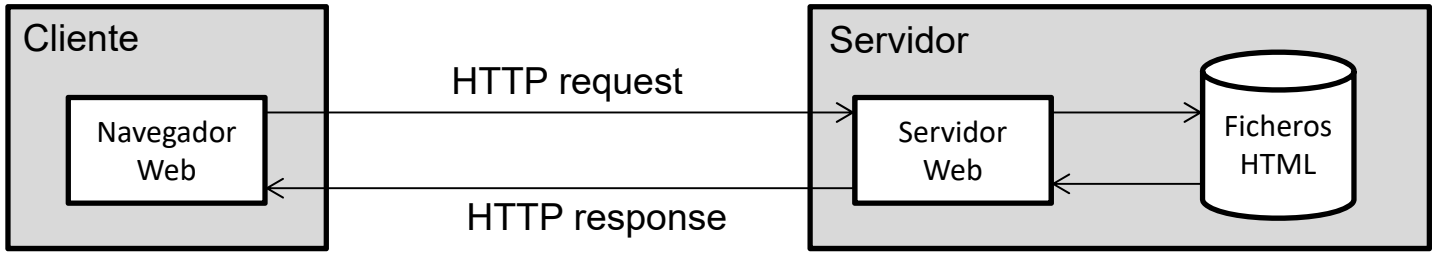
Índice

Parte I: HTML, CSS, HTTP

1. HTML

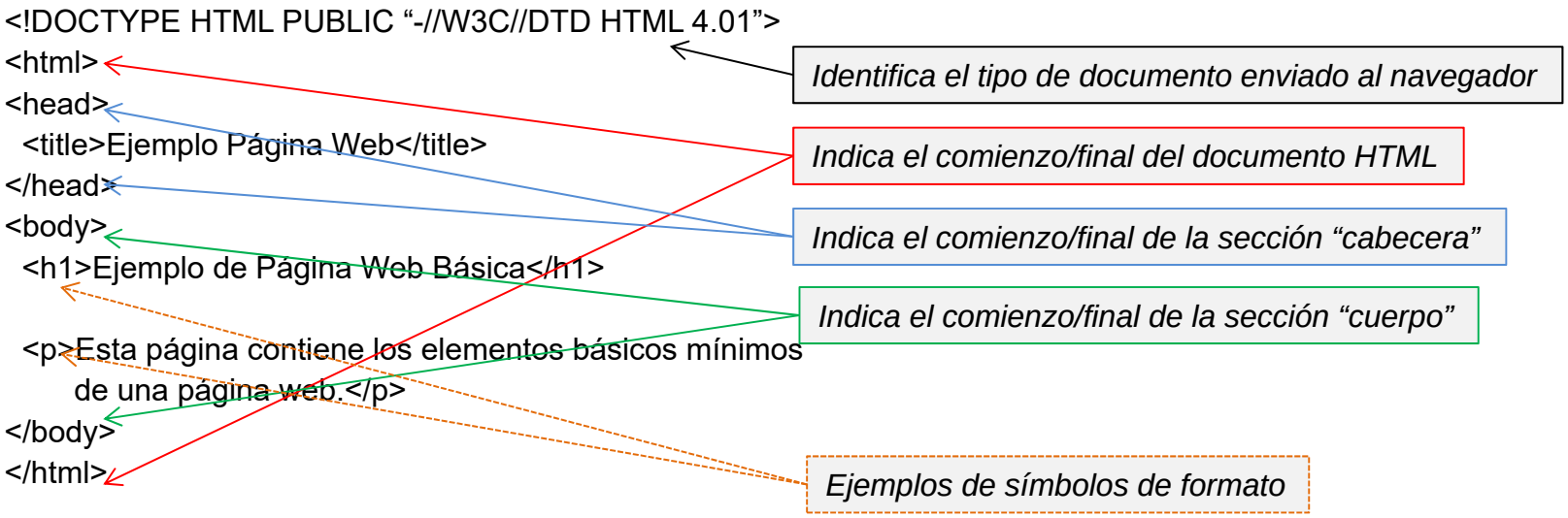
- HTML (Hypertext Markup Language): lenguaje de marcado usado para la construcción de páginas web.
  - Lenguaje de marcado: lenguaje usado para codificar un documento que incorpora, además del texto, etiquetas o marcas que indican el formato de presentación del documento.
- Página web estática: fichero HTML guardado en el servidor, tras ser pedido por el cliente es enviado mediante protocolo HTTP, siendo interpretado por el navegador.
  - Su contenido siempre es el mismo.
  - Son ficheros de texto.



