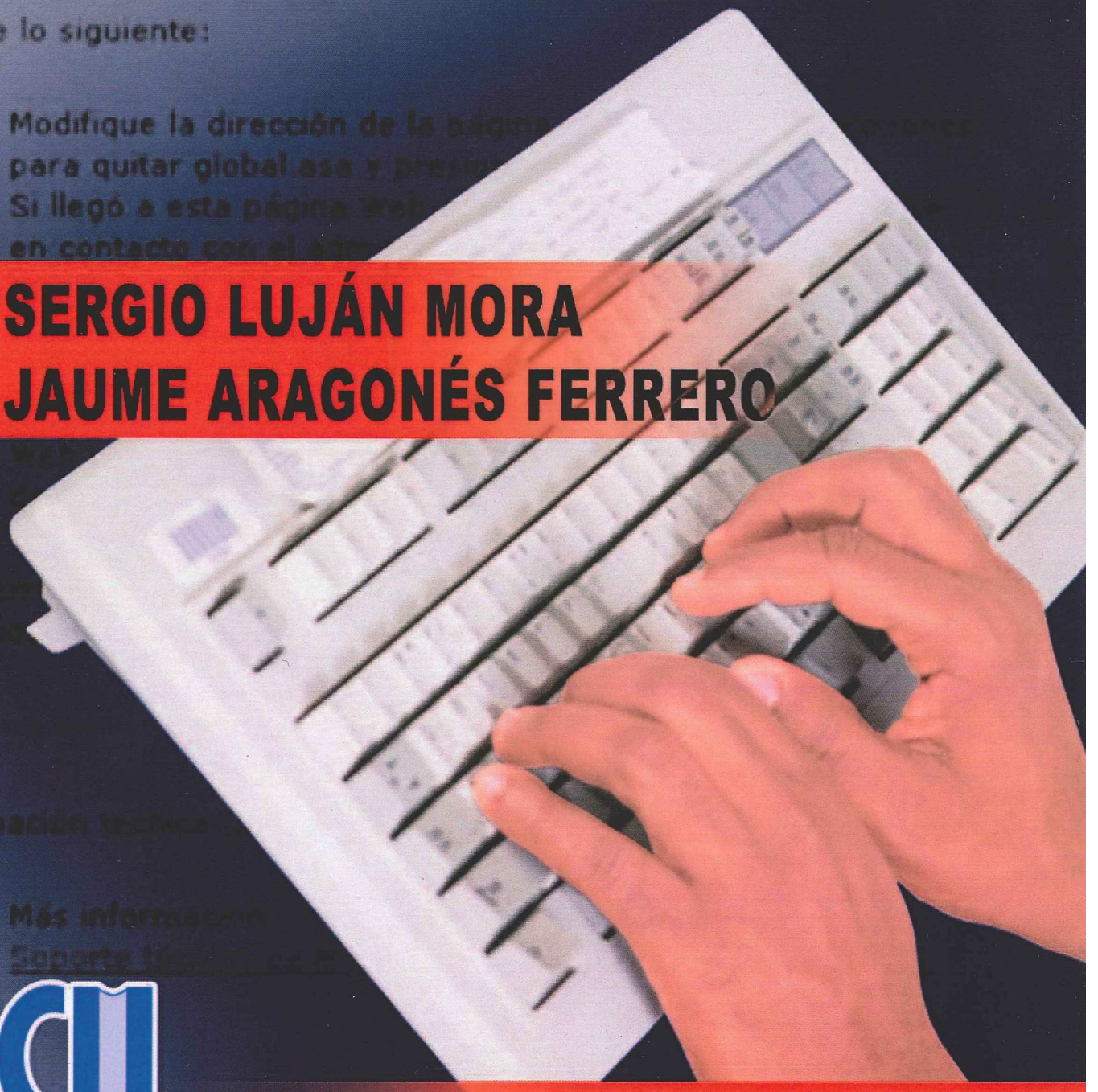
CUESTIONARIO BÁSICO SOBRE PROGRAMACIÓN EN INTERNET

SERGIO LUJÁN MORA
JAUME ARAGONÉS FERRERO



ECU

CUESTIONARIO BÁSICO SOBRE PROGRAMACIÓN EN INTERNET

El rápido desarrollo de Internet durante los últimos diez años ha modificado el desarrollo de las aplicaciones informáticas con la aparición del concepto de "aplicación web". Este novedoso concepto de aplicación ha propiciado la aparición de nuevas filosofías de desarrollo y la utilización de nuevas tecnologías, como HTML, JavaScript, VBScript, CGI, SSI, ASP, PHP, etc.

Este libro contiene casi 70 preguntas de tipo test (4 respuestas) que abarcan los principales aspectos de la programación de aplicaciones web: HTML (24 preguntas), JavaScript (10 preguntas), ASP (22 preguntas) y Java y JSP (12 preguntas). Cada una de las preguntas incluye sus respectivas respuestas correctas con una explicación razonada, lo que aporta un gran valor al libro.

Este libro es el complemento ideal para preparar cualquier tipo de prueba (examen, oposición, etc.) en la que se requieran conocimientos sobre la programación de aplicaciones web.

SERGIO LUJÁN MORA es Ingeniero en Informática por la Universidad de Alicante. Ha impartido diversos cursos sobre Internet y las diversas tecnologías que se emplean en la programación de aplicaciones web (HTML, JavaScript, ASP y Java).

Durante un año ha trabajado en el Laboratorio Multimedia (mmlab) de la Universidad de Alicante, desarrollando aplicaciones para Internet e intranets.

Ha escrito los libros "*Programación en Internet: Clientes Web*", "*Programación de servidores web con CGI, SSI, e IDC*" y "*Programación de aplicaciones web: historia, principios básicos y cliente web*" publicados por Editorial Club Universitario.

Desde 1999 forma parte del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Alicante. Las asignaturas que imparte en la actualidad son "Programación en Internet" y "Programación y Estructuras de Datos".

JAUME ARAGONÉS FERRERO es Ingeniero en Informática por la Universidad de Alicante. Ha impartido numerosos cursos sobre Internet y las diversas tecnologías que se emplean en la programación de aplicaciones web (HTML, JavaScript, ASP, PHP, JSP y Java).

Trabajó en el Laboratorio Multimedia durante dos años como responsable técnico de proyectos de desarrollo de aplicaciones en Internet e intranets para la Universidad de Alicante y durante tres años en la empresa privada como director de proyectos de aplicaciones intranets para empresas del sector de los medios de comunicación.

Desde 1999 forma parte del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Alicante. Las asignaturas que imparte en la actualidad son "Programación en Internet" y "Nuevas tecnologías aplicadas a la educación". Además, desde 2003 trabaja como analista-programador en el Servicio de Informática de la Universidad de Alicante, integrado en el equipo de desarrollo de la intranet educativa "Campus Virtual".

ECU®
EDITORIAL CLUB UNIVERSITARIO

I.S.B.N.: 84-8454-413-3

9 788484 544135

NOTA DEL AUTOR

Creative Commons

Creative Commons License Deed

Reconocimiento-NoComercial-SinObraDerivada 3.0 España (CC BY-NC-ND 3.0)

Usted es libre de:

copiar, distribuir y comunicar públicamente la obra

Bajo las condiciones siguientes:

Reconocimiento — Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciador (pero no de una manera que sugiera que tiene su apoyo o apoyan el uso que hace de su obra).

No comercial — No puede utilizar esta obra para fines comerciales.

Sin obras derivadas — No se puede alterar, transformar o generar una obra derivada a partir de esta obra.

Entendiendo que:

Renuncia — Alguna de estas condiciones puede no aplicarse si se obtiene el permiso del titular de los derechos de autor

Dominio Público — Cuando la obra o alguno de sus elementos se halle en el dominio público según la ley vigente aplicable, esta situación no quedará afectada por la licencia.

Otros derechos — Los derechos siguientes no quedan afectados por la licencia de ninguna manera:

- Los derechos derivados de usos legítimosu otras limitaciones reconocidas por ley no se ven afectados por lo anterior.
- Los derechos morales del auto;
- Derechos que pueden ostentar otras personas sobre la propia obra o su uso, como por ejemplo derechos de imagen o de privacidad.

- **Aviso** — Al reutilizar o distribuir la obra, tiene que dejar bien claro los términos de la licencia de esta obra.

Título: Cuestionario Básico sobre "Programación en Internet"
Autores: © Sergio Luján Mora y Jaume Aragonés Ferrero

I.S.B.N.: 84-8454-413-3
Depósito legal: A-177-2005

Edita: Editorial Club Universitario Telf.: 96 567 61 33
C/ Cottolengo, 25 – San Vicente (Alicante)
www.ecu.fm

Printed in Spain
Imprime: Imprenta Gamma Telf.: 965 67 19 87
C/. Cottolengo, 25 - San Vicente (Alicante)
www.gamma.fm
gamma@gamma.fm

Cuestionario básico sobre "Programación en Internet"

Sergio Luján Mora
Jaume Aragonés Ferrero

Índice general

Índice de figuras

Índice de cuadros

Índice de acrónimos

API *Application Program Interface*
Interfaz de programación de aplicaciones. Conjunto de constantes, funciones y protocolos que permiten programar aplicaciones. Una buena **API** facilita la tarea de desarrollar aplicaciones, ya que facilita todas las piezas y el programador sólo tiene que unirlas para lograr el fin que desea.

ARPA *Advanced Research Projects Agency*
Agencia de Proyectos de Investigación Avanzados. Agencia creada por el Departamento de Defensa de los Estados Unidos de Norteamérica en 1958. También conocida como **DARPA**. A lo largo de los años ha cambiado su nombre varias veces: en 1971 **DARPA**, en 1993 **ARPA** y en 1996 **DARPA** otra vez. El proyecto más conocido de los desarrollados por esta agencia es ARPANET (o ARPAnet), semilla de la actual Internet.

