

Notas da Aula 1: Introdução à Web  
Prof. Daniel Caetano

**Objetivo:** Apresentar a história da Web, "fases" da Web e diferenciá-la das GUIs.

### **1. Histórico da World Wide Web**

No início da década de 1990, começava a ser desenvolvido um novo paradigma no universo computacional: o das WUIs (Web User Interfaces - Interfaces Web com o Usuário). Mas esta história começou muito antes, e remonta o início do século XX, pois era baseado no paradigma do "hipertexto", pensado pela primeira vez no início do século XX por Paul Otlet e ligeiramente desenvolvido por H. G. Wells na década de 1930.

Entretanto, foi apenas na década de 1940 que o paradigma foi melhor formalizado (por Ted Nelson e Douglas Engelbart), tendo sido o termo "Hipertexto" criado apenas em 1965. O primeiro editor hipertexto da história foi criado em 1968, por Andries van Dam.

Os sistemas hipertexto foram bastante usados em diversas aplicações ao longo dos anos, mas seu uso tornou-se absolutamente mais expressivo quando Berners-Lee, um cientista do CERN (Organisation Européenne pour la Recherche Nucléaire - Organização Europeia para a Pesquisa Nuclear), criou a World Wide Web.

Um dos primeiros software para "navegação" pelo hipertexto da World Wide Web foi o ViolaWWW, criado no início da década de 1990. Pouco depois, entre 1992 e 1993, foi desenvolvido o Mosaic, da NCSA (National Center for Supercomputing Applications - Centro Nacional para Aplicações de Supercomputadores), que rapidamente tomou conta do mercado.

O domínio do Mosaic durou até o lançamento do Netscape Navigator, em 1994, que tornou-se o padrão de navegador web em praticamente todas as plataformas existentes. A então empresa Netscape começou a imaginar novas aplicações para seu "suite", independentes de plataforma, que permitiriam o usuário editar e manusear documentos em qualquer equipamento conectado, independente de seu sistema operacional... algo que nunca chegou a acontecer.

Observando o plano pretensioso da Netscape, a Microsoft temeu por seu já consagrado domínio do mercado de sistemas operacionais caseiros. Rapidamente tratou de desenvolver seu próprio navegador Web, o qual foi incluído pela primeira vez no Windows 95 OSR2, ainda no fim de 1995.

De 1995 a 1998 a "guerra dos navegadores" foi ferrenha. O uso de táticas questionáveis de marketing por parte da Microsoft tiveram como resultado a extinção da Netscape Communications e o completo domínio do Internet Explorer a partir de 1999. Entretanto, a necessidade para um novo navegador para outros sistemas operacionais (O

Internet Explorer só existia para Windows e MacOS), aliada à falta de segurança e problemas inerentes ao navegador da Microsoft, fizeram com que a comunidade OpenSource se movimentasse.

Aproveitando-se da liberação pública do código do antigo Netscape Communicator, formou-se a Mozilla Foundation, em 1998, para criar um novo navegador web gratuito, de código aberto e multiplataforma. Várias versões "Mozilla Internet Suite" foram lançadas de 1999 a 2003, até que em 2004 foi liberada a primeira versão oficial do novo navegador: Mozilla Firefox.

Desde o lançamento do Mozilla Firefox até os dias atuais, tem havido uma nova onda de migração do Internet Explorer 6 para Firefox 1.x. No fim de 2006 foram lançadas as novas versões de ambos navegadores: Internet Explorer 7 e Firefox 2, iniciando uma nova rodada na disputa pela liderança do mercado.

No momento atual, os primeiros betas do Firefox 3 já começaram a surgir, prometendo que muita coisa ainda deve mudar nos próximos anos.

## **2. Evolução da Web**

Durante toda esta evolução, os navegadores passaram por muitas mudanças tecnológicas... e como consequência, muito mudou também no desenvolvimento de web sites. Assim, antes de sentar e projetar um website, é interessante avaliar as **tecnologias** que estão **disponíveis** e quais serão **usuadas**. É possível classificar as evoluções da Web e suas tecnologias em **fases**:

**Web Estática**: Primeira fase da web, em que os **conteúdos eram estáticos**, ou seja, eram criados todos manualmente e quase nunca atualizados. Esta fase passou por dois estágios: um primeiro onde as formatações visuais eram mínimas e uma segunda, em que se introduziram tabelas, mudanças de cores e fontes, etc.

**Web Dinâmica**: Segunda fase da web, em que os **conteúdos** passaram a ser dinâmicos, ou seja, a **se modificarem sozinhos** com o tempo. As tecnologias acrescentadas, como forms, uso de linguagens script (Javascript, PHP, ASP, etc) em conjunto com bancos de dados tornaram possível a **criação de páginas personalizadas** e com características muito mais atrativas aos usuários.

**Web Ativa**: Fase atual, em que foram acrescentados **elementos que são executados no computador do usuário**, com informações sendo apresentadas e escolhidas sem a intervenção do servidor. Fazem parte desta era tecnologias como Java e Flash.

É importante lembrar que usar recursos de **Web Dinâmica** implicam em **exigências específicas com relação ao servidor** onde a página estará hospedada (como a existência de um interpretador PHP de uma determinada versão, por exemplo). Uma vez que esta exigência seja resolvida, todos os usuários terão acesso ao conteúdo.

No caso da **Web Ativa**, entretanto, a **exigência é com relação à existência de um determinado produto** (Flash, por exemplo) de uma determinada versão **no micro do cliente**. A vantagem quando os requisitos são atendidos é que o usuário tem uma experiência muito mais rica e rápida. A desvantagem é que o usuário pode não querer instalar o software em sua máquina (ou mesmo não poder instalá-lo, por uma série de razões) e seu site acabar ficando inacessível para estes usuários.

A dica aqui é sempre **usar o menor número de recursos necessário** para que o website seja atrativo e, sempre que for usado algum recurso extra de Web Dinâmica ou Ativa, procurar requisitar sempre a versão mais antiga possível de softwares a serem instalados no servidor ou computador do cliente, dado que a probabilidade de uma versão antiga ou mais nova já estar instalada é maior a cada versão do produto que entra no mercado.

### 3. Paradigma WUI

Um outro aspecto interessante de ser analisado antes de se pensar em projetar um web site, é a questão do "foco da web". As **Web User Interfaces** são interfaces **voltadas a facilitar a obtenção de informações** por parte do usuário, ao consultar um sistema qualquer. É importante ressaltar esta característica, em oposição ao objetivo de **facilitar o processamento de informações que é característico das GUIs**, pois muitos desenvolvedores confundem as duas coisas, criando interfaces WUI muito confusas e pouco práticas.

Vale lembrar, por exemplo, que o **usuário é diferente no paradigma WUI**. Enquanto **nas GUIs o usuário é mais ou menos conhecido** (embora não completamente), **nas WUIs os usuários são completos desconhecidos**. Em outras palavras, isso significa que no projeto das GUIs é possível determinar com alguma precisão o que os usuários esperam do software, mas no caso das WUIs essa determinação é quase impossível.

Além disso, existe uma outra questão envolvida no caso do usuário das WUIs. Enquanto as **GUIs normalmente são parte de softwares caros e que precisam ser adquiridos à priori**, o usuário acaba ficando cativo de um software por já ter gasto dinheiro no mesmo. Em outras palavras, nas GUIs o usuário só conhece a interface depois que comprou o produto. **No caso das WUIs a relação se inverte: a interface é usada para adquirir produtos**. Desta forma, o usuário não se fixa em um determinado site e, se não gostarem de sua interface, além de não adquirirem qualquer produto, com grande probabilidade jamais voltarão àquele site.

Uma quarta consideração é quanto à complexidade da navegação. Enquanto nas **GUIs a navegação é bem simples** (quando não inexistente), sendo basicamente direcionada por um menu suspenso, **nas WUIs a navegação é bastante complexa**, com elementos de navegabilidade em todos os cantos e sendo possível chegar a uma determinada página por dezenas de trajetos.

Como é possível observar, há **grandes diferenças** entre o "problema" de se projetar uma GUI ou uma WUI. É claro que **conceitos de ergonomia** (como mínima carga de memória, máxima realização, etc) **continuam válidos**. O **mesmo vale para os conceitos da interface de percepção**, como elementos de atenção, tempos de feedback e até mesmo projeto visual. Entretanto, as diferenças de funcionamento e dos usuários de cada uma das duas levam a ênfases em aspectos distintos.

As principais **diferenças entre GUIs e WUIs** podem ser **resumidas** da seguinte forma:

**GUIs:**

- 1) **Centradas em tarefas.**
- 2) **Anseios do usuário razoavelmente conhecidos.**
- 3) **Usuário "cativo".**
- 4) **Navegação simples.**

**WUIs:**

- 1) **Centradas na informação.**
- 2) **Usuários e seus anseios são desconhecidos.**
- 3) **Usuário "nômade".**
- 4) **Navegação complexa.**

## **6. Bibliografia**

DOMINGUES, D. G. O uso de metáforas na computação. Dissertação - Escola de Comunicações e Artes da Universidade de São Paulo. São Paulo, 2001.

NIELSEN, J. *Projetando Websites*. Ed. Campus, 2000.

FERREIRA, M.A.G.V. *Notas de Aula - Tópicos de Comunicação Homem-Máquina*, EPUSP, 2003.

Notas da Aula 02: Conceitos da World Wide Web - Fases da Web  
Prof. Daniel Caetano

**Objetivo:** Apresentar os diferentes tipos de site e suas características, vantagens e desvantagens.

### Introdução

Apesar das Interfaces Web com o Usuário (WUIs) serem relativamente mais jovens que as Interfaces Gráficas com o Usuário (GUIs), as **WUIs** também **já passaram por diversos estágios** de desenvolvimento, amadurecendo a cada geração em direção à sua vocação atual, que **combina a transmissão de informações**, presente na idéia original da Web, com o **comércio eletrônico**, uma área em larga expansão já há alguns anos.

Pode-se dizer que a web passou pelos **estágios** conhecidos como "**Web Estática**", "**Web Dinâmica**" e atualmente tenha surgido o conceito de "**Web Ativa**". Ainda que as **mudanças** sejam apresentadas como gerações distintas da web, é importante ter em mente que elas **ocorreram de forma contínua e suave** ao longo dos anos. Além disso, uma "geração" não substituiu a outra: pelo contrário, **sites** com características das mais diferentes etapas **de cada uma destas fases** estão ainda no ar e **convivem pacificamente**.

### 1. Páginas Web Estáticas

A **primeira fase** da Web, como idealizada por Tim Berners Lee, é chamada hoje de "Web Estática". As páginas Web Estáticas são **compostas** apenas **por elementos básicos** como **tabelas, imagens, listas e links**.

Seu **uso é bastante limitado**, servindo basicamente à **difusão de conteúdo que não muda** (ou muda muito pouco) ao longo do tempo, com **pouca capacidade de busca** (limitada à capacidade do navegador do usuário). A questão do conteúdo de pouca variação (daí o nome: estático), oriunda da **dificuldade de atualização de páginas** construídas neste formato, foi ignorada no começo e, como resultado, há uma infinidade de "páginas fantasma" na Web, com conteúdo bastante desatualizado e que não verão jamais uma atualização por todo o resto de sua existência.

Seu uso acertado foi basicamente para a **descrição dos produtos ou serviços** de uma empresa (coisas que, espera-se, não sofram tanta variação ao longo do tempo), além de **informações sobre objetivos de empresa** (que costumam mudar apenas em intervalos de alguns anos, quando a empresa tem sua diretoria ou presidência alterada).

Entretanto, a **difículdade de atualização e gerenciamento** (incluindo adição de novas informações), **difículdades para o cliente encontrar as informações** que procura (pela ausência de um sistema de busca adequado) e a **difículdade inerente modificar o layout** de um site (na primeira fase, as informações de visualização eram inseridas dentro das páginas) fizeram com que este **formato fosse evoluído em poucos anos**. A viabilidade comercial das páginas puramente estáticas teve uma vida de menos de dois anos.

Uma das inovações mais recentes da Web que poderiam ter trazido sobrevida à Web Estática foi o advento das **Cascade Style Sheets**, ou CSS.

A criação das Cascade Style Sheets trouxe um **novo conceito** ao projeto de páginas Web, em que o **conteúdo é separado** da definição **das características visuais** de uma página. Isso significa que a página descrita em **HTML não deve conter** qualquer informação sobre a **"forma" da página**: deve **apenas** especificar o **significado** do conteúdo (o que é título, o que é texto, qual é a figura a mostrar, o que é legenda, etc). Toda a **especificação da forma**, que é atrelada ao significado do conteúdo, **é definida** externamente **em um arquivo do tipo CSS**.

Embora esta tecnologia **não tire as limitações de uso das páginas web estáticas**, seu uso **traz maior flexibilidade visual**, permitindo a criação de layouts muito mais **elaborados**, bonitos e - muito importante - de **carregamento bastante rápido**. Além disso, a separação dos aspectos visuais do aspecto de conteúdo - algo que vem de encontro à arquitetura **MVC**, amplamente utilizada na Engenharia de Software - **facilita muito o gerenciamento de páginas e seu conteúdo**, bem como a **remodelagem visual de sites** inteiros.

Infelizmente, no que concerne a **negócios**, os **problemas** das páginas Web Estáticas vão além dos já citados, que são basicamente aspectos técnicos.

Primeiramente, as páginas web estáticas **não são capazes de receber entradas dos usuários**. Os contatos com as empresas (para adquirir produtos, por exemplo) precisam ocorrer por e-mail e, em geral, não podem ser automatizados. Além disso, elas **são pouco personalizáveis**, comprometendo a **usabilidade** e **dificultando a criação de usuários cativos**.

## 2. Páginas Web Dinâmicas

A primeira evolução da Web veio basicamente para **atender a** uma onipresente **necessidade de ferramentas de busca** mais poderosas do que as fornecidas pelos navegadores, além de **possibilitar o uso da web para negócios**.

O **ponto fundamental** desta primeira evolução foi a criação dos **"forms"**, os populares formulários, que nada mais são do que uma adaptação de **elementos de entrada de dados** de uma GUI para uso em uma WUI. Os formulários devem ser usados com cuidado,

**seguindo** basicamente as **regras** prescritas **para as GUIs**, ao mesmo tempo em que não deve ignorar as **regras de desenho das WUIs**.

As páginas Web Dinâmicas podem ser construídas de acordo com **3 conceitos distintos**, dependendo dos requisitos da aplicação/página sendo desenvolvida. Cada um destes conceitos possui tecnologias próprias, vantagens e defeitos:

- **Server Side, externo** à página: **CGIs, Servlets**. Usado quando se deseja uma **aplicação veloz e alta compatibilidade** com clientes.
- **Server Side, interno** à página: **ASP, JSP, PHP**. Usado quando se deseja **praticidade de desenvolvimento**, além da **alta compatibilidade** com clientes.
- **Client Side: JavaScript, VBScript**. Usando quando se deseja **reduzir a carga no servidor**, sob pena de uma **compatibilidade limitada**.

Adicionalmente, algumas outras tecnologias são utilizadas:

- **Banco de Dados**, para armazenar **perfil** de usuário, **conteúdos** em diversas línguas, últimos **lançamentos**, etc.
- **Cookies**, para armazenar **dados temporários** como **página visualizada** atualmente, **variáveis de sessão**, etc.

Talvez a **principal vantagem** das páginas web dinâmicas sobre as estáticas seja que as dinâmicas **permitem a realização de negócios por meio eletrônico**. Mas isso não é tudo, já que elas também podem ser usadas para **melhorar o tratamento do cliente** por parte da empresa, através de **tratamento personalizado**. Isso inclui não apenas **chamar o usuário pelo nome**, mas também **adaptar as informações** ao gosto do usuário: **listar os últimos produtos** comprados pelo usuário, **apresentar novos produtos** que podem ser de interesse dele, **propor promoções especiais** de acordo com seu perfil, etc.

Além disso, as tecnologias por trás da web dinâmica permitem uma **personalização das páginas** por parte do usuário, que pode **modificar o layout** de uma página, **escolher** quais **informações** serão apresentadas em sua página principal, **mudar o idioma** quando desejado, etc.

É importante lembrar que este tipo de característica **facilita a fidelização do cliente**, através de uma **melhor qualidade de serviço**, por **resolver os problemas do cliente de forma mais simples e rápida**, por **criar condições especiais para bons compradores**, etc.

Infelizmente a **Web Dinâmica também tem seus problemas**. As novas tecnologias e mecanismos de funcionamento fazem com que as **cargas no servidor sejam** bastante **maiores** que na web estática. Além disso, entre o comando do usuário e a resposta do servidor existe um tempo de tráfego das informações na rede, o que se traduz em uma **lentidão geral da interface**, perante o usuário. Para finalizar, as **interações são** bastante **limitadas**, resumindo-se ao preenchimento de formulários e seleções em mapas.

### 3. Páginas Web Ativas

A **segunda evolução** da Web veio basicamente para solucionar alguns problemas da web dinâmica: **reduzir os tempos de resposta e aumentar as possibilidades de interação**.

O **ponto fundamental** desta segunda evolução foi o uso dos "**plugins**", em especial os de máquina virtual, que possibilitaram a execução de programas na máquina do usuário, quase que de maneira independente do equipamento e sistema operacional usados pelo usuário. As **regras de uso e design** para estes casos **são as mais variadas** e dependem do tipo de site e aplicação que se deseja criar.

As páginas Web Ativas são **sempre "client-side"**, ou seja, rodam na máquina cliente. Isso significa que as respostas às **interações** não dependem mais do tempo da rede, tornando-as muito **mais rápidas**, além de **reduzir a carga no servidor**. As **tecnologias** básicas do desenvolvimento atual de páginas ativas são o **Java e Flash**. **Bancos de Dados** também são usados, embora de forma menos extensiva (sob pena de aumentar muito a carga no servidor).

As **vantagens das páginas Web Dinâmicas** estão praticamente todas presentes nas páginas Web Ativas. Além disso, são vantagens adicionais a **total flexibilidade de interação e layout**, o que combinado à **velocidade de resposta** às ações do usuário e **uso extensivo de áudio e vídeo** podem melhorar muito a experiência do usuário.

Estas tecnologias têm sido muito usadas na **criação de aplicativos sob demanda**, em que se paga pelo tempo de uso do aplicativo, por exemplo (ou não se paga nada, como nos milhares de jogos em flash disponíveis na internet).

Infelizmente a **Web Ativa também apresenta alguns problemas**. O principal deles, sob a óptica do usuário, é a **falta de padronização das páginas** feitas usando as tecnologias de Web Ativa. Cada página tem um tipo de interação, um tipo de visual, etc... obrigando que o usuário tenha de "aprender" a usar cada nova página que encontrar.

Além disso, como as aplicações rodam em "máquinas virtuais" no lado do computador cliente, é necessário que a máquina virtual adequada, da versão correta, esteja instalada no equipamento do usuário, o que nem sempre é possível, gerando um **problema de compatibilidade**. Por esta razão, é comum - e até mesmo desejável - que sites do tipo "ativo" possuam também uma versão do tipo "dinâmico".

O **uso de banco de dados** também **não costuma ser expressivo** nestas aplicações, para evitar o problema da sobrecarga do servidor e lentidão. Entretanto, esta postura costuma reduzir a qualidade deste tipo de site, sob a óptica de muitos usuários.



#### **4. Páginas Web Mistas**

A mais recente evolução, ainda em andamento, é na realidade um uso misto de web dinâmicas e ativa a um só tempo, tentando combinar o melhor de dois estágios de evolução da web.

O objetivo é prover toda a flexibilidade de interação e layout disponível com as páginas web ativas, mas mantendo uma alta compatibilidade, o que é alcançado com uma parte do processamento no servidor e, a parte que é processada no cliente, usa tecnologias padronizadas e amplamente disponíveis.

Ainda é cedo para dizer se essa nova "evolução" irá se firmar como um novo estágio de desenvolvimento da web. Entretanto, milhares de páginas usando tecnologias mistas (como Ajax) já estão disponíveis na web e normalmente o usuário sequer sabe que está executando uma página que usa esse tipo de tecnologia.

