1.2. Lenguajes de marcas en entornos web

Índice

Objetivos

En esta unidad perseguimos los siguientes objetivos:

- Conocer la **estructura y los elementos de los documentos de los lenguajes de marcas.**
- Conocer la **arquitectura cliente/servidor.**

Introducción

Metadatos

Los metadatos se usan **para implementar una forma de etiquetado, catalogación, descripción y clasificación de los recursos existentes.** Suelen ser datos sobre datos.

Berners y Lee, en la definición que dan sobre este concepto, afirman que "los metadatos son información inteligible para el ordenador sobre recursos web u otras cosas".

En algunos casos pueden estar asociados a un esquema de descripción. También pueden aportar información sobre elementos de datos o atributos, información sobre la estructura de los datos, información sobre un aspecto concreto, etc.

En los lenguajes de marcado los **metadatos sirven para complementar la información** contenida en el documento de marcas o para **añadir información** que no va a ser visualizada, pero que le da contexto al documento, e incluso indica cómo se debe interpretar el documento.

En los documentos HTML, por ejemplo, se incluyen metadatos en la etiqueta *<head>*, tales como el título de la página, especificación de caracteres a usar, palabras clave, estilos, etc.

Instrucciones de proceso

Una instrucción de procesamiento o **instrucciones de proceso (PI)** le dan al intérprete, que es el encargado de presentar el documento, instrucciones de **cómo deben ser procesadas las marcas** que se incluyen en él.

Un ejemplo de instrucción de procesamiento sería en el DOM (Document Object Model) y se denominaría Node. **PROCESSING_INSTRUCTION_NODE**, y pueden ser utilizados en Xpath y XQuery con el comando "*processing-instruction()*".

En SGML una "processing instruction" se encierra entre '<? y '>'

```
<?xml-stylesheet type="text/xsl" href="style.xsl"?>

<?xml-stylesheet type="text/css" href="style.css"?>
```

Codificación de caracteres

La codificación de caracteres consiste en la conversión de un carácter desde el lenguaje natural a un símbolo que pertenece a otro sistema de representación. Esta codificación se realiza mediante el uso de unas normas y reglas.

En este caso, la codificación serviría para **codificar caracteres** a través del lenguaje de marcas y su contenido de lenguaje escrito en los distintos alfabetos existentes.
Una de las primeras codificaciones fue el código **ASCII**, que con 8 caracteres codificó 128 caracteres del alfabeto latino.

Al confeccionar documentos HTML o XML se indica en el encabezado el juego de caracteres que queremos que use el navegador para decodificar la página o documento web.
La codificación de caracteres más moderna se realiza en **Unicode (UTF-8, UTF-16** o **UTF-32)**,que es un estándar definido por la ISO10646.

Por ejemplo, si queremos usar el UTF-8, en una página HTML podríamos incluir en la cabecera de nuestro documento la siguiente directiva:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
```

En XHTML.

```
<meta charset="UTF-8"/>
```

En HTML5.

Caracteres especiales

Cuando se necesitó codificar caracteres especiales, o los caracteres de lenguajes de todo el mundo, y representarlos en HTML, se crearon unas codificaciones que hacían que el navegador o el intérprete del documento los reconociera como caracteres especiales.

Tabla ASCII
Aquí podemos ver una tabla de códigos ASCII para caracteres y caracteres especiales.
https://ascii.cl/es/codigos-html.htm

También hubo que codificar de manera especial los caracteres que formaban parte del lenguaje de marcas, como “<” y “>”. Para ello se usaron **caracteres de escape**, que sirven para que el intérprete del documento conozca cuándo empieza y acaba la directiva o cuándo comienza el contenido.

Así, se crearon referencias específicas, de manera que esos caracteres pudiesen formar parte del contenido.

CÓDIGO	REPRESENTACIÓN
&lt;	<
&gt;	>
&amp;	&
&quot;	"

Referencias específicas para caracteres especiales.

Etiquetas o marcas

Una *tag* o etiqueta es la marca que usa el lenguaje para delimitar un objeto específico.

Las etiquetas o marcas están definidas por caracteres de escape que indican al software que lee el documento cuándo comienza y acaba el *tag* o etiqueta. **En lenguajes de marcado como HTML existen etiquetas o marcas que usan el carácter "<" , el "\" o el carácter ">".**

En HTML y en XML casi todas las marcas indican el comienzo de un elemento y finalizan con otra marca parecida, que marca el fin del contenido donde aplicamos esa marca.

Elementos

Un elemento es un componente individual que comienza con una marca de inicio y acaba con una marca de final.

