INSTITUTO SUPERIOR TÉCNICO

Sistemas Distribuídos

Tecnologias Base da World Wide Web

Miguel Filipe Leitão Pardal (miguel.pardal@dei.ist.utl.pt)

Fevereiro de 2003

Índice

Índice de Figuras

Objectivos

Pretende-se que no fim desta unidade, seja capaz de:

- Descrever a arquitectura da World Wide Web (cliente e servidor);
- Explicar o modelo de documentos da Web e o papel desempenhado pelos identificadores;
- Reconhecer os principais métodos do HyperText Transfer Protocol (HTTP) e descrever a sua utilização na Web, nomeadamente na passagem de argumentos entre cliente e servidor;
- Explicar as capacidades e limitações da HyperText Markup Language (HTML) para criação de documentos na Web;
- Descrever os objectivos da eXtensible Markup Language (XML), explicando as diferenças entre XML e HTML;
- Utilizar a XML para representar dados, construindo documentos que respeitem as regras da linguagem;
- Discutir as vantagens e desvantagens da XML em diferentes utilizações;
- Referir as tecnologias XML Schema e XSL, explicando a sua função.

1 – Introdução

A Internet é a maior rede de computadores do mundo, ligando actualmente milhões de computadores. A sua principal aplicação actual é a World Wide Web, formando um sistema distribuído de grande escala, baseado em documentos.

Nos últimos anos, a crescente utilização da Web por pessoas e organizações levou à construção de Web Sites com funcionalidades mais complexas e em maior número. Inicialmente a principal utilização da Web era consulta de documentos estáticos através de computadores pessoais. Hoje em dia, existem aplicações de maior interactividade com conteúdos dinâmicos, como é o caso do comércio electrónico, e existem também novas formas de acesso, através de dispositivos pequenos e móveis.

No seguimento desta evolução, começou a surgir a necessidade de interligar diferentes Web Sites para oferecer **serviços integrados** aos utilizadores. No entanto, a satisfação desta necessidade deparou-se com vários problemas, nomeadamente a não existência de mecanismos comuns de representação de informação e de controlo de transacções entre duas entidades. Surgiu então a iniciativa dos **Web Services**, com o objectivo de definir as normas necessárias para construir um **mercado electrónico de serviços**. Esta iniciativa é patrocinada por diferentes fabricantes da indústria informática, sendo um dos seus requisitos a interoperabilidade entre diferentes plataformas de serviços.

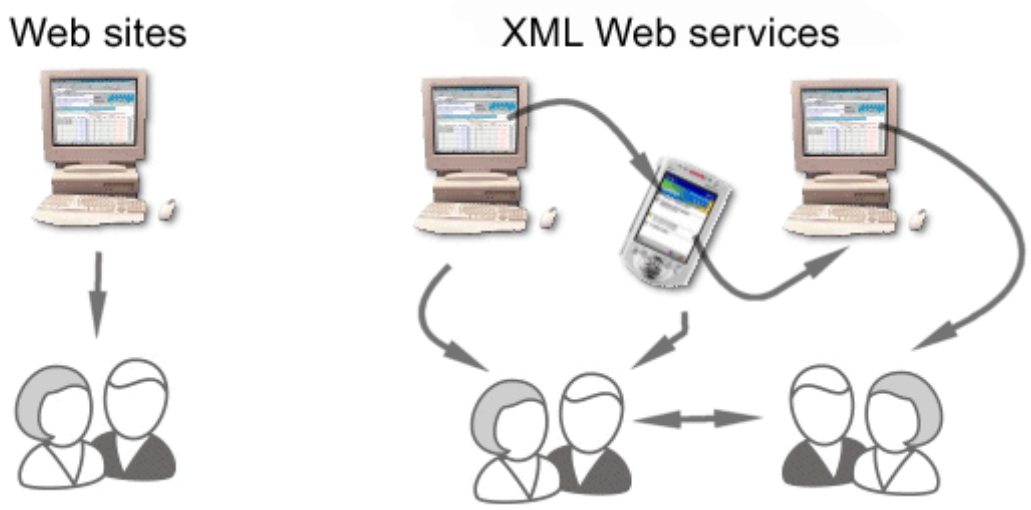

Figura 1 – Os Web Services como evolução dos Web Sites

A principal diferença conceptual entre ambas as utilizações da Web, é que os Sites são para pessoas, enquanto que os Services destinam-se a ser acedidos por outros serviços e programas de computador, sobretudo em transacções entre empresas (Business-to-Business).

A construção da infra-estrutura de Web Services partiu de um conjunto de tecnologias existentes na Web, por serem adequadas e por terem uma utilização generalizada. O **objectivo** deste texto é apresentar estas tecnologias base, descrevendo resumidamente a arquitectura da World Wide Web (WWW), o protocolo de comunicação HyperText Transfer Protocol (HTTP) e as linguagens HyperText Markup Language (HTML) e eXtensible Markup Language (XML). A compreensão destas tecnologias é necessária para a abordagem seguinte às tecnologias de Web Services.

2 – World Wide Web

Tal como já foi referido, a World Wide Web é actualmente a principal aplicação da Internet. Tanto é assim, que quando a maioria das pessoas se refere à Internet, está na realidade a referir-se à Web. A Web é uma aplicação da Internet entre outras, como o correio electrónico, por exemplo.

2.1 – Arquitectura

A Web pode ser vista como um sistema distribuído que permite o acesso a documentos, tendo uma **arquitectura** formada por clientes e servidores.

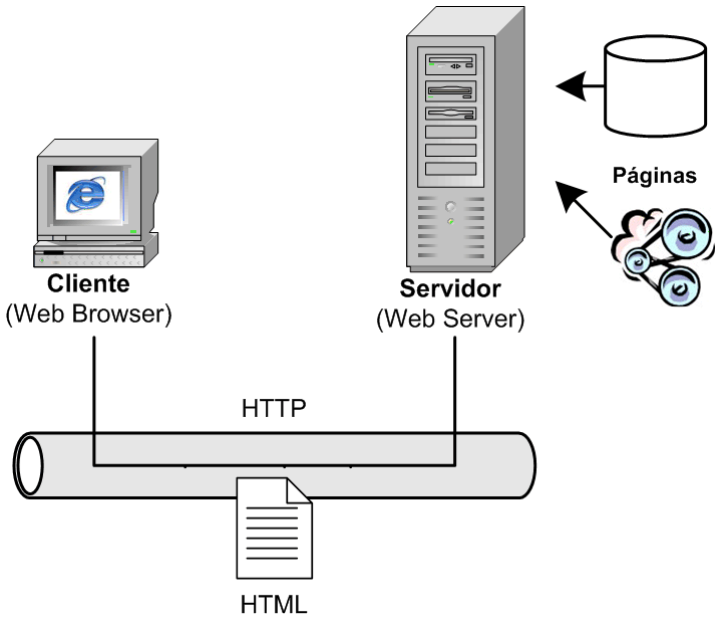

Figura 2 – Arquitectura da World Wide Web

Os **clientes**, normalmente designados por Web Browsers, apresentam os documentos aos utilizadores e permitem-lhes efectuar novos pedidos, sendo normalmente baseados em interfaces gráficas e amigáveis.
Exemplos: Microsoft Internet Explorer, Netscape.