#### **5.Bibliografia**

FOLEY NIELSEN, J. *Projetando Websites*. Ed. Campus, 2000.

TANENBAUM, A. S. *Redes de Computadores*. Editora Campus, 2003.

DUNCAN, C.B. *Flash MX: Criação e Desenvolvimento de Web Sites*. Editora Futura, 2003.

Notas da Aula 03: Web Estática: HTML Parte 1  
Prof. Daniel Caetano

**Objetivo:** Introduzir a linguagem HTML, estruturação de uma página e algumas tags iniciais.

### Introdução

Desde sua criação até os dias atuais, a **Web evoluiu** muito. Grande parte desta evolução só foi possível porque a **linguagem usada** para descrever as páginas da Web **também evoluiu**.

Com o surgimento de **novos recursos** nesta linguagem de descrição, a **maneira de utilizá-la corretamente** também **sofreu mudanças** profundas com o passar dos anos, fazendo com que muitos profissionais ficassem desatualizados, usando técnicas há muito desaconselhadas pelo W3C, o World Wide Web Consortium, responsável pela padronização da Web.

A partir desta aula, veremos a maneira "correta" de usar a linguagem de descrição de páginas Web para garantir que nossas páginas e web sites sejam flexíveis e facilmente administráveis.

### 1. A Origem da Linguagem de Descrição de Páginas Web

A linguagem de descrição de páginas Web foi inventada no início da década de 1990, por Tim Berners Lee, **baseada em uma linguagem chamada SGML** (Standard Generalized Markup Language - Linguagem de Marcação Geral Padronizada). A SGML serve para **criar documentos** que sejam **legíveis tanto por seres humanos quanto por máquinas**. O método para atingir este objetivo é indicar qual a função de cada trecho de texto através de **"tags"**, que são indicadores separados pelos caracteres "<" e ">", como por exemplo: **<QUOTE>Texto</QUOTE>**. Observe que existe um marcador de início e um marcador de fim, sendo que neste último um caractere "/" precede o nome do indicador.

Entretanto, como o próprio nome da SGML diz, ela é "geral", ou seja, serve para muitas coisas. Por esta razão, ela é de **"compreensão" relativamente complexa** tanto para humanos quanto para máquinas. Assim, quando Tim Berners Lee pensou na Web, ele fez uma **aplicação da SGML**, simplificando-a para os usos que ele tinha em mente: documentos em hipertexto. Por esta razão, Lee chamou essa **linguagem** de **HTML**: HyperText Markup Language - Linguagem de Marcação de HiperTexto.

A idéia do HTML é **usar "tags"** (que serão vistas em breve) **para indicar qual trecho de um texto é um título**, qual é um **subtítulo**, qual é um **parágrafo**, qual é um **endereço de e-mail**, qual é um trecho de um programa de computador, qual é uma citação, qual é um link... **e assim por diante**.

### 1. Estrutura de um Documento HTML

Antes de apresentarmos as tags citadas acima, é importante apresentar a **estrutura de um documento HTML**. Por quê? Porque se a estrutura de um documento HTML não estiver correta, corre-se o risco de o navegador não interpretá-la corretamente.

1) Todo documento HTML deve ser precedido pela tag **<HTML>** e seu fim deve ser indicado pela tag **</HTML>**. Em tese, tudo que estiver fora destes indicadores deve ser ignorado pelo navegador.

2) Todo documento HTML tem duas seções: a primeira delas, chamada de cabeçalho (**head**), contém informações sobre a página Web e a segunda delas, chamada corpo (**body**), contém o texto da página Web propriamente dita. A primeira seção é delimitada pelas tags **<HEAD>** e **</HEAD>**. A segunda seção é delimitada pelas tags **<BODY>** e **</BODY>**.

Assim, a página Web mais simples possível - que é uma página Web vazia - tem essa cara:

```
<HTML>
  <HEAD>
    [Informações sobre a Página]
  </HEAD>
  <BODY>
    [Texto da Página Propriamente Dita]
  </BODY>
</HTML>
```

O conteúdo da **seção <HEAD>** não é para o usuário, e sim para o programa navegador. Por esta razão, o conteúdo da seção **<HEAD>** não é mostrado para o usuário. Já o conteúdo da seção **<BODY>** é o conteúdo que o navegador mostrará para o usuário como sendo a página Web.

Observe que tanto a seção **<HEAD>** quanto a seção **<BODY>** estão inteiramente contidas entre as tags **<HTML>** e **</HTML>**. Caso isso não fosse respeitado, a página poderia apresentar problemas em alguns navegadores.

Esta ordenação de tags **<HTML>**, **<HEAD>** e **<BODY>** é a estrutura fundamental de uma página Web e toda página será iniciada exatamente com esta estrutura.

## 2. O Cabeçalho da Página

Na seção anterior apresentamos as duas principais seções de um documento HTML, como uma página Web. A primeira delas é o **Cabeçalho** e, como foi dito anteriormente, esta seção **contém informações para o navegador**, que pode ser seu Firefox, Internet Explorer ou Opera... mas também pode ser um navegador "spider" de algum serviço de busca como o Google.

Cada um destes navegadores busca coisas distintas neste cabeçalho, mas uma coisa que **todos eles investigam** é o *título da página*. Em HTML, identificamos o título da página, dentro do cabeçalho, pelas tags **<TITLE>** e **</TITLE>**. Por exemplo:

```
<HTML>
  <HEAD>
    <TITLE>Título da página Web!</TITLE>
  </HEAD>
  <BODY>
    Texto da página Web!
  </BODY>
</HTML>
```

Se este texto for gravado com o nome de "pagina.html" e este arquivo for aberto no navegador, você poderá observar que o texto "Título da página Web!" não aparece em lugar algum na página. Mas ele foi parar em um lugar especial: na **barra de título do navegador**.

Caso você faça um **"bookmark"** desta página, ou seja, coloque-a nos seus **"favoritos"**, também será este texto entre **<TITLE>** e **</TITLE>** que irá ser indicado nos favoritos.

Existem **outras indicações** que podem ser colocadas no cabeçalho, como o **nome do autor da página**, **data da última atualização**, indicações para o navegador recarregar a página depois de um certo número de segundos, indicações se o navegador deve usar cache para a página atual ou não... **etc.** Mas por enquanto vamos nos limitar à informação do título que, de qualquer forma, é a mais importante de todas as informações que podem figurar no cabeçalho.

## 3. O Corpo da Página

O corpo da página é o trecho delimitado pelas tags **<BODY>** e **</BODY>**. Tudo que aparece na página em si deve estar indicado nesta região, onde a função das tags é **explicar para o navegador qual é o conteúdo da página**.

Uma vez que o principal formato de conteúdo nas páginas Web é o formato texto, uma das mais importantes tags do HTML é a **tag de Parágrafo: <P>**. Como um parágrafo tem um início e um fim, então existe também uma **tag de fechamento de parágrafo: </P>**.

Assim, podemos indicar um parágrafo da forma correta em uma página, como no exemplo abaixo:

```
<HTML>
  <HEAD>
    <TITLE>Título da página Web!</TITLE>
  </HEAD>
  <BODY>
    <P>Texto da página Web!</P>
  </BODY>
</HTML>
```

Uma coisa importante que convém mencionar é que o **HTML ignora as quebras de linha**. Então, escrever:

```
<P>Texto da página Web!</P>
```

Tem o mesmo efeito de escrever:

```
<P>
Texto da página Web!
</P>
```

Nas próximas seções e aulas veremos muitas outras tags que podem ser usadas em um documento HTML.

#### 4. Usando Caracteres Acentuados

A **página** da seção **anterior**, apesar de simples, **tem um problema sério**, que a torna incompatível com muitos navegadores: ela **contém acentuação**. É importante lembrar que a língua usada em muitos países do mundo não faz uso de acentuação e, nestes lugares, os computadores não reconhecem essa acentuação.

Para evitar este problema, o **HTML tem uma maneira de indicar caracteres de acentuação**, de forma que os navegadores do mundo todo possam apresentar os símbolos de acentuação corretamente.

A maneira de indicar estes caracteres é através de uma indicação especial, que começa com o caractere **&**, é **seguido pela letra** que recebe o acento e, em seguida, temos **o nome do acento** - em francês, **terminando** a seqüência **com um caractere ;**. Por **exemplo**:

Tipo de Acento	Exemplo	Resultado
- Agudo: <b>&amp;_acute;</b>	<b>&amp;acute;</b>	é
- Crase: <b>&amp;_grave;</b>	<b>&amp;Agrave;</b>	À
- Circunflexo: <b>&amp;_circ;</b>	<b>&amp;ocirc;</b>	ô
- Trema: <b>&amp;_uml;</b>	<b>&amp;uuml;</b>	ü
- Tilde: <b>%_tilde;</b>	<b>&amp;atilde;</b>	ã
- Cedilha: <b>&amp;_cedil;</b>	<b>&amp;ccedil;</b>	ç

## 5. Mais Algumas Tags Básicas

**Manchetes/Títulos/SubTítulos:** O HTML tem tags para especificar trechos de um texto que são manchetes, títulos ou, ainda, subtítulos. Na realidade, **existem 6 níveis de títulos, sendo que o nível 1 tem maior evidência até o nível 6, de menor evidência**. Por **exemplo**:

```
<H1>1. Seção H1</H1>  
  <H2>1.1. Seção H2</H2>  
    <H3>1.1.1. Seção H3</H3>  
      <H4>1.1.1.1. Seção H4</H4>  
        <H5>1.1.1.1.1. Seção H5</H5>  
          <H6>1.1.1.1.1.1. Seção H6</H6>
```

Isso apareceria na tela mais ou menos assim:

**1. Seção H1**  
**1.1. Seção H2**  
**1.1.1. Seção H3**  
**1.1.1.1. Seção H4**  
**1.1.1.1.1. Seção H5**  
**1.1.1.1.1.1. Seção H6**

Estas tags **devem** ser usadas com sabedoria para **indicar as diversas seções do texto** da página. Bons navegadores poderão, no futuro, fazer **índices automáticos** com estas

seções, além de **manter uma compatibilidade** com os já existentes navegadores para cegos, que lêem estes títulos para que a pessoa escolha qual das seções quer ouvir primeiro.

**Quebra de Linha:** Apesar de praticamente não ter uso, existe uma **tag no HTML** para quando se deseja **forçar a quebra de uma linha** em uma determinada posição. Esta tag é **<BR>**, de line Break (quebra de linha, em inglês). Por se tratar de uma intervenção pontual, não existe tag de fechamento para <BR>. **Exemplo** de uso:

**<P>Este parágrafo será quebrado no meio, bem aqui<BR>e continua depois</P>**

O que será apresentado da seguinte forma:

Este parágrafo será quebrado no meio, bem aqui  
e continua depois

**Linha de Separação:** Um recurso muito usado para tornar claro quando uma seção termina e onde outra começa costuma ser a **linha de separação horizontal**. O HTML tem uma **tag para apresentar este tipo de separador: <HR>**, de Horizontal Ruler (Régua Horizontal em inglês). Exemplo de uso:

**<H1>1. Seção H1</H1>**  
    **<H2>1.1. Seção H2</H2>**  
**<HR>**  
**<H1>2. Seção H1</H1>**  
    **<H2>2.1. Seção H2</H2>**

Será apresentado da seguinte forma:

**1. Seção H1**  
**1.1. Seção H2**  
**1.1.1. Seção H3**

---

**2. Seção H1**  
**2.1. Seção H2**

## 6. Exercícios

Nesta aula, vimos o uso das seguintes tags:

<HTML>, <HEAD>, <TITLE>, <BODY>, <P>, <H1> a <H6>, <BR> e <HR>

Usemos estas tags para criar a página abaixo, que deve mostrar na barra de título do navegador o seguinte título: "Primeira página Web Estática". Ao criar a página, lembre-se de usar acentuação HTML. **Não** use <BR>.

Página a ser criada:

### **Primeira página HTML**

---

#### **Primeira Seção**

A primeira seção irá conter dois parágrafos. Note, também, que o título desta seção é em H2, sendo que o título da página foi em H1.

Este é o segundo parágrafo da primeira seção.

#### **Segunda Seção**

Esta é a segunda seção e também contém dois parágrafos. O título desta segunda seção também é em H2.

Este é o segundo parágrafo da segunda seção.



**Solução:**

```
<HTML>
<HEAD>
  <TITLE>Primeira página Web Estática</TITLE>
</HEAD>
<BODY>
  <H1>Primeira página HTML</H1>
  <HR>
  <H2>Primeira Seção</H2>
  <P>
    A primeira seção irá conter dois parágrafos. Note,
    também, que o título desta seção é em H2, sendo que o
    título da página foi em H1.
  </P>
  <P>
    Este é o segundo parágrafo da primeira seção.
  </P>
  <H2>Segunda Seção</H2>
  <P>
    Esta é a segunda seção e também contém dois
    parágrafos. O título desta seção também é em
    H2.
  </P>
  <P>
    Este é o segundo parágrafo da segunda seção.
  </P>
</BODY>
</HTML>
```

**7. Bibliografia**

RAMALHO, J.A. *HTML 4 Prático e Rápido*. Editora Berkeley, 1999.

BOENTE, A. *Programação Web Sem Mistérios*. Editora Brasport, 2006.

NIELSEN, J. *Projetando Websites*. Ed. Campus, 2000.

Notas da Aula 04: Web Estática: HTML Parte 2  
Prof. Daniel Caetano

**Objetivo:** Apresentar os tags fundamentais, de imagem e links.

### **Introdução**

Na aula passada foram apresentados tags muito importantes, mas talvez os dois tags mais importantes do HTML ainda não foram apresentados. Estes tags são aqueles relacionados à apresentação de imagens e links. Nesta aula, serão apresentados seus usos básicos, além de alguns outros tags adicionais.

### **1. As Tags de Links**

Como é de conhecimento geral, toda página na internet tem um "endereço". São endereços, por exemplo:

[www.uol.com.br](http://www.uol.com.br)  
[www.terra.com.br](http://www.terra.com.br)  
[www.w3.org](http://www.w3.org)

Na realidade, estes endereços não estão completos, pois não estamos nos referindo ao nome da homepage. No Terra, por exemplo, o endereço completo seria:

[www.terra.com.br/capa/index.html](http://www.terra.com.br/capa/index.html)

Neste caso, "www.terra.com.br" é o nome do servidor, "capa" é um diretório e "index.html" é o nome da página home, um arquivo em linguagem HTML.

Por padrão, ao navegar para um endereço, o navegador "ancorará" no topo da página, mas é possível também acrescentar "ancoradouros" no meio de uma página, permitindo que o navegador entre diretamente em pontos intermediários da mesma.

Como o "ancoradouro" padrão é no topo de uma página, quando a carregarmos, automaticamente ela será apresentada a partir de seu topo, ou seja, de seu início. Por exemplo, se entrarmos na página a seguir:

[pt.wikipedia.org/wiki/Engenharia\\_de\\_sistemas](http://pt.wikipedia.org/wiki/Engenharia_de_sistemas)

Seremos apresentados à página sobre "Engenharia de Sistemas" na WikiPedia, desde seu topo. Entretanto, se usarmos o endereço abaixo:

[pt.wikipedia.org/wiki/Engenharia\\_de\\_sistemas#Processo\\_de\\_Software](http://pt.wikipedia.org/wiki/Engenharia_de_sistemas#Processo_de_Software)

Iremos diretamente para a seção "Processo de Software" da página anteriormente indicada. Note que houve uma adição no endereço: **#Processo\_de\_Software**. Este é nome do "ancoradouro" intermediário para o qual eu desejo ir diretamente.

Bem, entendida a parte dos endereços e "ancoradouros"... o que são então os link? Ora, links nada mais são do que um "texto que indica outro ancoradouro" para o qual o browser possa navegar.

Isso significa que, com um link, podemos indicar:

- 1) O endereço do topo de uma outra página;
- 2) O endereço de um ancoradouro intermediário de uma outra página;
- 2) O endereço de um ancoradouro intermediário da página atual.

A tag do HTML usada para fazer isso é a tag "âncora" `<A>...</A>`. Esse nome vem da analogia já apresentada, do termo "navegação pela web".

Entretanto, se indicarmos algo como

`<A>Um Link</A>`

Isso será apresentado corretamente pelo navegador como:

[Um Link](#)

Entretanto, ao clicar no mesmo, nada acontece. Por quê? Bem, isso ocorre porque não definimos o endereço do próximo ancoradouro! Para isso, precisamos usar *um parâmetro* dentro da tag `<A>`. Este parâmetro, **HREF**, é usado para indicar uma referência para outro arquivo HTML na rede, onde o navegador irá ancorar. Assim, podemos colocar no corpo de uma página o seguinte:

```
<HTML>
  <HEAD>
    <TITLE>Teste de Link</TITLE>
  </HEAD>
  <BODY>
    <P>
      <A HREF="http://www.uol.com.br/">Vai para o UOL</A>:
      UOL, um dos maiores portais do Brasil na Internet.
    </P>
  </BODY>
</HTML>
```

O que será apresentado da seguinte forma:

[Vai para o UOL](http://www.uol.com.br/): UOL, um dos maiores portais do Brasil na Internet.

E clicando no link agora o navegador será redirecionado corretamente para o site do UOL! Bastou indicar corretamente o link e pronto!

Mas e se quiséssemos colocar o link para a seção de Processo de Software, na página de Engenharia de Sistemas da WikiPedia? O processo é o mesmo:

```
<HTML>
  <HEAD>
    <TITLE>Teste de Link 2</TITLE>
  </HEAD>
  <BODY>
    <P>
      <A HREF="pt.wikipedia.org/wiki/Engenharia_de_sistemas#Processo_de_Software">
        Processo de Software</A>: Direto para a WikiPedia.
    </P>
  </BODY>
</HTML>
```

O que será apresentado da seguinte forma:

[Processo de Software](pt.wikipedia.org/wiki/Engenharia_de_sistemas#Processo_de_Software): Direto para a WikiPedia.

Convém aqui fazer um comentário acerca do texto que é convertido em link. Muitas vezes vemos na internet links como "Clique Aqui" ou "Download". Todos que programam páginas Web fazem isso, mas é importante lembrar que isso deve ser evitado.

A primeira razão para isso é filosófica: na filosofia do hipertexto, você não deve ter que modificar um texto para inserir links. Em outras palavras, o texto deve ser escrito como se não existisse link algum, e os links deviam ser naturalmente associados à palavras já existentes.

A segunda razão para isso é social: navegadores para deficientes visuais e deficientes físicos, que dependem da voz para selecionar links, normalmente apresentam uma lista de links em separado (seja na forma textual, seja na forma verbal), e o usuário dita qual dos links quer entrar. Agora, imagine se a lista de links tiver essa cara:

Clique Aqui  
Clique Aqui  
Clique Aqui  
Download  
Clique Aqui  
Clique Aqui  
Download  
Clique Aqui  
Download

Fica um tanto complicado de selecionar, não é? Por estas razões, tentemos sempre usar textos elucidativos nos links. Acredite em uma coisa: se o usuário está habituado à internet (e hoje todos estão), se existe um trecho diferenciado no meio de um texto, ele já sabe que é um link e sabe que pode clicar lá. Ninguém precisa dizer para ele "clique aqui".

### 1.1. Definindo um Acoradouro no Meio de uma Página Web

Na seção anterior vimos como apontar para outra página ou até mesmo para um ancoradouro dentro de uma página. Mas como podemos inserir um acoradouro no meio de nossa própria página?