ASCII *American Standard Code for Information Interchange*
Código binario utilizado para representar letras, números, símbolos, etc. A cada carácter se le asigna un número del 0 al 127 (7 bits). Por ejemplo, el código **ASCII** para la A mayúscula es 65. Existen códigos **ASCII** extendidos de 256 caracteres (8 bits), que permiten representar caracteres no ingleses como las vocales acentuadas o la eñe. Los caracteres de la parte superior (128 a 255) de estos códigos **ASCII** extendidos varían de uno a otro. Por ejemplo, uno de los más extendidos es ISO Latin-1 (oficialmente ISO-8859-1).

ASP *Active Server Pages*
Tecnología propietaria de MICROSOFT que permite crear páginas web dinámicas en el servidor. Desarrollada con el objetivo de sustituir a la tecnología **CGI**, ofrece una serie de características que facilitan la programación de aplicaciones web. Las páginas **ASP** suelen estar programadas en *VBScript*, aunque también se pueden programar en otros lenguajes, como *JScript*.

BBS *Bulletin Board System*
Una **BBS** es un centro de intercambio de mensajes electrónicos. El funcionamiento básico de una **BBS** es muy sencillo: el usuario se conecta a través de

un módem, puede revisar los mensajes dejados por otros usuarios o puede dejar sus propios mensajes.

BMP *Bit-map*
Formato gráfico de mapa de bits estándar en los sistemas operativos Microsoft Windows. Almacena las imágenes en un formato llamado "mapa de bits independiente del dispositivo", que significa que el color de cada punto (*pixel*) se almacena de un modo independiente del método empleado por un dispositivo para representar el color. Existen diversos formatos: 1 bit (blanco y negro), 4 bits (16 colores), 8 bits (256 colores) y 24 bits (16 777 216 colores).

CERN *Conseil Européenne pour le Recherche Nucléaire*
Organización Europea para la Investigación Nuclear. Es el mayor centro científico a nivel mundial dedicado a la física de partículas. Su sede central se encuentra en Ginebra, Suiza. Fundado en 1954 por 12 países, actualmente está formado por 20 países, entre ellos España. Tim Berners-Lee, mientras trabajaba en él a principios de 1990, inventó la **WWW**.

CFML *ColdFusion Markup Language*
Lenguaje de etiquetas empleado para programar las páginas web dinámicas en el servidor basadas en la tecnología ColdFusion.

CGI *Common Gateway Interface*
Estándar que permite el intercambio de información entre un servidor y un programa externo al servidor. Un programa **CGI** es un programa preparado para recibir y enviar datos desde y hacia un servidor web según este estándar. Normalmente se programan en *C* o en *Perl*, aunque se puede usar cualquier lenguaje de propósito general.

CSP *Caché Server Pages*
Tecnología propietaria de InterSystems que permite crear páginas web dinámicas en el servidor. Se diferencia de otras tecnologías similares como **ASP** y **JSP** en que la lógica de negocio reside junto con la lógica de datos en el sistema gestor de bases de datos.

CSS *Cascading Style Sheets*
Tecnología empleada en la creación de páginas web, que permite un mayor control sobre el lenguaje **HTML**. Permite crear hojas de estilo que definen como cada elemento, como por ejemplo los encabezados o los enlaces, se tiene que mostrar. El término "en cascada" indica que diferentes hojas de estilo se pueden aplicar sobre la misma página. **CSS** ha sido desarrollada por **W3C**.

DARPA *Defense Advanced Research Projects Agency*
Ver **ARPA**.

DHTML *Dynamic HTML*
Conjunto de extensiones a **HTML** que permiten modificar el contenido de una página web en el cliente sin necesidad de establecer una nueva comunicación con el servidor. Se basa en el uso de **DOM** para acceder al contenido de la página.

DNS *Domain Name System*
También conocido como *Domain Name Service*, es un servicio de Internet que traduce los nombres de dominio (por ejemplo, `http://www.ua.es`) a direcciones IP (`193.145.233.8`). Como los nombres de dominio son palabras, son más fáciles de recordar que las direcciones IP. Sin embargo, Internet se basa en las direcciones IP. Cada vez que se emplea un nombre de dominio, el servicio de **DNS** traduce un nombre de dominio en su correspondiente dirección IP. Si un servidor de **DNS** no sabe cómo traducir un nombre de dominio, traslada la pregunta a otro servidor de **DNS**y así sucesivamente hasta que se obtenga la dirección IP correspondiente o una respuesta de error.

DOM *Document Object Model*
Especificación que define como se puede acceder a los objetos de un documento **HTML** (ventanas, imágenes, formularios) a través de un lenguaje de *script*. Básicamente define una jerarquía de objetos. **DOM** se encuentra en proceso de estandarización por **W3C**. **DHTML** depende de **DOM** para cambiar dinámicamente el contenido de una página web. Desgraciadamente, los dos navegadores mayoritarios poseen distintos modelos de objetos que en algunas partes son incompatibles entre sí.

GIF *Graphics Interchange Format*
Formato gráfico de mapa de bits desarrollado por CompuServe para su servicio de información. Sus principales características son: compresión de datos sin pérdidas (**LZW**), soporte de transparencias y de animaciones. Existen dos versiones de este estándar gráfico: 87A y 89A. Es el formato más adecuado para imágenes con pocos colores, dibujos sencillos o textos.

GNU *GNU is Not Unix*
GNU es un acrónimo recursivo de "*GNU is Not Unix*". El proyecto **GNU** nació en 1984 de la mano de Richard Stallman en el **MIT**, con el fin de desarrollar un sistema operativo del estilo de Unix, pero totalmente gratuito. El objetivo de**GNU** es la creación de software no propietario. Este software se puede descargar, usar y modificar libremente, siempre que se cumpla la **GPL**. La principal limitación que impone esta licencia es que no se puede limitar la distribución y uso de un software basado en **GNU**.

GPL *GNU General Public License*
GPL es la licencia que se emplea con algunos tipos de software de código abierto (*open source software*), que detalla bajo que condiciones el software y su correspondiente código fuente se puede copiar, distribuir y modificar.

HTML *HyperText Markup Language*
Lenguaje compuesto de una serie de etiquetas o marcas que permiten definir el contenido y la apariencia de las páginas web. Aunque se basa en el estándar **SGML**, no se puede considerar que sea un subconjunto de él. Existen cientos de etiquetas con diferentes atributos. **W3C** se encarga de su estandarización. El futuro sustituto de **HTML** es **XHTML**.

HTTP *HyperText Transfer Protocol*
Es el protocolo que emplea la **WWW**. Define como se tienen que crear y enviar los mensajes y que acciones debe tomar el servidor y el navegador en respuesta a un comando. Es un protocolo *stateless* (sin estado), porque cada comando se ejecuta independientemente de los anteriores o de los posteriores. Actualmente, la mayoría de los servidores soportan **HTTP** 1.1 (**RFC** 2616 de junio de 1999). Una de las principales ventajas de esta versión es que soporta conexiones persistentes: una vez que el navegador se conecta al servidor, puede recibir múltiples ficheros a través de la misma conexión, lo que aumenta el rendimiento de la transmisión hasta en un 20 %.

IAB *Internet Architecture Board*
Comité del **IETF**. En el **RFC** 2850 "Charter of the Internet Architecture Board (IAB)" se establece su organización y su misión. La dirección de su sitio web en Internet es la siguiente: `http://www.iab.org/`.

IANA *Internet Assigned Numbers Authority*
Comité del **IETF** encargado de estandarizar los distintos parámetros que se emplean en Internet, como son las direcciones **IP**, los nombres de dominio, los puertos empleados por los protocolos y otros tipos de nombres e identificadores. La dirección de su sitio web en Internet es `http://www.iana.org/`.

IDC *Internet Database Connector*
Conector de bases de datos de Internet. Tecnología propietaria de MICROSOFT que permite generar páginas web dinámicas a partir de la información almacenada en una base de datos. Es el precursor de **ASP**.

IETF *Internet Engineering Task Force*
Organización internacional formada por diseñadores de red, operadores de Internet y fabricantes, que vigila por la correcta evolución y funcionamiento de Internet. En el **RFC** 3160 "The Tao of IETF - A Novice's Guide to the Internet Engineering Task Force" se puede consultar su composición, funcionamiento y objetivos. La dirección de su sitio web en Internet es `http://www.ietf.org/`.

IP *Internet Protocol*
El protocolo **IP** especifica el formato de los paquetes (datagramas) y el esquema de direccionamiento en Internet. La versión actual de este protocolo es IPv4,

pero en la actualidad se está comenzando a implantar la nueva versión IPv6, también llamada IPng (*IP next generation*).

ISO *International Organization for Standards*
Organización fundada en 1946, cuyos miembros son las organizaciones nacionales de normalización (estandarización) correspondientes a los países miembros. Entre sus miembros se incluyen ANSI (Estados Unidos), BSI (Gran Bretaña), AFNOR (Francia), DIN (Alemania) y UNE (España).

ISOC *Internet Society*
La **ISOC** es una sociedad profesional establecida en 1992, que agrupa más de 150 organizaciones y 16.000 miembros individuales en más de 180 países. Esta organización agrupa otras organizaciones responsables de desarrollar los estándares de Internet, como la **IETF**, la **IANA** y la **IAB**. La dirección de su sitio web en Internet es `http://www.isoc.org/`.

JPEG *Joint Photographic Experts Group*
Nombre del comité de expertos que desarrolló el formato gráfico con el mismo nombre. Se trata de un formato gráfico de mapa de bits que incorpora compresión de datos con pérdidas y permite trabajar con 24 bits de color (color real o verdadero). El nivel de compresión es variable, por lo que se puede elegir entre mejor calidad y menor compresión o peor calidad y mayor compresión. Este formato se suele emplear con imágenes fotográficas o complejas, pero no es el adecuado para imágenes sencillas, dibujos o textos.

JPG *Joint Photographic Experts Group*
Ver **JPEG**.

JSP *Java Server Pages*
Tecnología de Sun Microsystems que permite crear páginas web dinámicas en el servidor. Equivale a la tecnología **ASP** de Microsoft. Se programan en *Java*.