Os **servidores**, normalmente designados por Web Servers, têm uma colecção de documentos, armazenados em ficheiros ou gerados dinamicamente para responder a pedidos de utilizadores. A principal função do servidor é responder a pedidos de documentos.
Exemplos: Microsoft Internet Information Server (IIS), Apache.

O protocolo de comunicação entre clientes e servidores utilizado na Web é o HyperText Transfer Protocol (HTTP), que é analisado na página 12.

Na Web tudo é um **documento**. Este modelo torna-se poderoso pela sua relativa simplicidade. Um documento pode ser composto por vários documentos mais simples, com diferentes tipos de informação (texto, imagem, som, entre outros).
Um dos aspectos mais importantes no modelo de documentos da Web, é a capacidade de definir ligações entre documentos, através de hyperlinks que utilizam identificadores. O conjunto das ligações entre vários documentos forma a "teia" que deu o nome à Web.

O formato utilizado para a maioria dos documentos na Web é o HyperText Markup Language (HTML), que é abordado na página 17. Outros tipos de formatos utilizados na Web são baseados na eXtensible Markup Language (XML), que é analisada na página 22.

2.2 – Identificadores

Os hyperlinks utilizam identificadores. Os identificadores da Web referenciam documentos e são designados Uniform Resource Identifiers (URIs). Um **URI** tem a seguinte estrutura sintáctica:

```
<scheme > : <scheme-specific-part>
```

Existem dois tipos de URIs:

- Uniform Resource Locators (URLs) – identificadores que incluem informação sobre como e onde aceder ao documento;
- Uniform Resource Name (URNs) – identificadores de documentos globalmente únicos, independentes da localização e persistentes.

O **URL** utiliza o `<scheme>` para definir como aceder ao documento, definindo o protocolo a utilizar. Alguns dos protocolos possíveis são http, ftp e telnet. A `<scheme-specific-part>` de um URL tem a seguinte estrutura:

```
//<user>:<password>@<host>:<port>/<url-path>
```

Os elementos são:

- `<user>` - nome de utilizador do site (opcional)
- `<password>` - senha de utilizador do site (opcional)
- `<host>` - nome do servidor ou endereço IP
- `<port>` - porto do servidor (opcional, quando não é especificado é utilizado o porto por omissão definido para o protocolo utilizado)
- `<url-path>` - define o restante caminho onde pode ser encontrado o documento.

A tabela seguinte apresenta exemplos de URLs válidos:

URL	Descrição
`http://www.ist.utl.pt/index.html`	Utiliza o protocolo HTTP (o porto utilizado por omissão é o 80)
`ftp://ftp.microsoft.com:2121/pub/file.zip`	Utiliza o protocolo FTP, contactando o servidor no porto 2121
`file://c:/tmp/text.txt`	Utiliza o sistema de ficheiros local

O **URN** utiliza é um identificador independente da localização, por isso utiliza um `<scheme>` específico: `urn`. A `<scheme-specific-part>` de um URN tem a seguinte estrutura:

```
<namespace-id> : <namespace-specific-string>
```

Os elementos são:

- `<namespace-id>` - espaço de nomes
- `<namespace-specific-string>` - parte do identificador com regras específicas em cada espaço de nomes

A tabela seguinte apresenta exemplos de URNs válidos:

URN	Descrição
`urn:isbn:0-13-349945-6`	Identificador de um livro, utilizando o código internacional ISBN (International Standard Book Number)
`urn:ietf:rfc:2648`	Identificador de um documento da IETF (Internet Engineering Task Force) que é responsável por desenvolver normas da Internet

Apesar da definição de URNs ser simples, o facto de a estrutura interna variar entre diferentes espaços de nomes, torna difícil a sua resolução para obter um documento. Por este motivo, a grande maioria dos identificadores utilizados na Web são URLs.

Fim do capítulo. Aspectos a reter:

World Wide Web:

- Arquitectura Cliente-Servidor (Web Browser – Web Server)
- Modelo de Documentos
- Documentos ligados por hyperlinks

Os identificadores de Documentos chamam-se Uniform Resource Identifiers – URIs, e têm dois tipos:

- Uniform Resource Locators – URLs, incluem a localização – como e onde aceder ao documento
- Uniform Resource Names – URNs, globalmente únicos, independentes da localização e persistentes

3 – HyperText Transfer Protocol

O HyperText Transfer Protocol (HTTP) é o protocolo utilizado na Web para transferir documentos entre cliente e servidor.

3.1 – Características

O HTTP é um protocolo de comunicação de nível **aplicacional**, que necessita de utilizar um protocolo de transporte fiável. Normalmente é utilizado o TCP/IP (Transport Control Protocol para protocolo de transporte e o Internet Protocol para protocolo de rede).

Figura 3 – Pilha de protocolos de comunicação normalmente associada ao HTTP

O TCP estabelece um **canal de comunicação com ligação** entre cliente e servidor, que apresenta as vantagens de ser fiável, bidireccional e garantir a sequencialidade da informação. As desvantagens são um maior consumo de recursos computacionais no servidor e o tempo de latência no estabelecimento do canal.

Outra característica do HTTP, é ser um **protocolo sem estado (stateless)**, o que significa que não são estabelecidas sessões com os clientes. Por outras palavras, o servidor não precisa de se "lembrar" dos clientes. Este facto permite poupar memória e simplificar o servidor, o que na Web é muito importante, pois permite uma maior escalabilidade do serviço (ou seja, ser capaz de atender muitos pedidos ao mesmo tempo sem perca significativa de desempenho).

3.2 – Versões

O protocolo HTTP tem duas versões: 1.0 e 1.1. A principal diferença entre ambas é a utilização mais eficiente de ligações TCP/IP efectuada pela versão 1.1. Enquanto que a versão 1.0 obriga a estabelecer uma ligação física diferente por cada documento pedido, a 1.1 aproveita uma mesma ligação física para efectuar uma sequência de pedidos. Dado que o tempo de estabelecimento de ligações é significativo e que um documento Web é normalmente composto por vários outros documentos (por exemplo, uma página pode ter várias imagens), o HTTP 1.1 obtém melhor desempenho pois consegue amortizar o custo da ligação.

3.3 – Métodos

O protocolo HTTP define um conjunto de **métodos**, correspondentes às operações que podem ser efectuadas.

Método	Descrição
GET	Pedido de documento
HEAD	Pedido apenas de cabeçalho de documento
PUT	Pedido para guardar um documento
POST	Fornecimento de informação para ser acrescentada ao documento
DELETE	Pedido para apagar um documento

Figura 4 – Métodos HTTP

O cliente executa um método HTTP construindo um pedido, enviando-o ao servidor, recebendo a resposta e tratando-a adequadamente. O servidor está continuamente à espera de receber pedidos para serem executados.

Um **pedido** é formado por: linha de pedido (com identificação do documento – URL), cabeçalho e corpo.

Uma **resposta** é formada por: linha de estado (com código de resultado), cabeçalho e corpo.