É bastante simples e para isso também se usa a tag **<A>**, entretanto neste caso não usamos o parâmetro HREF e sim o parâmetro **NAME**, que definirá o nome deste ancoradouro. Neste caso, não há a necessidade de "fechar" a tag com o **</A>**. Assim, para definir um ancoradouro chamado "Meu\_Perfil" no meio de uma página, use a seguinte construção HTML:

```
<HTML>
  <HEAD>
    <TITLE>Teste de Acoradouro</TITLE>
  </HEAD>
  <BODY>
    <P>
      Aqui come&ccedil;a a p&aacute;tina, com uma s&eacute;rie de
      informa&ccedil;&otilde;es interessantes que se estendem por uma
      centena de linhas e tal.
    </P>
    <A NAME = "Meu_Perfil">
      <P>
        Aqui come&ccedil;a o seu perfil.
      </P>
    </BODY>
</HTML>
```

O que será apresentado da seguinte forma, se for digitado seu endereço, como por exemplo <http://www.aluno.com/index.html> :

Aqui começa a página, com uma série de informações interessantes que se estendem por uma centena de linhas e tal.

Aqui começa o seu perfil.

Note que nada apareceu no lugar em que definimos o ancoradouro. Entretanto, se o usuário entrar na sua página com o endereço [http://www.aluno.com/index.html#Meu\\_Perfil](http://www.aluno.com/index.html#Meu_Perfil), o resultado será o apresentado abaixo: :

Aqui começa o seu perfil.

O usuário poderá rolar a tela para cima e ainda verá o texto introdutório, mas ao entrar, ele caiu diretamente em seu perfil.

## 1.2. Adicionando Títulos Explicativos aos Links

No caso de links que apontam outras páginas ou outros ancoradouros, é possível adicionar um texto explicativo ao link, de forma que este texto só seja apresentado se o usuário mantiver o ponteiro do mouse sobre o link por alguns instantes.

Este tipo de texto é interessante para que o usuário obtenha mais informações sobre o que vai encontrar "do outro lado do link", ou seja, no próximo ancoradouro, para saber se vale a pena navegar até lá. Para conseguir isso, usamos um outro parâmetro na tag `<A>...</A>`: o parâmetro **TITLE**. Ele é usado da seguinte forma:

```
<HTML>
  <HEAD>
    <TITLE>Teste de Link 3 - Com Título Explicativo</TITLE>
  </HEAD>
  <BODY>
    <P>
      <A HREF="http://www.uol.com.br/" TITLE="Universo OnLine"> Vai
      para o UOL</A>: UOL, um dos maiores portais do Brasil na Internet.
    </P>
  </BODY>
</HTML>
```

O que será apresentado da seguinte forma:

[Vai para o UOL](http://www.uol.com.br/): UOL, um dos maiores portais do Brasil na Internet.

Observe que a aparência é absolutamente a mesma de quando não definimos o título. Entretanto, se repousarmos o ponteiro do mouse por alguns segundos sobre este link, uma balãozinho de informações aparecerá com o texto "Universo OnLine", que definimos como o título do link.

## 2. Adicionando Imagens à Página Web

Um dos grandes atrativos da Web sempre foi sua capacidade de apresentar imagens. O uso de imagens torna uma página mais atrativa, pode auxiliar a explicar um assunto de forma mais elucidativa... Por outro lado, o uso exagerado e inadequado de imagens pode tornar o carregamento de uma página excessivamente lento. Assim, é importante saber usar as imagens e usá-las com parcimônia.

A tag usada para inserir uma imagem é a tag **<IMG>**. A tag `<IMG>` (de IMAge, ou IMAgem em português) não necessita um fechamento `</IMG>`, uma vez que por definição uma imagem já tem início e fim pré-definidos.

Entretanto, a tag de imagem exige pelo menos um parâmetro, que indica **onde** está essa imagem, que pode estar tanto no disco, juntamente com o arquivo HTML da página ou até mesmo em

outro lugar da web. O parâmetro usado para indicar a localização da imagem é o parâmetro **SRC**. Assim, o uso mais comum de uma figura é feito da seguinte forma:

```
<HTML>
  <HEAD>
    <TITLE>Teste de Imagem 1</TITLE>
  </HEAD>
  <BODY>
    <IMG SRC="http://www.caetano.eng.br/main/images/aflogo_horiz.gif">
  </BODY>
</HTML>
```

O que será apresentado da seguinte forma:



Da mesma maneira que as âncoras (links), também é possível definir um texto explicativo para uma figura, caso o usuário deixe o ponteiro do mouse sobre a figura por alguns instantes, usando o parâmetro **TITLE**.

Um outro parâmetro importante é o parâmetro **ALT**, que indica um texto a ser apresentado no lugar da figura, caso o navegador seja modo texto ou esteja com a apresentação de figuras desabilitada. O texto do **ALT** também é importante em navegadores para deficientes visuais, pois é o texto do **ALT** que é lido para descrever a figura. Assim, a página abaixo:

```
<HTML>
  <HEAD>
    <TITLE>Teste de Imagem 2</TITLE>
  </HEAD>
  <BODY>
    <IMG SRC="http://www.caetano.eng.br/main/images/aflogo_horiz.gif"
      TITLE="Empresa do professor" ALT="Amusement Factory Logo">
  </BODY>
</HTML>
```

Se for executada num navegador modo texto, será apresentada da seguinte forma:

[IMG] Amusement Factory Logo

## **2.1. Acelerando o Carregamento de Páginas com Imagens**

Muita vezes, quando o navegador vai carregar uma página com muitas imagens, ele não é capaz de mostrar **nada** da página até que carregue todas as imagens. Isso ocorre porque, para

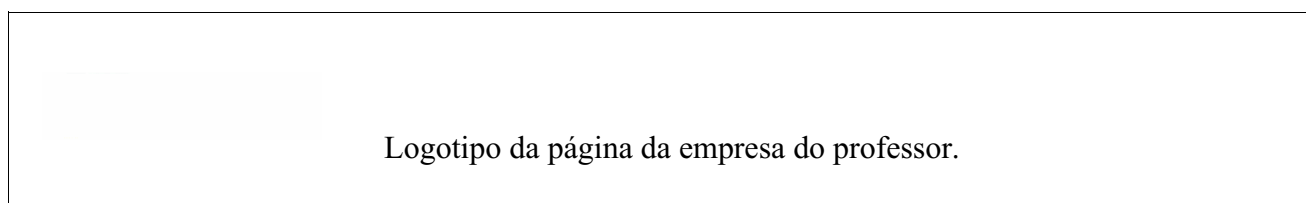
distribuir e desenhar o conteúdo textual da página, o navegador precisa primeiro saber qual será o tamanho e posição das figuras a serem apresentadas.

Entretanto, para saber o tamanho das figuras, ele precisa carregá-las primeiro, o que pode ser bem demorado, dependendo do tamanho das imagens. Porém, existe uma forma de ajudar o navegador a saber o tamanho das imagens, fazendo com que ele apresente o texto da página tão logo termine de carregar o arquivo .html, antes mesmo de baixar as imagens contidas na página, se serão preenchidas posteriormente.

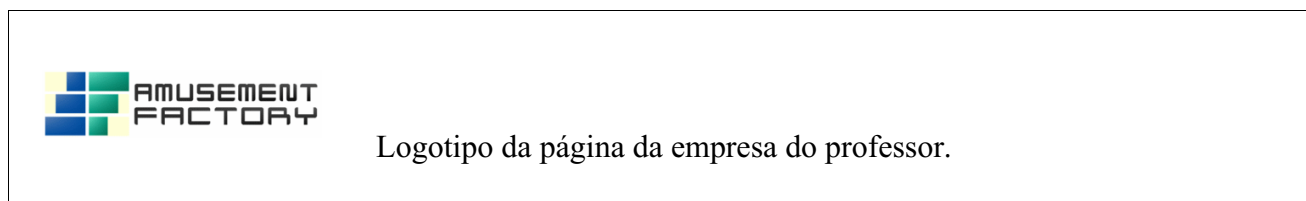
Para conseguir isso, basta indicar a largura da imagem pelo parâmetro WIDTH e a altura da imagem, pelo parâmetro HEIGHT. Vejamos o exemplo abaixo:

```
<HTML>
  <HEAD>
    <TITLE>Teste de Imagem 2</TITLE>
  </HEAD>
  <BODY>
    <P>
      <IMG SRC="http://www.caetano.eng.br/main/images/aflogo_horiz.gif"
      WIDTH="330" HEIGHT="80" TITLE="Empresa do professor"
      ALT="Amusement Factory Logo"> Logotipo da p&aacute;gina da
      empresa do professor.
    </P>
  </BODY>
</HTML>
```

Caso a página seja carregada em uma conexão lenta, será possível observar duas etapas. Na primeira delas, o seguinte conteúdo será apresentado:



Sendo, então, a imagem preenchida posteriormente:



É claro que, em uma página com uma única (e pequena) figura, a diferença de tempo entre a apresentação do texto é praticamente imperceptível. Entretanto, em páginas com muitas figuras grandes, a diferença é bastante sensível e recomenda-se que sempre sejam indicadas as dimensões da figura.



Mas o que acontece se usarmos as dimensões erradas? Neste caso, a imagem será redimensionada pelo navegador para aparecer do tamanho especificado. Por exemplo:

```
<HTML>
  <HEAD>
    <TITLE>Teste de Imagem 2</TITLE>
  </HEAD>
  <BODY>
    <P>
      <IMG SRC="http://www.caetano.eng.br/main/images/aflogo_horiz.gif"
      WIDTH="660" HEIGHT="80" TITLE="Empresa do professor"
      ALT="Amusement Factory Logo"> Logotipo da p&aacute;gina da
      empresa do professor.
    </P>
  </BODY>
</HTML>
```

Será apresentado como:



Logotipo da página da empresa do professor.

### 3. Exercícios

Nesta aula, vimos o uso das seguintes tags:

<IMG> e <A>

Usemos estas tags para acrescentar à página da aula anterior a imagem [http://www.caetano.eng.br/main/images/aflogo\\_horiz.gif](http://www.caetano.eng.br/main/images/aflogo_horiz.gif) no topo da primeira seção e um link para a página da disciplina <http://www.caetano.eng.br/aulas/fb/Web/> ao final da segunda seção.

O resultado final deve ser similar ao apresentado abaixo:

## Primeira página HTML

---

### Primeira Seção



A primeira seção irá conter dois parágrafos. Note, também, que o título desta seção é em H2, sendo que o título da página foi em H1.

Este é o segundo parágrafo da primeira seção.

### Segunda Seção

Esta é a segunda seção e também contém dois parágrafos. O título desta segunda seção também é em H2.

Este é o segundo parágrafo da segunda seção.

Mais informações na [página da disciplina Programação Web](#).

**Solução:**

```
<HTML>
<HEAD>
  <TITLE>Primeira página Web Estática</TITLE>
</HEAD>
<BODY>
  <H1>Primeira página HTML</H1>
  <HR>
  <H2>Primeira Seção</H2>
  <IMG SRC="http://www.caetano.eng.br/main/images/aflogo_horiz.gif"
    WIDTH="330" HEIGHT="80" TITLE="Empresa do professor"
    ALT="Amusement Factory Logo">
  <P>
    A primeira seção irá conter dois parágrafos. Note,
    também, que o título desta seção é em H2, sendo que o
    título da página foi em H1.
  </P>
  <P>
    Este é o segundo parágrafo da primeira seção.
  </P>
  <H2>Segunda Seção</H2>
  <P>
    Esta é a segunda seção e também contém dois
    parágrafos. O título desta seção também é em H2.
  </P>
  <P>
    Este é o segundo parágrafo da segunda seção.
  </P>
  <P>
    Mais informações na
    <A HREF="http://www.caetano.eng.br/aulas/fb/Web/" TITLE="Disciplina de
      Programação Web">página da disciplina
      Programação Web</A>.
  </P>
</BODY>
</HTML>
```

#### **4. Bibliografia**

RAMALHO, J.A. *HTML 4 Prático e Rápido*. Editora Berkeley, 1999.

BOENTE, A. *Programação Web Sem Mistérios*. Editora Brasport, 2006.

NIELSEN, J. *Projetando Websites*. Ed. Campus, 2000.

Notas da Aula 05: Web Estática: HTML Parte 3  
Prof. Daniel Caetano

**Objetivo:** Apresentar os tags adicionais do HTML.

### **Introdução**

Nas aulas anteriores foram apresentados os tags mais importantes do HTML. Entretanto, o HTML possui uma série de outros tags para indicar muitos trechos específicos de texto, bem como algumas estruturas especiais.

### **1. Tags para Listas**

Um dos recursos adicionais mais usados do HTML é a capacidade de exibir listas. Existem basicamente dois tipos de listas: as ordenadas (numeradas) e as não ordenadas (listas de "bullets"). Por exemplo:

Lista Ordenada:

1. Primeiro nível
- 1.1. Segundo nível.
2. Primeiro nível novamente.
- ...

Lista Não Ordenadas:

- Primeiro item
- Segundo item
- Terceiro item
- ...

Existem ainda as Listas de Definição <DL>...</DL> ou Listas de Definição de Termos <DT>...</DT>, onde cada item é marcado por um <DD>...</DD>. Seu uso é análogo às apresentadas abaixo.

#### **1.1. Listas Não Ordenadas**

Uma lista não ordenada sempre será demarcada pelas tags <UL> ... </UL> (UL = Unordered List, ou Lista Não Ordenada). Adicionalmente, cada elemento de lista deve ser delimitado pelas tags <LI> ... </LI> (LI = List Item, ou Item de Lista). Pode-se usar as tags <LH>...</LH> para marcar títulos de uma lista.

Assim, se quisermos indicar uma lista não ordenada, em HTML a especificaremos assim:

```
<UL>
  <LI>Um item.</LI>
  <LI>Outro item.</LI>
  <LI>Mais outro item.</LI>
</UL>
```

O que será apresentado assim:

- Um item.
- Outro item.
- Mais outro item.

É possível indicar uma lista dentro de outra:

```
<UL>
  <LI>Um item.</LI>
  <UL>
    <LI>Um sub-item.</LI>
    <LI>Outro sub-item.</LI>
  </UL>
  <LI>Mais outro item.</LI>
</UL>
```

O que será apresentado assim:

- Um item.
  - Um sub-item.
  - Outro sub-item.
- Mais outro item.

## **1.2. Listas Ordenadas**

As listas ordenadas são exatamente iguais às listas não-ordenadas, mas ao invés de serem demarcadas pelas tags `<UL> ... </UL>`, são demarcadas pelas tags `<OL> ... </OL>` (OL = Ordered List, ou Lista Ordenada). Os elementos de lista também devem ser

delimitados pelas tags **<LI>** ... **</LI>**. Assim, se quisermos indicar uma lista ordenada, em HTML a especificaremos assim:

```
<OL>
  <LI>Um item.</LI>
  <LI>Outro item.</LI>
  <LI>Mais outro item.</LI>
</OL>
```

O que será apresentado assim:

1. Um item.
2. Outro item.
3. Mais outro item.

É possível indicar uma lista dentro de outra:

```
<OL>
  <LI>Um item.</LI>
  <OL>
    <LI>Um sub-item.</LI>
  </OL>
  <LI>Mais outro item.</LI>
  <UL>
    <LI>Outro sub-item.</LI>
  </UL>
</OL>
```

O que será apresentado assim:

1. Um item.
  - 1.1 Um sub-item.
2. Mais outro item.
  - Outro sub-item.

## **2. Tabelas**

Um recurso muito útil e importante no HTML - porém freqüentemente muito mal utilizado - é o de apresentação de tabelas. Entretanto, para que uma tabela seja apresentada rapidamente pelo navegador, ela precisa estar completamente definida, algo que poucos programadores HTML fazem.

Uma tabela é basicamente demarcada pelas tags **<TABLE> ... </TABLE>**. Dentro destas tags, temos duas seções: a seção **<THEAD>...</THEAD>**, onde teremos a tag **<CAPTION> ... </CAPTION>**, que serve para indicar a legenda da tabela, e uma outra seção: **<TBODY> ... </TBODY>**, onde ficam as linhas de informação da tabela. Existe também os **<TFOOT>...</TFOOT>**, mas são pouco usados. Ou seja, até agora temos:

```
<TABLE>
  <THEAD>
    <CAPTION>Tabela 1: Uma tabela</CAPTION>
  </THEAD>
  <TBODY>
    ...
  </TBODY>
</TABLE>
```

Dentro do **<TBODY> ... </TBODY>**, cada linha da tabela tem sua estrutura própria e deve ficar demarcada pelas tags **<TR> ... </TR>** (Table Row, ou Linha de Tabela). Assim, se nossa tabela terá 3 linhas, podemos escrever sua estrutura da seguinte forma:

```
<TABLE>
  <THEAD>
    <CAPTION>Tabela 1: Uma tabela</CAPTION>
  </THEAD>
  <TBODY>
    <TR>
      ... [ linha 1 ]
    </TR>
    <TR>
      ... [ linha 2 ]
    </TR>
    <TR>
      ... [ linha 3 ]
    </TR>
  </TBODY>
</TABLE>
```

Dentro das linhas iremos colocar as "células" de nossa tabela. Uma célula pode ser de um de dois tipos: uma **célula título** ou uma **célula de dados**. No primeiro caso, delimitamos a informação com as tags **<TH> ... </TH>** (de Table Heading). No segundo, em caso de células de dados, delimitamos a informação com as tags **<TD> ... </TD>** (de Table Data).

Assim, se na primeira linha tivermos títulos de coluna e nas outras duas linhas tivermos dados, numa tabela com 2 colunas, o código fica:

```
<TABLE>
  <THEAD>
    <CAPTION>Tabela 1: Uma tabela</CAPTION>
  </THEAD>
  <TBODY>
    <TR>
      <TH>Título Coluna 1</TH>
      <TH>Título Coluna 2</TH>
    </TR>
    <TR>
      <TD>Coluna 1, Linha 1</TD>
      <TD>Coluna 2, Linha 1</TD>
    </TR>
    <TR>
      <TD>Coluna 1, Linha 2</TD>
      <TD>Coluna 2, Linha 2</TD>
    </TR>
  </TBODY>
</TABLE>
```

O que será apresentado da seguinte forma (lembrando que, por padrão, as linhas das tabelas não vão aparecer. Veremos como acrescentar as linhas posteriormente):

Tabela 1: Uma tabela

<b>Título Coluna 1</b>	<b>Título Coluna 2</b>
Coluna 1, Linha 1	Coluna 2, Linha 1
Coluna 1, Linha 2	Coluna 2, Linha 2

Lembrando aqui que é possível colocar uma tabela dentro de outra, como no código abaixo:

```
<TABLE>
  <THEAD>
    <CAPTION>Tabela 2: Uma tabela com outra dentro</CAPTION>
  </THEAD>
  <TBODY>
    <TR>
      <TH>Título Coluna 1</TH>
      <TH>Título Coluna 2</TH>
    </TR>
    <TR>
      <TD>
        <TABLE>
          <TBODY>
            <TR>
```



```

                <TH>Título Sub Coluna 1</TH>
                <TH>Título Sub Coluna 2</TH>
            </TR>
            <TR>
                <TD>Sub Coluna 1, Linha 1</TD>
                <TD>Sub Coluna 2, Linha 1</TD>
            </TR>
        </TBODY>
    </TABLE>
</TD>
<TD>Coluna 2, Linha 1</TD>
</TR>
<TR>
    <TD>Coluna 1, Linha 2</TD>
    <TD>Coluna 2, Linha 2</TD>
</TR>
</TBODY>
</TABLE>

```

E o resultado será como o apresentado abaixo:

Tabela 2: Uma tabela com outra dentro

Título Coluna 1	Título Coluna 2	
Coluna 1, Linha 1	Título Sub Coluna 1	Título Sub Coluna 2
	Sub Coluna 1, Linha 1	Sub Coluna 2, Linha 1
Coluna 1, Linha 2	Coluna 2, Linha 2	

Outra possibilidade é expandir uma linha por duas colunas, usando o parâmetro **COLSPAN** dentro da tag TD ou TH. ou seja: para obter a aparência abaixo, use COLSPAN como aparece no código em seguida:

Tabela 3: Uma tabela com coluna expandida

Título das Colunas	
Coluna 1, Linha 1	Coluna 2, Linha 1

Observe, no código a seguir, o uso de COLSPAN dentro da tag <TH>. O número 2 indica o número de colunas que aquela célula deve ocupar:

```

<TABLE>
    <THEAD>
        <CAPTION>Tabela 3: Uma tabela com coluna expandida</CAPTION>
    </THEAD>
    <TBODY>
        <TR>
            <TH COLSPAN="2">Título das Colunas</TH>
        </TR>

```

```
<TR>
  <TD>Coluna 1, Linha 1</TD>
  <TD>Coluna 2, Linha 1</TD>
</TR>
</TBODY>
</TABLE>
```

O mesmo vale para estender uma célula para ocupar mais de uma linha, bastando usar o parâmetro **ROWSPAN**.