LZW *Lempel Ziv Welch*
Esquema de compresión sin pérdidas empleado en el formato gráfico **GIF** de Compuserve. Desarrollado por J. Ziv and A. Lempel en 1977, y posteriormente mejorado por T. Welch. La patente de **LZW** la ostenta Unisys Corporation. Durante muchos años, Unisys Corporation permitió el uso de **LZW** sin cobrar un canon (la mayoría de la gente no sabía que había sido patentado en 1983). Sin embargo, a partir de 1995 decidió cobrar una tasa y se desató una gran controversia ya que se había extendido ampliamente su uso.

MIME *Multipurpose Internet Mail Extensions*
Estándar que permite la transmisión de cualquier tipo de fichero mediante correo electrónico. Además, los navegadores también emplean los tipos **MIME**

para identificar y visualizar distintos tipos de recursos que no están en formato **HTML**.

MIT *Massachusetts Institute of Technology*
Instituto Tecnológico de Massachusetts. Centro de investigación avanzado situado en los Estados Unidos. Famoso por su "Media Lab", en el cual trabajan investigadores de la talla de Nicholas Negroponte o Marvin Minsky.

MPEG *Moving Picture Experts Group*
Nombre del comité de expertos que desarrolló el formato de vídeo digital con el mismo nombre. En realidad, se trata de un conjunto de formatos de compresión de vídeo con pérdidas (MPEG-1, MPEG-2 and MPEG-4) con diferentes resoluciones (352 x 240, 720 x 480, etc.) y velocidades de refresco.

MPG *Moving Picture Experts Group*
Ver **MPEG**.

MVJ *Máquina Virtual Java*
Entorno de ejecución independiente de la plataforma que convierte los *bytecodes* del lenguaje *Java* al lenguaje máquina de la plataforma (arquitectura de microprocesador y sistema operativo) donde se ejecuta.

NCSA *National Center for Supercomputing Applications*
Centro Nacional para Aplicaciones de Supercomputación. Centro creado en la Universidad de Illinois en enero de 1986. Famoso porque uno de los primeros navegadores web gratuitos, NCSA Mosaic, se creó en sus instalaciones.

ODBC *Open DataBase Connectivity*
Conectividad abierta de bases de datos. **ODBC** es un estándar *de facto* para el acceso a bases de datos en entornos cliente/servidor. El objetivo de **ODBC** es facilitar el acceso a cualquier dato desde cualquier aplicación, independientemente del sistema gestor de bases de datos empleado. Para ello, en **ODBC**se inserta una capa intermedia, llamada controlador (*driver*) de la base de datos, entre la aplicación y el sistema gestor de bases de datos. El propósito de esta capa es traducir las consultas que genera la aplicación en comandos que entienda el sistema gestor de bases de datos. Por tanto, mediante **ODBC**, se puede cambiar la parte servidor (la base de datos) sin tener que cambiar el cliente, siempre que todas las partes sean compatibles con **ODBC**.

OSI *Open System Interconnection*
También conocido como el Modelo de Referencia **OSI** o el Modelo **OSI**. Se trata de un estándar de **ISO** que define un marco para implementar los protocolos de red en siete capas. Los siete niveles, desde el más inferior (1) al superior (7) son: físico, enlace, red, transporte, sesión, presentación y aplicación.

PNG *Portable Network Graphics*
Formato gráfico de mapa de bits similar a **GIF**. **W3C** ha desarrollado este formato gráfico con la idea de sustituir **GIF** por **PNG** debido a que el primero emplea un algoritmo que está patentado, mientras que **PNG** es totalmente gratuito. No permite crear animaciones, pero sí que permite definir distintos niveles de transparencia. Al igual que **GIF**, emplea un esquema de compresión sin perdidas que logra tasas de compresión mayores que **GIF**. Tanto Microsoft Internet Explorer como Netscape Navigator aceptan este formato, aunque no todas sus características.

RFC *Request for Comments*
Medio de publicar propuestas sobre Internet. Cada **RFC** recibe un número. Algunos se convierten en un estándar de Internet.

RGB *Red Green Blue*
Notación de los colores en la que cada color se representa como una combinación de los tres colores básicos (primarios) rojo (*red*), verde (*green*) y azul (*blue*). Se trata de un modelo aditivo (se parte del negro). Mediante la combinación adecuada de los tres colores básicos se consigue todo el espectro de colores. Además de **RGB** existen otras formas de representar los colores. Otra de las más corrientes es CMYK (*cyan*, *magenta*, *yellow*, *black*), que se trata de un modelo sustractivo.

RTB *Red de Telefonía Básica*
También llamada Red de Telefonía Conmutada. Es el servicio que los operadores de telefonía dan a los usuarios a través de plataformas terrestres y cableadas hasta el punto final.

SGML *Standard Generalized Markup Language*
Lenguaje que permite organizar y etiquetar los distintos elementos que componen un documento. Se emplea para manejar grandes documentos que sufren constantes revisiones y se imprimen en distintos formatos e idiomas. Desarrollado y estandarizado por **ISO** en 1986.

SSI *Server Side Include*
Directivas de inclusión del servidor. Comandos que se incluyen en una página **HTML** y que son ejecutados por el servidor web antes de transmitir la página al cliente. Permite generar páginas web dinámicas.

TCP/IP *Transmission Control Protocol/Internet Protocol*
Familia de protocolos que se emplean en las comunicaciones de Internet.

TIFF *Tagged Image File Format*
Formato gráfico de mapa de bits muy popular tanto en entorno Microsoft Windows como Apple Macintosh. Permite almacenar imágenes de cualquier resolución, en blanco y negro, escala de grises o color.

URL *Universal Resource Locator*
También conocido como *Uniform Resource Locator*. Sistema de direccionamiento de máquinas y recursos en Internet. Es decir, se trata de una dirección que permite localizar cualquier máquina o documento que se encuentre accesible a través de Internet.

VRML *Virtual Reality Modeling Language*
Lenguaje de Modelado de Realidad Virtual. Lenguaje para crear objetos en tres dimensiones en la Web. Los ficheros creados con este lenguaje poseen la extensión `.wrl` (de *world*) y para visualizarlos es necesario emplear un visor adecuado o que el navegador web disponga del correspondiente *plug-in*.

W3C *World Wide Web Consortium*
Consorcio internacional de compañías y organizaciones involucradas en el desarrollo de Internet y en especial de la **WWW**. Su propósito es desarrollar estándares y "poner orden" en Internet.

WWW *World Wide Web*
También conocida como "la Web" o "la Red". Sistema mundial de servidores web conectados a Internet (no todos los ordenadores conectados a Internet forman parte de la **WWW**). Su protocolo de comunicación es **HTTP**, su lenguaje de creación de documentos **HTML** y su sistema de direccionamiento de los recursos **URL**. Los navegadores web (*browsers*) permiten navegar por la web.

XHTML *eXtensible HyperText Markup Language*
HTML escrito según las normas que marca **XML**. Por tanto, se trata de una aplicación concreta de **XML** y no tienen que confundirse entre sí.

XML *Extensible Markup Language*
Metalenguaje de etiquetado basado en **SGML**. Diseñado específicamente para la **WWW** por **W3C**. Permite que un usuario diseñe sus propias etiquetas, con sus atributos y las reglas de construcción de documentos (sintaxis).

Capítulo 1

Introducción

En este capítulo se realiza una introducción del libro y se presenta el contenido de cada uno de los capítulos. Además, también se comentan las convenciones tipográficas empleadas para distinguir los acrónimos, nombres de programas, etc.

1.1. Introducción

Desde el curso 2001-2002, se imparte la asignatura "Programación en Internet" en la Universidad de Alicante. Esta asignatura pertenece al plan de estudios 2001 de las titulaciones de Ingeniería en Informática, Ingeniería Técnica en Informática de Sistemas e Ingeniería Técnica en Informática de Gestión de la Universidad de Alicante. En la Universidad de Alicante, esta asignatura la imparte el Departamento de Lenguajes y Sistemas Informáticos, adscrito a la Escuela Politécnica Superior de dicha Universidad.

Este libro trata de satisfacer el deseo de los alumnos de "Programación en Internet" de consultar las pruebas de evaluación de cursos anteriores, que emplean como medio de preparación a las pruebas de evaluación de conocimientos a las que se tendrán que enfrentar al final de la asignatura. Para ello, este libro contiene los tres exámenes de tipo test que se plantearon a lo largo del curso 2001-2002 (convocatorias de febrero, septiembre y diciembre). Además, se incluye un examen extra formado por preguntas de los otros tres exámenes. Cada uno de los exámenes incluye su correspondiente solución, que además incluye una explicación de las preguntas y respuestas.

1.2. Descripción de la asignatura "Programación en Internet"

La asignatura "Programación en Internet" es optativa y de duración cuatrimestral, con una carga docente de 6 créditos, repartidos entre 3 de teoría y 3 de prácticas. La descripción oficial de la asignatura, publicada en el Boletín Oficial del Estado número 230 de 25 de septiembre de 2001 es:

- Desarrollo y programación de sistemas de acceso a bases de datos de Internet.
- Planificación, diseño y administración de sitios Web.
- Migración de aplicaciones a entornos en Internet.
- Herramientas de desarrollo.
- Diseño y programación de elementos multimedia en Internet.

La asignatura "Programación en Internet" no posee prerrequisitos, pero sí las siguientes recomendaciones oficiales (no es necesario haber aprobado las siguientes asignaturas, pero sí recomendable haberlas cursado previamente):

- Fundamentos de Programación I (1er curso).
- Fundamentos de Programación II (1er curso).

- Bases de Datos I (2º curso).

Los objetivos principales de "Programación en Internet" son:

- Que el alumno conozca las características principales de las tecnologías empleadas en el desarrollo de aplicaciones web.
- Que el alumno conozca la estructura y funcionamiento de una aplicación web.
- Que el alumno adquiera los conocimientos y habilidades necesarios para programar aplicaciones destinadas a ser usadas en entornos Internet.
- Que el alumno conozca los recursos específicos (hardware y software) necesarios para poner en producción aplicaciones web.