- Última versión: HTML 5
  - Todavía en desarrollo: no todas sus características son implementadas por todos los navegadores.

1.1 HTML: Estructura Básica (I)

- La estructura básica, incluyendo las etiquetas básicas, es la siguiente



- Las etiquetas van entre "<…>".
  - Toda etiqueta abierta debe ser cerrada
  - Da igual ponerlas en mayúsculas o minúsculas
- Se pueden introducir espacios, tabuladores, retornos de carro, etc. para mejor legibilidad, ya que son ignorados al interpretar la página.

1.1 HTML: Estructura Básica (II)

- Secciones que componen el documento
  - !DOCTYPE (Document Type Declaration): Indica el tipo de documento y versión enviado al navegador
    - El tipo de documento y versión condiciona el tipo de etiquetas que se pueden usar

      (Ver, por ej.: http://www.w3schools.com/tags/ref_html_dtd.asp)

  - Head (cabecera). Puede contener:
    - Meta información, título de la página (como en el ejemplo), dónde encontrar hojas de estilo (CSS), scripts (pequeños programas a ejecutar por el navegador, por ejemplo escritos en javascript), …
    - Ejemplo interesante: cómo indicar el tipo de codificación usada en el envío de la página web

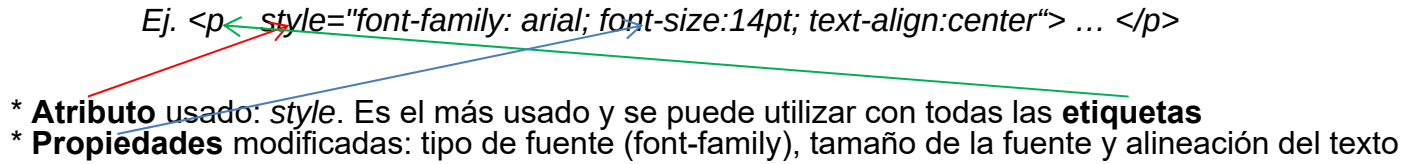
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"> (Unicode)

    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"> (ASCII para español)

  - Body (cuerpo): define el contenido de la página web
    - La mayoría de las etiquetas tienen sentido en esta parte del documento HTML

1.2 HTML: Etiquetas Básicas (I)

- Aquí sólo vamos a ver el formato básico de las etiquetas
  - La mayoría permiten modificar la interpretación estándar del navegador, personalizando la presentación de la información, mediante el uso de atributos, que modifican las propiedades de presentación.



  - Se deja para el alumno ampliar esta información.
  - Tanto etiquetas como los atributos pueden cambiar de una versión a otra.
- Etiquetas de comentario: Permiten insertar comentarios en el código
  - <!-- *Comentario* -->
- Etiquetas de formato:
  - <h1> </h1>, <h2> </h2>, …: Marcas de títulos, secciones, etc
  - : Marcas de párrafo con texto en formato normal
  -
: Salto de línea
  - : Texto en negrita
  - : Texto en cursiva o itálica
  - <u> </u>: Texto subrayado
  - : Permite modificar el formato de una parte de texto
  - : Junto con CSS permite agrupar partes de la página y darle características especiales de visualización.