### 3. Tags Diversas

Além das tags já apresentadas, existe ainda um importante conjunto das tags de formulários, mas estas serão vistas numa aula posterior. Abaixo seguem as tags complementares do HTML com uma breve descrição.

**<ABBR>...</ABBR>** - Usado para indicar uma abreviatura, como por exemplo, **<ABBR>Prof.</ABBR>**.

**<ACRONYM>...</ACRONYM>** - Usado para indicar uma sigla, como por exemplo, **<ACRONYM>GNU</ACRONYM>**.

**<ADDRESS>...</ADDRESS>** - Usado para marcar o endereço (de e-mail, por exemplo) do autor da página. Por exemplo: **<ADDRESS>Rua do Limoeiro, 37</ADDRESS>**.

**<BIG>...</BIG>** - Usado para fazer com que um trecho do texto seja apresentado em letras maiores, ressaltadas.

**<BLOCKQUOTE>...</BLOCKQUOTE>** - Usado para marcar citações exatas longas.

**<CITE>...</CITE>** - Usado para marcar um texto como uma citação.

**<CODE>...</CODE>** - Usado para marcar um texto como sendo um código de programação.

**<COMMENT>...</COMMENT>** ou **<!-- ... -->** - Usados para comentários que não devem ser exibidos pelo navegador.

**<DEL>...</DEL>** - Usado para marcar um trecho do texto como não sendo mais válido (riscado).

**<DIV>...</DIV>** - Usado para marcar logicamente uma seção dentro de uma página HTML. Seu uso é muito importante e será visto nas aulas seguintes.

**<EM>...</EM>** - Usado para marcar um texto de forma que ele seja enfatizado.

**<FRAME>...</FRAME>**, **<IFRAME>...</IFRAME>** - Servem para criar divisões físicas no documento. Seu uso não é muito indicado e, por esta razão, não serão analisados em profundidade neste curso.

**<INS>...</INS>** - Marca um texto que deve ser adicionado ao texto. Usado, normalmente, junto com os delimitadores **<DEL>...</DEL>**.

**<LANG>...</LANG>** - (tag de cabeçalho) Usado para indicar a língua e codificação de caracteres usadas na página.

**<META>...</META>** - (tag de cabeçalho) Usado para indicar informações sobre a página web.

**<NOBR>...</NOBR>** - Usado para indicar um trecho de texto que o navegador não deve quebrar no fim de linha.

**<PRE>...</PRE>** - Usado para marcar um texto pré-formatado. Dentro desta região, os "enters" do texto serão interpretados pelo navegador como quebras de linha.

**<Q>...</Q>** - Usado para citações exatas curtas.

**<SMALL>...</SMALL>** - Usado para fazer com que um trecho do texto seja apresentado em letras menores.

**<STRONG>...</STRONG>** - Usado para marcar que trecho de um texto deve estar bastante ressaltado.

**<SUB>...</SUB>** - Usado para colocar índices inferiores (subscrito).

**<SUP>...</SUP>** - Usado para colocar índices superiores (sobrescrito).

**<VAR>...</VAR>** - Usado para marcar uma palavra como uma variável de programa.

**<WBR>...</WBR>** - Usado para indicar locais possíveis de quebra de palavra. Usado normalmente dentro de um **<NOBR>...</NOBR>**.

#### 4. Exercícios

Nesta aula, vimos diversas tags. Usemos estas tags para acrescentar à página da aula algumas características. O resultado final deve ser similar ao apresentado abaixo:

## Primeira página HTML

---

Nesta página você encontra os seguintes tópicos:

- Primeira Seção
- Segunda Seção

### Primeira Seção



A primeira seção irá conter dois parágrafos. Note, também, que o título desta seção é em H2, sendo que o título da página foi em H1.

Este é o segundo parágrafo da primeira seção.

### Segunda Seção

Esta é a segunda seção e também contém dois parágrafos. O título desta segunda seção também é em H2.

Este é o segundo parágrafo da segunda seção.

Tabela 1: Este é o caption da tabela

Coluna 1 <sup>(1)</sup>	Coluna 2
Dado 1	Dado 2

(1) Este é um comentário explicativo sobre a coluna 1 em texto small.

Mais informações na [página da disciplina Programação Web](#).

**Solução:**

```
<HTML>
<HEAD>
  <TITLE>Primeira página Web Estática</TITLE>
</HEAD>
<BODY>
  <H1>Primeira página HTML</H1>
  <HR>
  <P>Nesta página você encontra os seguintes tópicos:</P>
  <UL>
    <LI>Primeira Seção</LI>
    <LI>Segunda Seção</LI>
  </UL>
  <H2>Primeira Seção</H2>
  <IMG SRC="http://www.caetano.eng.br/main/images/aflogo_horiz.gif"
    WIDTH="330" HEIGHT="80" TITLE="Empresa do professor"
    ALT="Amusement Factory Logo">
  <P>
    A primeira seção irá conter dois parágrafos. Note,
    também, que o título desta seção é em H2, sendo que o
    título da página foi em H1.
  </P>
  <P>
    Este é o segundo parágrafo da primeira seção.
  </P>
  <H2>Segunda Seção</H2>
  <P>
    Esta é a segunda seção e também contém dois
    parágrafos. O título desta seção também é em
    H2.
  </P>
  <P>
    Este é o segundo parágrafo da segunda seção.
  </P>
  <P>
    Mais informações na
    <A HREF="http://www.caetano.eng.br/aulas/fb/Web/" TITLE="Disciplina de
      Programações Web">página da disciplina
      Programações Web</A>.
  </P>
  <TABLE>
    <THEAD>
      <CAPTION>Tabela 1: Este é o caption da tabela</CAPTION>
    </THEAD>
```

```
<TBODY>
<TR><TH>Coluna 1<SUP>(1)</SUP></TH><TH>Coluna 2</TH></TR>
<TR><TD>Dado 1</TD><TD>Dado 2</TD></TR>
</TBODY>
</TABLE>
<P><SMALL>(1) Este é um comentário explicativo sobre a coluna 1
em texto small</SMALL></P>
</BODY>
</HTML>
```

#### **4. Bibliografia**

RAMALHO, J.A. *HTML 4 Prático e Rápido*. Editora Berkeley, 1999.

BOENTE, A. *Programação Web Sem Mistérios*. Editora Brasport, 2006.

NIELSEN, J. *Projetando Websites*. Ed. Campus, 2000.

Notas da Aula 06: Web Estática: CSS Parte 1  
Prof. Daniel Caetano

**Objetivo:** Apresentar os conceitos e introduzir as Folhas de Estilo em Cascata (CSSs).

### **Introdução**

Até a presente aula, vimos várias tags do HTML que servem para descrever a função estrutural de cada trecho do texto dentro da página HTML: o que é um título de seção, o que é um subtítulo, o que é parágrafo... e assim por diante.

Muitos alunos devem ter se perguntado por que um site feito desta forma é tão feio, e como é que outros sites na Web são tão coloridos e variados. A resposta para isso é que uma página Web atual não é feita apenas de HTML.

Além do HTML, que descreve os elementos existentes em uma página, é necessário também um arquivo de folhas de estilo, mais conhecido como arquivo CSS (Cascade Style Sheets ou, em bom português, Folhas de Estilo em Cascata).

Nesta e nas próximas aulas estaremos estudando como construir um arquivo CSS e como usá-lo para dar à nossa página a aparência que desejarmos.

### **1. O que é uma Folha de Estilo em Cascata (CSS)?**

Uma folha de estilo em cascata é nada mais do que um arquivo onde são definidas as características de apresentação de cada tipo de estrutura do HTML.

Alguns alunos podem se perguntar: "Mas por que não fazer isso dentro do próprio HTML?". Bem, originalmente era assim que as coisas eram feitas. Aliás, muita gente programa ainda daquela forma, embora a definição de apresentação dentro do HTML seja muito pouco recomendável hoje em dia. É preciso tomar cuidado, já que a grande maioria dos livros ainda ensina HTML "antigo", induzindo os novos programadores aos erros dos velhos programadores.

Mas, se antigamente se usava a codificação visual dentro do próprio HTML, por que hoje isso não é mais recomendado? Bem, as razões para isso chamam-se praticidade e eficiência. Quando separamos os detalhes de apresentação em um arquivo separado, podemos usar o mesmo arquivo de definição de apresentação para **todas** as páginas. Em outras palavras, apesar do trabalho inicial para criar o arquivo de estilos em separado, ele **economiza** digitação nas páginas seguintes, acelerando o desenvolvimento.

Além disso, com um (ou poucos) arquivo de estilos, é muito mais fácil alterar a aparência de todo um site, de forma consistente e **sem ter que editar todas as páginas do site!** Adicionalmente, os arquivos de CSS ficam no cache dos navegadores. Como ele é o mesmo para todas as páginas, acaba sendo uma economia de banda de transferência usá-lo, além de fazer com que o site fique mais rápido para o usuário.

### **1.1. Que Recursos as CSSs Possuem?**

Basicamente, as folhas de estilo CSS são capazes de modificar qualquer propriedade visual de qualquer coisa que apareça em uma página HTML.

As folhas de estilo permitem modificar as fontes de texto, os alinhamentos de texto, a posição das imagens, cores dos elementos da página, cores de botões, posição de tabelas... enfim, uma infinidade de recursos.

Nesta aula serão vistos alguns destes recursos, e outros seguirão nas próximas aulas.

### **1.2. Porque o nome "Em Cascata"**

As razões mais práticas para considerar o termo "Cascata" são:

1) Podemos ter três níveis de definição de estilo:

- a) Em um arquivo separado
- b) No próprio arquivo HTML, no topo da página
- c) No próprio arquivo HTML, dentro da tag

2) Se definirmos o estilo de uma tag como <H1> e depois definirmos um estilo derivado de <H1>, como por exemplo <H1.editorial>, este estilo derivado "herdará" todas as mudanças de estilo que foram realizadas na tag <H1>.

O primeiro conceito pode ser explicado assim: normalmente definimos estilos globais em arquivos separados, com a extensão .CSS. Isso permite o uso de todos os benefícios citados anteriormente para os arquivos CSS. Se quisermos que apenas em alguma página um dos estilos seja modificado, podemos redefinir este estilo no topo da página, sendo que a mudança terá efeito apenas nesta página. Se queremos ainda modificar um estilo apenas em uma tag, podemos redefinir o estilo dentro dela, e terá efeito apenas na tag. Em geral, usaremos o arquivo de estilo separado, por ser a maneira mais "correta" e limpa de usar folhas de estilo.

Com relação ao segundo conceito, sua assimilação é mais fácil com o uso. Na prática é como dizer que se mudamos a fonte do H1 para Arial (ao invés de Times), automaticamente o estilo <H1.editorial> passará a ser também em Arial.



### **1.3. Passos para a Criação de um Arquivo de Estilo (CSS)**

São três os passos básicos para criar um arquivo de estilo:

- 1) Criar arquivo para inserir as definições de estilo (normalmente com extensão .css)
- 2) Indicar este arquivo na página HTML, obviamente na seção <HEAD>...</HEAD>
- 3) Editar o arquivo de estilo e a página até que tudo fique como desejado.

## **2. Usando um Arquivo .CSS**

Quando vamos criar uma página com folhas de estilo, a primeira coisa é criar um arquivo texto vazio, com um nome qualquer (por exemplo: **estilos.css**), para armazenar os estilos de uma dada página.

Criado este arquivo, temos de indicá-lo no arquivo HTML, de forma que o navegador possa encontrá-lo e usar os estilos definidos no mesmo. Como a informação do arquivo de folhas de estilo é para o navegador, esta indicação virá dentro da seção <HEAD>...</HEAD>.

Esta indicação é feita da seguinte forma:

```
<LINK HREF="estilo.css" REL="stylesheet" TYPE="text/css">
```

Notando que "estilo.css" é o nome do arquivo de estilos criado pelo desenvolvedor. REL e TYPE são definições para que o navegador sabe o que fazer com as informações que encontrar neste arquivo.

### **2.1. Estrutura de um Arquivo .CSS**

Dentro do arquivo de estilo (que é um arquivo texto comum), devemos seguir estritamente uma estrutura para que ele funcione. Um erro neste arquivo e a página não funcionará corretamente.

A estrutura é a seguinte:

```
TAG {  
  propriedade1: valor1;  
  propriedade2: valor2;  
  ...  
  propriedadeN: valorN;  
}
```

Por exemplo:

```
H1 {  
  text-align: center;  
  font-size: 3em;  
}
```

Isto indica, no arquivo de estilo, que os textos do tipo <H1> no HTML serão centralizados na tela (text-align) e seu tamanho terá uma ênfase (em) de 300%, ou seja, ficará 3x maior.

É possível ainda modificar duas (ou mais) tags simultaneamente, da seguinte forma:

```
TAG, TAG2, TAGN {  
  propriedade1: valor1;  
  propriedade2: valor2;  
  ...  
  propriedadeN: valorN;  
}
```

Por exemplo:

```
H1, H2 {  
  font-family: verdana, arial, sans-serif;  
}
```

Que altera o tipo de fonte dos títulos e subtítulos simultaneamente.

## **2.2. Algumas Definições de Estilo**

Os estilos que podem ser definidos variam de acordo com a tag que está sendo modificada. Por exemplo: não faz sentido mudar a imagem de fundo da tag <H1>. Por esta razão, se fizermos isso, nada acontecerá.

Nesta parte serão indicados alguns tags e alguns modificadores que podem ser usados com os mesmos. Entretanto, esta lista não é, de forma alguma, exaustiva.

### **Tag BODY:**

É possível mudar, por exemplo, as margens (margin-left, margin-right, margin-top, margin-bottom), a cor do fundo (background-color), imagem de fundo (background-image), se a imagem de fundo estará repetida (background-repeat), dentre outros.

Por exemplo, vamos definir um fundo com marge dem 20 pixels em cada lateral da tela, com um fundo creme claro:

```
BODY {  
  margin-left: 20px;  
  margin-right: 20px;  
  background-color: #FFFFD0;  
}
```

### **Tags H1, H2, H3, H4, H5 e H6:**

É possível mudar o alinhamento (text-align), o tamanho da fonte (font-size), o tipo de fonte (font-family), realce de fonte (font-weight), margem (margin-left, margin-right, margin-top, margin-bottom), cor (color), dentre outras.

Por exemplo, vamos redefinir H1 centralizado, com fonte 1.6x maior que o normal, usando uma fonte sem serifa (verdana, arial ou fonte sem serifa padrão do navegador), com texto em realce (mais grosso) e com cor azul escuro:

```
H1 {  
  text-align: center;  
  font-size: 1.6em;  
  font-family: verdana, arial, sans-serif;  
  font-weight: bold;  
  color: #000030;  
}
```

```
H2 {  
  font-size: 1.3em;  
  font-family: verdana, arial, sans-serif;  
  font-weight: bold;  
  color: #000030;  
}
```

### **Tag P:**

É possível mudar muitas coisas da tag de parágrafo, dentre elas: o alinhamento (text-align), o tamanho da fonte (font-size), o tipo de fonte (font-family), realce de fonte (font-weight), margem (margin-left, margin-right, margin-top, margin-bottom), cor (color)...

Por exemplo, vamos redefinir P como texto "justificado", com fonte 1.2x maior que o normal, usando uma fonte com serifa (garamond, times new roman ou fonte com serifa padrão do navegador), em cor azul desbotado:

```
P {  
  text-align: justify;  
  font-size: 1.2em;  
  font-family: garamond, times new roman, serif;  
  color: #004080;  
}
```

Existem alguns "subcomandos" no CSS que são úteis em alguns casos. Na tag <P> é comum queremos modificar o comportamento da primeira letra, de forma que ela fique com um espaço adicional a partir da borda esquerda da tela. Neste caso usamos o seguinte, para colocar 200% do tamanho de uma letra de "margem adicional":

```
P:first-letter {  
  margin-left: 2em;  
}
```

### **Tag A:**

É possível mudar várias coisas também na tag de âncora (links), e é muito comum que se mude a aparência de um link. Dentre as coisas possíveis de modificar estão: o tamanho da fonte (font-size), o tipo de fonte (font-family), realce de fonte (font-weight), margem (margin-left, margin-right, margin-top, margin-bottom), cor (color), a decoração da fonte (text-decoration)...

Por exemplo, vamos redefinir A como usando uma fonte com serifa (garamond, times new roman ou fonte com serifa padrão do navegador), em letra mais forte (negrito) e sem sublinhado, mas em cor azul mais claro:

```
A {  
  font-family: garamond, times new roman, serif;  
  font-weight: bold;  
  text-decoration: none;  
  color: #3030D0;  
}
```

### **Tag HR:**

Também é possível mudar muitas coisas na linha divisória. Dentre as coisas possíveis de modificar estão: a altura da barra (height), a largura da barra (width), a cor da barra (background-color), a borda (border), a imagem de fundo (background-image), a repetição da imagem (image-repeat) e assim por diante.

Por exemplo, vamos redefinir HR como 7 pixel de altura, 75% da largura da tela, cor de fundo azul escuro, sem borda 3D:

```
HR {  
  height: 7px;  
  width: 75%;  
  background-color: #000480;  
  border: none;  
}
```

### **Centralizando uma Image: Truque:**

Por enquanto, não temos uma forma mais adequada de centralizar uma imagem, mas podemos fazer o seguinte: definir um novo tipo de parágrafo <P CLASS="centered"> que será centralizado, e colocar a imagem dentro deste parágrafo.

Primeiramente definiremos a nova classe de parágrafo:

```
P.centrado {  
  text-align: center;  
}
```

E em seguida alteramos o documento original, indicando esta versão alternativa de parágrafo no parágrafo da figura:

```
<P CLASS="centrado">  
<IMG SRC="nome_da_imagem.jpg" ALT="">  
</P>
```

### 3. Exercício

Nesta aula, vimos a mudança de estilo para diversas tags. Usemos estas tags para fazer com que a página fique com essa aparência:

## Primeira página HTML

Nesta página você encontra os seguintes tópicos:

- Primeira Seção
- Segunda Seção

### Primeira Seção



A primeira seção irá conter dois parágrafos. Note, também, que o título desta seção é em H2, sendo que o título da página foi em H1.

Este é o segundo parágrafo da primeira seção.

### Segunda Seção

Esta é a segunda seção e também contém dois parágrafos. O título desta segunda seção também é em H2.

Este é o segundo parágrafo da segunda seção.

Tabela 1: Este é o caption da tabela

Coluna 1 <sup>(1)</sup>	Coluna 2
Dado 1	Dado 2

(1) Este é um comentário explicativo sobre a coluna 1 em texto small.

Mais informações na [página da disciplina Programação Web](#).