Además, no se trata sólo de aprender habilidades técnicas, sino de dotar de conocimientos de fondo para formar profesionales flexibles capaces de trabajar con cualquier tecnología con una curva de aprendizaje mínima. Las tecnologías que se emplean en Internet están en continua evolución, por lo que no tiene sentido especializarse en una única tecnología, sino ofrecer una visión más amplia del estado actual de las tecnologías de desarrollo en Internet.

1.3. Temario de la asignatura

La asignatura está dividida en tres módulos principales que tratan aspectos generales de la programación en Internet, la programación de la parte cliente y la programación de la parte servidor:

- Módulo I: Introducción a la programación en Internet
  - Presentación y objetivos de la asignatura
  - Historia de Internet
  - Arquitecturas cliente/servidor
  - ¿Qué es una aplicación web?
  - Estructura de un sitio web
- Módulo II: Programación de clientes web
  - HTML
  - Guía de estilo
  - Lenguajes de script
  - JavaScript

    - Modelo de objetos de documento

- Módulo III: Programación de servidores web
    - CGI
    - SSI
    - IDC
    - Conceptos comunes de las tecnologías de programación de servidor web: ASP, ColdFusion, JSP y PHP
    - Active Server Pages (ASP)
    - Java Server Pages (JSP)
    - Otras tecnologías: PHP, ColdFusion y Caché

1.4. Estructura del libro

Este libro se compone de 4 capítulos, un apéndice con bibliografía recomendada y varios índices (figuras, acrónimos, etc.) que facilitan la búsqueda de información.

En el Capítulo 2 (**Exámenes sin solución**), se presentan los exámenes sin solución empleados durante el curso 2001-2002, más un examen extra formado por preguntas de los otros tres.

En el Capítulo 3 (**Exámenes con solución**), aparecen los mismos exámenes presentados en el Capítulo 2, pero con la respuesta correcta indicada. Además, aparece una referencia a la explicación de la pregunta que aparece en el Capítulo 4.

En el Capítulo 4 (**Explicaciones**), se explica la solución de cada una de las preguntas de los exámenes presentados. Aquellas preguntas que tratan un tema similar aparecen relacionadas entre sí. Además, las explicaciones aparecen ordenadas según el tema que tratan. Las preguntas planteadas para cada tema son:

- Internet (11): 6, 10, 26, 29, 35, 51, 52, 53, 77, 78, 95.
- Arquitecturas cliente/servidor (1): 1.
- Aplicaciones web (7): 5, 9, 14, 42, 54, 55, 80, 91.
- HTML (24): 2, 19, 20, 28, 30, 32, 36, 38, 39, 41, 43, 45, 48, 56, 57, 58, 59, 60, 82, 84, 86, 89, 93, 96.
- JavaScript (10): 15, 21, 33, 49, 61, 62, 63, 64, 88, 98.
- VBScript (2): 24, 67.
- CGI (5): 12, 23, 40, 65, 90.

- SSI (5): 17, 44, 66, 81, 100.
- ASP (22): 3, 7, 13, 16, 18, 22, 25, 31, 37, 47, 50, 68, 69, 70, 71, 72, 76, 79, 83, 87, 92, 97.
- Java (3): 4, 34, 75.
- JSP (9): 8, 11, 27, 46, 73, 74, 85, 94, 99.

1.5. Convenciones tipográficas

Con el fin de mejorar la legibilidad del texto, distintas convenciones tipográficas se han empleado a lo largo de todo el libro.

Los ejemplos, que normalmente están completos y por tanto se pueden escribir y probar, aparecen destacados de la siguiente forma (el texto de los ejemplos emplea un `tipo de letra de paso fijo como Courier`):

```
<!DOCTYPE HTML PUBLIC ".//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
Cabecera de la página
</HEAD>
<BODY>
Cuerpo de la página
</BODY>
</HTML>
```

Los estilos empleados a lo largo del texto son:

- Los acrónimos y siglas que figuran en el índice de acrónimos aparecen siempre destacados en **negrita**. Ejemplo: **ASP**, **HTML**, **URL**, etc.
- Cuando un acrónimo aparece por primera vez, se muestra el nombre completo en *cursiva* y entre paréntesis y en **negrita** el acrónimo. Ejemplo: *Graphics Interchange Format* (**GIF**), *World Wide Web* (**WWW**), etc.
- Las palabras no escritas en castellano aparecen destacadas en *cursiva*. Ejemplo: *bookmarks*, *browser*, *plug-in*, etc.
- Cuando se hace referencia a un programa, el texto de los interfaces que se ven en pantalla aparece en **negrita**. Además, se emplea una flecha para indicar una secuencia de acciones o pulsaciones en un programa. Ejemplo: **Aceptar**, **Inicio** → **Programas** → **Accesorios**.

- Los nombres de las compañías se muestran con un tipo de letra de MAYÚSCULAS PEQUEÑAS. Ejemplo: MICROSOFT, NETSCAPE, etc.
- Los nombres de los programas se muestran con un tipo de letra sin palo (sans serif). Ejemplo: Microsoft Paint, Netscape Navigator, Opera, etc.
- Los lenguajes informáticos se muestran con un tipo de letra *inclinada*. Ejemplo: *C*, *Java*, *Perl*, etc.
- Las extensiones de los ficheros, las palabras clave de los lenguajes de programación y el código incluido dentro del texto se muestra con un `tipo de letra de paso fijo como Courier`. Ejemplo: `.html`, `<IMG>`, `var ciudad = "Elche"`, etc.

Capítulo 2

Exámenes sin solución

En este capítulo se presenta una serie de exámenes sin solución realizados en la asignatura "Programación en Internet" de la Universidad de Alicante durante el curso académico 2001-2002. El último examen no se ha empleado en ningún examen real y se basa en preguntas de los otros tres exámenes.

Cada examen se compone de 25 preguntas. Las preguntas aparecen tal como aparecieron en los exámenes y no poseen ningún orden, ya que así es como se presentaron a los alumnos.

Cada pregunta posee dos numeraciones:

- La primera indica el número de orden de la pregunta dentro del examen. Para cada examen, esta numeración empieza desde 1.
- La segunda numeración es única para todos los exámenes e indica la posición de la pregunta en el conjunto de todas las preguntas.

Todas las preguntas poseen cuatro respuestas, de las cuales únicamente una es la correcta.

2.1. Febrero de 2002

1. (1) En las arquitecturas cliente/servidor:
   a) La parte cliente se conoce como front-end y la servidor como back-end.
   b) La parte cliente se conoce como back-end y la servidor como front-end.
   c) La parte cliente se conoce como display-end y la servidor como process-end.
   d) Las anteriores respuestas no son correctas.
2. (2) Respecto a XHTML, señala cuál de las siguientes afirmaciones es falsa:
   a) Las etiquetas y atributos tienen que escribirse siempre en minúsculas.
   b) El valor de los atributos tiene que ir entre comillas.
   c) No se admiten atributos sin valor.
   d) No existen etiquetas vacías.
3. (3) En la siguiente petición, "`pag.asp?id=123`", ¿cómo obtenemos el valor de "`id`" en la página ASP "`pag.asp`"?:
   a) `Request.Form("id")`.
   b) `Request.QueryString("id")`.
   c) `Request("id")`.
   d) La b) y la c) son correctas.
4. (4) De las siguientes afirmaciones sobre Java, cuál no es cierta:
   a) El propósito inicial fue el desarrollo de aplicaciones en Internet.
   b) Es un lenguaje multiplataforma.
   c) Es un lenguaje fuertemente tipado.
   d) Incorpora un recolector de basura.
5. (5) De las siguientes tecnologías, ¿cuál se emplea para programar un cliente web?:
   a) CGI.
   b) ASP.
   c) HTML.
   d) ColdFusion.
6. (6) El sistema que se emplea en Internet para traducir un nombre de dominio en dirección IP se llama:

a) Name Resolution System.

b) Direct Name System.

c) Domain System Name.

d) Domain Name System.

7. (7) Respecto a la petición "`file:///localhost/p.asp`" ejecutada en un ordenador que posee el servidor Microsoft Personal Web Server:

   a) El código ASP de la página se ejecuta si el servidor web está iniciado.

   b) El código ASP de la página no se ejecuta.

   c) El código ASP de la página no se ejecuta si el servidor web no está iniciado.

   d) La a) y la c) son correctas.

8. (8) Cuando un servicio web recibe una petición sobre una página JSP:

   a) Se interpreta el código de la página JSP y se devuelve una respuesta.

   b) Se ejecuta directamente el servlet correspondiente a la página JSP.

   c) Se busca la página JSP en la caché y se interpreta su código.

   d) Las anteriores respuestas no son correctas.

9. (9) De las siguientes tecnologías, ¿cuál no se emplea para programar un servidor web?:

   a) Servlets.

   b) Applets.

   c) SSI.

   d) ColdFusion.

10. (10) El protocolo HTTP fue inventado por:

    a) Tim Berners-Lee.

    b) Marc Andresseen.

    c) Linus Torvalds.

    d) Steve Jobs.

11. (11) Respecto a la programación con JSP, señala cuál no es cierta:

    a) Las expresiones son simples volcados de datos a la página.

    b) Las acciones únicamente tienen sintaxis basada en XML.

    c) Es necesario hacer una operación de conversión (cast) cuando en una expresión se devuelvan datos que no sean de tipo `String`.

d) Las directivas tienen como propósito enviar instrucciones al motor de JSP.

12. (12) Un programa CGI:

    a) Sólo puede recibir información por la entrada estándar.
    b) Se tiene que programar en un lenguaje compilado.
    c) Mantiene automáticamente el estado (sesión) entre una conexión y otra.
    d) Las anteriores respuestas no son correctas.

13. (13) El tiempo de vida de una sesión en ASP se puede configurar con:

    a) `Session.Timeout`.
    b) `Session.SetTimeout`.
    c) `Session.ScriptTimeout`.
    d) Las anteriores respuestas no son correctas.