Os **cabeçalhos** contêm informação de controlo, para descrever aspectos relativos ao cliente, ao servidor ou ao documento. Por exemplo, existem campos de cabeçalho para indicar o nome e a versão do Web Browser, o prazo de validade do documento, entre outros.

Os dois métodos HTTP mais utilizados são o GET e o POST, seguidos pelo HEAD. O PUT e o DELETE são menos utilizados, porque é raro existir um grau de confiança nos clientes que lhes permita guardar ou apagar documentos no servidor.

3.4 – Passagem de Argumentos

Em algumas situações o cliente necessita de enviar dados para o servidor, especificando um conjunto de parâmetros. Por exemplo, para aceder à sua conta pessoal num site de comércio electrónico, o utilizador tem que enviar o seu nome e a sua senha de acesso. Existem três maneiras de efectuar a passagem de parâmetros: POST, GET e Cookies.

O método **POST** permite ao cliente enviar dados ao servidor. Quando recebidos, os dados são normalmente passados a um programa auxiliar que os utiliza para produzir uma resposta dinâmica. Existem várias tecnologias para implementar estes programas auxiliares, como por exemplo os CGI (Common Gateway Interface), os Java Servlets e as Microsoft ASP (Active Server Pages).
O pedido de POST inclui o URL do documento que vai receber os dados enviados, cujo nome é mapeado para o programa auxiliar. Os dados são passados no corpo, em pares nome=valor e que são invisíveis para o utilizador. À partida não existe limite para o volume de dados que pode ser enviado desta forma.
A limitação associada a este método é que tem que estar associado a um formulário, o que impõe restrições na interacção com o utilizador.

Uma alternativa para enviar dados para o servidor é utilizar método **GET**, utilizando o URL para enviar os argumentos. Os valores são anexados no fim do URL, segundo a sintaxe:

```
URL?name_arg1=value_arg1&name_arg2=value_arg2&...
```

O ponto de interrogação ('?') separa o URL dos argumentos. Os vários argumentos são separados entre si pelo caracter '&'.
Por exemplo, para enviar o nome e senha de utilizador para o programa auxiliar login.cgi, poderia ser usado o seguinte URL:

```
http://www.server.net/login.cgi?nome=joao&senha=segredo
```

Um aspecto que salta à vista neste exemplo, é que os valores têm visibilidade directa para o utilizador final, pois o URL surge na barra de endereço do Web Browser. Neste caso, em que se transmite uma senha de utilizador, esta situação é claramente indesejável.
Utilizando o método GET o volume de dados que pode ser transmitido é limitado pelo comprimento máximo do URL (que normalmente não excede os 4 Kilobytes).

A última alternativa para passar parâmetros para o servidor é a utilização de **Cookies**. Os cookies são pequenos ficheiros que são armazenados localmente no cliente e que registam informação sobre o utilizador. Um cookie está associado ao site que o criou. Sempre que o cliente efectua um pedido a esse site, o conteúdo do cookie é enviado no cabeçalho. Os cabeçalhos relacionados com cookies são:

- set-cookie – criação (servidor–>cliente);
- cookie – envio de conteúdo (cliente–>servidor).

Normalmente é utilizando este mecanismo que os Web Sites mantêm a identidade do utilizador entre diferentes visitas, o que permite apresentar mensagens personalizadas como "Olá João, bem-vindo de volta à sua conta pessoal", por exemplo.
O principal problema dos cookies é que podem ser utilizados abusivamente, sem consentimento do utilizador, possibilitando eventuais perdas de privacidade.

Fim do capítulo. Aspectos a reter:

O HyperText Transfer Protocol (HTTP) permite transferir documentos na Web entre cliente e servidor.

Principais características do HTTP:

- Protocolo de comunicação aplicacional;
- Utiliza TCP/IP para garantir canal fiável;
- Protocolo sem estado (stateless);
- Estruturado em métodos.

A versão HTTP 1.1 permite melhor desempenho que a 1.0.

Um método efectua uma operação, com pedido e resposta.
Os cabeçalhos dos pedidos e das respostas permitem trocar informação de controlo.

A passagem de argumentos entre cliente e servidor pode ser feita por:

- POST;
- GET;
- Cookies.

4 – HyperText Markup Language

A maior parte dos documentos na Web utilizam a HyperText Markup Language (HTML), que permite construir páginas de texto com imagens e outros conteúdos, permitindo também a definição de referências entre documentos através de hyperlinks.

4.1 – Linguagem de Etiquetas

A HTML é uma aplicação específica da Standard Generalized Markup Language (SGML). A SGML é uma linguagem muito poderosa, mas também muito complexa, e é utilizada sobretudo na área das publicações profissionais.
Tal como a SGML, a HTML é uma **linguagem baseada em etiquetas** ou marcas (tags ou markup). As etiquetas são utilizadas para anotar texto, acrescentando-lhe significado adicional. No caso da HTML, são normalmente utilizadas para formatar o texto.

Exemplo:
Vamos supor que queríamos escrever a frase seguinte numa página HTML:

```
O programador terminou o software.
```

No entanto, pretendia-se escrever "software" em itálico. Para conseguir este efeito é utilizada a etiqueta "`i`" da seguinte forma:

```
O programador terminou o <i>software</i>.
```

A abertura da etiqueta `i` (`<i>`) significa que o texto que se segue deve ser formatado em itálico até que surja a etiqueta de fecho correspondente (`</i>`). Depois de correctamente interpretada por um Browser, a frase seria apresentada como:

O programador terminou o *software*.

É possível combinar várias etiquetas para obter combinações de formatação. Por exemplo, podemos utilizar as etiquetas "`i`" e "`u`" para que o texto fique simultaneamente em itálico e sublinhado:

```
O programador terminou o <u><i>software</i></u>.
```

O programador terminou o ***software***.

Uma **etiqueta** pode ter três formas:

- Abertura (`<tag-name>`) – a partir desse ponto, o texto está abrangido pelo significado da etiqueta;
- Fecho (`</tag-name>`) – termina o significado de uma etiqueta aberta anteriormente;
- Vazia (`<tag-name />`) – maneira compacta de abrir e fechar imediatamente uma etiqueta. Não contém texto e serve para marcar uma posição no texto ou para fazer surgir um elemento específico. Por exemplo, o elemento "`hr`" define uma linha divisória de secção numa página na seguinte forma `<hr />`.

O conjunto formado pela etiqueta (abertura e fecho) e pelo seu contéudo (texto e outras etiquetas) é designado por **elemento**.

Uma etiqueta pode ter também **atributos**, que parametrizam o significado da etiqueta. A sintaxe utilizada é:

```
<tag-name attribute-name="attribute-value">
```

Uma etiqueta pode ter vários atributos.
Apenas as etiquetas vazias ou de abertura podem atributos.

Exemplo:

A etiqueta "p" define um parágrafo de texto. Este elemento pode ter o atributo "id" para associar um identificador ao parágrafo. Este atributo pode ser usado para referir este parágrafo no resto da página.

```
<p id="introducao">O programador terminou o software.</p>
```

4.2 – Principais Elementos

Os elementos HTML são vários e permitem definir a **formatação** e a **estrutura** de uma página.

Para uma apresentação eficaz dos principais elementos da linguagem, sugere-se o tutorial HTML de [W3Schools03].