1.2 HTML: Etiquetas Básicas (II)

- Introducir enlaces
  - Etiqueta: <a> *texto del enlace* . Al hacer clic sobre el texto del enlace se realizará la petición correspondiente al servidor (request).
  - Atributo: href

Ej. Enlace a página web externa: Web Escuela de Informática
Ej. Enlace a página web dentro del servidor: Catálogo 1
Ej. Enlace a fichero dentro del servidor: Programación asignatura

- Mostrar imágenes
  - Etiqueta: <img>
  - Atributos: src, height, width, alt, border, hspace, vspace, align

Ej. fichero en el servidor: <img src="images/conf1.jpg" alt="Configuración 1" height="142" width="100">
Ej. fichero externo: <img src="http://www.ejemplo.com/img.gif" align="left">

- Dividir documento en bloques
  - Etiqueta
    - Permite dividir el documento en bloques o secciones
    - Por si sólo no tiene efecto en la visualización, salvo que por defecto los navegadores suelen incluyen una línea de corte antes y después del bloque
    - Adquiere su potencial al ser usado conjuntamente con CSS
    - Atributo interesante: id. Permite identificar cada bloque.

1.3 HTML: Tablas

- Permiten representar la información en filas y columnas.
- Etiquetas básicas:
  - <table> </table>: Marcan el principio y final de la tabla
    - Algunos atributos: border, cellspacing (espacio entre celdas), cellpadding (márgenes internos de la celda), width, height
  - <th> </th>: Permite definir encabezamientos de la tabla
  - <tr> </tr>: Marcan el principio y el final de cada fila de la tabla
    - Algunos atributos: align, valign (alineamiento vertical del texto), bgcolor
  - <td> </td>: Macan el principio y el final de cada celda de la fila
    - Algunos atributos: align, valign, colspan (permite juntar celdas consecutivas dentro de la columna), rowspan (permite juntar celdas consecutivas en la fila), bgcolor, height, width

Ejemplo 1:

```
<table border="1">
  <tr>
    <th>Fila 1</th>
    <th>Fila 2</th>
  </tr>
  <tr>
    <td>celda 11</td>
    <td>celda 12</td>
  </tr>
  <tr>
    <td colspan="2">celdas juntas</td>
  </tr>
  <tr>
     <td rowspan="2">Filas juntas</td>
     <td> celda 32 </td>
  </tr>
  <tr>
      <td> celda 42 </td>
  </tr>
 </table>
```

1.4 HTML: Formularios (I)

- Permiten introducir información al usuario de diversas maneras (también información oculta) y enviarla al servidor.
- Etiquetas básicas
  - <form> </form>: Marcan el principio y final del formulario
    - Atributos importantes:
      - action: específica el URL del fichero que será llamado al hacer clic sobre el botón “enviar”
        - » Este fichero debe permitir recibir y procesar la información del formulario enviada. Por ejemplo, puede ser: JSP, servlet, PHP, etc.
      - method: especifica el método HTTP que usará el navegador para enviar la información del formulario al servidor en la correspondiente “HTTP request” generada al hacer clic en el botón “enviar”.
        - » Tipos: get y post. Por defecto se usa get. Se explicarán más adelante.
    - Otro atributos: name (identifica el formulario)
  - <input>: Permite especificar un campo de entrada de datos para el usario.
    - Atributos importantes:
      - name: nombre del campo.
      - value: valor del campo.
      - size: tamaño del campo en los tipos de texto.
      - maxlength: máximo número de caracteres que se podrán introducir en un campo de tipo texto
      - type: especifica la forma de presentar e introducir información al usuario.