### Solução:

Arquivo **pagina.html** :

```
<HTML>
<HEAD>
  <TITLE>Primeira página Web Estática</TITLE>
  <LINK HREF="estilo.css" REL="stylesheet" TYPE="text/css">
</HEAD>
<BODY>
  <H1>Primeira página HTML</H1>
  <HR>
  <P>Nesta página você encontra os seguintes tópicos:</P>
  <UL>
    <LI>Primeira Seção</LI>
    <LI>Segunda Seção</LI>
  </UL>
  <H2>Primeira Seção</H2>
  <IMG SRC="http://www.caetano.eng.br/main/images/aflogo_horiz.gif"
    WIDTH="330" HEIGHT="80" TITLE="Empresa do professor"
    ALT="Amusement Factory Logo">
  <P>
    A primeira seção irá conter dois parágrafos. Note,
    também, que o título desta seção é em H2, sendo que o
    título da página foi em H1.
  </P>
  <P>
    Este é o segundo parágrafo da primeira seção.
  </P>
  <H2>Segunda Seção</H2>
  <P>
    Esta é a segunda seção e também contém dois
    parágrafos. O título desta seção também é em
    H2.
  </P>
  <P>
    Este é o segundo parágrafo da segunda seção.
  </P>
  <P>
    Mais informações na
    <A HREF="http://www.caetano.eng.br/aulas/fb/Web/" TITLE="Disciplina de
      Programações Web">página da disciplina
      Programações Web</A>.
  </P>
</BODY>
<TABLE>
```

```
<THEAD>
<CAPTION>Tabela 1: Este é o caption da tabela</CAPTION>
</THEAD>
<TBODY>
<TR><TH>Coluna 1<SUP>(1)</SUP></TH><TH>Coluna 2</TH></TR>
<TR><TD>Dado 1</TD><TD>Dado 2</TD></TR>
</TBODY>
</TABLE>
<P><SMALL>(1) Este é um comentário explicativo sobre a coluna 1
em texto small</SMALL></P>
</BODY>
</HTML>
```

Arquivo **estilo.css**:

```
BODY {
  margin-left: 20px;
  margin-right: 20px;
  background-color: #FFFFD0;
}

H1 {
  text-align: center;
  font-size: 1.6em;
  font-family: verdana, arial, sans-serif;
  font-weight: bold;
  color: #000030;
}

H2 {
  font-size: 1.3em;
  font-family: verdana, arial, sans-serif;
  font-weight: bold;
  color: #000030;
}

P {
  text-align: justify;
  font-size: 1.2em;
  font-family: garamond, times new roman, serif;
  color: #004080;
}

P:first-letter {
  margin-left: 2em;
}
```



```
A {  
  font-family: garamond, times new roman, serif;  
  font-weight: bold;  
  text-decoration: none;  
  color: #3030D0;  
}
```

```
HR {  
  height: 7px;  
  width: 75%;  
  background-color: #000480;  
  border: none;  
}
```

### **5. Bibliografia**

Cascade Style Sheets, level 2 revision 1: CSS 2.1 Specification - W3C Working Draft 06 November 2006. Disponível em < <http://www.w3.org/TR/CSS21/> >. Visitado em 21 de Dezembro de 2006.

RAMALHO, J.A. *HTML 4 Prático e Rápido*. Editora Berkeley, 1999.

BOENTE, A. *Programação Web Sem Mistérios*. Editora Brasport, 2006.

NIELSEN, J. *Projetando Websites*. Ed. Campus, 2000.

Notas da Aula 07: Web Estática: CSS Parte 2  
Prof. Daniel Caetano

**Objetivo:** Apresentar o CSS como instrumento para construção de layout de página.

### **Introdução**

Na aula passada vimos como usar as CSSs para alterar a aparência de nosso documento. Muitas propriedades do documento podem ser alteradas, como cor de fundo, cor de texto, dentre outras.

Entretanto, o mecanismo visto para modificação de propriedades é um tanto quanto limitado com relação ao posicionamento dos diversos elementos de uma página: podemos mudar alinhamentos e, talvez, a ordem de alguns elementos. Entretanto, se quisermos posicionar o conteúdo de alguma forma menos tradicional, não será possível apenas com o que já vimos até agora.

O objetivo desta aula é, então, apresentar elementos que permitam posicionar elementos de uma página Web no lugar desejado e, mais uma vez, isso é feito pelos arquivos CSS, com poucas modificações do documento HTML.

### **1. Tag de Divisão de Documento**

O primeiro conceito a ser compreendido é o da divisão lógica de documento. Se queremos posicionar ou modificar as características de um pequeno trecho do documento, precisamos indicar para o navegador qual é esse trecho (ou seja, onde ele começa e onde termina) e também qual é o nome que utilizaremos para modificá-lo... sendo esta a única mudança que realizaremos no código HTML em si.

Em outras palavras, existe uma tag que usaremos para marcar uma região da página que, de alguma forma, poderá receber uma cor de fundo diferente, um parágrafo diferente ... ou mesmo uma posição diferente da convencional. Esta tag é **<DIV> ... </DIV>**.

Como podemos ter várias divisões dentro de um mesmo documento (cada uma delas devendo ter propriedades distintas e, eventualmente, posições distintas), é adequado dar um **nome** para cada uma destas seções. Isso é feito usando-se o parâmetro **ID**:

```
<DIV ID="nome_da_divisao">  
...  
</DIV>
```

Um comentário importante é que a tag DIV só terá efeito visual no navegador se realizarmos mudanças efetivas na mesma, por meio do arquivo CSS. Caso contrário, será como se o navegador simplesmente tivesse ignorado a tag. Na página definida na aula anterior, podemos definir várias seções. Por exemplo:

<DIV ID="titulo">

## Primeira página HTML

</DIV>

<DIV ID="indice">

Nesta página você encontra os seguintes tópicos:

- Primeira Seção
- Segunda Seção

</DIV>

<DIV ID="secao1">

### Primeira Seção



A primeira seção irá conter dois parágrafos. Note, também, que o título desta seção é em H2, sendo que o título da página foi em H1.

Este é o segundo parágrafo da primeira seção.

</DIV>

<DIV ID="secao2">

### Segunda Seção

Esta é a segunda seção e também contém dois parágrafos. O título desta segunda seção também é em H2.

Este é o segundo parágrafo da segunda seção.

Tabela 1: Este é o caption da tabela

Coluna 1 <sup>(1)</sup>	Coluna 2
Dado 1	Dado 2

(1) Este é um comentário explicativo sobre a coluna 1 em texto small.

Mais informações na [página da disciplina Programação Web](#).

</DIV>

Ao carregar esta página no navegador, ela será exibida exatamente como antes. Entretanto, isso só ocorre porque não realizamos qualquer mudança no arquivo CSS. Mas o que deveríamos modificar no arquivo CSS? Resumidamente, devemos indicar o que estas seções devem possuir de diferente!

## **2. Modificando Seções Usando CSS**

Como agora não iremos mais modificar apenas a apresentação de algumas tags do HTML e sim como serão apresentados pedaços inteiros do documento, a definição no arquivo CSS é um pouco diferente daquela que vimos antes.

O formato geral é:

```
#secao {  
    ...  
}
```

Como uma regra geral, também aqui o CSS é muito chato com a sintaxe. Um pequeno erro de digitação, um ponto-e-vírgula faltando ou algo do gênero pode ser responsável por sua página não aparecer corretamente. Por esta razão, muita atenção ao digitar o arquivo .CSS!

Nas próximas páginas serão apresentados alguns exemplos de modificação de propriedades e posicionamento através de CSS.

Pegando como exemplo a página da aula anterior, para modificar a seção de título (definida como no item anterior) de forma que ela tenha um fundo azul claro e margens diferentes, devemos indicar da seguinte forma no arquivo estilo.css:

```
#titulo {  
    position: relative;  
    background-color: #C0C0E0;  
    margin-left: 15%;  
    margin-right: 15%;  
}
```

O resultado será o apresentado a seguir. Observe que o <HR> agora não traça mais uma linha divisória de 75% da página, e sim de 75% da região definida dentro do <DIV>!



Um outro exemplo: para fazer com que o índice fique limitado a uma faixa esquerda da tela, com uma cor de fundo branca:

```
#indice {  
    position: absolute;  
    left: 0%;  
    right: 75%;  
    top: 15%;  
    background-color: #FFFFFF;  
}
```

O resultado é apresentado a seguir. Note que o índice ficou agora sobreposto ao texto da seção 1!

# Primeira página HTML

## Primeira Seção

Nesta página você encontra os seguintes tópicos:

- Primeira Seção
- Segunda Seção

conter dois parágrafos. Note, também, que o título desta seção é em H2, a primeira foi em H1.

Este é o segundo parágrafo da primeira seção.

## Segunda Seção

Esta é a segunda seção e também contém dois parágrafos. O título desta seção também é em H2.

Este é o segundo parágrafo da segunda seção.

Mais informações na [página da disciplina Programação Web](#).

Tabela 1: Este é o caption da tabela

Coluna 1 <sup>(1)</sup>	Coluna 2
Dado 1	Dado 2

(1) Este é um comentário explicativo sobre a coluna 1 em texto small

Vamos movimentar agora a seção 1 para a direita, com uma borda arredondada azul, de 6 pixels de largura:

```
#secao1 {  
  position: relative;  
  left: 30%;  
  right: 5%;  
  border-color: #303060;  
  border-width: 6px;  
  border-style: ridge;  
}
```

O resultado é apresentaod a seguir. Observe que o box não coube na tela.



É possível controlar a largura do box diretamente. Aqui queremos que ele tenha até uns 65% da tela (100%, descontado 25% do menu, dá 75%... Descontados 5% de margem em cada lado, 65%). Modificamos o CSS assim:

```
#secao1 {  
  position: relative;  
  left: 30%;  
  right: 5%;  
  width: 65%;  
  border-color: #303060;  
  border-width: 6px;  
  border-style: ridge;  
}
```

E o resultado será:



Um último passo, agora, vamos colocar a seção dois abaixo da seção 1, da seguinte forma:

```
#secao2 {  
  position: relative;  
  left: 30%;  
  right: 5%;  
  width: 65%;  
  border-color: #303060;  
  border-width: 6px;  
  border-style: ridge;  
}
```

Sendo o resultado igual a esse:





Ainda não está bom. Repare que os textos estão muito colados nas margens. Podemos acertar isso com a diretiva **padding**, como por exemplo em:

```
#indice {  
  position: absolute;  
  left: 0%;  
  right: 75%;  
  top: 15%;  
  padding: 2px;  
  background-color: #FFFFFF;  
}
```

Seria também interessante mover o indice de forma que ele não fique tão grudado na esquerda:

```
#indice {  
  position: absolute;  
  left: 5%;  
  right: 75%;  
  top: 15%;  
  padding: 2px;  
  background-color: #FFFFFF;  
}
```

### 3. Exercício


Nesta aula, vimos como posicionar elementos de uma página usando o CSS. Use o que foi ensinado para reproduzir a página abaixo:

## Primeira página HTML

Nesta página  
você encontra os  
seguintes tópicos:

- Primeira Seção
- Segunda Seção

### Primeira Seção



A primeira seção irá conter dois parágrafos. Note, também, que o título desta seção é em H2, sendo que o título da página foi em H1.

Este é o segundo parágrafo da primeira seção.

### Segunda Seção

Esta é a segunda seção e também contém dois parágrafos. O título desta seção também é em H2.

Este é o segundo parágrafo da segunda seção.

Mais informações na [página da disciplina Programação Web](#).

Tabela 1: Este é o  
caption da tabela

Coluna 1 <sup>(1)</sup>	Coluna 2
Dado 1	Dado 2

(1) Este é um comentário explicativo sobre a coluna 1 em texto small

### Solução:

Arquivo **pagina.html** :

```
<HTML>
<HEAD>
  <TITLE>Primeira página Web Estática</TITLE>
  <LINK HREF="estilo.css" REL="stylesheet" TYPE="text/css">
</HEAD>
<BODY>
<DIV ID="titulo">
  <H1>Primeira página HTML</H1>
  <HR>
</DIV>
<DIV ID="indice">
  <P>Nesta página você encontra os seguintes tópicos:</P>
  <UL>
    <LI>Primeira Seção</LI>
    <LI>Segunda Seção</LI>
  </UL>
</DIV>
<DIV ID="secao1">
  <H2>Primeira Seção</H2>
  <IMG SRC="http://www.caetano.eng.br/main/images/aflogo_horiz.gif"
    WIDTH="330" HEIGHT="80" TITLE="Empresa do professor"
    ALT="Amusement Factory Logo">
  <P>
    A primeira seção irá conter dois parágrafos. Note,
    também, que o título desta seção é em H2, sendo que o
    título da página foi em H1.
  </P>
  <P>
    Este é o segundo parágrafo da primeira seção.
  </P>
</DIV>
<DIV ID="secao2">
  <H2>Segunda Seção</H2>
  <P>
    Esta é a segunda seção e também contém dois
    parágrafos. O título desta seção também é em
    H2.
  </P>
  <P>
    Este é o segundo parágrafo da segunda seção.
  </P>
```

```
<P>
Mais informa&ccedil;&otilde;es na
<A HREF="http://www.caetano.eng.br/aulas/fb/Web/" TITLE="Disciplina de
  Programa&ccedil;&atilde;o Web">p&aacute;gina da disciplina
  Programa&ccedil;&atilde;o Web</A>.
</P>
<TABLE>
  <THEAD>
    <CAPTION>Tabela 1: Este &eacute; o caption da tabela</CAPTION>
  </THEAD>
  <TBODY>
    <TR><TH>Coluna 1<SUP>(1)</SUP></TH><TH>Coluna 2</TH></TR>
    <TR><TD>Dado 1</TD><TD>Dado 2</TD></TR>
  </TBODY>
</TABLE>
<P><SMALL>(1) Este &eacute; um coment&aacute;rio explicativo sobre a coluna 1 em
texto small</SMALL></P>
</DIV>
</BODY>
</HTML>
```

Arquivo **estilo.css**:

```
BODY {
  margin-left: 20px;
  margin-right: 20px;
  background-color: #FFFFD0;
}

H1 {
  text-align: center;
  font-size: 1.6em;
  font-family: verdana, arial, sans-serif;
  font-weight: bold;
  color: #000030;
}

H2 {
  font-size: 1.3em;
  font-family: verdana, arial, sans-serif;
  font-weight: bold;
  color: #000030;
}
```

```
P {  
    text-align: justify;  
    font-size: 1.2em;  
    font-family: garamond, times new roman, serif;  
    color: #004080;  
}
```

```
P:first-letter {  
    margin-left: 2em;  
}
```

```
A {  
    font-family: garamond, times new roman, serif;  
    font-weight: bold;  
    text-decoration: none;  
    color: #3030D0;  
}
```

```
HR {  
    height: 7px;  
    width: 75%;  
    background-color: #000480;  
    border: none;  
}
```

```
#titulo {  
    position: relative;  
    background-color: #C0C0E0;  
    margin-left: 15%;  
    margin-right: 15%;  
}
```

```
#indice {  
    position: absolute;  
    left: 5%;  
    right: 75%;  
    top: 15%;  
    padding: 2px;  
    background-color: #FFFFFF;  
}
```

```
#secao1 {  
  position: relative;  
  top: 25%;  
  left: 30%;  
  right: 5%;  
  width: 65%;  
  padding: 2px;  
  border-color: #303060;  
  border-width: 6px;  
  border-style: ridge;  
}
```

```
#secao2 {  
  position: relative;  
  left: 30%;  
  right: 5%;  
  width: 65%;  
  padding: 2px;  
  border-color: #303060;  
  border-width: 6px;  
  border-style: ridge;  
}
```

#### **4. Bibliografia**

Cascade Style Sheets, level 2 revision 1: CSS 2.1 Specification - W3C Working Draft 06 November 2006. Disponível em < <http://www.w3.org/TR/CSS21/> >. Visitado em 21 de Dezembro de 2006.

RAMALHO, J.A. *HTML 4 Prático e Rápido*. Editora Berkeley, 1999.

BOENTE, A. *Programação Web Sem Mistérios*. Editora Brasport, 2006.

NIELSEN, J. *Projetando Websites*. Ed. Campus, 2000.

Notas da Aula 08: Noções de Projeto de Sites Web  
Prof. Daniel Caetano

**Objetivo:** Apresentar os estilos de projetos de Navegação Web.

### Introdução

Até este momento aprendemos detalhes de como criar páginas isoladas. Apesar de ter sido apresentado o conceito das âncoras e links, ainda não foi feito qualquer tipo de explicação sobre como estruturar várias páginas de um mesmo site, ou seja, em quais páginas colocar links para quais outras.

Apesar de parecer uma tarefa trivial, é fácil produzir uma página em que o usuário se sente perdido, quando não há um planejamento adequado. Por esta razão, antes de partirmos para as páginas web dinâmicas, é interessante apresentar algumas considerações sobre a estrutura de navegabilidade do site e também algumas características que devem ser respeitadas no desenvolvimento de um site para obtenção de adequada usabilidade.

Convém lembrar que, para projetarmos a estrutura de um site web, temos de sempre ter em mente seu foco na informação. Além disso, é preciso evitar seguir cegamente alguns dos "hábitos" de desenho de algumas páginas web existentes, pois muitas são impregnadas por maus hábitos de projetistas que iniciaram na "arte" de produzir páginas web muito antes de qualquer estudo sobre usabilidade ter sido realizado.

### 1. O Projeto do Site

Em primeiro lugar, é preciso deixar claro que a **Engenharia para a Web** não é um clone perfeito da Engenharia de Software, embora procure seguir os mesmos princípios. Neste espírito, as **características** mais importantes (mas não únicas) que norteiam um projeto são:

- **Confiabilidade**
- **Usabilidade**
- **Adaptabilidade**

**Busca-se um desenvolvimento bem sucedido, bem como uma implementação e manutenção simplificados, ao mesmo tempo em que se obtém um resultado de alta qualidade sob o ponto de vista do usuário.**

A **ponto de vista de Lowe (1999)** facilita a compreensão da maneira como devemos enxergar o desenvolvimento de um web site: "O desenvolvimento de Websites está muito mais relacionado à criação de uma infraestrutura (as linhas mestras do jardim) e depois ao

"cultivo" da informação que cresce e floresce dentro deste jardim. Ao longo do tempo, o jardim (ou seja, o site na Web) vai continuar a evoluir, a se modificar e a crescer. Uma boa arquitetura inicial deve permitir que este crescimento ocorra de forma controlada e consistente."

Existem algumas outras **diretivas** que impulsionam o processo de projeto de um WebSite:

- **Imediatismo**, uma necessidade de que o site em poucos dias ou semanas no ar.
- **Segurança**, para limitar o acesso a algumas informações para alguns tipos de usuários.
- **Estética**, fundamental para vender produtos ou idéias, fundamental para o sucesso.

Segundo Pressman (2002), os principais **atributos de qualidade** de um site são:

- **Usabilidade**, relativa à compreensão geral do site, ajuda online, interface e estética.
- **Funcionalidade**, relativo à buscas, navegação e características da aplicação.
- **Confiabilidade**, relativo a links corretos, recuperação de erros, validação de entradas de usuário.
- **Eficiência**, relativo ao tempo de resposta, tempo de renderização da página e gráficos.
- **Manutibilidade**, relativo à facilidade de correção, adaptabilidade e extensibilidade.

### 1.1. Diferenças no Ciclo de Projeto de Sites

Em geral, **o ciclo de projeto de um Site é exatamente o mesmo de qualquer aplicação**, com **planejamento, análise, engenharia, testes, avaliação** e assim por diante. Da mesma forma que em projetos de software usual temos uma equipe cuidando das funções básicas do aplicativo e uma outra equipe cuidando da interface com o usuário, **no projeto Web também temos duas frentes**.

A **primeira** destas frentes cuida da **arquitetura** do site, o **projeto da navegação** e da **interface**. A **segunda**, deve se preocupar com o **projeto de conteúdo** e a **produção**.

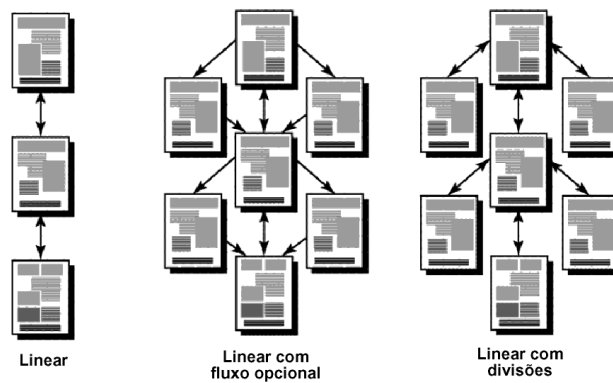
## 2. Dicas de Projeto do Site como um Todo

### ESTRUTURA DO SITE

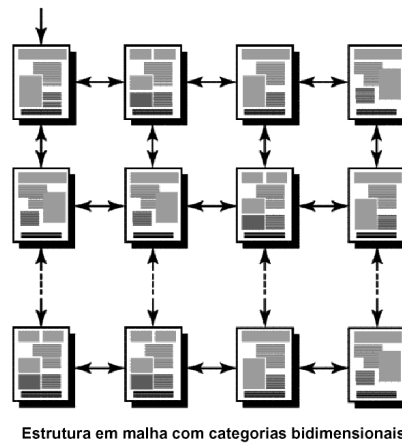
Talvez a primeira coisa a ser definida, deve ser a estrutura do site. De forma geral, esta estrutura é baseada no **conteúdo** a ser apresentado, nas **classes de usuários** que devem acessar a página e na **filosofia de navegação** proposta. As **filosofias** de navegação podem ser: **lineares, em malha, hierárquicas, em teia, dentre outras** mais caóticas.