14. (14) Una Intranet es:

    a) Una red global que conecta millones de ordenadores por todo el mundo.
    b) Una red privada basada en TCP/IP.
    c) Una red pública basada en TCP/IP.
    d) Las anteriores respuestas no son correctas.

15. (15) JavaScript es un lenguaje de programación:

    a) Orientado a objetos.
    b) Basado en objetos.
    c) Sin objetos.
    d) Las anteriores respuestas no son correctas.

16. (16) Desde una página ASP, para enviar información al cliente se emplea el objeto:

    a) `Request`.
    b) `Response`.
    c) `Session`.
    d) `Sender`.

17. (17) En SSI, para insertar en una página la fecha de la última modificación de un archivo se emplea:

    a) `<!-- #flastmod file="archivo" -->`.

b) `<!-- #flastmod virtual="archivo" -->`.

c) `<!-- #flastmod src="archivo" -->`.

d) La a) y la b) son correctas.

18. (18) Respecto al fichero `Global.asa`, señala cuál de las siguientes afirmaciones es falsa:

    a) En él se declaran los eventos de inicio y finalización de los objetos `Application` y `Session`.

    b) Es un fichero obligatorio en cualquier aplicación web realizada con ASP.

    c) No es accesible desde el cliente.

    d) El código incluido en él no puede escribir datos en la página devuelta.

19. (19) El esqueleto básico de una página HTML es:

    a) `<HTML>`, `<HEAD>`, `<BODY>`, `</BODY>`, `</HTML>`.

    b) `<HTML>`, `<HEAD>`, `<BODY>`, `</HEAD>`, `</BODY>`.

    c) `<HTML>`, `<HEAD>`, `<BODY>`, `</BODY>`, `</HEAD>`.

    d) Las anteriores respuestas no son correctas.

20. (20) La etiqueta `<P> ... </P>` se emplea para:

    a) Marcar párrafos de texto.

    b) Destacar el texto con un formato especial.

    c) Aumentar la sangría del texto.

    d) Las anteriores respuestas no son correctas.

21. (21) En JavaScript, si queremos mostrar un mensaje de alerta, usaremos:

    a) `window.alert("mensaje")`.

    b) `window.alert = "mensaje"`.

    c) `document.alert("mensaje")`.

    d) `windows.alert("mensaje")`.

22. (22) Desde una página ASP se puede acceder:

    a) Al sistema de archivos en el servidor.

    b) Al sistema de archivos en el cliente.

    c) A ambos sistemas de archivos.

    d) Las anteriores respuestas no son correctas.

23. (23) Para obtener la información que un cliente web envía con una URL al servidor web, un programa CGI:

    a) Consulta la variable de entorno `QUERY_STRING`.

    b) Consulta la variable de entorno `CONTENT`.

    c) Consulta la variable de entorno `PATH_INFO`.

    d) La a) y la c) son correctas.

24. (24) En VBScript, a la hora de concatenar cadenas, el mejor operador es:

    a) "`+`".

    b) "`&`".

    c) "`.`".

    d) No existe un operador, se tiene que usar una función.

25. (25) Cuando se programa una página ASP:

    a) VBScript es el único lenguaje que se puede emplear.

    b) JavaScript es el único lenguaje que se puede emplear.

    c) En la instalación por defecto (estándar) se pueden emplear tanto VBScript como JavaScript.

    d) Las anteriores respuestas no son correctas.

2.2. Septiembre de 2002

1. (26) Cuando nos referimos a Internet y a la Web:
   a) Internet incluye a la Web.
   b) La Web incluye a Internet.
   c) Nos estamos refiriendo a lo mismo.
   d) Las anteriores respuestas no son correctas.
2. (27) Si deseamos finalizar o abortar la sesión de un usuario en JSP se emplea:
   a) `session.invalidate`.
   b) `session.abandon`.
   c) `server.invalidate`.
   d) `response.abandon`.
3. (28) La estructura básica de una página web es:
   a) `<Html><Head> ... </Head><Body> ... </Body></Html>`.
   b) `<html><head> ... </head><body> ... </body></html>`.
   c) `<HTML><HEAD> ... </HEAD><BODY> ... </BODY></HTML>`.
   d) Todas las respuestas son correctas.
4. (29) El protocolo HTTP fue inventado por:
   a) Larry Roberts.
   b) Marc Andresseen.
   c) Vinton Cerf.
   d) Las anteriores respuestas no son correctas.
5. (30) Para insertar un salto de línea en una página HTML se emplea la etiqueta:
   a) `<BR>`.
   b) ` `.
   c) `<BK>`.
   d) Las anteriores respuestas no son correctas.
6. (31) Para emplear variables globales a todos los usuarios, en ASP se utiliza:
   a) El objeto `Session`.
   b) El objeto `Application`.

c) El objeto `Server`.

d) La a) y la b) son correctas.

7. (32) ¿Cuáles de las siguientes etiquetas se emplean para crear tablas?

   a) `<table>`, `<tr>` y `<tt>`.

   b) `<table>`, `<tf>` y `<td>`.

   c) `<tr>`, `<td>` y `<th>`.

   d) `<tr>`, `<td>` y `<tt>`.

8. (33) En JavaScript, las cadenas literales (constantes) se escriben:

   a) Como secuencias de caracteres encerrados entre comillas dobles (`" ... "`).

   b) Como secuencias de caracteres encerrados entre llaves (`{ ... }`).

   c) Como secuencias de caracteres encerrados entre comillas simples (`' ... '`).

   d) La a) y la c) son correctas.

9. (34) De las siguientes afirmaciones sobre Java, cuál no es cierta:

   a) El propósito inicial fue el desarrollo de aplicaciones en Internet.

   b) Es un lenguaje multiplataforma.

   c) Es un lenguaje fuertemente tipado.

   d) Incorpora un recolector de basura.

10. (35) Cuál de los siguientes es el origen de Internet:

    a) ARCNET.

    b) ETHERNET.

    c) MILNET.

    d) ARPANET.

11. (36) Para insertar una imagen en una página web se emplea la etiqueta:

    a) `<IMG> ... </IMG>`.

    b) `<A> ... </A>`.

    c) `<IMAGE> ... </IMAGE>`.

    d) Las anteriores respuestas no son correctas.

12. (37) ¿Qué significa ASP?:

    a) Active Standard Pages.

b) Active Server Pages.

c) Active Software Pages.

d) Las anteriores respuestas no son correctas.

13. (38) ¿Cómo se indicaría en una página web que la función de un enlace es enviar un correo electrónico? En la dirección del enlace se escribiría:

    a) La dirección de correo:
    `xxx@alu.ua.es`

    b) La palabra `mailto:` seguida de la dirección de correo:
    `mailto:xxx@alu.ua.es`

    c) La dirección de correo seguida del texto que se quiere enviar:
    `xxx@alu.ua.es "Este es el correo que te envío"`

    d) Todas las respuestas son correctas.

14. (39) Los formatos de imágenes que aceptan la mayoría de los navegadores son:

    a) GIF, JPG y BMP.

    b) GIF, TIFF y PNG.

    c) GIF, JPG y MPG.

    d) Las anteriores respuestas no son correctas.

15. (40) Cuando un cliente web envía la información de un formulario al servidor:

    a) Las distintas parejas `campo=valor` se separan por espacios en blanco.

    b) Las distintas parejas `campo=valor` se separan por "`&`".

    c) Las distintas parejas `campo=valor` se separan por "`+`".

    d) Las distintas parejas `campo=valor` se separan por "`%`".

16. (41) Si en una tabla se modifica el color de fondo de la tabla y el color de fondo de una celda:

    a) El color de fondo de la tabla domina sobre el color de fondo de la celda.

    b) El color de fondo de la celda domina sobre el color de fondo de la tabla.

    c) El color de fondo de una celda no se puede definir.

    d) Las anteriores respuestas no son correctas.

17. (42) De las siguientes tecnologías, ¿cuál se emplea para programar un cliente web?:

    a) CGI.

b) ASP.

c) HTML.

d) ColdFusion.

18. (43) La etiqueta `<TD> ... </TD>` define:

    a) Una fila de una tabla.

    b) Una celda de una tabla.

    c) Una celda que es encabezado de una tabla.

    d) Las anteriores respuestas no son correctas.

19. (44) En SSI, para insertar el valor de una variable de entorno en una página se emplea:

    a) `<!-- #echo var="variable" -->`.

    b) `<!-- #print var="variable" -->`.

    c) `<!-- #getenv var="variable" -->`.

    d) Las anteriores respuestas no son correctas.

20. (45) Para cambiar el color de fondo de una página HTML se emplea:

    a) El atributo `COLOR` de la etiqueta `<BODY> ... </BODY>`.

    b) El atributo `BGCOLOR` de la etiqueta `<BODY> ... </BODY>`.

    c) El atributo `COLOR` de la etiqueta `<HTML> ... </HTML>`.

    d) El atributo `BGCOLOR` de la etiqueta `<HTML> ... </HTML>`.

21. (46) Si deseamos desde una página JSP redirigir al cliente a una nueva URL, usaremos:

    a) `session.redirect()`.

    b) `request.redirect()`.

    c) `response.redirect()`.

    d) `response.sendRedirect()`.

22. (47) Para finalizar la ejecución de una página ASP se emplea:

    a) `Response.Clear`.

    b) `Response.End`.

    c) `Response.Stop`.

    d) Las anteriores respuestas no son correctas.

23. (48) Cuál de los siguientes enlaces es correcto:

    a) `<a href="http://www.ua.es/indice.html">UA</a>`.

    b) `<a url="http://www.ua.es/indice.html">UA</a>`.

    c) `<a name="http://www.ua.es/indice.html">UA</a>`.

    d) Las anteriores respuestas no son correctas.

24. (49) En JavaScript, ¿cómo se escribe una sentencia condicional para comprobar que la variable "`i`" es igual a "`5`"?:

    a) `if (i = 5)`.

    b) `if (i == 5)`.

    c) `if i == 5 then`.

    d) `if i = 5 then`.

