteúdo prático à disciplina, cobrindo a área de desenvolvimento de aplicações *web* do lado servidor, como fizemos. Dessa forma, a disciplina se mostra extremamente útil e adequada à formação do nosso curso.

Hipermídia na Internet – Linguagens de Texto e Formatação

Linguagens Textuais Pré-Internet: SGML

Historicamente importante, SGML (*Standard Generalized Markup Language*) é uma linguagem através da qual se pode definir outras linguagens de marcação para documentos. Ela descende da GML (*Generalized Markup Language*) da IBM, desenvolvida na década de 60 por Charles Goldfarg, Edward Mosher e Raymond Lorie.

Inicialmente concebido para possibilitar a partilha de documentos que permitissem a leitura por máquinas em projetos de grande dimensão governamentais e na indústria aeroespacial, SGML prove uma variedade de sintaxes de marcação que podem ser usadas por várias aplicações.

Exemplo de sintaxe SGML:

```
<QUOTE TYPE="exemplo">
   Tipicamente algo como <ITALICS>isto</ITALICS>
</QUOTE>
```

Normatizado em 1986, o SGML propiciou o surgimento do HTML e do XML, sendo este último um subconjunto específico do SGML na tentativa de simplificá-lo para aplicações de uso geral. XML tem sido usado num largo número de aplicações, como XHTML, RSS, XML-RPC, AJAX, SOAP, etc.

Uma derivação do SGML chamada HyTime (*Hypermedia/Time-based Structuring Language*) define um conjunto de tipos de elementos orientados a hipertexto que efetivamente suplementam o SGML e permitem aos autores de documentos construir apresentações de hipermídia de forma padronizada. A primeira edição do HyTime foi oficialmente publicada em 1992 e alguns dos seus conceitos foram incorporados posteriormente em versões de HTML e XML.

HTML

Já usamos HTML (*HyperText Markup Language)* no passado e sabemos que é uma linguagem textual, exibida em *browsers* (navegadores), mas agora vamos além, no que diz respeito à sua evolução e desdobramentos.

História

Historicamente, o HTML surge do legado de SGML e HyTime, vistos acima. Tim Berners-Lee criou o HTML original em 1989 (e outros protocolos associados como o HTTP). Na época, a linguagem não era uma especificação, mas uma coleção de ferramentas para resolver um problema específico de Tim: a comunicação e disseminação das pesquisas entre ele e seu grupo de colegas. Sua solução, combinada com a então emergente rede pública (que se tornaria a Internet) ganhou atenção mundial.

A linguagem foi definida formalmente na década de 1990, inspirada nas propostas originais de Tim Berners-Lee. A primeira publicação foi esboçada por ele e Dan Connolly em 1993 na IETF (*Internet Engineering Task Force*, uma comunidade aberta de pesquisa e desenvolvimento para a Internet) como uma aplicação formal para o SGML. A IETF criou um grupo de trabalho para o HTML no ano seguinte, e publicou o HTML 2.0 em 1995. Desde 1996, as especificações HTML vêm sendo mantidas, com o auxílio de fabricantes de software, pela W3C (*World Wide Web Consortium*, um consórcio de diversas empresas que buscam estabelecer padrões para a internet). Em 2000, a linguagem tornou-se também uma norma internacional. A especificação atualmente em uso é a HTML 4.01, lançada pela W3C no final de 1999. Uma errata foi lançada em 2001. Atualmente, está em fase de desenvolvimento pelo W3C o HTML 5.0 (que alguns querem ver fundido com o XHTML, que veremos adiante).

As primeiras versões do HTML foram definidas com regras sintáticas flexíveis, o que ajudou aqueles sem familiaridade com linguagens de computador. Atualmente tenta-se tornar a sintaxe do HTML mais rígida, permitindo um código mais preciso. Mas, por questões históricas os navegadores ainda hoje conseguem interpretar (sem erros aparentes) páginas web que não seguem os padrões atuais – ou até com erros óbvios, como já observamos. Desde a publicação do HTML 4.0 no final de 1997, o grupo de trabalho da W3C tem focado no desenvolvimento do XHTML,

uma especificação HTML baseada em XML que é considerada pela W3C como um sucessor do HTML. O XHTML faz uso de uma sintaxe mais rigorosa e menos ambígua para tornar o HTML mais simples de ser processado e estendido.