1.4 HTML: Formularios (II)

- Etiquetas básicas (continuación 1)
  - <input> (continuación)
    - Tipos de campos más importantes (valores del atributo “type”)
      - text (<input type=“text”…>): para introducir texto
      - password (<input type=“password”…>): de tipo texto, pero el texto es ocultado
      - hidden (<input type=“hidden”…>): oculto al usuario. Sirve para mandar información (valor del atributo “value”) al servidor de manera oculta al usuario.
      - Botones de acción (el atributo “value” es el texto que se muestra):
        - » submit (<input type=“submit”…>): ejecuta el atributo “action” del formulario
        - » reset (<input type=“reset”…>): vacia el formulario
        - » buttom (<input type=“buttom” value=“valor” onclick=“metodo_javascript()”>): ejecuta el método Javascript indicado en el atributo “onclick”
      - checkbox (<input type=“checkbox”…>): permite seleccionar o no una opción
        - » Atributo adicional: checked. El “checkbox” aparece seleccionado por defecto
        - » No tiene atributo “value”.
      - radio (<input type=“radio”…>): permite realizar un selección entre varias opciones
        - » Al servidor se manda el valor de la opción seleccionada.
        - » Atributo adicional: checked. Indica que opción aparece seleccionada por defecto.

1.4 HTML: Formularios (III)

- Etiquetas básicas (continuación 2)
  - <select> + <option>: permite seleccionar uno (combo box) o varios elementos (list box) de una lista
    - Ej. Combo box
```
<select name="pais">
    <option value="France">Francia
    <option value="Spain" selected>España
    <option value="Italy">Italia
</select>
```
List box
```
<select name="pais" multiple>
    <option value="France">Francia
    <option value="Spain">España
    <option value="Italy">Italia
</select>
```
    - La etiqueta "selected" permite realizar una selección por defecto
  - <textarea> </textarea>: permite introducir múltiples líneas de texto
    - Atributos: rows (número de líneas del área), cols (número de columnas)

1.4 HTML: Formularios (IV)


```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01">
<html>
<head>
 <title>Ejemplo de formulario</title>

 <SCRIPT language="Javascript" type="text/JavaScript">
  function validar() {
   if (document.ejemplo.usuario.value != "Invitado") {
    if (alfanumerico(document.ejemplo.clave.value)==0) {
      alert ("Introduzca la clave.");
      document.ejemplo.clave.focus();
      return;
      }
    }
 document.ejemplo.submit();
 }

  function alfanumerico(txt) {
// Devuelve 0 si la cadena esta vacia, 1 si es numerica
//o 2 si es alfanumerica
   var i;
   if (txt.length!=0) {
    for (i=0;i<txt.length;i++){
     if (txt.substr(i,1)<"0" || txt.substr(i,1)>"9") {
      return 2;
      }
     }
    return 1;
   }
   else
    return 0;
  }
 </script>
</head>
<body>
 <h1>Ejemplo de Formulario y Tipos de Campos</h1>
 <p>Introduce nombre de usuario y clave</p>
 <form action="enviado.html" name="ejemplo" method="get">
  Username: <input type="text" name="usuario“ value="Invitado"><br>
  Password: <input type="password" name="clave"><br><br>
            <input type="hidden" name="control" value="formulario">

  <input type="checkbox" name="registrado"> Ya estoy registrado<br>

  <p>Modo de contacto</p>
  <input type="radio" name="ModoContacto" value="correo">correo<br>
  <input type="radio" name="ModoContacto" value="email" checked>email<br>
  <input type="radio" name="ModoContacto" value="ambos">ambos<br><br>

  <p>Tipo de usuario</p>
  <select name="TipoUsuario">
   <option value="tipo1" selected>Particular
   <option value="tipo2">Profesional
  </select><br><br>

  <p>Comentarios:</p>
  <textarea name="comentarios" rows="5" cols="20">
  </textarea><br><br>

  <input type="submit" value="Enviar (sin revisar)">
  <input type="reset" value="Vaciar">
  <input type="button" value="Enviar (revisando)" onclick="validar()">
 </form>
</body>
</html>
```