4.3 – Estrutura de uma Página

Uma página HTML tem uma estrutura composta por um cabeçalho (head) e por um corpo (body). O cabeçalho descreve a página. O corpo define o conteúdo da página.

Exemplo de página HTML:

```
<html>
      <head>
            <title>Titulo da Pagina</title>
      </head>
      <body>
            <h1>Texto...</h1>
            <p>Primeiro paragrafo</p>
      </body>
</html>
```

Resultado da interpretação da página anterior num Browser:

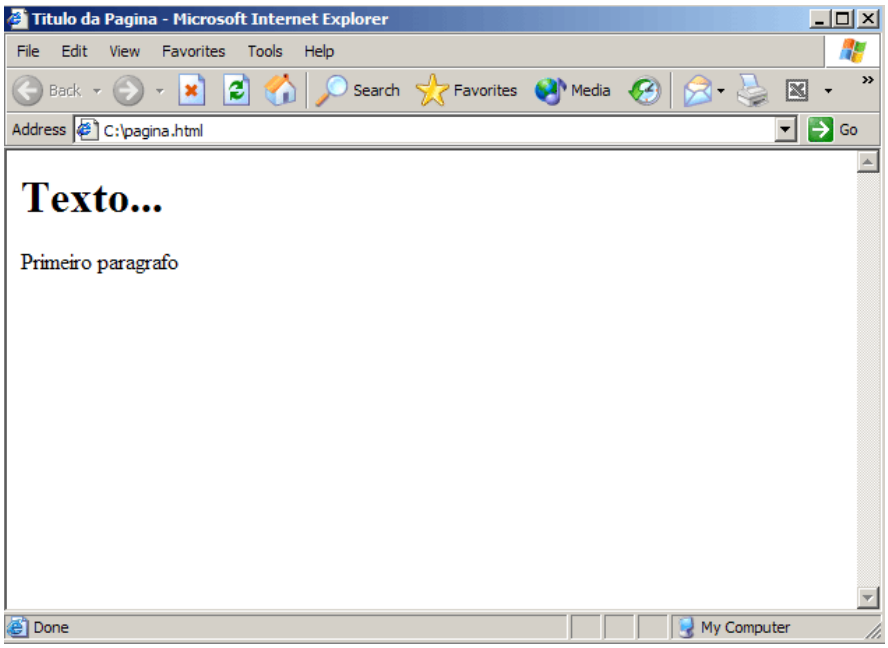

Figura 5 – Exemplo de uma página HTML interpretada

Neste exemplo, o cabeçalho (`head`) contém apenas o título da página. Esse título é apresentado no campo superior esquerdo da janela.
O corpo (`body`) é formado por um título de nível (`h1`) e por um parágrafo (`p`).

Os elementos `html`, `head` e `body` são utilizados para estruturar o documento em diferentes secções.

4.4 – Problemas

A utilização da HTML em páginas Web apresenta alguns problemas:

- Mistura entre apresentação e conteúdo da informação;
- Utilização de extensões específicas a determinados browsers (resultado da concorrência comercial);
- Validação "relaxada" da linguagem HTML pelos browsers, o que permite que sejam aceites documentos ligeiramente incorrectos (o que torna a interpretação de páginas ambígua e mais complexa do que seria necessário).

O problema da mistura entre apresentação e conteúdo é o mais grave, na óptica da construção de Web Services. A utilização da eXtensible Markup Language (XML), descrita no próximo capítulo, permite ultrapassar este problema.
Os restantes problemas são causados, sobretudo, pelo grande número de extensões que já foram efectuadas à linguagem, com desvios em relação ao seu objectivo inicial.

Fim do Capítulo. Aspectos a reter:

As páginas da Web utilizam a HyperText Markup Language (HTML), que permite construir páginas de texto com imagens e outros conteúdos, incluindo hyperlinks.

A HTML é uma linguagem baseada em etiquetas. As etiquetas atribuem significados ao texto que envolvem. Os elementos e atributos são formas complementares de associar informação ao conteúdo do documento.

Os elementos HTML definem a formatação e a estrutura da página.

O principal problema da HTML, na óptica de construção de Web Services, é a mistura entre apresentação e conteúdo da informação.

5 – eXtensible Markup Language

5.1 – Objectivos

A eXtensible Markup Language (XML) é uma linguagem concebida para **descrever informação**, baseada em etiquetas. Os **objectivos** da XML são diferentes dos da HTML, apesar de serem ambas linguagens baseadas em etiquetas. A XML descreve os dados (o que são) enquanto que a HTML apresenta os dados (que aspecto têm).

Exemplo de documento XML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<mensagem>
      <de>João</de>
      <para>Carla</para>
      <assunto>Reunião urgente</assunto>
</mensagem>
```

Neste exemplo, os elementos XML descrevem as diferentes partes da mensagem (remetente, destinatário e assunto). Estes elementos não estão definidos à partida na linguagem XML, tendo sido criados especificamente para este tipo de documento. Esta característica é o que dá **extensibilidade** à linguagem.

5.2 – Regras de Construção

As regras de **sintaxe** da XML são bastante simples, mas são seguidas rigorosamente pelos interpretadores (parsers). Deste modo, estes programas podem ser menos complexos, evitando situações de interpretação ambígua.

Sendo uma linguagem baseada em etiquetas. Os documentos em XML têm que respeitar rigorosamente as regras de construção de elementos e atributos, descritas em 4.1 – Linguagem de Etiquetas (na página 17).

Vamos utilizar o exemplo da página 22 para ilustrar as principais regras de construção de documentos XML. O documento inicia-se com a **declaração XML**, seguindo-se o início do **elemento raiz** do documento, que se chama `mensagem`. As três linhas seguintes contêm três **elementos filhos** de `mensagem` (`de`, `para` e `assunto`). Cada um destes elementos poderia ter outros sub-elementos e assim sucessivamente. O documento termina com o **fecho do elemento raiz**.

- A **declaração XML** pode surgir opcionalmente no início do documento:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

Esta declaração define a **versão** da XML e a **codificação de caracteres** utilizada. No exemplo, a versão é a 1.0 (que é a única existente actualmente) e o código de caracteres é o ISO-8859-1 (Latin-1/West European).
A declaração XML faz parte da estrutura do documento, mas não é um elemento. Para diferenciar, a sintaxe da declaração utiliza "`<? ... ?>`" em vez de "`< ... >`". Uma outra diferença é que a declaração não tem uma etiqueta de fecho correspondente.
O código de caracteres utilizado define a maneira como os diferentes caracteres que compõem o documento XML são representados no ficheiro. Caso sejam utilizados caracteres fora do código definido, o interpretador gera um erro ao ler o ficheiro. Alguns valores frequentes para esta definição são "UTF-8", "UTF-16" (correspondentes a Transformações Unicode – ver [Unicode03] para mais pormenores) e o já referido "ISO-8859-1".

- Os documentos XML tem que ter um **elemento raiz**, e esse elemento tem que ser único. Os restantes elementos têm que estar dentro do elemento raiz. Todos os elementos podem ter sub-elementos (**elementos filhos**):