25. (50) En ASP, cuál de las siguientes afirmaciones es cierta:

    a) Todos los usuarios comparten el mismo objeto `Session` y `Application`.

    b) Cada usuario tiene su objeto `Session` y `Application`.

    c) Cada usuario tiene su objeto `Session`, pero todos los usuarios comparten el mismo objeto `Application`.

    d) Las anteriores respuestas no son correctas.

2.3. Diciembre de 2002

1. (51) Los orígenes de Internet se sitúan en:
   a) Las BBS.
   b) ARPANET.
   c) MILNET.
   d) FidoNet.
2. (52) Al decir TCP/IP, estamos hablando de:
   a) Un conjunto de protocolos.
   b) El sistema de direcciones de máquinas en Internet.
   c) Un protocolo para transmitir páginas web.
   d) Un sistema para conectarse a Internet con un módem.
3. (53) La estandarización de la Web es tarea de:
   a) Microsoft y sus compañías asociadas.
   b) Netscape y sus compañías asociadas
   c) W3C.
   d) IEEE.
4. (54) De las siguientes tecnologías, ¿cuál no se emplea para programar un cliente web?:
   a) HTML.
   b) SSI.
   c) JavaScript.
   d) VRML.
5. (55) Una Intranet es:
   a) Una red global que conecta millones de ordenadores por todo el mundo.
   b) Una red privada basada en TCP/IP.
   c) Una red pública basada en TCP/IP.
   d) Las anteriores respuestas no son correctas.
6. (56) La etiqueta `<OL>` define:
   a) Listas no ordenadas.

b) Listas ordenadas.

c) Elementos en una lista.

d) Esta etiqueta no existe.

7. (57) ¿Cómo se puede abrir un enlace en una ventana nueva "siempre"?:

   a) Hace falta emplear código de JavaScript.

   b) `<a href="url" target="_blank">`.

   c) `<a href="url" target="blank">`.

   d) `<a href="url" target="_top">`.

8. (58) Para alinear a la izquierda el contenido de una celda de una tabla se emplea:

   a) `<td alignleft>`.

   b) `<td align="left">`.

   c) `<td valign="left">`.

   d) Las anteriores respuestas no son correctas.

9. (59) Para colocar en un formulario un área de texto multilínea, se emplea la etiqueta:

   a) `<input type="text">`.

   b) `<input type="textarea">`.

   c) `<input type="multiline">`.

   d) Las anteriores respuestas no son correctas.

10. (60) El esqueleto básico de una página HTML es:

    a) `<HTML>`, `<HEAD>`, `<BODY>`, `</BODY>`, `</HTML>`.

    b) `<HTML>`, `<HEAD>`, `<BODY>`, `</HEAD>`, `</BODY>`.

    c) `<HTML>`, `<HEAD>`, `<BODY>`, `</BODY>`, `</HEAD>`.

    d) Las anteriores respuestas no son correctas.

11. (61) En JavaScript, a la hora de concatenar cadenas, el mejor operador es:

    a) "`+`".

    b) "`&`".

    c) "`.`".

    d) No existe un operador para concatenar cadenas, se tienen que emplear funciones del objeto `String`.

12. (62) En JavaScript, los índices para los caracteres de una cadena empiezan en:

    a) -1.

    b) 0.

    c) 1.

    d) 2, porque en la posición 1 se almacena la longitud de la cadena.

13. (63) Cuál es la forma correcta de hacer referencia un fichero externo que contiene código de JavaScript:

    a) `<script src="fichero.js">`.

    b) `<script href="fichero.js">`.

    c) `<script link="fichero.js">`.

    d) Las anteriores respuestas no son correctas.

14. (64) En JavaScript, ¿cómo se abre una ventana nueva del navegador?:

    a) `window.open("pagina.html", "nueva")`.

    b) `window.new("pagina.html", "nueva")`.

    c) `document.open("pagina.html", "nueva")`.

    d) `document.new("pagina.html", "nueva")`.

15. (65) Para obtener la información que un cliente web envía al servidor web, un programa CGI:

    a) Consulta la variable de entorno `REQUEST`.

    b) Consulta la variable de entorno `CONTENT`.

    c) Consulta la variable de entorno `QUERY`.

    d) Las anteriores respuestas no son correctas.

16. (66) En SSI, para insertar el tamaño de un archivo en una página se emplea:

    a) `<!-- #size file="archivo" -->`.

    b) `<!-- #exec file="archivo" -->`.

    c) `<!-- #include size="archivo" -->`.

    d) Las anteriores respuestas no son correctas.

17. (67) En VBScript, respecto a la declaración de variables:

    a) Siempre hay que declarar las variables.

    b) Se puede forzar la declaración con `Option Implicit`.

c) Se declaran con la palabra reservada `Dim`.

d) La b) y la c) son correctas.

18. (68) ¿Qué hora mostrará la instrucción `<% Response.Write Time%>` en una página ASP?

    a) La hora en el servidor.

    b) La hora en el cliente.

    c) Depende de la directiva `<@ RUNAT="Server|Client" @>`.

    d) Depende del valor de `Request.Time`.

19. (69) Si deseamos desde una página ASP redirigir al cliente a una nueva URL, usaremos:

    a) `Session.Redirect`.

    b) `Request.Redirect`.

    c) `Server.Redirect`.

    d) `Response.Redirect`.

20. (70) Respecto al `Global.asa`, señala cuál de las siguientes afirmaciones es falsa:

    a) En él se declaran los eventos de inicio y finalización de los objetos `Application` y `Session`.

    b) Es un fichero obligatorio en cualquier aplicación web realizada con ASP.

    c) No es accesible desde el cliente.

    d) El código incluido en él no puede volcar datos en la página devuelta.

21. (71) Para acceder a las variables de entorno desde una página ASP se emplea:

    a) `Request.ServerVariables`.

    b) `Request.GetVariables`.

    c) `Application.ServerVariables`.

    d) `Application.GetVariables`.

22. (72) En ASP, ¿cómo se obtienen los datos enviados desde un formulario mediante el método `GET`?:

    a) Mediante `Request.QueryString`.

    b) Mediante `Request.Form`.

    c) Mediante `Request.GetForm`.

    d) Las anteriores respuestas no son correctas.

23. (73) Cuál de las siguientes afirmaciones no es cierta:

    a) JSP y servlet son tecnologías basadas en Java.

    b) Tanto JSP como servlet son tecnología de scripting (permiten integrar código de cliente y de servidor).

    c) JSP es una tecnología con más prestaciones que servlets.

    d) Tanto JSP como servlet permiten utilizar componentes de software (JavaBeans).

24. (74) Cuál de las siguientes sintaxis no se emplea en los comentarios de código de JSP:

    a) `' comentario`.

    b) `// comentario`.

    c) `/* comentario */`.

    d) `<%-- comentario --%>`.

25. (75) De las siguientes afirmaciones sobre Java, cuál es cierta:

    a) Ha sido desarrollado por W3C.

    b) Sólo se soporta en las siguiente plataformas: Windows, Linux y MacOS.

    c) Es un lenguaje basado en objetos.

    d) Es un lenguaje fuertemente tipado.

2.4. Examen extra

1. (76) ¿Qué significa ASP?:
   a) Active Standard Pages.
   b) Active Server Pages.
   c) Active Software Pages.
   d) Las anteriores respuestas no son correctas.
2. (77) Los orígenes de Internet se sitúan en:
   a) Las BBS.
   b) ARPANET.
   c) MILNET.
   d) FidoNet.
3. (78) El protocolo HTTP fue inventado por:
   a) Tim Berners-Lee.
   b) Marc Andresseen.
   c) Linus Torvalds.
   d) Steve Jobs.
4. (79) Si deseamos desde una página ASP redirigir al cliente a una nueva URL, usaremos:
   a) `Session.Redirect`.
   b) `Request.Redirect`.
   c) `Server.Redirect`.
   d) `Response.Redirect`.
5. (80) De las siguientes tecnologías, ¿cuál no se emplea para programar un servidor web?:
   a) Servlets.
   b) Applets.
   c) SSI.
   d) ColdFusion.
6. (81) En SSI, para insertar el tamaño de un archivo en una página se emplea:
   a) `<!-- #size file="archivo" -->`.

b) `<!-- #exec file="archivo" -->`.

c) `<!-- #include size="archivo" -->`.

d) Las anteriores respuestas no son correctas.

7. (82) Respecto a XHTML, señala cuál de las siguientes afirmaciones es falsa:

   a) Las etiquetas y atributos tienen que escribirse siempre en minúsculas.

   b) El valor de los atributos tiene que ir entre comillas.

   c) No se admiten atributos sin valor.

   d) No existen etiquetas vacías.

8. (83) Para emplear variables globales a todos los usuarios, en ASP se utiliza:

   a) El objeto `Session`.

   b) El objeto `Application`.

   c) El objeto `Server`.

   d) La a) y la b) son correctas.

9. (84) La etiqueta `<P> ... </P>` se emplea para:

   a) Marcar párrafos de texto.

   b) Destacar el texto con un formato especial.

   c) Aumentar la sangría del texto.

   d) Las anteriores respuestas no son correctas.

10. (85) Si deseamos finalizar o abortar la sesión de un usuario en JSP se emplea:

    a) `session.invalidate`.

    b) `session.abandon`.

    c) `server.invalidate`.

    d) `response.abandon`.

11. (86) Para insertar un salto de línea en una página HTML se emplea la etiqueta:

    a) `<BR>`.

    b) ` `.

    c) `<BK>`.

    d) Las anteriores respuestas no son correctas.

12. (87) Para acceder a las variables de entorno desde una página ASP se emplea:

a) `Request.ServerVariables`.

b) `Request.GetVariables`.

c) `Application.ServerVariables`.

d) `Application.GetVariables`.

13. (88) En JavaScript, si queremos mostrar un mensaje de alerta, usaremos:

    a) `window.alert("mensaje")`.

    b) `window.alert = "mensaje"`.

    c) `document.alert("mensaje")`.

    d) `windows.alert("mensaje")`.

14. (89) Para alinear a la izquierda el contenido de una celda de una tabla se emplea:

    a) `<td alignleft>`.

    b) `<td align="left">`.

    c) `<td valign="left">`.

    d) Las anteriores respuestas no son correctas.

