Guía HTML

Introducción

Funcionamiento del protocolo HTTP:

El protocolo **HTTP (Hypertext Transfer Protocol)** es el lenguaje fundamental que permite la comunicación entre navegadores web y servidores, posibilitando la transmisión de información y la visualización de páginas web. Su funcionamiento se basa en un esquema de **petición-respuesta**:

1. Solicitud del cliente: El usuario escribe una dirección URL en su navegador. El navegador analiza la URL y la convierte en una **solicitud HTTP**. La solicitud incluye información como el método (GET, POST, etc.), la URL del recurso solicitado y encabezados adicionales. El navegador envía la solicitud al servidor web correspondiente, utilizando el protocolo TCP/IP.

2. Respuesta del servidor: El servidor recibe la solicitud y la procesa. Si la solicitud es válida y el recurso existe, el servidor genera una **respuesta HTTP**. La respuesta incluye un código de estado (200 para éxito, 404 para no encontrado, etc.), encabezados con información del recurso y el contenido solicitado (por ejemplo, una página HTML, una imagen o datos JSON). El servidor envía la respuesta al navegador a través del protocolo TCP/IP.

3. Recepción y procesamiento de la respuesta: El navegador recibe la respuesta HTTP del servidor. Interpreta el código de estado y los encabezados para determinar si la solicitud fue exitosa. Si la respuesta es exitosa, el navegador procesa el contenido y lo muestra al usuario en la forma correspondiente (por ejemplo, renderizando una página web, descargando una imagen o procesando datos JSON).

PARTE 1: HTML (HyperText Markup Language) es el lenguaje estándar para crear páginas web. Permite estructurar el contenido de una página web utilizando diferentes etiquetas. En esta guía, aprenderemos los conceptos básicos de HTML y cómo crear tu primera página web.

Paso 1: Configuración del Documento

Todo documento HTML debe comenzar con una estructura básica. Aquí tienes un ejemplo de cómo iniciar un documento HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Título de la Página</title>
</head>
<body>

</body>
</html>
```

<!DOCTYPE html>: Define la versión de HTML utilizada.

<html lang="es">: Define el idioma del documento.

<head>: Contiene metadatos y enlaces a recursos externos.

<meta charset="UTF-8">: Define el conjunto de caracteres utilizado (UTF-8 es estándar).

<meta name="viewport" content="width=device-width, initial-scale=1.0">: Configura la escala inicial y el tamaño de la pantalla para dispositivos móviles.

<title>: Define el título de la página, visible en la pestaña del navegador.

<body>: Contiene el contenido visible de la página.

Paso 2: Estructura Básica

Dentro del elemento **<body>**, puedes comenzar a agregar contenido utilizando diferentes etiquetas. Aquí hay algunas etiquetas básicas que puedes usar:

<h1>, <h2>, <h3>, <h4>, <h5>, <h6>: Encabezados de diferentes niveles.

<p>: Párrafos de texto.

<a href="#">: Enlaces (ancla) a otras páginas o recursos.

<img src="ruta/a/la/imagen.jpg" alt="Descripción de la Imagen">: Inserta imágenes.

** y <ol>:** Listas no ordenadas (viñetas) y listas ordenadas (números), respectivamente.

<li>: Elementos de la lista.

```
<body>
    <h1>Título Principal</h1>
    <p>Este es un párrafo de texto.</p>
    <a href="#">Enlace a otra página</a>
    <img src="ruta/a/la/imagen.jpg" alt="Descripción de la
Imagen">
    <ul>
        <li>Elemento de lista 1</li>
        <li>Elemento de lista 2</li>
    </ul>
</body>
```

Paso 3: Estructura Avanzada

HTML también permite crear estructuras más complejas utilizando etiquetas como ** ** para dividir el contenido en secciones y **** para aplicar estilos a partes específicas del texto.