```
<raiz>
   <filho>
      <neto>...</neto>
   </filho>
</raiz >
```

- Todos os elementos de um documento têm que conter o **fecho** das etiquetas respectivas:

```
<x>Incorrecto

<x>Correcto</x>
```

- Os elementos estão **correctamente aninhados** quando as etiquetas abrem e fecham em sequências inversas:

```
<x><y>Incorrecto</x></y>

<x><y>Correcto</y></x>
```

No primeiro exemplo: abrir x, abrir y, fechar x, fechar y –> Incorrecto.
No segundo exemplo: abrir x, abrir y, fechar y, fechar x –> Correcto.
Os sub-elementos têm que estar correctamente aninhados dentro do seu elemento pai.

- As etiquetas XML diferenciam entre maiúsculas e minúsculas (são **case-sensitive**):

```
<Mensagem>Errado</mensagem>

<mensagem>Correcto </mensagem>

<Mensagem>Correcto </Mensagem>
```

O primeiro exemplo é incorrecto porque a etiqueta de abertura não corresponde à etiqueta de fecho (diferem na primeira letra). Isso já não acontece nos dois outros exemplos.

- Os valores dos **atributos** têm que estar envolvidos em aspas:

```
<mensagem numero="13">
```

- O XML preserva o **espaço branco** (espaços, mudanças de linha, tabulações, etc). Este espaço não é reduzido a um único caracter.

- A sintaxe para escrever **comentários** em documentos XML é a apresentada no exemplo seguinte:

```
<!-- Comentário -->
```

Se um documento respeitar todas estas regras de construção, diz-se **bem-formado** (well-formed) e pode ser utilizado por qualquer interpretador XML.
Além de bem-formado, um documento pode ser também **válido**. Um elemento diz-se válido se é bem formado e respeita uma gramática definida por DTD ou XML Schema (apresentadas nas páginas 34 e 35, respectivamente).

5.3 – Modelos de Processamento

Existem dois modelos de processamento de documentos XML:

- Simple API for XML (SAX);
- Documento Object Model (DOM).

A escolha do modelo de processamento num programa XML não depende só das suas características, mas depende também da funcionalidade disponível na biblioteca de programação utilizada.

O SAX processa documentos em série, lendo os dados XML e produzindo eventos de vários tipos: início de documento, início de elemento, caracteres, fim de elemento, fim de documento, etc. Este modelo de processamento é mais rápido e utiliza menos memória que o DOM, devendo ser utilizado, por exemplo, no lado servidor de aplicações em rede. No entanto, o SAX necessita de mais programação para definir o tratamento dos eventos lançados pelo interpretador durante a leitura dos dados XML. Outra restrição é que não é possível "voltar atrás" no documento, devido ao processamento em série.

O DOM lê o documento e constrói uma estrutura em árvore, em que cada nó é um componente do documento XML. Os nós podem ser elementos, atributos, texto e até comentários. Os dois tipos mais comuns são elementos e texto. Utilizando o DOM é

possível criar nós, remover nós, alterar o seu conteúdo e percorrer o nó hierarquicamente (pai, filhos e irmãos). O principal problema deste modelo de processamento é que necessita de bastante memória, o que pode ser impeditivo de manipular documentos de grande dimensão. Por este motivo, o DOM é sobretudo utilizado em aplicações cliente orientadas a utilizador, que tipicamente correm em máquinas com poucos problemas de memória.

5.4 – Representação de Dados

A linguagem XML pode ser utilizada para representar diferentes tipos de dados. Os dados podem ser simples ou compostos.

Os dados **simples** representam valores que podem ter vários tipos, incluindo texto, numérico e datas.
Uma vez que um documento XML é um ficheiro de texto, quer os numéricos, quer as datas são representados por caracteres. Um **problema** que se coloca com frequência, é a necessidade de definir formatos padrão para estes tipos de informação. Se ignorado, este problema pode provocar problemas complicados de resolver em fases avançadas de desenvolvimento de aplicações, nomeadamente se as formatações utilizadas forem decididas implicitamente (por exemplo, através das definições regionais da máquina). Uma das formas de evitar este problema é definir à partida um conjunto de formatos padrão, a respeitar por todas as partes envolvidas no desenvolvimento. Como base de trabalho para esta situação, sugerem-se os formatos definidos em [W3Schools03].
Os dados **compostos** são formados pela combinação de vários dados simples. São normalmente estruturas de dados (registos com vários campos) ou vectores (vários valores semelhantes indexados), em que cada elemento é um dado simples.
O XML pode representar dados **binários**, utilizando vectores de bytes (representados por caracteres).

A representação de **nulos** normalmente é feita de uma das duas maneiras seguintes:

- Omitindo o elemento nulo ou;
- Criando um atributo "`null`" no elemento e preenchendo o seu valor com "1" para indicar a ausência de valor.

Uma questão que se coloca muitas vezes durante a modelação de estruturas de dados em XML é a **escolha entre** utilizar **sub-elementos ou atributos** para representar um campo de uma estrutura XML. Os dois exemplos seguintes ilustram as duas alternativas aplicadas a uma estrutura:

Campos como atributos:

```
<mensagem de="João" para="Carla" assunto="Reunião urgente" />
```

Campos como elementos:

```
<mensagem>
      <de>João</de>
      <para>Carla</para>
      <assunto>Reunião urgente</assunto>
</mensagem>
```

Nalguns casos, as próprias características dos elementos e atributos **forçam a escolha** de uma das alternativas:

- Se os dados contêm uma sub-estrutura, então têm que ser modelados como elementos, uma vez que os atributos estão limitados a representar dados simples;
- Se o dado contém várias linhas, então também tem que ser modelado como elemento. Um atributo não tem limite de tamanho, mas torna-se pouco legível e utilizável;
- Se os dados têm cardinalidade n (múltiplas ocorrências), então também têm que ser elementos. Um único atributo apenas pode representar um dado, mas podem existir vários elementos iguais, um para cada valor.

Caso a escolha não seja forçada, podem ser utilizadas **heurísticas**:

- Os dados visíveis para o utilizador final devem ser representados como elementos;
- Os elementos devem ser utilizados para representar entidades menores contidas em entidades mais complexas. Os atributos devem ser utilizados para

representar propriedades de uma entidade, que não fazem sentido separadamente.

Relativamente aos elementos, os atributos têm a vantagem de serem mais concisos e de ocuparem menos recursos (quer na dimensão do documento, quer na árvore DOM carregada em memória).

5.5 – Espaços de Nomes

Os espaços de nomes XML (namespaces) são um mecanismo para evitar **conflitos de nomes** entre elementos. Um conflito de nomes acontece quando existem dois elementos com o mesmo nome, mas com estrutura diferente.
Um outro tipo de conflito, que não é detectado pelo interpretador, acontece quando os elementos têm a mesma estrutura mas significado diferente.

O elemento `nome` tem uma estrutura diferente nos dois exemplos seguintes:

```
<aluno>
      <nome>João Filipe</nome>
      <numero>11</numero>
</aluno>

<funcionario >
      <nome>
            <primeiro>António</primeiro>
            <ultimo>Silva</ultimo>
      </nome>
</funcionario>
```

Caso um documento XML combinasse os elementos `aluno` e `funcionario`, surgiria um conflito. Uma maneira de resolver o problema seria dar um **prefixo** diferente aos nomes dos elementos, usando `a` no primeiro exemplo e `f` no segundo, por exemplo:

```
<a:aluno>
      <a:nome>João Filipe</a:nome>
      <a:numero>11</a:numero>
</a:aluno>
```

```
<f:funcionario >
      <f:nome>
            <f:primeiro>António</f:primeiro>
            <f:ultimo>Silva</f:ultimo>
      </f:nome>
</f:funcionario>
```

O conflito deixou de existir porque ambos os documentos utilizam etiquetas diferentes para representar o elemento nome.

Em vez de se utilizarem apenas prefixos, pode utilizar-se um atributo especial **xmlns**, permitindo associar o prefixo a um **espaço de nomes**, criando o que se designa por um nome qualificado (qualified name). O atributo xmlns pode surgir na etiqueta de abertura de um elemento e tem a seguinte sintaxe:

```
xmlns:namespace-prefix="namespace"
```

Utilizando este atributo, os exemplos ficariam com o seguinte aspecto:

```
<a:aluno xmlns:a="http://www.escola.pt">
      <a:nome>João Filipe</a:nome>
      <a:numero>11</a:numero>
</a:aluno>

<f:funcionario xmlns:f="http://www.empresa.com/rechumanos">
      <f:nome>
            <f:primeiro>António</f:primeiro>
            <f:ultimo>Silva</f:ultimo>
      </f:nome>
</f:funcionario>
```

Quando o espaço de nomes é definido na etiqueta de abertura de um elemento, todos os elementos filhos com o mesmo prefixo ficam associados ao mesmo espaço de nomes.

O espaço de nomes é definido a partir de um **identificador Internet** (URI), que pode ser um URL ou um URN (descritos em 2.2 – Identificadores na página 9).
O identificador do espaço de nomes não é utilizado para obter informação a partir da localização especificada, pois o seu único objectivo é fornecer um nome único. No entanto é frequente serem utilizados URLs que referenciam uma página ou aplicação Web associada à definição do formato XML.
Os espaços de nomes são únicos, desde que os identificadores utilizados sejam reais e estejam devidamente registados, e não sejam fictícios como os apresentados nos exemplos anteriores.

O **nome completo de um elemento XML** é formado pela conjugação do seu espaço de nomes e do seu nome individual. Tal como não é possível mudar o nome individual de um elemento XML, também não é possível mudar o espaço de nomes que lhe está associado (mudar o valor do atributo xmlns também não resulta, pois é um atributo especial).

Um documento pode utilizar elementos definidos em diferentes espaços de nomes, sendo utilizado um prefixo próprio para cada um.
Um elemento filho pode ter um espaço de nomes diferente do seu pai e antecessores, bastando para isso definir um valor próprio para xmlns ou então utilizar um prefixo já associado a um espaço de nomes diferente.

O espaço de nomes pode ser definido por omissão, evitando a utilização dos prefixos em todos os elementos filhos. A sintaxe para definir o espaço de nomes por omissão é a seguinte:

```
<element xmlns="namespace">
```

Utilizando uma definição de **espaço de nomes por omissão**, o exemplo do aluno fica com o seguinte aspecto:

```
<aluno xmlns="http://www.escola.pt">
      <nome>João Filipe</nome>
      <numero>11</numero>
</aluno>
```

Os espaços de nomes são utilizados em várias aplicações XML, como XML Schema e XSL, que são apresentadas em 5.8 – Tecnologias baseadas em XML, na página 34.

5.6 – Entidades Referenciais e CDATA

Todo o texto num documento XML é **interpretado** durante a leitura do ficheiro, para permitir a detecção de inícios e fins de etiquetas.

Existem alguns **caracteres** que são **ilegais** em XML, pois a sua utilização gera documentos com interpretações ambíguas. Estes caracteres têm que ser substituídos por **entidades referenciais** (começadas pelo caracter "`&`" e terminadas pelo caracter "`;`"). Os caracteres ilegais XML são `<` e `&` que devem ser substituídos por `<` e `&`, respectivamente. Existem outros caracteres, que apesar de não serem ilegais, também devem ser substituídos para evitar dificuldades de leitura. São `>`, `'` e `"` que devem ser substituídos por `>`, `&apos;` e `&quot;`, respectivamente.

Exemplo:

```
<condicao> se salario < 1000 entao </condicao>
```

Tem que ser substituído por:

```
<condicao> se salario &lt; 1000 entao </condicao>
```

O **CDATA** permite definir secções de texto do documento XML que não são interpretadas, e que portanto não restringem a utilização de caracteres ilegais. Uma secção CDATA começa com "<![CDATA[" e termina com "]]>":

Exemplo:

```
<script>
      <![CDATA[
            if (a < b && a < 0) then
                  return 1;
            else
                  return 0;
      ]]>
</script>
```

As secções CDATA não podem conter outras secções CDATA, ou seja, não podem conter as sequências de caracteres "<![CDATA[" ou "]]>".
A sequência "]]>" também não pode ter espaços nem mudanças de linha no seu interior.

5.7 – Utilizações Possíveis

A XML é uma ferramenta que permite transmitir dados de uma forma independente de plataformas de software e hardware. Pode ser utilizada de diferentes maneiras:

- Troca de informação entre sistemas incompatíveis – a XML é um formato padrão, o que permite reduzir a complexidade e permitir a leitura dos dados por diferentes tipos de aplicações;
- Ilhas de Dados (Data Islands) em páginas HTML – permitindo a separação entre o conteúdo e apresentação;
- Criação de novas linguagens – utilização das capacidades de extensibilidade da XML para criar linguagens específicas, por exemplo, para diferentes sectores de actividade económica.

Estas características da XML tornaram-na na linguagem escolhida como base para a representação de informação na infra-estrutura de Web Services.

A principal **vantagem** da XML é que os dados passam a poder ser processados por muitas ferramentas em sistemas de informação, aumentando o valor potencial da informação guardada neste formato.
Uma outra característica positiva é o facto dos documentos XML serem baseados em texto, podendo ser compreendidos e editados por programas, mas também por pessoas. Além disto, o documento pode tornar-se praticamente auto-descritivo, se as etiquetas forem criteriosamente escolhidas.

A principal **desvantagem** da utilização de XML é o facto de produzir ficheiros de dados com grande percentagem de informação de controlo. Os exemplos seguintes ilustram o problema:

Exemplo em XML:

```
<mensagem>
      <de>João</de>
      <para>Carla</para>
      <assunto>Reunião urgente</assunto>
</mensagem>
```

Exemplo equivalente em formato específico:

```
de=João;para=Carla;assunto=Reunião urgente
```

O documento em XML tem 86 caracteres no total, 62 de controlo (72%) e 24 de dados (**28%**). O documento no formato específico tem 42 caracteres no total, 18 de controlo (43%), e 24 de dados (**57%**).
Este cálculo permite ter uma ideia aproximada do "preço" a pagar pelas vantagens da utilização de XML. Esta questão pode perder importância relativa, caso a rede utilizada tenha bastante largura de banda disponível. No entanto, é uma característica da XML que não deve ser ignorada durante a concepção de sistemas.

5.8 – Tecnologias baseadas em XML

Actualmente já existem inúmeras linguagens definidas com base na XML. Algumas delas acrescentam funcionalidades muito úteis ao ambiente de programação em XML, alargando o seu âmbito.

5.8.1 – Document Type Definition

O objectivo de uma Document Type Definition (DTD) é construir uma gramática que define a estrutura e elementos válidos de um tipo de documento XML. Os documentos que respeitam essa DTD dizem-se válidos.
A DTD é efectuada numa linguagem própria com sintaxe própria, diferente da XML.
O documento XML pode conter uma DTD embebida ou usar uma referência externa.

Exemplo de documento XML que referencia DTD:

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "mensagem.dtd">
<?xml version="1.0" encoding="ISO-8859-1"?>
<mensagem>
      <de>João</de>
      <para>Carla</para>
      <assunto>Reunião urgente</assunto>
</mensagem>
```

Exemplo de DTD (mensagem.dtd):

```
<!ELEMENT mensagem (de,para,assunto)>
<!ELEMENT de (#PCDATA)>
<!ELEMENT para (#PCDATA)>
<!ELEMENT assunto (#PCDATA)>
```

Quando o documento XML é lido por um interpretador, se a opção de validação estiver activa, a DTD é utilizada para validar o documento. Caso o documento não respeite a DTD, gera-se um erro.

5.8.2 – XML Schema

A linguagem XML Schema, também conhecida por XML Schema Definition (XSD), é uma alternativa à DTD, mas as suas regras sintácticas são baseadas nas da XML. Por este motivo, a XML Schema é uma aplicação da XML.

Um XML Schema define:

- Elementos que podem aparecer num documento;
- Atributos que podem aparecer num documento;
- Elementos que são elementos filhos;
- A ordem dos elementos filhos;
- O número de elementos filhos;
- Se um elemento é vazio ou pode incluir texto;
- Tipos de dados para elementos e atributos;
- Valores por omissão ou fixos para elementos e atributos.

A utilização de DTD tem tendência a ser substituída por XML Schema, porque esta última apresenta várias vantagens:

- É extensível;
- É mais expressiva que a DTD;
- É escrita em XML;
- Suporta tipos de dados;
- Suporta espaços de nomes.

Exemplo de documento XML que referencia um Schema:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<shiporder orderid="889923"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="shiporder.xsd">
 <orderperson>John Smith</orderperson>
 <shipto>
  <name>Ola Nordmann</name>
  <address>Langgt 23</address>
  <city>4000 Stavanger</city>
```

```
  <country>Norway</country>
 </shipto>
 <item>
  <title>Empire Burlesque</title>
  <note>Special Edition</note>
  <quantity>1</quantity>
  <price>10.90</price>
 </item>
 <item>
  <title>Hide your heart</title>
  <quantity>1</quantity>
  <price>9.90</price>
 </item>
</shiporder>
```

Exemplo de um XML Schema:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="shiporder">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="orderperson" type="xs:string"/>
   <xs:element name="shipto">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="address" type="xs:string"/>
      <xs:element name="city" type="xs:string"/>
      <xs:element name="country" type="xs:string"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="item" maxOccurs="unbounded">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="note" type="xs:string" minOccurs="0"/>
```

```
      <xs:element name="quantity" type="xs:positiveInteger"/>
      <xs:element name="price" type="xs:decimal"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
  <xs:attribute name="orderid" type="xs:string"
use="required"/>
 </xs:complexType>
</xs:element></xs:schema>
```

5.8.3 – eXtensible Stylesheet Language

A eXtensible Stylesheet Language (XSL) é uma linguagem para definir folhas de estilos, que especificam como um documento deve ser apresentado ao utilizador. É constituída por três componentes:

- XSLT – linguagem para transformar documentos XML;
- XPath – linguagem para referenciar partes de um documento XML;
- Objectos de Formatação XSL – vocabulário para formatar documentos XML.

Utilizando estas componentes, é possível utilizar a XSL para:

- Transformar documentos XML noutros formatos XML (incluindo XHTML, uma versão do HTML com sintaxe rigorosa baseada em XML);
- Filtrar e ordenar dados em XML;
- Formatar dados XML;
- Apresentar o mesmo documento de formas diferentes, dependendo do dispositivo de destino (ecrã, impressão, telefone móvel, etc).

Exemplo XSL:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"><xsl:template
match="/">
  <html>
  <body>
```

```
    <h2>My CD Collection</h2>
    <table border="1">
    <tr bgcolor="#9acd32">
      <th align="left">Title</th>
      <th align="left">Artist</th>
    </tr>
    <xsl:for-each select="catalog/cd">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
    </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

5.8.4 – Outras

A XML Schema e a XSL são bastante genéricas, sendo úteis em diversas situações. Existem muitas outras linguagens baseadas em XML que têm objectivos mais específicos. Uma tendência crescente actualmente, é utilizar XML para ficheiros de configuração de software. O exemplo seguinte apresenta um ficheiro de configuração de uma aplicação Web, no ambiente do Web Server TOMCAT.

Exemplo de ficheiro de aplicação Web TOMCAT (web.xml):

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
2.3//EN"
    "http://java.sun.com/j2ee/dtds/web-app_2_3.dtd">