1.5 HTML: Traduciendo (I)

- Una vez se carga el fichero HTML (página web) en el navegador, éste lo tiene que traducir para su presentación gráfica en pantalla
- Este proceso se conoce con la anglicismo "*renderizar*"
- El navegador analizar la estructura de la página web, aplica las reglas y formatos indicados y lo transforma a valores de píxeles sobre la pantalla
  - Traduce texto del fichero HTML a información gráfica sobre la pantalla
- **IMPORTANTE** en esta traducción (renderizado):
  - **PARA UN NAVEGADOR, TODO ES UNA CAJA**
  - Todo texto o imagen está dentro de una caja, que estará a su vez dentro o al lado de otra caja y así sucesivamente
  - Es importante tener en cuenta que el contenido de una página web se estructura u organiza en cajas (rectángulo)
  - Cada caja tendrá
    - Un tamaño. Se puede modificar de por defecto con los atributos *width* y *length*.
    - Un borde (que limita la caja). Se puede modificar su formato con los atributos *border*
    - Un margen externo (separación de las cajas que la rodean). Se puede modificar con los atributos *margin*
    - Un margen interno (separación del texto que contiene con respecto a los bordes). Se puede modificar con los atributos *padding*
- Etiquetas como *div* o *span* permiten dar formato u organizar, siguiendo este esquema de cajas, el documento en partes
  - *div*: etiqueta de bloque. Comienza en nueva línea y ocupa todo el espacio (tanto en ancho como en alto) posible
  - *span: etiqueta de línea. No comienza en nueva línea y ocupa el espacio mínimo necesario para el contenido a mostrar*

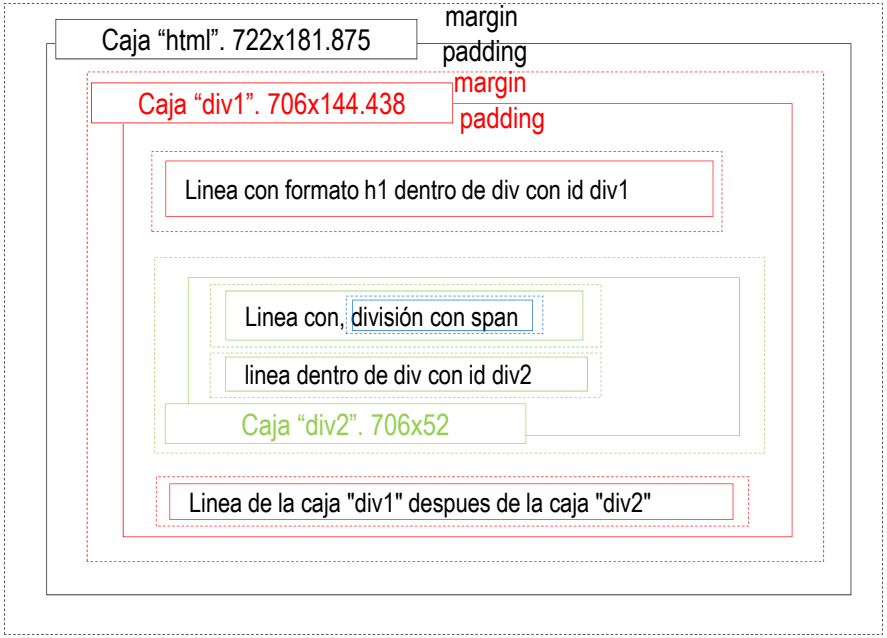
1.5 HTML: Traduciendo (II)