```
<etiqueta atributo1="valor1" atributo2="valor2">contenido</etiqueta>
```

Sintaxis de una etiqueta.

- **Elemento *void:*** solo contiene una etiqueta de entrada, lleva atributos y puede no contener ningún elemento hijo como texto.
- **Elementos de texto sin procesar (*raw text elements*):** se construyen con una etiqueta, atributos, algún contenido de texto, sin elementos, y con una etiqueta de fin.
- **Elementos normales:** se construyen con una etiqueta, atributos, algún contenido de texto y una etiqueta de fin.

Atributos

Los atributos sirven para **dotar al elemento de propiedades** que tienen que ver con la marca en la que están contenidos.

La forma de representar dichas propiedades es a través de pares nombre-valor, separados por un signo de igual **"="**. Normalmente se escriben después de la etiqueta de comienzo de un elemento.

El valor que va asociado al atributo suele ir encerrado entre comillas dobles o simples.

```
<etiqueta atributo1="valor1" atributo2="valor2">contenido</etiqueta>
```

Existe una **clasificación para los atributos:**

Atributos básicos

Se pueden utilizar prácticamente en todas las etiquetas HTML.

Atributos para internacionalización

Los utilizan las páginas que muestran sus contenidos en varios idiomas.

Atributos de eventos

Solo se utilizan en las páginas web dinámicas creadas con JavaScript.

Atributos de foco

Relacionados principalmente con la accesibilidad de los sitios web.

Comentarios

Los comentarios sirven para **incluir información extra que no es interpretada por el navegador** y que puede aparecer en cualquier parte de un documento. Para definirlos en XHTML se usa la siguiente etiqueta:

```
<!-- Esto es un comentario -->
```

Por razones de compatibilidad con algunos navegadores anteriores a 1995 **los contenidos de estilo y *script* se mantienen entre delimitadores de comentarios**.

No se permite ningún espacio en blanco entre el delimitador de apertura de declaración de etiqueta ("<!") y el delimitador de apertura de comentario ("--"), pero sí se permite entre el delimitador de cierre de comentario ("--") y el delimitador de cierre de declaración de etiqueta (">").

Un error común es incluir una cadena de guiones ("---") dentro de un comentario.

Arquitectura cliente/servidor

Aplicaciones web

Las **aplicaciones web tienen su naturaleza en Internet.** Esto estructura su funcionamiento siguiendo un modelo **cliente/servidor**, en donde el servidor es multi-plataforma y el cliente, aunque soportado por varias marcas de navegador, es uni-plataforma.

Esto quiere decir que el estándar de la parte cliente, implantado a través de los navegadores, se unifica a través de HTML5, CSS3 y JavaScript, mientras que en el servidor viven distintas plataformas no estandarizadas, como Java, PHP, Python, CGI, .NET, etc.

Siempre que vayamos a crear un software para desplegar en Internet deberá tener varios componentes que trabajarán en conjunto para satisfacer las necesidades y los objetivos de dicha aplicación. Tendremos en cuenta:

La interfaz del cliente

Soportará la parte visual e interactiva; en ella el usuario trabaja con contenidos, enlaces, formularios etc., y estos producirán información.

El módulo de servidor

Recibirá esa información y, según la lógica de la aplicación, lanzará procesos en bases de datos u otros módulos y dará una respuesta adecuada al cliente.

En esta arquitectura el cliente tiene la presentación visual y cierta lógica que se ejecuta en la parte de interfaz de usuario y emite peticiones a la parte servidor.

Características

En cuanto a la construcción de estas aplicaciones y su mantenimiento podemos indicar:

Son fáciles de mantener

Como cada aplicación tiene sus propios módulos y se programan abstrayéndose del resto de capas, el mantenimiento de cada una de las capas se realiza de manera independiente, por lo

que se pueden realizar mejoras y correcciones sin que afecte al resto de módulos de las otras capas.

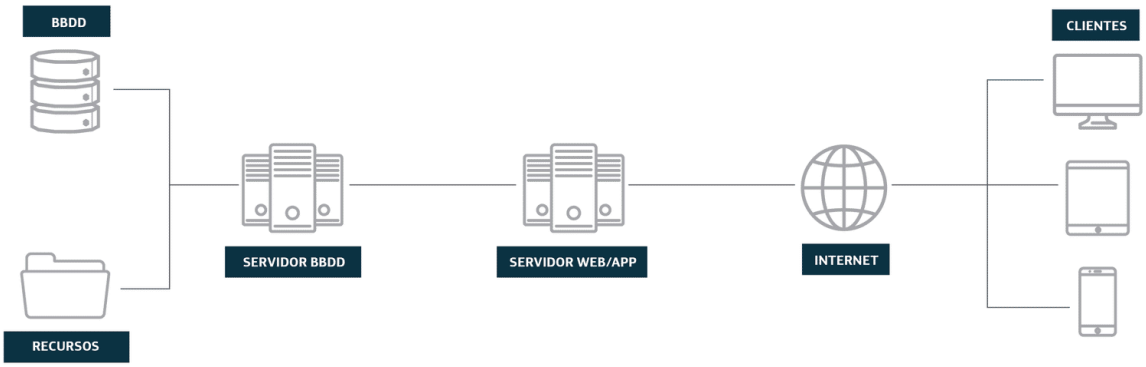
Son fáciles de escalar
Por el mismo motivo, el escalamiento de la aplicación hacia afuera es razonablemente sencillo.

Son reusables
Los módulos se pueden construir para que puedan ser usados por otras aplicaciones o para la generación de nuevos módulos.

Disponibilidad
Al tener una estructura modular, la sustitución o puesta en marcha de partes de la aplicación es muy rápida, lo que deriva en su disponibilidad.

Arquitectura hardware cliente/servidor

Como hemos visto, un software para Internet debe tener varios componentes que trabajan en conjunto.



Base de datos
Tablas con los datos de la aplicación.

Gestor BBDD
El gestor de la base de datos es la aplicación encargada de procesar y responder las peticiones de información. Los más utilizados son los basados en SQL u Oracle.

HTTP
Servicio del servidor encargado de responder las peticiones del cliente. Los más extendidos son el servidor Apache para sistemas Linux, o IIS (Internet Information Service) para servidores Windows.

Este servicio devuelve los archivos procesados en el servidor al cliente, como los códigos HTML, CSS, etc.

Internet
La red.

Archivos
Archivos de imágenes, PDF, o librerías que la aplicación puede necesitar externamente.

Clientes
Los clientes envían las peticiones mediante URL e interpretan los archivos devueltos.

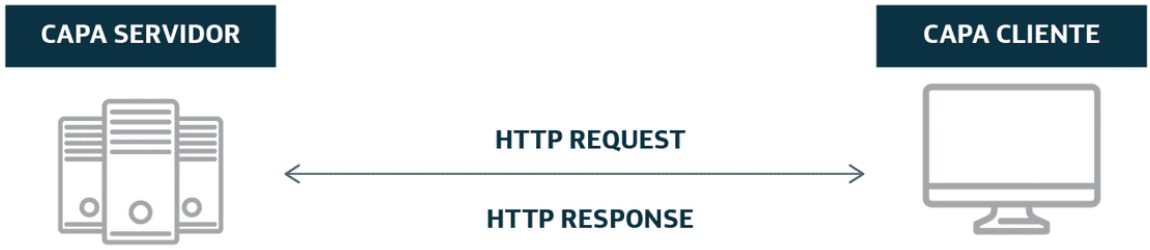
Normalmente son archivos HTML junto a otros como las hojas de estilo CSS, códigos JavaScript, imágenes, etc.

HTTP

Cuando el cliente hace una **petición** al servidor, este le devuelve el mensaje con la información solicitada. Para esta comunicación, en la mayoría de los casos se usa el **protocolo HTTP** a través de *request-response.*

Veamos el proceso.

1. **Un cliente HTTP inicia una solicitud** mediante el establecimiento de una conexión de Protocolo de Control de Transmisión (TCP) a un puerto determinado en un servidor (normalmente el puerto 80).
2. **Hay un servidor HTTP escuchando en ese puerto** en espera de un mensaje de petición de un cliente.
3. Al recibir la petición **el servidor envía una línea de estado, como "HTTP/1.1 200 OK"**, y un mensaje de su cuenta.



Proceso *request-response*.

El contenido que se envía en un mensaje HTTP es:

- Una línea de petición.
- Encabezado *request*.
- Una línea vacía.
- Un cuerpo del mensaje opcional.

La línea de solicitud y los encabezados de todos deben terminar con *<CR> <LF>*, es decir, un carácter de retorno de carro seguido de un carácter de salto de línea.
La línea vacía debe consistir solo en *<CR> <LF>* y ningún otro espacio en blanco.

```
GET /index.html HTTP/1.1  Host: www.example.com
```

Ejemplo de petición.

Contenido de un mensaje *response*:

- Una línea de estado.
- Encabezado *request*.
- Una línea vacía.
- Un cuerpo del mensaje opcional.

La línea de estado y los encabezados de todos deben terminar con *<CR> <LF>*, es decir, un carácter de retorno de carro seguido de un carácter de salto de línea. La línea vacía debe consistir solo de *<CR> <LF>* y ningún otro espacio en blanco.

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
ETag: "3f80f-1b6-3e1cb03b"
Content-Type: text/html; charset=UTF-8
Content-Length: 131
Connection: close
<html>
<head> <title>An Example Page</title> </head>
<body> Hello World, this is a very simple HTML document. </body>
</html>
```

Ejemplo de un contenido mensaje.