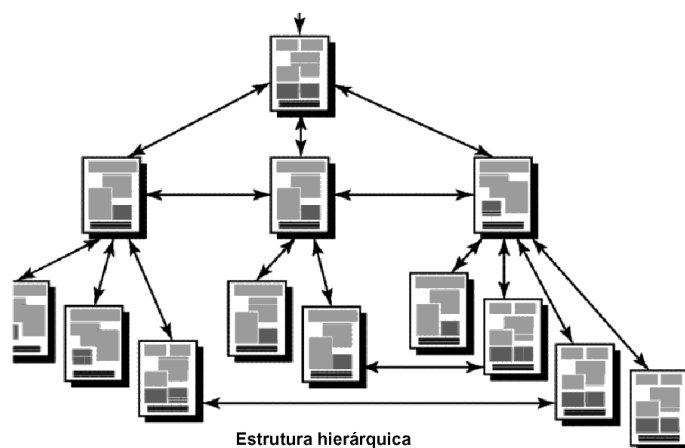
Estruturas **lineares** são de **fácil compreensão**, porém **pouco flexíveis**.



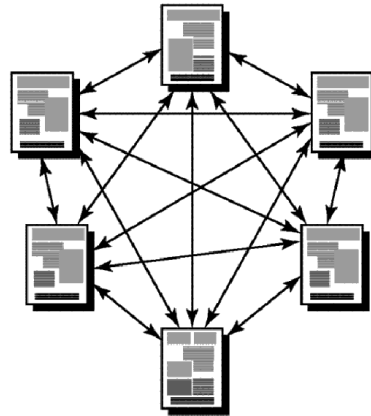
Estruturas **em malha** são de **de compreensão um pouco complexa**, porém **mais flexíveis**.



Estruturas **hierárquicas** são de **de compreensão complexa**, porém são **flexíveis**.



Estruturas **em teia** são de **de compreensão muito complexa**, porém são **totalmente flexíveis**.



Estrutura em Teia (Web)

- **A estrutura escolhida deve ser o menos confusa possível**, além de **refletir como o usuário enxerga o conteúdo** do site e não a forma como a empresa é dividida.

- **Não use telas Splash** (aquelas telas de abertura), a menos que necessário. Exemplos destas situações são mudanças do endereço principal, páginas com conteúdos inapropriados para menores, etc.

- **Não force o usuário a entrar pela homepage**. Os links das páginas internas devem estar **sempre** acessíveis e, dentro do possível, não devem ser alterados.

### DESENHO GLOBAL

- **A largura da página deve ser dinâmica**, sempre que possível. Se não puder, ela deve ser corretamente apresentada mesmo em baixas resoluções, ou seja, para boa **visualização em torno de 600 pixels**.

- **Cuidado com o uso de metáforas** no projeto visual geral! Metáforas equivocadas podem ser "bonitinhas" mas dificultarem o uso por parte do usuário. Lembre-se, por exemplo: Web não é TV! Metáforas consagradas (como a do carrinho de compras, por exemplo) devem ser usadas, entretanto.

- **Região de navegação clara**, indicando "onde o usuário está", "onde estava" e "onde pode ir", fazendo indicações com relação à Web e ao Site como um todo.

### DESENHO DA HOMEPAGE

A homepage é a **página principal**, onde o usuário entra se digitar o link principal de sua página.

- **Deve ter design diferenciado**, visando captar a atenção do usuário.
- **Não deve ter botão "home"**, mas se tiver, deve ficar "desligado".
- **Deve ter um logotipo maior**, respondendo à pergunta "onde estou" que o usuário possa se fazer ao chegar em seu site.
- **Deve passar a idéia do que o site faz**, mas nunca através das enfadonhas "missões" que são apresentadas em alguns sites.
- **Área de notícias restrita**, a menos que seu site seja um site de notícias. A maioria dos usuários entra em uma página buscando uma informação específica, não as últimas novidades que ocorreram em sua empresa.
- **Deve ter um mecanismo de navegação por menus**, se possível indicando também um mapa do site. Muitos usuários preferem a navegação link-a-link ou por mapa de site.
- **Deve ter um mecanismo de busca**, incluindo busca avançada. Nem todo usuário gosta de navegação link-a-link e sempre vai direto no mecanismo de busca.

### DESENHO DAS PÁGINAS INTERIORES

Diferentemente da homepage, as páginas interiores são direcionadas à **apresentação de conteúdo**.

- **O logotipo deve ser menos proeminente**, já que aqui ele é apenas uma informação adicional e não o foco da página interior.
- **O logotipo deve linkar para a homepage**, para que o usuário que entrou "pelo meio" da página, por um mecanismo de busca, possa descobrir mais sobre a empresa que criou aquela página.
- **Deve mostrar conteúdo específico e direto**, não ficar aborrecendo o usuário com mensagens de boas vindas.

### DESENHO DE SUBSITES

Subsites são uma **forma interessante de categorizar informações** de maneira que apenas usuários realmente interessados as acessem. Entretanto, alguns cuidados adicionais devem ser ressaltados.

- **Podem ter diferenças do site principal**, mas é bom evitar diferenças demais.
- **Devem ter coerência visual com o site principal**, para manter a identidade do site como um todo.
- **Devem ter navegação similar**, para não frustrar o usuário por ter que aprender a navegar em mais um site diferente.
- **As buscas devem oferecer opções de escopo**, deixando claro quando uma busca é local (no subsite) ou global (no site todo).

### DESENHO DE URLS

A URL é a parte que é mais divulgada de seu site, de forma que as **pessoas possam acessá-lo**. Por esta razão, vale tomar alguns cuidados:

- **Escolha nomes claros**, sem caracteres malucos.
- **Escolha nomes breves**, já que nomes grandes são mais difíceis de guardar e mais propensos a erros de digitação
- **Não use mistura de caixa alta e baixa**. Dentro do possível, use todo o nome em letras minúsculas e não use, em hipótese alguma, acentuação.
- **Use endereços estáveis**. Esta dica é importante para sites com conteúdo que mudam em determinados períodos de tempo. Apesar de ser "bonitinho" ter um link como "edicaoatual.html", isso é péssimo para quem quer linkar um artigo. A pessoa faz um bookmark para um artigo no mês de agosto e ao tentar reler o mesmo artigo no mês de setembro ele sumiu.
- **Produtos devem indicar a URL de sua página web**. Embora seja comum as empresas colocar o site da empresa em seus produtos, é tão ou mais importante indicar a URL da página do produto na embalagem/manual. Poucas coisas irritam mais um usuário do que ter que "achar" a página do produto num site, quando precisa de drivers ou suporte.

## **6. Bibliografia**

NIELSEN, J. *Projetando Websites*. Ed. Campus, 2000.

PRESSMAN, R. *Engenharia de Software*. Ed. McGraw-Hill, 2002.

FERREIRA, M.A.G.V. *Notas de Aula - Tópicos de Comunicação Homem-Máquina*, EPUSP, 2003.

Notas da Aula 09: Noções de Design de Páginas Web  
Prof. Daniel Caetano

**Objetivo:** Apresentar dicas de design de páginas que respeitam os critérios de usabilidade.

### Introdução

Como visto anteriormente, o usuário é, muitas vezes, desconhecido. Entretanto, uma característica marcante é conhecida: o usuário de Web busca informações. Sendo assim, o **projeto das páginas e de seu conteúdo** deve ser **focado na facilidade de compreensão e da navegabilidade**, sendo a **linguagem visual** utilizada **muito importante** para uma boa aceitação da mesma.

Em última análise, **essas são características que praticamente definem o sucesso ou fracasso de** uma determinada **página ou site**. Por esta razão, continuando as dicas da aula anterior, serão apresentadas dicas relativas ao desenho da página, em termos de diretrizes gerais de desenho) e do conteúdo como um todo.

Lembre-se que **credibilidade é tudo** e, portanto, se a página a ser criada for séria, nada de firulas como ícones animados para mandar e-mail e coisas do tipo, se quiser que sua página tenha sucesso.

### 1. Dicas para o Design da Página

O **design da página** envolve basicamente **como os elementos da página estão dispostos e a finalidade de cada uma das áreas**.

#### ÁREA DA TELA E LAYOUT

- **Deixe a maior parte da tela para conteúdo.** Lembre-se que o usuário não entra em uma página por causa da beleza do logotipo, nem mesmo pelo incrível mecanismo que foi criado para os menus... Muito menos para ver propagandas.

- **Não abarrotar a tela com informações**, em especial se elas forem inúteis. Se há muitas informações a serem apresentadas, segmente-as em conjuntos coerentes e apresente-as de maneira separada. Abarrotar a tela com informações só fará com que o usuário perca interesse no conteúdo como um todo.

- **Evite gastar área da tela com propaganda.** Embora nem sempre seja possível, o ideal é nem existir propaganda alguma. Se for decidido que haverá propagandas, nunca coloque mais que uma, pois elas disputarão atenção (e espaço) entre si e contra o conteúdo da

página, irritando o usuário. Se for usá-las, procure colocar propagandas que tenham alguma correlação com o conteúdo da página.

- **O layout deve ser independente de resolução**, ou seja, deve ser redimensionável, como já dito anteriormente. Caso não seja possível, deve ser fixo em um tamanho adequado para visualização no máximo de dispositivos.

- **O layout deve ficar bom em diferentes navegadores e plataformas**, por isso, realize testes em todas elas! Não há nada mais desagradável do que um usuário visualizar uma página toda distorcida porque a equipe que a projetou só testou em um único navegador. Página distorcida ou não funcional significa usuário perdido... e usuário perdido significa negócios não concretizados!

- **Lembre-se dos softwares antigos**. Muitos usuários ainda usam versões antigas de navegadores e extensões. Procure desenvolver seu site para duas versões mais antigas que a recente. Se fizer para a "última moda" em navegador ou plugin, lembre-se de criar uma versão que funcione bem com versões antigas dos mesmos.

- **Separe programação de layout do conteúdo**. Lembre-se: conteúdo é especificado no HTML, que indica o significado de cada elemento. O layout é definido nos arquivos de CSS (Cascade Style Sheets), ou seja, separados do conteúdo. Isso facilita a manutenção do site, tanto na alteração de conteúdo quanto na criação de novos layouts.

- **Não use frames**. Embora em alguns raros casos o uso de frames seja aceitável (e talvez até mesmo requeridos), os frames criam problemas sérios de navegação, dificuldades para linkar partes específicas do site, além de terem uma visualização muito dificultada em dispositivos de navegação com baixa resolução.

- **Lembre-se da impressão**, ou seja, que os usuários podem querer imprimir sua página. Se sua página tiver fundo escuro com texto claro, crie uma versão alternativa de CSS para impressão, com fundo claro e texto escuro, sem figuras de fundo, de preferência. Se não quiser, adicione links para a matéria em formato postscript (.PS) ou portable data file (.PDF).

### TEMPOS E TAMANHOS

- **Os tempos de resposta devem ser mínimos**, já que ninguém gosta de ficar esperando olhando para uma ampulheta.

- **0,1s é o ideal**, para que o usuário sinta uma navegação em tempo real.
- **1s é razoável** para que não atrapalhe o fluxo de pensamento do usuário.
- **10s é o limite** de usabilidade.

- **Pense nos usuários de modem**. Estes tempos não são relativos ao seu próprio micro ou à uma conexão banda larga. Desprezar usuários de modem é desprezar uma grande parcela da população.

- **Cuidado com figuras e textos muito grandes**, pois eles aumentam **muito** o tempo de download. Os tamanhos ideais são:

CONEXÃO	IDEAL	MÁXIMO
- Modem:	até 2KB	34KB.
- ADSL até 1Mbps:	até 8KB	150KB
- ADSL rápidas:	até 100KB	2MB.

### USO DE LINKS

- **Use links de navegação estrutural ao longo do texto**, que são aqueles que interligam partes de uma única página. Eles facilitam ao usuário encontrar o que procuram e ajudam a manter os textos enxutos, já que explicações detalhadas de alguns aspectos podem ter suas próprias páginas.

- **Use links associativos ao longo do texto**, que ligam partes de sua página com páginas que expliquem termos ou assuntos citados. Não incorra no erro de querer prender o usuário em sua página. Não é assim que se conquista usuários de internet.

- **Use links de referência adicional ao fim do texto**, ligando sua página a páginas que falem de assuntos semelhantes e que possam ser de interesse de pessoas que tenham lido sua página do começo ao fim.

- **Use palavras significativas nos links**, não use palavras como "clique aqui" ou algo assim. Isso dificulta o uso por deficientes visuais e dificulta a geração de um menu do tipo "Links" onde o usuário possa navegar sem precisar ler a página.

- **Use títulos nos links**, dentro do possível, explicando de forma resumida para onde os links levam e qual o conteúdo da página indicada.

- **Evite mudar as cores padrão dos links**, já que os usuários geralmente estão acostumadas a elas. Se mudá-las, seja coerente.

- **Use sempre a mesma URL na indicação de links para uma mesma página**, de forma que se ela já houver sido visitada (mesmo que através de outro link), o link fique em cor diferente.

- **Diferencie links externos**, se possível, indicando-os com um estilo diferente ou alguma marca (por exemplo, escrevendo em itálico).



## **2. Dicas para o Projeto do Conteúdo**

### **TEXTO**

O texto é, em geral, a **parte mais importante** de um site. Entretanto, a **leitura na tela tem seus problemas** e medidas especiais devem ser adotadas para tornar a experiência do usuário mais agradável.

- **Seja sucinto**, é a principal dica com relação ao texto. A leitura na tela é cerca de 25% mais lenta que no papel, devido à maior dificuldade de leitura oriunda do brilho e a baixa resolução da tela. Por esta razão, textos longos são cansativos e desagradáveis. Na web os textos devem ter em torno de 50% do tamanho que eles teriam em uma publicação impressa.

- **Use e abuse do hipertexto**. Isso significa que se uma dada palavra do seu texto merece um "artigo" para explicá-la, transforme-a em um link para uma página onde exista a explicação para a mesma. Lembre-se: web não é livro. O uso de hipertexto facilita a redução do tamanho dos textos, já que os termos não serão explicados detalhadamente dentro do texto principal, como ocorreria na mídia impressa.

- **Divida o conteúdo em trechos**, apresentados em páginas diferentes. Observe que isso deve ser feito com critério e não simplesmente cortando um texto no meio, em qualquer parte, apenas para dividir em mais de uma página.

- **Evite erros ortográficos**, em especial em páginas de empresa. Embora em páginas pessoais isso não seja grave, há poucas coisas que tiram mais credibilidade de uma empresa que uma página web cheia de erros ortográficos.

- **Facilite a leitura...** os usuários raramente lêem uma página toda. O uso de elementos especiais ajudam ao usuário focar naquilo que é mais importante. Ou seja: destaque o que é mais importante.

- **Estruture o texto em 2 ou 3 níveis de títulos.**
- **Use títulos significativos**, ao invés de títulos "bonitinhos".
- **Quebre blocos de texto em trechos menores**. Bullets ajudam nisso.
- **Enfatize palavras importantes**, usando negrito, mudanças de cor, etc.

- **Use linguagem simples**, uma vez que o usuário pode não ser um especialista no assunto sendo tratado. Como em qualquer interface, evite ao máximo o uso de termos computacionais.

- **Cuidado com o uso de humor**, alguns usuários podem não compreender ou se ofender com as brincadeiras e trocadilhos.

## TÍTULOS

- **Seja descritivo e sucinto**, usando de 2 a 6 palavras. Lembre-se que os títulos são, muitas vezes, apresentados fora de contexto, em bookmarks ou em sites de busca.
- **Use linguagem simples**, sem trocadilhos que possam dificultar usuários de outras nacionalidades.
- **Evite atrair usuários que não são público alvo**, ou seja, evite títulos que possam ter uma interpretação dúbia e trazer usuários que não se interessariam por sua página. Tais usuários certamente se decepcionarão com o conteúdo encontrado.
- **Pule artigos definidos e indefinidos iniciais** do título, lembrando que muitos sites de diretório e busca organizam os links para os sites pelo título e você não gostaria que o seu site estivesse misturado em meio a um monte de sites começando com "O", "A", "Um" ou "Uma", por exemplo.
- **Use uma primeira palavra bem descritiva**, já que, como dito acima, muitos sites de diretório e busca organizam os links pelo título dos sites. Usando uma primeira palavra indicativa, você facilitará ao usuário encontrar seu site.
- **Não use o mesmo título para todas as páginas** do site, e evite que todas as páginas comecem com título igual. Lembre-se que os sites de busca apresentam o **título** da página como representação do seu conteúdo.

## LEGIBILIDADE

- **Use cores de alto contraste**, ou seja, se usar fundo claro, use texto escuro. Se usar fundo escuro, use texto claro.
- **Evite figuras de fundo**, mas se usá-las, use figuras com padrões sutis.
- **Use fontes de tamanho suficiente**, e nunca fixe o tamanho da fonte, permitindo que usuários com deficiências ampliem o tamanho da mesma.
- **Mantenha os textos parados!** Embora isso pareça estranho, há pessoas que adoram usar "marquees", com textos correndo, subindo e descendo, piscando... isso atrapalha **muito** a legibilidade.
- **Não use textos "TUDO EM MAIÚSCULA"**. Além de ter uma aparência agressiva, o ser humano lê com muito menos velocidade textos totalmente em maiúsculas, causando um maior cansaço visual e irritação.

## MULTIMÍDIA

- **Evite vídeos**, já que eles são demasiado lentos para as redes atuais. Entretanto, quando usá-los, indique o tamanho e tempo estimado de download.

- **Reduza as imagens de forma adequada**, para reduzir seu tamanho sem perda do significado. Normalmente ela deve ser cortada (crop), ressaltando a área de interesse, e depois redimensionada (resize) para reduzi-la a um tamanho adequado.

- **Evite animações**, pois elas costumam tirar a seriedade do site, além de aumentar os tempos de download. Seu uso excessivo também pode causar irritação no usuário.

- **Use áudio com parcimônia**, e sempre permita que o usuário o desligue. Evite música de fundo automática e efeitos sonoros que não possam ser desligados. Audioclipes podem, entretanto, dar uma maior dimensão à apresentação de conteúdo, mostrando um exemplo de um trecho de um CD ou uma ópera. Nestes casos, indique o tamanho e o tempo estimado de download.

- **Evite o uso de navegação 3D**, a menos que isso **realmente** facilite a compreensão. Usar 3D para ficar "bonito" em geral tem o péssimo efeito de tornar a navegação muito lenta. Navegação por ambientes virtuais (uma cidade onde cada prédio é uma página) é também uma péssima idéia, pois é um tipo de navegação muito lento, o que vai na contra-mão dos interesses do usuário típico da web.

## 3. Bibliografia

NIELSEN, J. *Projetando Websites*. Ed. Campus, 2000.

Notas da Aula 10: Web Dinâmica: Introdução ao PHP  
Prof. Daniel Caetano

**Objetivo:** Apresentar algumas tecnologias de desenvolvimento de Web Dinâmica e realizar uma introdução ao PHP, suas variáveis e funções mais básicas.

### Introdução

Nas aulas anteriores já vimos como construir uma página estática usando HTML e CSS. Também foi visto como projetar a navegação de um site e também alguns conselhos com relação ao projeto da aparência do site.