- Ejemplo 2

  Código

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01">
<html>
<head>
  <title>Ejemplo de todo es caja</title>
</head>
<body>
<div id="div1">
   <h1>Linea con formato h1 dentro de div con id div1</h1>
    <div id="div2">
       Linea con, <span class=“span1">division con span</span>
       <p>linea dentro de div con id div2</p>
    </div>
    <p>Linea de la caja "div1" despues de la caja "div2"</p>
</div>
```

Cajas (dimensiones de Chrome)



2. CSS

- Cascading Style Sheets: lenguaje para crear hojas de estilo donde especificar el formato de los distintos elementos HTML que puede contener la página web.
  - También se puede aplicar a otros lenguajes de marcado como XML
- No forma parte oficial de las especificaciones de HTML, pero evoluciona junto con éste.
- Permite separar la estructura del documento de su presentación
  - Estructura: ficheros HTML, presentación: ficheros CSS
  - Se puede añadir el formato de presentación en cada etiqueta, pero si la página web es compleja, es más flexible y eficiente separar ambos.
  - Su uso permite, además, separar la programación del diseño de la web.
- Se puede sobrescribir el estilo indicado en la hoja de estilo mediante
  - Etiqueta <style>
  - El atributo “style” en la etiqueta correspondiente.
- Se referencia en el documento HTML mediante la etiqueta <link> en la parte de cabecera (head).
  - Ej. <link rel=“stylesheet” type=“text/css” href=“estilos/fichstyle.css”>
- Aquí sólo vamos a ver una mínima introducción.

2.1 CSS: Sintaxis (I)

- Sintaxis básica: *elemento (o selector)* {

  propiedad1: *valor;* *(misma sintaxis que en atributo style)*

  propiedad2: valor;

  …

  }

  – Donde

  - *elemento* puede ser:
    – Una etiqueta HTML referenciada mediante su palabra clave. Ej. p, h1, h2, a, body, etc
      » Afecta a todas la etiquetas de ese tipo en el documento
    – Referencia por el atributo “id”. La sintaxis de *elemento* será: #valor_id
      » Permite particularizar estilos
      » El valor de este atributo no puede ser duplicado. Ha de ser único en todo el documento.
      Ej. Etiqueta en el documento HTML: * Texto ejemplo *
      Definición de estilo en CSS: *#texto1{font-family: arial; font-size:14pt; text-align:center}*
      Este estilo sólo afectará a **la** etiquetas con *id=“texto1”*
    – Referencia por el atributo “class”. La sintaxis será: *.valor_class*
      » Similar a “id”, pero éste puede tener el mismo valor en varios elementos del documento
      Ej. Etiqueta en el documento HTML: * Texto 1 *
      * Texto 2 *
      Definición de estilo en CSS: *.texto1{font-family: arial; font-size:14pt; text-align:center}*
      Este estilo sólo afectará a **las** etiquetas con *class=“texto1”*
      » Si etiquetas distintas tienes igual valor de atributo “class”, se puede particularizar:
      Ej. p.texto1 {…..}, li.menu{….}, …

2.1 CSS: Sintaxis (II)

- Donde (continuación 1)
  - elemento puede ser (continuación):
    - Una etiqueta dentro de un bloque

    Ej. Código HTML:
    ```
    <div id="nav" >
      <ul>
        <li><a href="#ord">Ordenadores</a></li>
        <li><a href="#tab">Tabletas</a></li>
      </ul>
    </div>
    ```

    Código CSS: #nav ul {….} → Hace referencia a las etiquetas "ul" dentro del bloque con id=nav

    #nav ul li {….} → Hace referencia a las etiquetas "li" del bloque "ul" del bloque con id=nav
    - Se puede particularizar si el ratón está o no sobre la etiqueta.

    Ej. Código CSS: a {color:#699065; text-decoration:underline; } → Estilo normal

    a:hover { color:#999999; text-decoration:none;} → Estilo si el ratón está sobre el enlace
    - …
  - propiedad puede ser:
    - Cualquiera de las características de visualización asociadas a una determinada etiqueta
    - La lista completa excede los objetivos de la asignatura (se deja para el alumno), pero algunas de las más habituales son:
      - » Referentes a texto: color, font-size, font-family, text-align, font-weight, text-decoration, …