<web-app>
   <display-name>SupplierApplication</display-name>
```

```
   <description>Coffee Supplier Application</description>
   <servlet>
      <servlet-name>JAXMPriceListEndpoint</servlet-name>
      <servlet-class>com.sun.cb.PriceListServlet</servlet-
class>
       <load-on-startup>1</load-on-startup>
   </servlet>
   <servlet>
      <servlet-name>JAXMOrderConfirmationEndpoint</servlet-
name>
      <servlet-class>com.sun.cb.ConfirmationServlet</servlet-
class>
       <load-on-startup>2</load-on-startup>
   </servlet>
   <servlet-mapping>
      <servlet-name>JAXMPriceListEndpoint</servlet-name>
      <url-pattern>/getPriceList</url-pattern>
   </servlet-mapping>
   <servlet-mapping>
      <servlet-name>JAXMOrderConfirmationEndpoint</servlet-
name>
      <url-pattern>/orderCoffee</url-pattern>
   </servlet-mapping>
   <session-config>
      <session-timeout>60</session-timeout>
   </session-config>
</web-app>
```

Fim do Capítulo. Aspectos a reter:

A eXtensible Markup Language (XML) é uma linguagem extensível para descrever informação, que é baseada em etiquetas.

A XML descreve os dados (o que são) enquanto que a HTML apresenta os dados (que aspecto têm).

A sintaxe XML é simples mas é rigorosamente interpretada.
Um documento XML tem que ter um elemento único na raiz.
Um documento XML é bem-formado (well-formed) se está sintacticamente correcto.
Um documento XML é válido se é bem-formado e respeita uma gramática definida por DTD ou XML Schema.

Existem dois modelos de processamento XML:

- SAX – processamento em série, baseado no tratamento de eventos;
- DOM – manipulação de árvore construída em memória.

A representação de dados em XML depende se os dados são simples ou são compostos.
Os dados simples são valores escalares, que podem ter vários tipos, incluindo texto, numérico e datas.
Os dados compostos são estruturas ou vectores de dados simples.
A opção entre representação de estruturas de dados utilizando sub-elementos ou atributos depende de vários factores, que podem ser forçados pelos próprios dados ou condicionados por heurísticas.

Os espaços de nomes XML permitem evitar conflitos de nomes, permitindo combinar elementos de diversas proveniências num único documento XML.

A XML pode ser utilizada genericamente para:

- Trocar informação entre sistemas incompatíveis;
- Criar ilhas de dados (data islands) em páginas HTML;
- Criar novas linguagens.

As principais tecnologias baseadas em XML são:

- XML Schema – define gramáticas que definem a estrutura, elementos válidos e tipos de dados utilizados num formato XML.
- eXtensible Stylesheet Language (XSL) – define várias alternativas de processamento e transformação de XML para permitir diferentes apresentações ao utilizador.

6 – Síntese Conclusiva

A World Wide Web é um sistema distribuído baseado em documentos, que na actualidade é a aplicação mais utilizada da Internet.
A evolução das aplicações Web, bem como as novas formas de acesso através de novos dispositivos, fizeram surgir a necessidade de construir uma infra-estrutura de **Web Services**, com vista a criar um mercado electrónico de serviços na Web. Esta infra-estrutura foi e continua a ser construída tendo como ponto de partida as tecnologias base da Web, que foram analisadas nesta unidade.

A Web tem uma **arquitectura** cliente-servidor e está baseada num modelo de documentos com hyperlinks, que permitem formar uma "teia" de ligações entre documentos. Estas ligações são definidas através de **identificadores** (URI), sendo os mais utilizados os URLs, que localizam os documentos na Internet.

O **HTTP** é o protocolo aplicacional de comunicação entre clientes e servidores na Web, que utiliza normalmente a pilha de protocolos TCP/IP para garantir um canal de comunicação fiável. O HTTP tem vários métodos, sendo os mais utilizados os GET e POST. Estes dois métodos permitem enviar informação nos dois sentidos, entre cliente e servidor Web.

A linguagem **HTML** é utilizada para construir grande parte dos documentos Web. As páginas HTML utilizam etiquetas para definir a sua formatação e estrutura. O principal problema da HTML é não permitir a separação entre representação e apresentação da informação.

A linguagem **XML** tem como objectivo a descrição de informação, através de etiquetas que são extensíveis. Os documentos XML são usados para representar informação, que pode ser partilhada entre diferentes tipos de sistema. Devido a esta característica, a XML é apropriada para ser utilizada na infra-estrutura de Web Services.

A XML serve como base para criar novas linguagens, como a XML Schema e a XSL, que têm aplicação genérica.

A **XML Schema** define gramáticas que definem a estrutura, elementos válidos e tipos de dados utilizados num formato XML.

A **XSL** define várias alternativas de processamento e transformação de XML, permitindo criar diferentes apresentações da mesma informação para o utilizador.

Fim do Capítulo e da Unidade

Referências

As referências utilizadas incluem livros, artigos e páginas Web. Durante a leitura do texto em formato electrónico, cada referência na forma [...] pode ser utilizada para acesso rápido a esta secção.

[Ferreira00] – Paulo Ferreira, *Suporte de Aplicações Distribuídas – Draft*, 2000

[IETF91] – Internet Engineering Task Force, *RFC 1180 - A TCP/IP Tutorial*, Janeiro de 1991
http://www.ietf.org/rfc/rfc1180.txt

[IETF94] – Internet Engineering Task Force, *RFC 1630 - Universal Resource Identifiers in WWW*, Junho de 1994
http://www.ietf.org/rfc/rfc1630.txt

[IETF94b] – Internet Engineering Task Force, *RFC 1738 - Uniform Resource Locators (URL)*, Dezembro de 1994
http://www.ietf.org/rfc/rfc1738.txt

[IETF95] – Internet Engineering Task Force, *RFC 1808 - Relative Uniform Resource Locators*, Junho de 1995
http://www.ietf.org/rfc/rfc1808.txt

[IETF99] – Internet Engineering Task Force, *RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1*, Junho de 1999
http://www.ietf.org/rfc/rfc2616.txt

[MarquesGuedes98] – José Alves Marques e Paulo Guedes, *Tecnologia de Sistemas Distribuídos – 1ª edição*, FCA, Maio de 1998

[TanenbaumSteen02] – Andrew S. Tanenbaum, Maarten van Steen, *Distributed Systems – Principles and Paradigms*, Prentice Hall, 2002

[Unicode03] – Unicode Consortium, *Unicode Home Page*, Fevereiro de 2003
http://www.unicode.org/

[W3C00] - World Wide Web Consortium, *Extensible Markup Language (XML) 1.0 (Second Edition)*, 6 October 2000
http://www.w3.org/TR/REC-xml

[W3C01] – World Wide Web Consortium, *XML Schema Part 2: Datatypes*, 2 de Maio de 2001
http://www.w3.org/TR/xmlschema-2/

[W3Schools03] – World Wide Web Consortium, *W3Schools Online Web Tutorials*
http://www.w3schools.com

Abreviaturas

Abreviatura	Significado
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DTD	Document Type Definition
FTP	File Transfer Protocol
HTML	HyperText Markup Language
HTTP	Hyper Text Transfer Protocol
HTTPS	HTTP sobre SSL
IP	Internet Protocol
J2EE	Java 2 Enterprise Edition
LDAP	Lightweight Directory Access Protocol
SMTP	Simple Mail Transfer Protocol
SSL	Secure Sockets Layer
TCP	Transport Control Protocol
TPS	Transaction Processing System
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
XML	eXtensible Markup Language
Xpath	Xml Path
XSL	eXtensible Stylesheet Language
XSLT	eXtensible Stylesheet Language Transformations

Anexo A – Organizações Reguladoras da Internet e da World Wide Web

A **Internet Engineering Task Force (IETF)** é uma comunidade internacional de grande dimensão que junta engenheiros de redes, operadores, fabricantes e investigadores preocupados com a operação e evolução da arquitectura da Internet. Está aberta a quaisquer indivíduos interessados.
O trabalho técnico é feito em grupos de trabalho, organizados por assunto de investigação. Os principais meios de comunicação entre os grupos são listas de correio electrónico.

Os documentos Requests for Comments (RFC) são notas técnicas e organizacionais sobre a Internet que se iniciaram em 1969. Estas notas discutem muitos aspectos de redes de computadores, incluindo protocolos, procedimentos, programas e conceitos, bem como notas de reuniões e opiniões.
Tecnologias: HTTP, URI (URL e URN), TCP, IP entre outras
Endereço: http://www.ietf.org

O **World Wide Web Consortium (W3C)** foi criado em Outubro de 1994 para potenciar a World Wide Web, desenvolvendo protocolos comuns que promovam a sua evolução e garantam a sua interoperabilidade. O W3C tem aproximadamente 450 organizações membro e já fez importantes contribuições para o crescimento da Web.
Tecnologias: HTML, XML, XML Schema, XSL entre outras
Endereço: http://www.w3.org/