Apesar de todas as qualidades do que já aprendemos, nosso site ainda é estático; a partir desta aula teremos uma breve introdução sobre uma das tecnologias mais poderosas para Web Dinâmica na atualidade: o PHP.

### 1. O que são Páginas Web Dinâmicas

Já vimos anteriormente o que é a Web Dinâmica, mas o que são "Páginas Web Dinâmicas"? Bem, "Páginas Web Dinâmicas" são páginas que sempre se modificam: são recriadas sempre que o usuário as solicita.

A parte importante aqui é que elas não são recriadas por qualquer motivo: elas são recriadas para se adaptar às necessidades do usuário. E para que isso seja possível, tais páginas, via de regra, possuem um pequeno programa.

As linguagens são as mais diversas. Algumas delas podem ser vistas no quadro abaixo:

Nome	Empresa	Tipo	Similar à	Execução
Javascript	Sun	Interpretada	Java/C	Cliente
VBscript	Microsoft	Interpretada	BASIC	Cliente/Servidor
PHP	opensource	Interpretada	C/Java	Servidor
ASP	Microsoft	Interpretada	BASIC	Servidor
JSP	Sun	Interpretada	Java/C	Servidor
Java Servlets	Sun	Compilada	Java	Servidor
ASP .Net	Microsoft	Compilada	BASIC .Net	Servidor

Existem outras, mas de qualquer forma não existe uma grande variedade de produtores de linguagens para Web Dinâmica. Existem aquelas que rodam do lado do cliente, aquelas que podem rodar do lado do cliente ou do servidor e aquelas que rodam apenas no lado do servidor.

Dentre as que rodam no lado do servidor temos aquelas que são interpretadas e aquelas que são compiladas. Entretanto, vale notar que o desempenho das linguagens de Web Dinâmica compiladas têm um desempenho comparável às interpretadas, fazendo diferença apenas em sites com milhões de acessos diários.

É importante notar que todas as linguagens server-side têm uma característica em comum: para o navegador, do lado do cliente, é como se não existisse nada além de um servidor web de páginas estáticas do outro lado. Isso ocorre porque **nenhum** código da linguagem sendo usada será enviado ao navegador: apenas o **resultado** do processamento de um código será enviado ao navegador. Por exemplo, o código abaixo:

```
print("<H1>T&iacute;tulo da p&aacute;gina</H1>\n");  
print("<P>Texto da página!</P>\n");
```

Após ser processado no servidor, chegará ao navegador do cliente da seguinte forma:

```
<H1>T&iacute;tulo da p&aacute;gina</H1>  
<P>Texto da página!</P>
```

A idéia é a seguinte:

Página	_____ \	Interpretador	_____ \	HTML	_____ \	Navegador
HTML+PHP	/	PHP	/	Puro	/	(Cliente)

Por esta razão, o nível de compatibilidade que se consegue com uma linguagem server-side é altíssimo.

## 2. O funcionamento da Linguagem PHP

Mas como e quando tudo isso ocorre? Bem, quando o navegador solicita o download de uma página .PHP, o servidor Web pensa: "Ei, eu não entendo esse tal de PHP... eu só entendo HTML! Vou chamar um intérprete!".

Nesta hora, ele chama os serviços do PHP, que é um programa que vai ler o arquivo .PHP e interpretá-lo, transformando-o em um arquivo .HTML padrão, e dizer pro servidor Web: "Olha! Lembra aquele PHP que você pediu que eu interpretasse? Então, está pronto. É esse arquivo HTML aqui."

Assim, o servidor Web pega o HTML e, reconhecendo o formato, envia todo o conteúdo para o navegador, que receberá um arquivo HTML puro, feito como se algum ser humano o tivesse criado naquele instante... e o apresentará normalmente para o usuário.

Mas como é um arquivo .PHP por dentro? É algo tão complexo assim? Na verdade, não. Um arquivo .PHP costuma ser um arquivo HTML normal, mas contendo alguns trechos de código embutido. A aparência geral de uma página com código PHP é a seguinte:

```
<HTML>
  <HEAD>
    <TITLE>
      Teste de PHP
    </TITLE>
  </HEAD>
  <BODY>
    <?php
      [... Código PHP ...]
    ?>
  </BODY>
</HTML>
```

A primeira função que veremos será para avaliar se o PHP está instalado corretamente no servidor: esta função é **phpinfo()**.

```
<HTML>
  <HEAD>
    <TITLE>
      Teste de PHP
    </TITLE>
  </HEAD>
  <BODY>
    <?php
      phpinfo();
    ?>
  </BODY>
</HTML>
```

Ao gravar isso em um arquivo com extensão **php**, index.php por exemplo, ao chamá-lo pelo navegador iremos ver uma página inteiramente gerada pelo PHP, mostrando toda a configuração da máquina onde o script está sendo executado.

### 3. Usando Variáveis

Antes de apresentar as variáveis, é interessante introduzir a função **print**, que serve para apresentar qualquer informação na tela. Ela já apresentada em um código anterior desta aula, agora apresentaremos uma página completa usando **print**:

```
<HTML>
  <HEAD>
    <TITLE>
      Teste de PHP
    </TITLE>
  </HEAD>
  <BODY>
    <?php
      print("<H1>Título da Página</H1>\n");
      print("<P>Texto da Página</P>\n");
    ?>
  </BODY>
</HTML>
```

Mas, para que colocar o código HTML dentro do PHP deste jeito? De fato, não faz muito sentido. Mas, quando modificamos um pouco o código, inserindo uma variável de controle, **\$page** por exemplo, pode ser que isso faça sentido:

Arquivo **teste.php** :

```
<HTML>
  <HEAD>
    <TITLE>
      Teste de PHP
    </TITLE>
  </HEAD>
  <BODY>
    <?php
      $page = 1;
      if ($page == 1)
      {
        print("<H1>Página 1</H1>\n");
        print("<P>Texto da Página 1</P>\n");
      }
      else
      {
        print("<H1>Página 2</H1>\n");
        print("<P>Texto da Página 2</P>\n");
      }
    ?>
  </BODY>
</HTML>
```

Neste trecho aparecem duas coisas novas: primeiro, a variável **\$page**. Tudo que começar com **\$** em PHP será considerado uma variável. É possível criar ou usar uma variável em qualquer lugar de uma página.

A segunda coisa nova é a estrutura de controle **if**. Este tipo de estrutura é bastante comum em qualquer linguagem, e no PHP não é diferente. A estrutura **if** serve para decidir quando realizar determinadas ações ou não: o código no bloco (delimitado por { }) após o **if** será executado apenas se a condição do **if** (entre parênteses ( ) ) for verdadeira. Ou seja, no código acima:

```
if ($page == 1)
{
  print("<H1>Página 1</H1>\n");
  print("<P>Texto da Página 1</P>\n");
}
```

Significa: "Se o valor da variável **\$page** for igual a 1, imprima os textos "<H1>Página 1</H1>" e "<P>Texto da Página 1</P>".

A palavra que vem em seguida, **else** indica o bloco que deve ser executado caso a afirmação do **if** não seja verdadeira.

Ao chamarmos esta página **teste.php** pelo navegador, ele a solicitará ao servidor Web que, por sua vez, solicitará que o interpretador PHP a interprete, antes de enviar o resultado ao navegador.

Da forma como a página foi escrita, o interpretador PHP responderá o seguinte texto, que será enviado ao navegador:

```
<HTML>
  <HEAD>
    <TITLE>
      Teste de PHP
    </TITLE>
  </HEAD>
  <BODY>
    <H1>Página 1</H1>
    <P>Texto da Página 1</P>
  </BODY>
</HTML>
```

Se, no código, modificarmos a linha:

```
$page = 1;
```

Para algo como:

```
$page = 2;
```

O resultado será:

```
<HTML>
  <HEAD>
    <TITLE>
      Teste de PHP
    </TITLE>
  </HEAD>
  <BODY>
    <H1>Página 2</H1>
    <P>Texto da Página 2</P>
  </BODY>
</HTML>
```

E ... se mudarmos aquela mesma linha para

```
$page = 0;
```

O resultado será, igualmente,

```
<HTML>
  <HEAD>
    <TITLE>
      Teste de PHP
    </TITLE>
  </HEAD>
  <BODY>
    <H1>Página 1</H1>
    <P>Texto da Página 1</P>
  </BODY>
</HTML>
```

Pois este resultado é para valores de `$page` diferentes de 1! Mas e se quisermos imprimir um número diferente para cada página? Há diversas formas. Há a forma "ruim", que seria inserir quantos ifs fossem necessários, como por exemplo em:



```
<HTML>
  <HEAD>
    <TITLE>
      Teste de PHP
    </TITLE>
  </HEAD>
  <BODY>
    <?php
      $page = 1;
      if ($page == 1) print("<H1>P&aacute;gina 1</H1>\n");
      else if ($page == 2) print("<H1>P&aacute;gina 2</H1>\n");
      else if ...
      ...
      else print("<H1>P&aacute;gina N</H1>\n");
    ?>
  </BODY>
</HTML>
```

Mas, se tudo que queremos é colocar o número na página, é mais fácil indicar o nome da variável na chamada da função **print**:

```
<HTML>
  <HEAD>
    <TITLE>
      Teste de PHP
    </TITLE>
  </HEAD>
  <BODY>
    <?php
      $page = 1;
      print("<H1>P&aacute;gina $page</H1>\n");
    ?>
  </BODY>
</HTML>
```

Agora, com um código muito menor, a página funciona corretamente para qualquer valor que coloquemos em \$page!

Lembrando que em PHP não dizemos os tipos de variáveis, ou seja, qual é um número, qual é um texto... o PHP "adivinha" sozinho. Por esta razão, cuidado com a execução de operações "estranhas", como por exemplo:

```
$var1 = "2";
$var2 = "2";
$var3 = $var1+$var2;
```

Qual será o resultado? O resultado é 4, mas observe que não era óbvio. O resultado poderia ser "22". Isso ocorre porque quando é realizada uma operação aritmética, o PHP sempre tenta converter o valor para um número... e isso pode inclusive conduzir a resultados estranhos. Por exemplo:

```
$var1 = "teste";
$var2 = "2";
$var3 = $var1+$var2;
```

Resulta em \$var3 valendo "2" (já que a conversão de teste para um número não é possível, o PHP considera o valor zero). Por outro lado, se desejamos a **concatenação** das strings, devemos usar o operador .= . Ele é usado da seguinte forma:

```
$var1 = "teste";  
$var2 = "2";  
$var3 = $var1;  
$var3 .= $var2;
```

Primeiro copiamos \$var1 para \$var3 e depois concatenamos àquele texto o conteúdo de \$var2.

#### 4. Laços for e while

Assim como na maioria das outras linguagens de programação, para conseguir execuções repetidas de alguns trechos de código usamos laços for e while. A sintaxe destes laços em PHP é:

```
for (definição_inicial; verificação_de_continuidade; regra_de_atualização)  
{  
    ....  
}  
  
while (verificação_de_continuidade)  
{  
    ...  
}  
  
do {  
    ...  
} while (verificação_de_continuidade);
```

Por exemplo, para escrever um texto "Linha N" 9 vezes na tela, onde N é o número da linha, usamos o seguinte código:

```
<HTML>  
  <HEAD>  
    <TITLE>  
      Teste de PHP  
    </TITLE>  
  </HEAD>  
  <BODY>  
    <?php  
      for ($i=1; $i<=9; $i++) print ("<P>Linha $i</P>\n");  
    ?>  
  </BODY>  
</HTML>
```

Que resulta em:

Linha 1  
Linha 2  
Linha 3  
Linha 4  
Linha 5  
Linha 6  
Linha 7  
Linha 8  
Linha 9

### **5. Bibliografia**

MUTO, C.A. PHP & MySQL: Guia Introdutório. Rio de Janeiro: Brasport, 2006.

RATSCHILLER, T; GERKEN, T; Desenvolvendo aplicações Web com PHP 4.0. Ed. Ciência Moderna, 2000.

Notas da Aula 11: Web Dinâmica: Funções e Parâmetros Básicos do PHP  
Prof. Daniel Caetano

**Objetivo:** Apresentar a forma GET de passar parâmetros e algumas funções básicas do PHP.

### Introdução

Na aula passada, foi apresentada a linguagem PHP e algumas de suas características. Entretanto, com o que vimos na aula passada, nossa página era ainda muito limitada: para modificar seu conteúdo ainda precisávamos modificar o código!

Nesta aula, veremos que isso nem sempre é necessário, se soubermos usar **parâmetros** corretamente. Há várias formas de passar parâmetros para uma página; nesta aula nos veremos a primeira delas.

Adicionalmente, veremos algumas funções básicas do PHP, que possibilitem, de alguma forma, incrementar um pouco o visual de nossas páginas.

## 1. O que são Parâmetros?

Na aula anterior, vimos que podíamos modificar o comportamento de uma página inteira simplesmente modificando um valor de variável (no caso, era a variável **\$page**). Isso, quando é feito no código da página, é interessante mas muito pouco cômodo. Seria muito interessante se pudéssemos modificar o valor de uma variável **sem** precisar modificar o código da página. É exatamente para isso que os parâmetros servem!

Resumidamente, a passagem de parâmetros para uma página é uma forma de modificar o conteúdo de determinadas variáveis antes que a página seja executada (de forma que ela se comporte como desejamos) e sem ter que alterar o código da página em si.

### 1.1. Como Passar Parâmetros?

A primeira dúvida que surge é: ok, eu posso modificar algumas variáveis sem ter de mexer no código da página... mas como é que eu faço isso?".

Há diversas formas de realizar esta tarefa; na aula de hoje, entretanto, veremos apenas uma delas: através da URL, ou seja, do endereço da página.

Algumas aulas anteriores vimos que uma URL completa tinha as seguintes partes:

`www.servidor.com/diretorio/pagina.html#secao`

Bem, é possível complementar essa URL, indicando valores de variáveis que serão passadas para a página. Para isso, devemos usar o separador `?`, e em seguida definir o nome da variável e o seu valor. Por exemplo, se quero definir a variável **page** como tendo o valor 1, basta indicar:

`www.servidor.com/diretorio/pagina.html#secao?page=1`

ou, dependendo da página (se ela não tiver ancoradouros intermediários), pode ser algo simplesmente assim:

`www.servidor.com/diretorio/pagina.html?page=1`

Mas... e se quisermos passar **mais de um** parâmetro? Basta separar seus valores com o separador **&**, como apresentado abaixo:

`www.servidor.com/diretorio/pagina.html?page=1&user=1345`

Onde são definidos os valores de **page** como **1** e de **user** como **1345**.

## 1.2. Como Receber Parâmetros?

"Ok, eu fiz isso mas minha página continuou sem saber o valor das variáveis \$page e \$user..."

Sim, isso é esperado, por questões de segurança. Perceba que se fosse um mecanismo automático, um cracker qualquer poderia modificar **qualquer** variável de sua página, modificando o comportamento da mesma e conseguindo acesso onde não deveria.

Assim, o PHP exige que você **selecione** especificamente as variáveis que deseja receber pela URL. Para isso, o PHP coloca todas as variáveis definidas na URL em um lugar especial e tudo que você precisa fazer é copiar o valor deste lugar para a variável que desejar.

Este lugar especial é um "vetor", que é como se fosse uma variável com vários compartimentos, cada um deles guardando um valor e tendo um nome diferente. O nome deste vetor que guarda os parâmetros da URL é **\$\_GET**. Para lermos o valor de uma variável, temos que indicar o nome dela no vetor, por exemplo: **\$\_GET['page']**.

Assim, podemos ler os valores de page e user, enviados pela URL, da seguinte forma:

```
$var1 = $_GET['page'];  
$var2 = $_GET['user'];
```

Ou, como é mais usual:

```
$page = $_GET['page'];  
$user = $_GET['user'];
```

Desta forma, podemos obter as variáveis enviadas pela URL, mas apenas aquelas que desejarmos. Se o usuário redefinir uma outra variável **authorized** como **1**, este valor será ignorado, porque nossa página não terá requisitado este valor.

A partir do momento que colhemos os valores da URL usando \$\_GET, basta usar a variável em que colocamos os valores (no exemplo anterior, \$var1 e \$var2 ou \$page e \$user) como se ela tivesse sido definida dentro do próprio documento PHP.

## **2. Funções Básicas**

Há algumas funções e instruções no PHP que são muito usadas, pela sua praticidade. Aqui serão apresentadas algumas delas, dentro de funções prontas e úteis.

### **2.1. A Função Date**

A função **date** retorna uma data completa, de acordo com o formato indicado:

```
string date ( string $formato [, int $carimbo_de_hora] )
```

Por exemplo:

```
$hoje = date('d-m-Y');  
print ("<P>Hoje &acute; $hoje</P>\n");
```

Para mostrar também as horas:

```
$agora = date('d-m-Y H:i:s');  
print ("<P>Agora &acute; $agora</P>\n");
```

## 2.2. A Instrução For

A instrução **for**, já apresentada anteriormente, é bastante usada, sobretudo para a criação de menus, por exemplo. É possível fazer a seguinte construção:

```
// Define as opções do menu em uma matriz
$menu[0][0] = "Home";
$menu[0][1] = "http://www.caetano.eng.br/";
$menu[1][0] = "Ajuda";
$menu[1][1] = "http://www.caetano.eng.br/ajuda/";
$menu[2][0] = "Contato";
$menu[2][1] = "mailto:daniel@caetano.eng.br";

// Pega o número de entradas da Matriz
$count = count($menu);

// Para cada uma das entradas da matriz...
for ($i=0; $i<$count; $i++)
{
    // Imprime o link
    $link = $menu[$i][0];
    $url = $menu[$i][1];
    print("<A HREF=\"$url\">$link</A><BR>\n");
}
```

## 2.3. A Função Strlen

A função **strlen** também é bastante usada, e serve para contar quantos caracteres tem uma string. Sua forma de uso é:

```
int strlen ( string $string )
```

Exemplo:

```
$str = 'abcdef';
echo strlen($str);           // Echo é similar à print, mas pode imprimir
                             // valores de funções, qualquer tipo de variável, etc.
```

Que responderá "6".

```
$str = ' ab cd ';
echo strlen($str);
```

Que responderá "7".

#### 2.4. A Função strcmp

A função **strcmp** compara duas strings e responde 0 se ambas são iguais, um número negativo se a primeira é menor que a segunda e um número positivo se a segunda é menor que a primeira. Sua forma de uso é:

```
int strcmp ( string str1, string str2 )
```

Por exemplo:

```
$str1 = 'ab cd ';  
$str2 = 'ab cd ';  
echo strcmp($str1, str2);
```

Responderá 0.

```
$str1 = 'ab cd ';  
$str2 = 'zb cd ';  
echo strcmp($str1, str2);
```

Responderá um número positivo.

#### 2.5. A Função substr

A função **substr** pega "um pedaço" de uma string. Sua forma de uso é:

```
string substr ( string string, int início [, int comprimento] )
```

Por exemplo:

```
$str = 'abcdefgh';  
echo substr($str, 2, 4);
```

Responderá: cdef.

#### 2.6. A "instrução" Include

A "instrução" **include** serve para adicionar um arquivo dentro de outro. É bastante comum seu uso e sua forma é:

```
include "nome_de_arquivo"
```



Este nome de arquivo pode ser um outro arquivo .php, um arquivo .html ou mesmo um .txt. Por exemplo, suponhamos que temos um arquivo **menu.php** onde exista o código que desenha o menu da página. Para adicionar este menu em uma página qualquer, basta usar o código:

```
include "menu.php"
```

### **3. Como Criar Minha Própria Função?**

Neste ponto talvez alguém se pergunte: "Mas como é que eu posso criar minha própria função?" É bastante simples. Basta usar uma "instrução" chamada **function**. A forma de realizar esta atividade é:

```
function nome_da_função($parametro1, $parametro2, ..., $parâmetro n)
{
}
```

Por exemplo:

```
function ImprimeDataCompleta()
{
    $agora = date('d-m-Y H:i:s');
    print ("<P>Agora &eacute; $agora</P>\n");
}
```

### **4. Exercício**

Implemente a impressão de um menu e de data e hora na página que vínhamos desenvolvendo ou em sua própria página.