      - » Referentes a posición de bloque: margin, padding, position, width, hight, rigth, top, …
      - » Otros: background, border, display, content, …
  - valor: será alguno de los posibles asociados a cada propiedad
    - Se deja para el alumno profundizar en este campo.

3. HTTP (I)

- Hypertext Transfer Protocol: protocolo usado para pedir un recurso desde el cliente al servidor y para retornar desde éste una respuesta al cliente.
  - Petición del recurso por parte del cliente: request
  - Repuesta del servidor: response
- Es el protocolo usado en cada transacción de la WWW (World Wide Web)
- Estructura básica de cada transacción (request/response):
  - Encabezado + línea en blanco (opcional) + Datos
    - Encabezado: contiene información acerca de la transacción
    - Datos:
      - Request: información enviada al servidor mediante la acción HTTP POST
      - Response: será la información pedida por el cliente, habitualmente el contenido en formato HTML de la página web pedida.
- No guarda estado: no se guarda información de transacción pasadas.
  - Veremos como se puede realizar esto, cuando se necesite

3. HTTP (II)

- Localización de recursos:
  - URI (Uniform Resource Identifier): Cadena de caracteres para identificar un recurso abstracto o físico (W3C RFC3918). Puede ser:
    - URL (Uniform Resource Locator): tipo de URI que identifica el recurso por su localización. Ej. localizar persona por su dirección
    - URN (Uniform Resource Name): tipo de URI que identifica el recurso por nombre. Ej. localizar persona por su nombre, el ISBN de los libros
    - Ambos.
    - URI es un concepto más general que URL
    - Estructura general: *esquema* ":" *parte-jerarquica* ["?" *petición*] ["#" *fragmento*]
      - *esquema*: indica el protocolo que se va a usar, condicionando la interpretación del resto.
        - » Ej. esquemas registrados: http, https, ftp, mailto, data, file, … No registrados: ssh, …
      - *parte-jerárquica*: es la más compleja. En los recursos web contiene información del dominio (servidor) y la ruta en el servidor para acceder al recurso.
        - » Ej. //www.infor.uva.es/~cevp/index.html
      - *petición*: variables que se pasan al recurso, en nuestro caso, página web
        - » Ej. …/index.php?variable1=valor1&variable2=valor2
      - *fragmento*: permite identificar un recurso secundario y añadir información adicional
        - » Ej. http://www.infor.uva.es/~cevp/index.php#otros
    - Otros ejemplos:
      - ftp://ftp.is.co.za/rfc/rfc1808.txt
      - ldap://[2001:db8::7]/c=GB?objectClass?one
      - mailto:John.Doe@example.com
      - news:comp.infosystems.www.servers.unix

3.1 HTTP: Request (I)

- Mensaje enviado desde el cliente al servidor para pedir un recurso.
- Estructura (W3C RFC 2616):
  1. Request *Line: request-method request-URI request-protocol*
     - Métodos de petición típicos (request_method): GET y POST, pero existen otros.

     Ej. GET http://www.murach.com/email/join_email_list.html HTTP/1.1
  2. Request Headers: *request-header*: *valor*
     - contienen información acerca del cliente que realiza la petición, por lo tanto, puede ser ligeramente diferente en distintos navegadores.

     Ej. user-agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
     - Cabeceras (*request-header*) más comunes:
       - accept: Orden de tipos MIME aceptados por el cliente (navegador).
       - accept-encoding: Tipos de compresión aceptados por el navegador.
       - accept-charset: Conjuntos de caracteres aceptados por el navegador.
       - accept-language: Códigos de los lenguajes preferidos para la respuesta. Ej. “en”, “en-us”, …
       - connection: Tipo de conexión usada por el navegador.
       - cookie: Especifica cualquier cookie que hubiera sido previamente enviada por el servidor.
       - host: Host y puerto del servido al que se pide el recurso.
       - referer: URI (URL) de la página referenciada.
       - user-agent: Tipo de navegador.