Tags

Como já sabemos, um documento HTML precisa das *tags* (etiquetas), elementos entre parênteses angulares (sinais de maior e menor) (< e >) para ser formatado. A maioria das *tags* tem sua correspondente de fechamento:

```
<etiqueta>...</etiqueta>
```

Isso é necessário porque as etiquetas servem para definir a formatação de uma porção do documento, e assim marcamos onde começa e termina o texto com a formatação especificada por ela. Alguns elementos são chamados "vazios", pois não marcam uma região de texto, apenas inserem algum elemento no documento:

```
<etiqueta>
```

Uma tag é formada por comandos, atributos (parâmetros) e valores. Os atributos modificam os resultados padrões dos comandos e os valores caracterizam essa mudança. Exemplo:

```
<HR color="red">
```

Cada comando tem seus atributos possíveis e seus valores. Um exemplo é o atributo `SIZE` que pode ser usado com o comando `FONT e` com o `HR`, mas que não pode ser usado com o comando `BODY`. Isso quer dizer que devemos saber exatamente quais os atributos e valores possíveis para cada comando.

Estrutura básica de um documento

Como já sabemos, a estrutura mínima de um documento HTML apresenta os seguintes componentes:

```
<html>
<head>
  <title>Título do Documento</title>
</head>
<body>
  ...
</body>
</html>
```

As etiquetas HTML não são sensíveis à caixa (*case sensitive*), portanto tanto faz escrever `<HTML>`, `<Html>`, `<html>` ou `<HtMl>`.

Cabeçalho (head)

Dentro do cabeçalho podemos encontrar os seguintes comandos:

- `<title>`: define o título da página, que é exibido na barra de título dos navegadores.
- `<style>`: define formatação em CSS.
- `<script>`: define programação de certas funções em página com scripts, podendo adicionar funções de JavaScript, por exemplo.
- `<link>`: define ligações da página com outros arquivos como feeds, CSS, scripts, etc.
- `<meta>`: define propriedades da página, como codificação de caracteres, descrição da página, autor, etc. São meta informações sobre documento. Tais campos são muitos usados por motores de busca para obterem mais informações sobre o documento, a fim de classificá-lo melhor. Por exemplo, pode-se adicionar o código `<meta name="description" content="descrição da sua página" />` no documento HTML para indicar ao motor de busca que texto de descrição apresentar junto com a ligação para o documento.

Obs: as etiquetas <style> e <script> servem tanto para delimitar o espaço usado pelos códigos na pagina quanto para invocar códigos existentes em outros arquivos externos.

Corpo (*body*)

Dentro do corpo podemos encontrar outras várias etiquetas, por exemplo:

- <h1>, <h2>,... <h6>: cabeçalhos e títulos no documento em diversos tamanhos. (quanto menor for o numero, maior sera o tamanho da letra)
- <p>: novo parágrafo.
-
: quebra de linha.
- <table>: cria uma tabela (linhas são criadas com <TR> e novas células com <TD>. Já os cabeçalhos de coluna são criados com a etiqueta <TH>.)
- : determina uma divisão na página a qual pode possuir variadas formatações.
- <font>: forma um texto (fonte, cor e tamanho) de um trecho do texto.
- <b>, <i>, <u> e <s>: negrito, itálico, sublinhado e riscado, respectivamente.
- <img>: imagem.
- <a>: hiper-ligação para um outro local, seja uma página, um e-mail ou outro serviço.
- <textarea>: caixa de texto (com mais de uma linha); estas caixas de texto são muito usadas em blogs, elas podem ser auto selecionáveis e conter outros códigos a serem distribuídos.
- <acronym>: acrônimo (sigla)
- <cite>: citação
- <adress>: endereço

Edição de documentos HTML

Os documentos em HTML são arquivos de texto comuns. Tão comuns que muitas vezes usamos um simples editor de texto (como o bloco de notas) para escrevê-los. Mas, para facilitar a produção de documentos, principalmente quando uma formatação mais complexa é necessária, existem editores HTML de texto ou mesmo gráficos. Exemplos:

- Dreamweaver
- Bluefish
- Edit Plus
- Emacs
- FirstPage
- Frontpage

XML

XML (*eXtensible Markup Language*) é uma recomendação da W3C para gerar linguagens de marcação para necessidades especiais. É um subtipo de SGML, capaz de descrever diversos tipos de dados. Seu propósito principal é a facilidade de compartilhamento de informações através da Internet. Entre linguagens baseadas em XML incluem-se XHTML, RDF, SDMX ,SMIL, MathML (formato para expressões matemáticas), NCL, XBRL, XSIL e SVG (formato gráfico vetorial).