15. (90) Un programa CGI:

    a) Sólo puede recibir información por la entrada estándar.

    b) Se tiene que programar en un lenguaje compilado.

    c) Mantiene automáticamente el estado (sesión) entre una conexión y otra.

    d) Las anteriores respuestas no son correctas.

16. (91) Una Intranet es:

    a) Una red global que conecta millones de ordenadores por todo el mundo.

    b) Una red privada basada en TCP/IP.

    c) Una red pública basada en TCP/IP.

    d) Las anteriores respuestas no son correctas.

17. (92) En la siguiente petición, "`pag.asp?id=123`", ¿cómo obtenemos el valor de "`id`" en la página ASP "`pag.asp`"?:

    a) `Request.Form("id")`.

    b) `Request.QueryString("id")`.

    c) `Request("id")`.

d) La b) y la c) son correctas.

18. (93) Cuál de los siguientes enlaces es correcto:

    a) `<a href="http://www.ua.es/indice.html">UA</a>`.

    b) `<a url="http://www.ua.es/indice.html">UA</a>`.

    c) `<a name="http://www.ua.es/indice.html">UA</a>`.

    d) Las anteriores respuestas no son correctas.

19. (94) Cuando un servicio web recibe una petición sobre una página JSP:

    a) Se interpreta el código de la página JSP y se devuelve una respuesta.

    b) Se ejecuta directamente el servlet correspondiente a la página JSP.

    c) Se busca la página JSP en la caché y se interpreta su código.

    d) Las anteriores respuestas no son correctas.

20. (95) Cuando nos referimos a Internet y a la Web:

    a) Internet incluye a la Web.

    b) La Web incluye a Internet.

    c) Nos estamos refiriendo a lo mismo.

    d) Las anteriores respuestas no son correctas.

21. (96) ¿Cómo se indicaría en una página web que la función de un enlace es enviar un correo electrónico? En la dirección del enlace se escribiría:

    a) La dirección de correo:
    `xxx@alu.ua.es`

    b) La palabra `mailto:` seguida de la dirección de correo:
    `mailto:xxx@alu.ua.es`

    c) La dirección de correo seguida del texto que se quiere enviar:
    `xxx@alu.ua.es "Este es el correo que te envío"`

    d) Todas las respuestas son correctas.

22. (97) Respecto a la petición "`file:///localhost/p.asp`" ejecutada en un ordenador que posee el servidor Microsoft Personal Web Server:

    a) El código ASP de la página se ejecuta si el servidor web está iniciado.

    b) El código ASP de la página no se ejecuta.

    c) El código ASP de la página no se ejecuta si el servidor web no está iniciado.

    d) La a) y la c) son correctas.

23. (98) En JavaScript, las cadenas literales (constantes) se escriben:

    a) Como secuencias de caracteres encerrados entre comillas dobles (`" ... "`).

    b) Como secuencias de caracteres encerrados entre llaves (`{ ... }`).

    c) Como secuencias de caracteres encerrados entre comillas simples (`' ... '`).

    d) La a) y la c) son correctas.

24. (99) Cuál de las siguientes sintaxis no se emplea en los comentarios de código de JSP:

    a) `' comentario`.

    b) `// comentario`.

    c) `/* comentario */`.

    d) `<%-- comentario --%>`.

25. (100) En SSI, para insertar el valor de una variable de entorno en una página se emplea:

    a) `<!-- #echo var="variable" -->`.

    b) `<!-- #print var="variable" -->`.

    c) `<!-- #getenv var="variable" -->`.

    d) Las anteriores respuestas no son correctas.

Capítulo 3

Exámenes con solución

En este capítulo se presentan los mismos exámenes que los presentados en el Capítulo 2, pero con la correspondiente solución. Además, cada pregunta contiene una referencia a la explicación que indica cuál es la respuesta correcta y por qué. Las explicaciones se presentan en el siguiente capítulo.

3.1. Febrero de 2002

1. (1) En las arquitecturas cliente/servidor:
   a) **(✓) La parte cliente se conoce como front-end y la servidor como back-end**.
   b) La parte cliente se conoce como back-end y la servidor como front-end.
   c) La parte cliente se conoce como display-end y la servidor como process-end.
   d) Las anteriores respuestas no son correctas.
   (Explicación 9, página 69)

2. (2) Respecto a XHTML, señala cuál de las siguientes afirmaciones es falsa:
   a) Las etiquetas y atributos tienen que escribirse siempre en minúsculas.
   b) El valor de los atributos tiene que ir entre comillas.
   c) No se admiten atributos sin valor.
   d) **(✓) No existen etiquetas vacías**.
   (Explicación 14, página 72)

3. (3) En la siguiente petición, "`pag.asp?id=123`", ¿cómo obtenemos el valor de "`id`" en la página ASP "`pag.asp`"?:
   a) `Request.Form("id")`.
   b) `Request.QueryString("id")`.
   c) `Request("id")`.
   d) **(✓) La b) y la c) son correctas**.
   (Explicación 47, página 96)

4. (4) De las siguientes afirmaciones sobre Java, cuál no es cierta:
   a) **(✓) El propósito inicial fue el desarrollo de aplicaciones en Internet**.
   b) Es un lenguaje multiplataforma.
   c) Es un lenguaje fuertemente tipado.
   d) Incorpora un recolector de basura.
   (Explicación 62, página 105)

5. (5) De las siguientes tecnologías, ¿cuál se emplea para programar un cliente web?:
   a) CGI.
   b) ASP.

c) **(✓) HTML**.

d) ColdFusion.
(Explicación 10, página 69)

6. (6) El sistema que se emplea en Internet para traducir un nombre de dominio en dirección IP se llama:

   a) Name Resolution System.

   b) Direct Name System.

   c) Domain System Name.

   d) **(✓) Domain Name System**.
   (Explicación 1, página 63)

7. (7) Respecto a la petición "`file:///localhost/p.asp`" ejecutada en un ordenador que posee el servidor Microsoft Personal Web Server:

   a) El código ASP de la página se ejecuta si el servidor web está iniciado.

   b) **(✓) El código ASP de la página no se ejecuta**.

   c) El código ASP de la página no se ejecuta si el servidor web no está iniciado.

   d) La a) y la c) son correctas.
   (Explicación 48, página 97)

8. (8) Cuando un servicio web recibe una petición sobre una página JSP:

   a) Se interpreta el código de la página JSP y se devuelve una respuesta.

   b) Se ejecuta directamente el servlet correspondiente a la página JSP.

   c) Se busca la página JSP en la caché y se interpreta su código.

   d) **(✓) Las anteriores respuestas no son correctas**.
   (Explicación 64, página 106)

9. (9) De las siguientes tecnologías, ¿cuál no se emplea para programar un servidor web?:

   a) Servlets.

   b) **(✓) Applets**.

   c) SSI.

   d) ColdFusion.
   (Explicación 11, página 70)

10. (10) El protocolo HTTP fue inventado por:

    a) **(✓) Tim Berners-Lee**.

b) Marc Andresseen.

c) Linus Torvalds.

d) Steve Jobs.
(Explicación 2, página 63)

11. (11) Respecto a la programación con JSP, señala cuál no es cierta:

a) Las expresiones son simples volcados de datos a la página.

b) Las acciones únicamente tienen sintaxis basada en XML.

c) **(✓) Es necesario hacer una operación de conversión (cast) cuando en una expresión se devuelvan datos que no sean de tipo** `String`.

d) Las directivas tienen como propósito enviar instrucciones al motor de JSP.
(Explicación 65, página 106)

12. (12) Un programa CGI:

a) Sólo puede recibir información por la entrada estándar.

b) Se tiene que programar en un lenguaje compilado.

c) Mantiene automáticamente el estado (sesión) entre una conexión y otra.

d) **(✓) Las anteriores respuestas no son correctas**.
(Explicación 40, página 90)

13. (13) El tiempo de vida de una sesión en ASP se puede configurar con:

a) **(✓)** `Session.Timeout`.

b) `Session.SetTimeout`.

c) `Session.ScriptTimeout`.

d) Las anteriores respuestas no son correctas.
(Explicación 49, página 97)

14. (14) Una Intranet es:

a) Una red global que conecta millones de ordenadores por todo el mundo.

b) **(✓) Una red privada basada en TCP/IP**.

c) Una red pública basada en TCP/IP.

d) Las anteriores respuestas no son correctas.
(Explicación 12, página 71)

15. (15) JavaScript es un lenguaje de programación:

a) Orientado a objetos.

b) **(✓) Basado en objetos**.

c) Sin objetos.

d) Las anteriores respuestas no son correctas.
(Explicación 30, página 83)

16. (16) Desde una página ASP, para enviar información al cliente se emplea el objeto:

a) `Request`.

b) **(✓)** `Response`.

c) `Session`.

d) `Sender`.
(Explicación 50, página 98)

17. (17) En SSI, para insertar en una página la fecha de la última modificación de un archivo se emplea:

a) `<!-- #flastmod file="archivo" -->`.

b) `<!-- #flastmod virtual="archivo" -->`.

c) `<!-- #flastmod src="archivo" -->`.

d) **(✓) La a) y la b) son correctas**.
(Explicación 44, página 93)

18. (18) Respecto al fichero `Global.asa`, señala cuál de las siguientes afirmaciones es falsa:

a) En él se declaran los eventos de inicio y finalización de los objetos `Application` y `Session`.

b) **(✓) Es un fichero obligatorio en cualquier aplicación web realizada con ASP**.

c) No es accesible desde el cliente.

d) El código incluido en él no puede escribir datos en la página devuelta.
(Explicación 51, página 99)

19. (19) El esqueleto básico de una página HTML es:

a) `<HTML>`, `<HEAD>`, `<BODY>`, `</BODY>`, `</HTML>`.

b) `<HTML>`, `<HEAD>`, `<BODY>`, `</HEAD>`, `</BODY>`.

c) `<HTML>`, `<HEAD>`, `<BODY>`, `</BODY>`, `</HEAD>`.