```
<body>
    <header>
        <h1>Encabezado de la Página</h1>
        <nav>
            <ul>
                <li><a href="#">Inicio</a></li>
                <li><a href="#">Acerca</a></li>
                <li><a href="#">Contacto</a></li>
            </ul>
        </nav>
    </header>
    <main>
        <section>
            <h2>Sección 1</h2>
            <p>Contenido de la sección 1...</p>
        </section>
        <section>
            <h2>Sección 2</h2>
            <p>Contenido de la sección 2...</p>
        </section>
    </main>
    <footer>
        <p>Derechos de autor © 2024 - Mi Sitio Web</p>
    </footer>
</body>
```

Paso 4: Comentarios y Anidamiento

Puedes agregar comentarios en tu código HTML para hacerlo más legible para otros desarrolladores o para ti mismo en el futuro. Los comentarios no se muestran en la página web y comienzan con **<!--** y terminan con **-->**.

```
<body>
    <!-- Esto es un comentario -->
    <h1>Título Principal</h1>
    <!-- <p>Este párrafo no se mostrará en la página.</p> -->
</body>
```

Además, puedes anidar elementos HTML dentro de otros elementos para crear una estructura más compleja y organizada.

Paso 5: Validación y Pruebas

Es importante validar tu código HTML para asegurarte de que cumpla con los estándares y evitar posibles errores. Puedes utilizar herramientas en línea como el validador de W3C **https://validator.w3.org/** para verificar la validez de tu código. También debes probar tu página

web en diferentes navegadores y dispositivos para asegurarte de que se vea correctamente en todas partes.

PARTE 2: Implementar un aplicativo que responda a solicitudes de conexión

Implementar un servidor en Java que responda a solicitudes de conexión enviando una página HTML. En este ejemplo, el servidor escuchará en el puerto 8080 y responderá con una página HTML simple cuando se le haga una solicitud desde un navegador web:

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;

public class SimpleHTTPServer {
    public static void main(String[] args) throws IOException {
        ServerSocket serverSocket = new ServerSocket(8080);
        System.out.println("Servidor escuchando en el puerto 8080...");

        while (true) {
            Socket clientSocket = serverSocket.accept();
            System.out.println("Nueva conexión entrante: " + clientSocket);

            // Leer contenido HTML desde un archivo
            StringBuilder htmlContent = new StringBuilder();
            try (BufferedReader br = new BufferedReader(new FileReader("pagina.html"))) {
                String line;
                while ((line = br.readLine()) != null) {
                    htmlContent.append(line);
                }
            } catch (FileNotFoundException e) {
                System.err.println("Archivo HTML no encontrado");
                return;
            }

            // Enviar respuesta HTTP con contenido HTML del archivo
            String httpResponse = "HTTP/1.1 200 OK\r\n\r\n" + htmlContent.toString();
            OutputStream outputStream = clientSocket.getOutputStream();
            outputStream.write(httpResponse.getBytes("UTF-8"));
            outputStream.close();
        }
    }
}
```

Explicación del código:

1. Creamos un **ServerSocket** que escucha en el puerto **8080**.
2. Esperamos continuamente nuevas conexiones entrantes utilizando un bucle **while(true).**
3. Cuando se recibe una conexión entrante, aceptamos el socket del cliente (**clientSocket**).
4. Enviamos una respuesta **HTTP** simple al cliente, que incluye una página **HTML**.
5. Cerramos el **OutputStream** después de enviar la respuesta.

Para probar este servidor, puedes ejecutar el código Java y luego abrir cualquier navegador web. Luego, en la barra de direcciones del navegador, ingresa **http://localhost:8080**. Esto enviará una solicitud al servidor en el puerto 8080 de tu máquina local. El servidor responderá con la página HTML que hemos definido, y verás en tu navegador la página de la **Parte 1.**

La solicitud que hace el navegador al servidor es una solicitud HTTP GET, donde especifica la dirección (**/**) y el protocolo HTTP (**HTTP/1.1**). El servidor responde con el código de estado **200 OK**, indicando que la solicitud fue exitosa, seguido del contenido de la página HTML.