Características do XML

Estimulado pela insatisfação com os formatos existentes (padronizados ou não), o W3C começou a trabalhar em meados da década de 1990 em uma linguagem de marcação que combinasse a flexibilidade da SGML com a simplicidade da HTML. O principio do projeto era criar uma linguagem que pudesse ser lida por software, e integrar-se com as demais linguagens. Sua filosofia seria incorporada por vários princípios importantes:

- Separação entre conteúdo e formatação.
- Simplicidade e legibilidade, tanto para humanos quanto para computadores.
- Possibilidade de criação de *tags* sem limitação.
- Criação de arquivos para validação de estrutura, o DTD (*Document Type Definition* – uma adição ao arquivo que funciona como uma espécie de glossário, “explicando” sua estrutura).

- ? Interligação de bancos de dados distintos.
- ? Concentração na estrutura da informação, e não na sua aparência.

O XML é considerado um bom formato para a criação de documentos com dados organizados de forma hierárquica, como se vê frequentemente em documentos de texto formatados, imagens vetoriais ou bancos de dados.

Pela sua portabilidade, uma aplicação pode ler dados de um banco e escrevê-los em um arquivo XML e então passá-los a um outro banco distinto.

Exemplos

Codigo XML descrevendo um Curriculum Vitae:

```
<?xml version="1.0" encoding="UTF-8"?>
<curriculo>
   <InformacaoPessoal>
      <DataNascimento>23-07-68</DataNascimento>
      <Nomecompleto>...</Nomecompleto>
      <Contatos>
         <Morada>
              <Rua>R.Topazio</Rua>
              <Num>111</Num>
              <Cidade>nome_cidade</Cidade>
              <Pais>nome_país</Pais>
         </Morada>
         <Telefone>9999-9999</Telefone>
         <CorreioEletronico>email@email.com</CorreioEletronico>
      </Contatos>
      <Nacionalidade>brasileiro</Nacionalidade>
      <Sexo>M</Sexo>
   </InformacaoPessoal>
   <objetivo>Atuar na area de TI</objetivo>
   <Experiencia>
      <Cargo>Suporte tecnico</Cargo>
      <Empregador>Empresa, Cidade - Estado</Empregador>
   </Experiencia>
   <Formacao>Superior Completo</Formacao>
</curriculo>
```

Outro exemplo demonstrando a sintaxe flexível do XML sendo usada para descrever uma receita de pão:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<receita nome="pão" tempo_de_preparo="5 minutos" tempo_de_cozimento="1 hora">
  <titulo>Pão simples</titulo>
  <ingredientes>
    <ingrediente quantidade="3" unidade="xícaras">Farinha</ingrediente>
    <ingrediente quantidade="7" unidade="gramas">Fermento</ingrediente>
    <ingrediente quantidade="1.5" unidade="xícaras"
estado="morna">Água</ingrediente>
    <ingrediente quantidade="1" unidade="colheres de chá">Sal</ingrediente>
  </ingredientes>
  <instrucoes>
    <passo>Misture todos os ingredientes, e dissolva bem.</passo>
    <passo>Cubra com um pano e deixe por uma hora em um local morno.</passo>
    <passo>Misture novamente, coloque numa bandeja e asse num forno.</passo>
  </instrucoes>
</receita>
```

O XML se tornou um padrão para transferência de dados, embora sua estrutura seja pesada algumas vezes, pois pode haver uma grande quantidade de informação repetida. Por isso, outros formatos têm surgido como o Properties e o JSON, mas nenhum chega à popularidade do XML.

CSS

CSS (*Cascading Style Sheets*) é uma linguagem de estilo utilizada para definir a apresentação de documentos escritos em uma linguagem de marcação, como HTML ou XHTML. Seu principal benefício é prover a separação entre o formato e o conteúdo de um documento.

É possível colocar a formatação de estilo dentro de uma página, como no exemplo abaixo:

```
<html>
  <head>
    <style type="text/css">
      h1 {color: blue; background-color: silver; }
      h3 {color: yellow; background-color: black; }
    </style>
  </head>
  <body>
    Isto é um texto comum<br>
    <h1>Isto é um texto em h1</h1>
    <h2>Isto é um texto em h2</h2>
    <h3>Isto é um texto em h3</h3>
    <h4>Isto é um texto em h4</h4>
    <h5>Isto é um texto em h5</h5>
    <h6>Isto é um texto em h6</h6>
  </body>
</html>
```

Definimos a cor padrão para <h1> e <h3>. Porém, ao invés de embutir a formatação dentro do documento, pode ser mais interessante criar um link (ligação) para uma página CSS que contenha os estilos que podem ser definidos para cada *tag* de formatação HTML. Fazendo isso em todas as páginas de um *site*, se quisermos alterar a aparência do portal inteiro, basta modificar um arquivo.

É importante notar que os *browsers* não suportam igualmente as definições do CSS. Desta forma, o *webdesigner* deve sempre testar suas folhas de estilo em *browsers* de vários fabricantes, e preferencialmente em mais de uma versão, para se certificar de que o que foi codificado realmente seja apresentado da forma desejada. O Internet Explorer 6, por exemplo, tem suporte total a CSS1 e praticamente nulo a CSS2. Navegadores mais modernos como Opera, Internet Explorer 7 e Firefox tem suporte maior, inclusive até a CSS 3, ainda em desenvolvimento. A interpretação dos *browsers* pode ser avaliada com os testes "Acid" (Acid1, Acid2, Acid3), que se tornaram uma forma padrão de revelar quão eficiente (em relação ao padrão W3C) é o suporte de CSS (e de HTML também).

Exemplo de definição em CSS

```
/* comentário em CSS, semelhante ao da linguagem c */
body
{
   font-family: Arial, Verdana, sans-serif;
   background-color: #000FFF;
   margin: 5px 10px;
}
```

O código acima define fonte padrão Arial, caso não exista substitui por Verdana, caso não exista define qualquer fonte sem serifa. Define também a cor de fundo do corpo da página e margens.

Exemplo de CSS aplicado a HTML

Arquivo "exemplo1.html" com ligação para uma folha de estilos em cascata:

```
<html>
<head>
  <title>Exemplo 1: CSS com HTML</title>
  <link rel="stylesheet" type="text/css" href="exemplo1.css" />
</head>
<body>
  <h1>Exemplo de CSS com HTML</h1>
</body>
</html>
```

O arquivo “exemplo1.css” que formata o HTML anterior:

```
body
{
   font-family: Arial, Verdana, sans-serif;
   background-color: #000FFF;
   margin: 5px 10px;
}

h1 {
   color: yellow;
}
```

Uma boa fonte de referência de CSS é: http://www.bitpt.com/index.php/content/category/3/19/60/ .

XHTML

XHTML (*eXtensible Hypertext Markup Language*) é uma reformulação da linguagem de marcação HTML baseada em XML. Combina as *tags* de marcação HTML com regras da XML; este processo de padronização tem em vista a exibição de páginas web em diversos dispositivos (televisão, palm, celular, etc) melhorando a acessibilidade e tornando a linguagem mais robusta e regrada (por exemplo: as *tags* devem ser escritas em minúsculas).

O XHTML consegue ser interpretado por qualquer dispositivo, independentemente da plataforma utilizada. O HTML, por não ter regras tão rígidas, não consegue isto sempre. No entanto, não existem muitas diferenças entre o HTML e o XHTML. Para verificar se uma página XHTML está bem construída, o melhor método é validar o código através de uma aplicação Web disponibilizada pela W3C em http://validator.w3.org/. Quase todos os bons editores de HTML suportam XHTML.

XHTML poderá ser o sucessor do HTML. Assim, muitos consideram que XHTML é a atual ou mais nova versão do HTML. Porém, XHTML é uma recomendação separada.

As principais diferenças entre XHTML e HTML

- todas as tags e parâmetros devem ser escritos em letras minúsculas;
- as tags devem estar convenientemente aninhadas;
- os documentos devem ser bem formados;
- o uso de tags de fechamento é obrigatório, mesmo em tags únicas (ex:
);
- elementos vazios devem ser fechados;
- regras para os atributos (estar entre aspas, por exemplo).

Um exemplo de documento XHTML válido (exemplo2.xhtml):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
 <title>Teste de XHTML</title>
 <link rel="stylesheet" type="text/css" href="exemplo2.css" />
 </head>
 <body>
    <form action="" method="get">
     <fieldset>
     <legend>Dados para Cadastro</legend>
     <label for="nome">Nome:</label>
     <input type="text" name="nome" id="nome" />
     <label for="tipo">Tipo:</label>
     <input type="text" name="tipo" id="tipo" />
          <label for="data">Data:</label>
     <input type="text" name="data" id="data" />
          <input type="submit" value="enviar" class="botao" />
     </fieldset>
   </form>
</body>
```

```
</html>
```

