Sistemas de Información

Resumen

Dorian Wozniak

Índice

Capítulo 1

INTRODUCCIÓN

Capítulo 2

EVOLUCIÓN DE LA WEB

Capítulo 3

WEB ESTÁTICA

3.1 HTML

Es un **lenguaje de marcado** basado en **<etiquetas>** que encierran secciones del código. Las etiquetas pueden contener **atributos**, no contener contenido, y no son sensibles al tipo de letra, aunque es recomendado usar solo minusculas.

Hay varios estándares de HTML, siendo el mas reciente **HTML5** (2012)

3.1.1 Estructura básica

```
<!DOCTYPE html>
<!-- Comentario -->
<html lang="es">
    <head>
        <title> Título </title>
        <meta charset="UTF-8"/>
    </head>
    <body>
        <h1> Cabecera </h1>
        <p> Párrafo 1</p>

        <p>
            <strong> Resaltado </strong>
            <br/> <!-- Salto de línea -->
            <em> Énfasis </em>
        </p>
    </body>
</html>
```

3.1.2 Enlaces y multimedia

```
<!-- Metainformación sobre elementos -->
<meta name="elem" content="descripcion"/>
<!-- Enlaces (elemento al que se enlaza se denomina anchor) -->
<a href="https://pagina.web"> Un enlace </a>
<!-- Imagenes -->
<img src="ruta" alt="texto alternativo"/>
<!-- Audio -->
<audio src="ruta" preload="none||auto||metadata" controls autoplay loop muted> </audio>
<audio preload autoplay controls loop>
    <source src="ruta.mp3" type="audio/mpeg"/>
    <source src="ruta.ogg" type="audio/ogg"/>
    Si ves este mensaje, tu navegador no soporta audio
</audio>
<!-- Video -->
```

```
<video controls>
    <source src="ruta.mp4" type="video/mp4" codecs="avc1.42E01E,mp4a.40.2"/>
    <source src="ruta.ogg" type="video/ogg"/>
    Si ves este mensaje, tu navegador no soporta video
</video>
```

3.1.3 Tablas

```
<table>
    <tr> <th>Cabecera 1</th>    <th>Cabecera 2</th> </tr>
    <tr> <td>F1C1</td>          <td>F1C2</td> </tr>
    <tr> <td>F2C1</td>          <td>F2C2</td> </tr>
</table>
```

3.1.4 Listados

```
<!-- Listas no ordenadas-->
<ul>
    <li>Item 1</li>
    <li>Item 2</li>
</ul>
<!-- Listas ordenadas-->
<ol>
    <li>Item 1</li>
    <li>Item 2</li>
</ol>
<!-- Listas de definiciones-->
<dl>
    <dt>Item 1</dt> <dd>Definición 1</dd>
    <dt>Item 2</dt> <dd>Definición 2</dd>
</dl>
```

3.1.5 Formularios

```
<form name="ejemplo" action="procesar.do" method="get">
    <!-- Introducción de texto -->
    <label for="campo1"> Nombre </label>
    <input type="text" name="nombre" id="campo1"/>

    <label for="campo2"> Contraseña </label>
    <input type="password" name="contraseña"/>

    <!-- Selección única -->
    <input type="radio" name="sex" value="H"/>
    <input type="radio" name="sex" value="M"/>

    <!-- Selección múltiple -->
    <input type="checkbox" name="vehiculo" value="coche"/>
    <input type="checkbox" name="vehiculo" value="moto"/>
    <input type="checkbox" name="vehiculo" value="bici"/>

    <!-- Botones-->
    <input type="submit"/>                              <!-- Envío -->
    <input type="reset"/>                               <!-- Reinicio-->
    <input type="button" onclick="función"/>            <!-- Botón genérico-->
    <input type="hidden" name="variable" value="valor"/> <!-- Campos ocultos -->

    <!-- Listas desplegables -->
    <select name="Marca_Vehiculo">
        <option value="volvo"> Volvo </option>
        <option value="saab"> Saab </option>
```

```
        <option value="fiat"> Fiat </option>
    </select>

    <!-- Introducción de texto mas grande -->
    <textarea name="resupesta" rows="4" columns="50">
        Escriba aquí su respuesta
    </textarea>
</form>
```

3.1.6 Secciones

```
<style>
    .myDiv {
        border: 5px outset red;
        background-color: lightblue;
        text-align: center;
    }
</style>

<div class="myDiv">
    <!-- Contenido -->
</div>
```

Las secciones por defecto no se visualizan, pero se pueden aplicar estilos sobre estos. Son importantes al poder desplazarse respecto a otras secciones.

3.1.7 Estilos y scripts

El código HTML genera un Modelo de Objetos del Documento (DOM) al ser interpretado por el navegador, el cual muestra y aplica estilos y *scripts* sobre la página generada

```
<!-- Definición de estilos -->
<style>
    h1 {
        text-align:center;
        font-size:12;
    }
</style>
<h1 style="text-align:center" > Ejemplo </h1>                    <!-- Inline -->
<!-- Definición de scripts -->
<script> </script>
<!-- Enlace a recurso externo -->
<link rel="stylesheet" href="style.css" type="text/css"/>
```

3.2 CSS

Es un lenguaje de definición de **hojas de estilo**. Cada estilo se compone de un **selector** y una serie de **propiedades**.

```
/* Selector básico (sobre etiqueta) */
h1 {
    text-align:center;
    font-size:12;
}
/* Selector de clase */
.efecto1 {
    text-align:left;
    font-size:24;
}
/* Selector de identificador (para un único elemento de la página) */
#id1 {
    text-align:center;
```

```
    font-size:12;
}
```

Capítulo 4

WEB DINÁMICA

4.1 Patrón VO/DAO (Value Objetct/Data Access Object)

Es un patrón que utiliza clases **VO** (*Value Object*) para representar entidades de información equivalentes a las de una base de datos persistente; y una interfaz **DAO** (*Data Access Object*) entre la aplicación y la base de datos para abstraer detalles de la segunda.

Los elementos constructivos de un VO son:

- Propiedades, que representan los atributos de la BBDD
- Metodos set/get para acceder y modificar estas propiedades
- Un constructor que permite generar un nuevo objeto VO a partir de unas propiedades dadas

La DAO la forman clases que implementan el acceso a datos entre la BBDD y la aplicación. Las operaciones se abstraen utilizando el modelo conceptual de la base en vez del implementado.

4.2 Servlets

Un **servlet** es un ejecutable escrito en Java que se ejecuta normalmente como respuesta a una petición HTTP, procesandola y generando una respuesta en tiempo de ejecución. Se diferencia de los CGI en que se ejecutan en hilos diferentes del proceso servidor, y son independientes de plataforma.

- Cada servlet se asocia a URLs, y el contenedor de aplicaciones las asocia
- El método **init()** se ejecuta cuando el servidor de aplicaciones carga el servlet en memoria, y **destroy()** cuando decide eliminarlo
- El método **service()** se ejecuta cuando se requiere del servicio del servlet
  - La petición se almacena en una clase **ServletRequest**
  - La respuesta se almacena en una clase **ServletResponse**
- Un servlet HTTP contiene los métodos **doGet()** y **doPost()**, que se basan en **service()**
  - La petición se almacena en una clase **HttpServletRequest**
  - La respuesta se almacena en una clase **HttpServletResponse**

4.3 Java Server Pages