  3. Línea vacía
  4. Cuerpo del mensaje. Es opcional.
     - Aparece, por ejemplo, si se usa POST como método de petición

3.1 HTTP: Request (II)

- Ejemplo

  GET http://www.murach.com/email/join_email_list.html HTTP/1.1
  host: www.murach.com
  connection: Keep-Alive
  user-agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
  referer: http://www.murach.com/murach/index.html
  accept: image/gif, image/jpeg, text/html, application/vnd.ms-excel
  accept-encoding: gzip, deflate
  accept-language: en-us
  cookie: emailCookie=joel@murach.com; firstNameCookie=Joel

  – Para ver cabeceras reales:
    - URL: http://websniffer.cc/
      – En el formulario que aparece escribir la URL que se quiera y hacer clic en botón “Snif page”
        » Esa web nos permite ver la cabecera del HTTP request generado
        » También aparece HTTP response, que lo veremos más adelante
        » Se pueden cambiar parámetros de la petición, para ver cómo cambia la cabecera.

    - También se pueden usar las herramientas de desarrollador de los navegadores. Ejemplo Mozilla Firefox: Menu->Desarrollador Web->Red (Ctrl+May+E)

3.2 HTTP: GET vs POST

- HTTP define 8 métodos para especificar la acción a realizar sobre el recurso especificado.
  - GET y POST: métodos típicamente usados.
- HTTP GET
  - Método típicamente usada para realizar una petición de recurso al servidor.
  - Permite enviar datos al servidor añadiéndolos como parámetros de la URI

  Ej. http://www.greidi.uva.es/Web_LEEMUSICA/index.php?page=LeeMusica&part=parte1
    - Si se recarga la página, los datos son nuevamente enviados al servidor
      - Esta forma de enviar datos al servidor puede ser usada cuando esto no suponga un problema
    - Los datos enviados se muestran en la URI, es decir, pueden ser vistos por el usuario
    - Ejecución un poco más rápida que el POST
- HTTP POST
  - Método típicamente usado para enviar datos desde el cliente al servidor
  - Los datos se añaden en el cuerpo de la petición (después de las cabeceras)
    - Los datos no se muestran al usuario
    - Si se refresca la página se avisa al usuario que los datos van a ser renviados
    - Mejor si el tamaño de los datos a enviar es elevado

3.3 HTTP Response (I)

- Mensaje enviado desde el servidor al cliente como respuesta a una petición de recurso.
- Estructura (W3C RFC 2616):

1. Status Line: *HTTP-Version Status-Code Mensaje-Asociado-al-Status-Code*

Ej. HTTP/1.1 200 OK

- Status-Code: entero de 3 dígitos que codifica la respuesta del servidor
  - Ej. 200-299: petición realizada con éxito. (200: código por defecto para todo correcto)

    400-499: error en la petición realizada por el cliente (404: URL no encontrada)

    500-599: error en el servidor (500: error interno del servidor)

2. Response Headers: *response-header*: *valor*

Ej. content-type: text/html

- Algunas cabeceras de la respuesta:
  - content-type: Tipo MIME del documento resultado de la petición
  - cache-control: controla cuándo y cómo un navegador gestiona la caché
  - content-length: tamaño en bytes del cuerpo de la respuesta
  - …

3. Línea en blanco
4. Response entity o response body (cuerpo de la respuesta): respuesta al recurso pedido.

Ej. documento html, documento xml, texto plano, una imagen o video, fichero pdf, etc

3.3 HTTP Response (II)

- Ejemplo

```
HTTP/1.1 200 OK
date: Sat, 17 Mar 2008 10:32:54 GMT
server: Apache/2.2.3 (Unix) PHP/5.2.4
content-type: text/html
content-length: 172
last-modified: Fri, 16 Aug 2007 12:52:09 GMT

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01">
<html>
  <head>
    <title>Murach's Java Servlets and JSP</title>
  </head>
  <body>
    <h1>Join our email</h1>
   </body>
</html>
```

- Para verlo de manera práctica:
  - Usar http://websniffer.cc/ fijándonos, ahora, en la parte de HTTP response
    Ej. Página web: http://www.greidi.infor.uva.es
  - Probar con páginas que no existen y ver respuesta con error.
    Ej. http://www.uva.es/problemas.html