O codigo CSS (exemplo2.css) que formata esse XHTML:

```
<style>
form fieldset {
  float: left;
  margin-right: 7px;
  width: 40%;
  border: solid black 1px;
  padding: 3%;
  margin-bottom: 10px;
}
form legend {
  padding: 6px;
  margin: 10px;
  border: solid black 1px;
  font-size: 90%;
  font-weight: bold;
  background-color: #e8e8e8;
}
form label {
  display: block;
  font-size: 11px;
}
form input {
  width: 100%;
  border: solid #ccc 1px;
  font-size: 11px;
  font-family: 'Trebuchet MS', Verdana, Tahoma, Serif;
}
form input.botao {
  display: block;
  width: auto;
  float: right;
}
</style>
```

A Recomendação XHTML original da W3C, XHTML 1.0, foi simplesmente uma reformulação do HTML 4.01 em XML. A mais recente Recomendação XHTML da W3C é o *XHTML 1.1: Module-based XHTML*, que é uma reformulação do XHTML 1.0 Strict, com pequenas modificações. Todas as ferramentas consideradas obsoletas de HTML, como framesets, atributos `lang` e o atributo de âncoras `name`, que ainda eram permitidos no XHTML 1.0 Strict, foram removidos desta versão. A apresentação é controlada puramente pelas Folhas de Estilo em Cascata (CSS).

Outros membros da família XHTML:

- XHTML Basic: Uma versão especial "light" do XHTML para serviços que não podem suportar os grandes e complicados dialetos XHTML, principalmente celulares. Este deverá ser o substituto da WML e C-HTML.
- XHTML Mobile Profile: Baseado em XHTML Basic, essa OMA (*Open Mobile Alliance*) tem como alvo fones de mão.
- XHTML+Voice: XHTML combinado com Voice XML para suportar interações visuais e sonoras.

DHTML

DHTML (*Dynamic HTML*) não é uma linguagem, mas um conjunto de tecnologias para criar sites interativos: uma combinação de linguagem de marcação de texto (como HTML), uma linguagem de script (como Javascript) e uma linguagem de definição de apresentação (como CSS).

Uma página DHTML é qualquer página na qual um script *client-side* altere o conteúdo da linguagem de apresentação, o que fará com que o visual do HTML mude APÓS a página já ter sido totalmente carregada e durante o processo de sua visualização. A característica dinâmica do DHTML deve ser entendida como a forma que ele funciona enquanto a página é vista e não sua habilidade de gerar uma página diferente a cada visita.

Ao contrário, uma página web gerada dinamicamente é um conceito mais amplo: qualquer página gerada de forma diferente para cada requisição ou carregamento. Isso não é DHTML e sim páginas criadas por script no cliente ou no servidor (como no PHP ou JSP) onde o se gera o conteúdo antes de enviá-lo para exibição.

O termo caiu em desuso nos últimos anos, por terem surgido novas tecnologias e por terem as páginas "carimbadas" como DHTML problemas freqüentes de compatibilidade entre navegadores.

Tableless

Tableless é uma forma de desenvolvimento de páginas que não utiliza tabelas para disposição de conteúdo, pois defende que o código HTML deveria ser usado para o propósito que foi criado, sendo que tabelas foram criadas para exibir dados tabulares, não para formatar páginas. Para isto, o recomendado seria usar CSS e outras técnicas. Dentre as possíveis vantagens da metodologia, estão a diminuição do peso da página e a melhora da legibilidade.

Quem quiser ver alguns exemplos e uma discussão maior, pode acessar a página abaixo:

http://www.marcelotoledo.org/stuff/artigos/padroes_web_e_tableless/padroes_web_e_tableless.html