Tabla de métodos *request*

HTTP tiene un conjunto de métodos para realizar las peticiones e indicar la acción que se desea realizar para un recurso determinado. Estos métodos implementan una semántica diferente, pero características similares: *Get*, *Head*, *Post*, *Put*, *Delete*, *Trace*, *Options*, *Connect*.

Veamos cada uno de ellos:

GET

Solicita una **representación del recurso especificado**.

Por seguridad no debería ser usado por aplicaciones que causen efectos, ya que transmite información a través de la URI agregando parámetros a la URL.

HEAD

Solicita una **respuesta idéntica a la que correspondería a una petición GET**, pero sin el cuerpo de la respuesta.

Esto es útil para la recuperación de meta-información escrita en los encabezados de respuesta, sin tener que transportar todo el contenido.

POST

Solicita que se sometan los datos a ser procesados para el recurso identificado**. Los datos se incluirán en el cuerpo de la petición**.

Esto puede derivar en la creación de un nuevo recurso, en las actualizaciones de los recursos existentes o en ambas cosas.

PUT

Sube, carga o realiza un *upload* de un recurso especificado (archivo). Es el camino más eficiente para subir archivos a un servidor, porque en POST utiliza un mensaje multiparte y el mensaje es decodificado por el servidor.

DELETE

Elimina el recurso especificado.

TRACE

Este método solicita al servidor que **envíe de vuelta un mensaje de respuesta**, en la sección del cuerpo de entidad, con todos los datos que reciba del mensaje de solicitud.

Se utiliza con fines de comprobación y diagnóstico.

OPTIONS

Devuelve los **métodos HTTP que el servidor soporta para una URL específica.**

Puede usarse para comprobar la funcionalidad de un servidor web mediante petición, en lugar de un recurso específico.

CONNECT

Se utiliza para saber si se **tiene acceso a un *host***, y la petición no necesariamente llega al servidor.

Este método se utiliza principalmente para saber si un *proxy* nos da acceso a un *host* bajo condiciones especiales, como por ejemplo "corrientes" de datos bidireccionales encriptadas (como lo requiere SSL).

XML en el intercambio de información

El estándar **XForms** se basa en dividir el formulario en tres partes:

1. El modelo de XForms.
2. Los datos.
3. La interfaz de usuario.

Por lo tanto, se separan claramente los datos de la visualización, lo que permite una gran flexibilidad de opciones en la presentación.

En los archivos **HTML o XHTML se definirán, en la cabecera, el tipo de XForms** a utilizar y en el cuerpo, los controles que se utilicen.

El fichero aparte XML describe el formulario que permite modificar un gran número de opciones de los controles implementados en el HTML. Cuando el formulario se debe enviar, el procesador de XForms es el encargado de recoger los datos y enviarlos en forma de archivos XML. Esta separación hace que los formularios sean reutilizables, ya que no están ligados al documento HTML. Además, al definirse en XML, los formularios no solo son aplicables a HTML o XHTML, sino a cualquier lenguaje de marcado.

Ficheros de configuración basados en XML

En muchas aplicaciones, ya sea web o de escritorio, se utilizan ficheros XML para configurar los elementos personalizables de las aplicaciones.

Un ejemplo de esto es el fichero de configuración de un servidor web mediante la tecnología J2EE. En este archivo de configuración (*web.xml*) se describe el contenido de despliegue de la aplicación web, las páginas web, *sevlets*, páginas *jsp*, etc.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
    <servlet>
      <servlet-name>HolaMundoServlet</servlet-name>
      <servlet-class>org.prueba.servlet.HolaMundoServlet</servlet-class>
    </servlet>
```

Otro ejemplo es el *framework* .NET, en el que se puede configurar el comportamiento de la aplicación así como el motor de ejecución en sí con archivos XML (*web.config*).

Aunque es posible editar estos archivos para configurar la aplicación, contienen librerías especializadas para extraer y modificar los datos de configuración de los archivos XML.

Despedida

Resumen

Has terminado la lección. Repasemos los puntos más importantes que hemos tratado.

- Existen diferentes elementos que conforman un documento realizado con lenguajes de marcas. Son las **etiquetas**, los **elementos**, los **atributos** y los **comentarios.**
- Cada uno de ellos tiene una funcionalidad distinta y su propia sintaxis.
- Las **aplicaciones web y el protocolo de comunicación que utilizan** para la transmisión de datos (HTTP) son herramientas fundamentales de los lenguajes de marcas en los entornos web.