### **5. Bibliografia**

MUTO, C.A. PHP & MySQL: Guia Introductório. Rio de Janeiro: Brasport, 2006.

RATSCHILLER, T; GERKEN, T; Desenvolvendo aplicações Wev com PHP 4.0. Ed. Ciência Moderna, 2000.

Notas da Aula 12: Web Dinâmica: Passando Parâmetros com Formulários  
Prof. Daniel Caetano

**Objetivo:** Apresentar a forma POST de passar parâmetros.

### Introdução

Na aula passada foi visto como passar parâmetros pelo modo GET, ou seja, pela barra de endereços do navegador. Entretanto, apesar de ser uma forma extremamente simples de passar parâmetros, esta é uma maneira deselegante e limitada.

É deselegante porque permite que o usuário veja toda a "sujeira" por trás do funcionamento de sua página, permitindo inclusive que ele altere facilmente alguns valores que não deveria modificar.

Além disso, é limitada, porque impede que alguns tipos de dados sejam passados: o endereço web tem limitação de tamanho. Por esta razão, imagens e outros arquivos grandes não podem ser transmitidos pelo método GET.

Nesta aula, veremos como resolver estes problemas, passando parâmetros através de formulários.

### 1. O que são Formulários?

**Formulários são conjuntos de campos de entrada de dados** que permitem que o conteúdo destes campos sejam **enviados para um endereço específico, onde eles serão processados**.

Um exemplo de um formulário:



Instant Messenger v2.02

Digite aqui sua mensagem...\*

Uma mensagem

Nome\*: nome

E-mail: e@mail.com

Link: http://www.server.com/

Espaço: 16 (máx 4096)

Enviar

A estrutura básica de um formulário é:

```
<FORM NAME="nome_do_formulário" METHOD="post" ACTION="pagina_php_destino.php">  
    [ ... campos do formulário ]  
</FORM>
```

Note que a tag <FORM>...</FORM> delimita a área do formulário.

Um formulário sempre deve possuir um botão "Enviar", "Aplicar", "Atualizar"... ou seja, um botão que acionará a ação que, invariavelmente, culmina com o envio dos dados para um determinado endereço - normalmente uma nova página PHP - onde os dados serão analisados e processados. Além deste botão, um formulário pode conter diversos outros elementos, que serão apresentados a seguir.

### **1.1. Campos de Texto**

Os campos de texto são compostos por uma linha onde é possível digitar um texto qualquer, sendo que cada campo de texto deve ter seu próprio nome. A sintaxe é a seguinte:

```
<INPUT TYPE="text" NAME="nome1" VALUE="valor_inicial">
```

Isso faz aparecer na página o seguinte campo:

O nome deste campo será "nome1". Veremos a importância deste fato na seção 2.

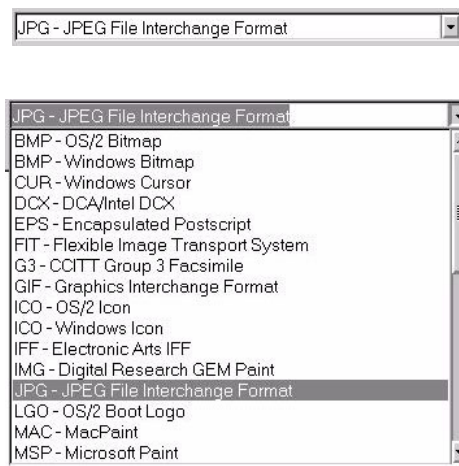
### **1.2. Lista de Seleção (combobox)**

As listas de seleção são aquelas listas "drop-down", que apresentam apenas um valor mas, se clicarmos no botão à direita delas, uma lista maior é mostrada para que selecionemos um dos valores existentes. Para criar uma lista deste tipo, a sintaxe é a seguinte:

```
<SELECT NAME="nome2">  
    <OPTION VALUE="valor1">Valor 1: a</OPTION>  
    ...  
    <OPTION VALUE="valor2">Valor 2: b</OPTION>  
</SELECT>
```

Cada linha <OPTION>...</OPTION> será uma opção da lista. O "VALUE" de cada option é o valor que será associado à lista de seleção (valor selecionado) quando o conteúdo do formulário for enviado. O texto entre <OPTION>...</OPTION> é o texto que será apresentado na lista.

Abaixo, um exemplo de uma lista de seleção:



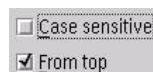
### **1.3. Caixa de Seleção (checkbox)**

As caixa de seleção são controles que permitem você marcar algo como "sim" ou "não", ou seja, indicar opções que você deseja e desmarcar opções que você não deseja. A sintaxe básica é:

```
<INPUT TYPE="checkbox" NAME="nome3" VALUE="valor" [CHECKED]>
```

Onde o "CHECKED" é um parâmetro adicional que indica se, inicialmente, a opção estará marcada ou não. O valor VALUE será associado ao controle apenas se a caixinha estiver selecionada. Caso contrário, o valor associado ao controle será vazio.

Exemplo de caixas de seleção:



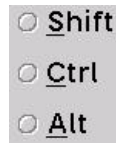
### **1.4. Botões de Opção (radiobox)**

Os botões de opção são usados quando temos um pequeno número de alternativas fixas para uma mesma opção, e apenas uma delas pode ser selecionada de cada vez. Um exemplo de sintaxe seria:

```
<INPUT TYPE="radio" NAME="sexo" VALUE="fem" CHECKED>Feminino  
<INPUT TYPE="radio" NAME="sexo" VALUE="masc">Masculino
```

Note que ambos os campos possuem o mesmo nome (NAME), já que ambos se referem à mesma seleção. O próprio navegador se encarrega de garantir que apenas um deles está selecionado de cada vez. Mais uma vez, a propriedade CHECKED existe para indicar qual opção estará marcada inicialmente.

Um exemplo de botões de opção pode ser visualizado abaixo:



### **1.5. Botões de Envio (submit)**

Como dito inicialmente, praticamente todos os formulários precisam ter um botão de envio, para que o usuário indique quando quer que as informações sejam transmitidas. Para incluir um botão deste tipo, sintaxe básica é:

```
<INPUT TYPE="submit" VALUE="texto_do_botao">
```

### **1.6. Outros Tipos de Controle**

Além destes 5 tipos básicos de controle, existem os tipos de controle "de bloco de dados", como TEXTAREA e FILE. A primeira serve para colar um grande bloco de texto e a segunda para enviar um arquivo. As sintaxes básicas são:

```
<TEXTAREA ROWS="5" COLS="35" MAXLENGTH="10240">Texto</TEXTAREA>
```

ROWS indica o número de linhas, COLS o número de colunas e MAXLENGTH o número máximo de caracteres que é possível digitar na área de texto.

Já a sintaxe do FILE é:

```
<INPUT TYPE="file" NAME="nome5">
```

No caso do file, entretanto, é preciso acrescentar um parâmetro na tag FORM, para que a codificação do arquivo seja feita corretamente e ele chegue intacto ao servidor:

```
<FORM NAME="nome" METHOD="post" ENCTYPE="multipart/form-data" ACTION="pagina.php">
```

## **2. Como Receber Parâmetros?**

Bem, vimos muitas formas de enviar dado pelo formulário, mas... como recebê-las? O processo é bem parecido com o que fazíamos com o método GET (que, aliás, pode ser usado com os formulários, alterando o parâmetro "METHOD" para "get" ao invés de "post").

Quando um formulário é enviado para uma página (indicada no parâmetro ACTION), o PHP coloca todos os valores do formulário específico dentro de um "vetor" (aquela variável com vários compartimentos, em que cada compartimento tem um nome). O nome deste vetor que guarda os parâmetros de um formulário POST é **\$\_POST**. Para lermos o valor de um campo, temos que indicar o nome dele no vetor. Por exemplo: **\$\_POST['campo1']**.

Assim, podemos ler os valores do campo nome e password de um formulário da seguinte forma:

```
$nome = $_POST['nome'];  
$pass = $_POST['password'];
```

Desta forma, podemos obter, em variáveis, os valores dos campos enviados por um formulário. Como estes nomes de campos e valores não são apresentados em lugar algum visível pelo usuário, acaba por ser um método mais elegante.

Adicionalmente, este método não tem limite de tamanho dos dados enviados. O PHP define um limite inicial de 8MB, mas é possível aumentar este valor através do arquivo de configuração do PHP.

### **3. Exercício**

Implemente um campo de busca em sua própria página, redirecionando para um arquivo **resultado\_busca.php**. Não precisa programar a busca.

### **4. Bibliografia**

MUTO, C.A. PHP & MySQL: Guia Introdutório. Rio de Janeiro: Brasport, 2006.

RATSCHILLER, T; GERKEN, T; Desenvolvendo aplicações Web com PHP 4.0. Ed. Ciência Moderna, 2000.

Notas da Aula 13: Web Dinâmica: Integrando PHP com SQL  
Prof. Daniel Caetano

**Objetivo:** Apresentar a forma de acessar bancos de dados SQL usando PHP.

### Introdução

Nas aulas anteriores, vimos uma base sobre como usar o PHP para construir páginas web dinâmicas. Entretanto, a parte de conteúdo ainda está muito "estática". A única maneira que vimos de facilitar o manuseio de conteúdo é o uso da diretiva **include**, mas isso está longe de ser uma solução adequada.

A razão para isso é que usando arquivos separados para o conteúdo, inserindo-os na página através de includes, não permite que façamos uma busca adequada. Para que possamos ter sistemas de busca eficientes, é necessário usar banco de dados.

O PHP fornece muitas formas de acesso a banco de dados, a diversos tipos de bancos de dados diferentes. Em especial, fornece até mesmo um sistema de abstração de bancos de dados.

Neste curso veremos como usar o banco de dados MySQL que, em sua versão 5.x, deixa muito pouco ou nada a desejar para grandes bancos de dados comerciais, com a vantagem de ser um produto de código aberto e uso livre.

### 1. Qual o Processo de Acesso ao MySQL com PHP?

O processo básico de acesso é:

- Conexão com o Servidor SQL
- Seleção de Banco de Dados
- Buscas (queries)
- Desconexão

É um processo bem simples. Vejamos como cada um deles é executado.

#### Conexão com o Servidor SQL

Antes de mais nada, obviamente, precisamos conectar com o servidor. O PHP já possui uma função pronta para realizar essa conexão: basta que forneçamos os dados para que a conexão ocorra. O nome desta função é **mysql\_connect**.

A sintaxe básica desta função é:

*recurso* **mysql\_connect** ([string \$servidor [, string \$nome\_usuario [, string \$password ]]])

O significado de servidor, nome\_usuario e password são mais ou menos óbvios: são os parâmetros da conexão. Entretanto, o retorno *recurso* exige alguma explicação. Quando chamamos esta função, uma conexão é estabelecida entre o micro onde o PHP roda e o servidor SQL. Este *recurso* retornado é o *número identificador* desta conexão.

Assim, o uso mais comum seria:

```
$sqlcon = mysql_connect("localhost", "usuario_web", "senha");
```

Feito isso, considerando que existe um usuário "usuario\_web" cuja senha é "senha" e que pode acessar a partir do computador local, \$sqlcon conterá o identificador da conexão com o SQL atual.

O próximo passo é selecionar o banco de dados.

### **Seleção de Banco de Dados**

A seleção de banco de dados é simples, também, já que há uma função pronta para realizar esta atividade no PHP, chamada **mysql\_select\_db**. Sua sintaxe básica é:

*bool* **mysql\_select\_db** ( string \$nome\_do\_database [, resource \$identificador\_sql] )

O "nome\_do\_database" refere-se exatamente ao banco de dados que queremos selecionar e o identificador\_sql é justamente o identificador retornado pela função mysql\_connect. Caso tenhamos apenas uma conexão ativa, não é necessário especificar o identificador\_sql.

O retorno desta função (bool) é um valor "falso" ou "verdadeiro", indicando o fracasso ou sucesso da seleção, respectivamente. A seleção pode falhar por diversos fatores, como por exemplo: banco de dados não existe, usuário não tem acesso àquele banco de dados, etc.

Para visualizar este (e outros erros), podemos usar a função **mysql\_error**, que sempre devolverá o erro ocorrido na última instrução para o mysql. Seu uso, normalmente, é feito da seguinte forma:

```
$texto_do_erro = mysql_error ( [resource $identificador_sql] )  
print ("Erro no SQL: " . $texto_do_erro);
```

Considerando que a seleção do banco de dados ocorreu com sucesso, é hora de realizar uma busca.



### **Buscas (queries)**

O ponto fundamental de um banco de dados são as buscas (SELECTs), ou seja, a recuperação das informações. Isso é feito também através de uma função do PHP, mas o comando SELECT deve ser construído anteriormente. A função que envia o comando SELECT para o banco de dados e pega sua resposta é a função **mysql\_query**. Sua sintaxe básica é:

```
resource mysql_query ( string $query [, resource $identificador_sql] )
```

O retorno desta função é um ponteiro para os resultados e, assim, deve ser armazenado. O uso mais comum desta função é:

```
$query = "SELECT id, titulo, text FROM database WHERE text LIKE '%$busca%'";  
$result = mysql_query ($query);
```

Caso a busca encontre um erro, o valor de \$result será o booleano "FALSE". Caso contrário, precisamos arrumar uma maneira de ler os dados da resposta. Para isso, usamos a função **mysql\_fetch\_array**. Sua sintaxe básica é:

```
matriz mysql_fetch_array ( resource $result )
```

Continuando o exemplo anterior, poderíamos ter:

```
$query = "SELECT id, titulo, text FROM database WHERE text LIKE '%$busca%'";  
$result = mysql_query ($query);  
$linha = mysql_fetch_array ($result);
```

E, como consequência, \$linha['id'] passaria a conter o valor da coluna 'id' da resposta, \$linha['titulo'] passaria a conter o valor da coluna 'titulo' da resposta e \$linha['text'] passaria a conter o valor da coluna 'text' da resposta.

Poderíamos finalizar o código da seguinte forma:

```
$query = "SELECT id, titulo, text FROM database WHERE text LIKE '%$busca%'";  
$result = mysql_query ($query);  
$linha = mysql_fetch_array ($result);  
print ("ID: " . $linha['id'] . "<BR>\n");  
print ("T&iacute;tulo: " . $linha['titulo'] . "<BR>\n");  
print ("Texto: " . $linha['text'] . "<BR>\n");
```

Mas isso funcionaria bem se a resposta fosse única, ou seja, apenas **uma** das entradas do banco de dados fosse respondida pelo SELECT. O que fazer caso as respostas sejam muitas? Simples: realizaremos vários mysql\_fetch\_array. Mas como saber quando parar? Simples: mysql\_fetch\_array retorna **NULL** quando não há mais respostas a ler. Assim,

podemos colocar tudo aquilo dentro de um loop, em que todas as respostas são coletadas e apresentadas:

```
$query = "SELECT id, titulo, text FROM database WHERE text LIKE '%$busca%'";
$result = mysql_query ($query);
$linha = mysql_fetch_array ($result);
while ($linha != NULL)
{
    print ("ID: " . $linha['id'] . "<BR>\n");
    print ("T&iacute;tulo: " . $linha['titulo'] . "<BR>\n");
    print ("Texto: " . $linha['text'] . "<BR>\n");
    print ("<BR>\n");
    $linha = mysql_fetch_array ($result);
}
```

Finalizada a consulta, o próximo passo é a desconexão do banco de dados.

### **Desconexão**

Assim que todas as consultas são finalizadas, devemos desconectar do banco de dados para liberar a conexão. Isso pode ser feito usando a função **mysql\_close**, que tem a seguinte sintaxe:

```
bool mysql_close ( [resource $identificador_sql] )
```

O parâmetro de identificador especifica qual conexão você quer fechar (se houver mais de uma). O retorno é se a conexão foi fechada com sucesso ou não, indicando TRUE ou FALSE, respectivamente.

## **2. Exercício**

Na aula passada implementamos um formulário de busca que enviava seus resultados para um outro arquivo, **resultado\_busca.php**. Agora criaremos um pequeno banco de dados, e realizaremos uma busca nele.

### **Criando o Banco de Dados**

Para criar o banco de dados, entre no prompt do MySQL, digitando

```
mysql -uroot
```

dentro do diretório \bin do MySQL. Dentro do prompt, vamos primeiro criar o banco de dados:

```
CREATE DATABASE webpage;
```

Agora, vamos autorizar um usuário "webuser" a usar este banco de dados, com permissão total acessando pelo computador local, com password "umpassword":

```
GRANT ALL PRIVILEGES ON webpage.* TO 'webuser'@'localhost' IDENTIFIED  
BY 'umpassword';
```

Agora, vamos selecionar o banco de dados que criamos e criar uma tabela:

```
USE webpage;
```

```
CREATE TABLE webdata (id INT NOT NULL AUTO_INCREMENT, titulo  
VARCHAR (255), text BLOB, PRIMARY KEY(id));
```

Agora vamos inserir uns dados neste banco de dados:

```
INSERT INTO webdata VALUES (NULL, "Pagina 1", "Texto da pagina 1");  
INSERT INTO webdata VALUES (NULL, "Pagina 2", "Texto da pagina 2");  
INSERT INTO webdata VALUES (NULL, "Outro titulo", "Um texto completo");
```

Você pode verificar se está tudo ok com o select abaixo:

```
SELECT * FROM webdata;
```

Verificando que todos os dados estão lá, basta sair do banco de dados:

```
EXIT
```

### **Acessando o banco de dados pela página Web de busca**

Vamos usar o código desenvolvido na aula para apresentar as informações do banco de dados... mas como a informação da busca veio de um formulário, dentro da página **resultado\_busca.php** devemos ter algo do seguinte tipo:

```
// Pega o valor indicado pelo usuário na página para a busca.  
$busca = $_POST['busca'];  
  
// Define a query  
$query = "SELECT id, titulo, text FROM database WHERE text LIKE '%$busca%'";  
// Colhe os resultados  
$result = mysql_query ($query);
```

```
// Recupera a primeira linha da resposta
$linha = mysql_fetch_array ($result);
// Enquanto houver linhas na resposta
while ($linha != NULL)
{
    // Imprime os dados coletados
    print ("ID: " . $linha['id'] . "<BR>\n");
    print ("T&iacute;tulo: " . $linha['titulo'] . "<BR>\n");
    print ("Texto: " . $linha['text'] . "<BR>\n");
    print ("<BR>\n");
    $linha = mysql_fetch_array ($result);
}
```

Pronto! Agora vá até o formulário que você criou na aula passada, que deve conter o seguinte:

```
<FORM METHOD="post" ACTION="resultado_busca.php">
    <INPUT TYPE="text" NAME="busca">
    <INPUT TYPE="submit" VALUE="Procurar">
</FORM>
```

E tente buscas diferentes, e veja os resultados.

Perceba também que você pode usar este tipo de estrutura para armazenar suas páginas. Se colocar texto da página em "text", "id" pode ser o identificador de número de página. Assim, é possível fazer uma página da seguinte forma:

```
// Pega número da página, passado por Get (na URL)
$page = $_GET['page'];
// Define a query
$query = "SELECT titulo, text FROM database WHERE id = $page";
// Colhe os resultados
$result = mysql_query ($query);
// Recupera a primeira (e única, já que ID é chave primária) linha da resposta
$linha = mysql_fetch_array ($result);
// Imprime a página
print ("T&iacute;tulo: " . $linha['titulo'] . "<BR>\n");
print ("Texto: " . $linha['text'] . "<BR>\n");
```

E note que agora é possível fazer busca em toda a página (em seu conteúdo), já que o conteúdo está no banco de dados.

Combinando os elementos transmitidos nas últimas aulas, é possível criar páginas extremamente interessantes. Na realidade, a maior parte do conteúdo da

### **3. Bibliografia**

MUTO, C.A. PHP & MySQL: Guia Introductório. Rio de Janeiro: Brasport, 2006.

RATSCHILLER, T; GERKEN, T; Desenvolvendo aplicações Web com PHP 4.0. Ed. Ciência Moderna, 2000.