d) **(✓) Las anteriores respuestas no son correctas**.
(Explicación 15, página 72)

20. (20) La etiqueta `<P> ... </P>` se emplea para:

    a) **(✓) Marcar párrafos de texto**.

    b) Destacar el texto con un formato especial.

    c) Aumentar la sangría del texto.

    d) Las anteriores respuestas no son correctas.
    (Explicación 16, página 73)

21. (21) En JavaScript, si queremos mostrar un mensaje de alerta, usaremos:

    a) **(✓)** `window.alert("mensaje")`.

    b) `window.alert = "mensaje"`.

    c) `document.alert("mensaje")`.

    d) `windows.alert("mensaje")`.
    (Explicación 31, página 83)

22. (22) Desde una página ASP se puede acceder:

    a) **(✓) Al sistema de archivos en el servidor**.

    b) Al sistema de archivos en el cliente.

    c) A ambos sistemas de archivos.

    d) Las anteriores respuestas no son correctas.
    (Explicación 52, página 99)

23. (23) Para obtener la información que un cliente web envía con una URL al servidor web, un programa CGI:

    a) Consulta la variable de entorno `QUERY_STRING`.

    b) Consulta la variable de entorno `CONTENT`.

    c) Consulta la variable de entorno `PATH_INFO`.

    d) **(✓) La a) y la c) son correctas**.
    (Explicación 41, página 90)

24. (24) En VBScript, a la hora de concatenar cadenas, el mejor operador es:

    a) "+".

    b) **(✓) "&"**.

    c) ".".

    d) No existe un operador, se tiene que usar una función.
    (Explicación 38, página 89)

25. (25) Cuando se programa una página ASP:

   a) VBScript es el único lenguaje que se puede emplear.

   b) JavaScript es el único lenguaje que se puede emplear.

   c) **(✓) En la instalación por defecto (estándar) se pueden emplear tanto VBScript como JavaScript**.

   d) Las anteriores respuestas no son correctas.
   (Explicación 53, página 101)

3.2. Septiembre de 2002

1. (26) Cuando nos referimos a Internet y a la Web:
   a) (✓) **Internet incluye a la Web**.
   b) La Web incluye a Internet.
   c) Nos estamos refiriendo a lo mismo.
   d) Las anteriores respuestas no son correctas.
   (Explicación 3, página 64)

2. (27) Si deseamos finalizar o abortar la sesión de un usuario en JSP se emplea:
   a) (✓) `session.invalidate`.
   b) `session.abandon`.
   c) `server.invalidate`.
   d) `response.abandon`.
   (Explicación 66, página 109)

3. (28) La estructura básica de una página web es:
   a) `<Html><Head> ... </Head><Body> ... </Body></Html>`.
   b) `<html><head> ... </head><body> ... </body></html>`.
   c) `<HTML><HEAD> ... </HEAD><BODY> ... </BODY></HTML>`.
   d) (✓) **Todas las respuestas son correctas**.
   (Explicación 15, página 72)

4. (29) El protocolo HTTP fue inventado por:
   a) Larry Roberts.
   b) Marc Andresseen.
   c) Vinton Cerf.
   d) (✓) **Las anteriores respuestas no son correctas**.
   (Explicación 4, página 65)

5. (30) Para insertar un salto de línea en una página HTML se emplea la etiqueta:
   a) (✓) `<BR>`.
   b) ` `.
   c) `<BK>`.
   d) Las anteriores respuestas no son correctas.
   (Explicación 17, página 73)

6. (31) Para emplear variables globales a todos los usuarios, en ASP se utiliza:

   a) El objeto `Session`.

   b) **(✓) El objeto `Application`.**

   c) El objeto `Server`.

   d) La a) y la b) son correctas.
   (Explicación 54, página 101)

7. (32) ¿Cuáles de las siguientes etiquetas se emplean para crear tablas?

   a) `<table>`, `<tr>` y `<tt>`.

   b) `<table>`, `<tf>` y `<td>`.

   c) **(✓) `<tr>`, `<td>` y `<th>`.**

   d) `<tr>`, `<td>` y `<tt>`.
   (Explicación 18, página 73)

8. (33) En JavaScript, las cadenas literales (constantes) se escriben:

   a) Como secuencias de caracteres encerrados entre comillas dobles (`" ... "`).

   b) Como secuencias de caracteres encerrados entre llaves (`{ ... }`).

   c) Como secuencias de caracteres encerrados entre comillas simples (`' ... '`).

   d) **(✓) La a) y la c) son correctas.**
   (Explicación 32, página 84)

9. (34) De las siguientes afirmaciones sobre Java, cuál no es cierta:

   a) **(✓) El propósito inicial fue el desarrollo de aplicaciones en Internet.**

   b) Es un lenguaje multiplataforma.

   c) Es un lenguaje fuertemente tipado.

   d) Incorpora un recolector de basura.
   (Explicación 62, página 105)

10. (35) Cuál de los siguientes es el origen de Internet:

    a) ARCNET.

    b) ETHERNET.

    c) MILNET.

    d) **(✓) ARPANET.**
    (Explicación 5, página 65)

11. (36) Para insertar una imagen en una página web se emplea la etiqueta:

    a) `<IMG> ... </IMG>`.

    b) `<A> ... </A>`.

    c) `<IMAGE> ... </IMAGE>`.

    d) **(✓) Las anteriores respuestas no son correctas**.
    (Explicación 19, página 74)

12. (37) ¿Qué significa ASP?:

    a) Active Standard Pages.

    b) **(✓) Active Server Pages**.

    c) Active Software Pages.

    d) Las anteriores respuestas no son correctas.
    (Explicación 55, página 102)

13. (38) ¿Cómo se indicaría en una página web que la función de un enlace es enviar un correo electrónico? En la dirección del enlace se escribiría:

    a) La dirección de correo:
    `xxx@alu.ua.es`

    b) **(✓) La palabra `mailto:` seguida de la dirección de correo:**
    `mailto:xxx@alu.ua.es`

    c) La dirección de correo seguida del texto que se quiere enviar:
    `xxx@alu.ua.es "Este es el correo que te envío"`

    d) Todas las respuestas son correctas.
    (Explicación 20, página 74)

14. (39) Los formatos de imágenes que aceptan la mayoría de los navegadores son:

    a) GIF, JPG y BMP.

    b) GIF, TIFF y PNG.

    c) GIF, JPG y MPG.

    d) **(✓) Las anteriores respuestas no son correctas**.
    (Explicación 21, página 75)

15. (40) Cuando un cliente web envía la información de un formulario al servidor:

    a) Las distintas parejas `campo=valor` se separan por espacios en blanco.

    b) **(✓) Las distintas parejas `campo=valor` se separan por "`&`"**.

    c) Las distintas parejas `campo=valor` se separan por "+".

d) Las distintas parejas `campo=valor` se separan por "`%`".
(Explicación 42, página 90)

16. (41) Si en una tabla se modifica el color de fondo de la tabla y el color de fondo de una celda:

    a) El color de fondo de la tabla domina sobre el color de fondo de la celda.

    b) **(✓) El color de fondo de la celda domina sobre el color de fondo de la tabla**.

    c) El color de fondo de una celda no se puede definir.

    d) Las anteriores respuestas no son correctas.
    (Explicación 22, página 75)

17. (42) De las siguientes tecnologías, ¿cuál se emplea para programar un cliente web?:

    a) CGI.

    b) ASP.

    c) **(✓) HTML**.

    d) ColdFusion.
    (Explicación 10, página 69)

18. (43) La etiqueta `<TD> ... </TD>` define:

    a) Una fila de una tabla.

    b) **(✓) Una celda de una tabla**.

    c) Una celda que es encabezado de una tabla.

    d) Las anteriores respuestas no son correctas.
    (Explicación 23, página 76)

19. (44) En SSI, para insertar el valor de una variable de entorno en una página se emplea:

    a) **(✓)** `<!-- #echo var="variable" -->`.

    b) `<!-- #print var="variable" -->`.

    c) `<!-- #getenv var="variable" -->`.

    d) Las anteriores respuestas no son correctas.
    (Explicación 45, página 94)

20. (45) Para cambiar el color de fondo de una página HTML se emplea:

    a) El atributo `COLOR` de la etiqueta `<BODY> ... </BODY>`.

b) (✓) **El atributo** `BGCOLOR` **de la etiqueta** `<BODY> ... </BODY>`.

c) El atributo `COLOR` de la etiqueta `<HTML> ... </HTML>`.

d) El atributo `BGCOLOR` de la etiqueta `<HTML> ... </HTML>`.
(Explicación 24, página 76)

21. (46) Si deseamos desde una página JSP redirigir al cliente a una nueva URL, usaremos:

a) `session.redirect()`.

b) `request.redirect()`.

c) `response.redirect()`.

d) (✓) `response.sendRedirect()`.
(Explicación 67, página 109)

22. (47) Para finalizar la ejecución de una página ASP se emplea:

a) `Response.Clear`.

b) (✓) `Response.End`.

c) `Response.Stop`.

d) Las anteriores respuestas no son correctas.
(Explicación 56, página 102)

23. (48) Cuál de los siguientes enlaces es correcto:

a) (✓) `<a href="http://www.ua.es/indice.html">UA</a>`.

b) `<a url="http://www.ua.es/indice.html">UA</a>`.

c) `<a name="http://www.ua.es/indice.html">UA</a>`.

d) Las anteriores respuestas no son correctas.
(Explicación 25, página 76)

24. (49) En JavaScript, ¿cómo se escribe una sentencia condicional para comprobar que la variable "`i`" es igual a "`5`"?:

a) `if (i = 5)`.

b) (✓) `if (i == 5)`.

c) `if i == 5 then`.

d) `if i = 5 then`.
(Explicación 33, página 85)

25. (50) En ASP, cuál de las siguientes afirmaciones es cierta:

a) Todos los usuarios comparten el mismo objeto `Session` y `Application`.

b) Cada usuario tiene su objeto `Session` y `Application`.

c) **(✓) Cada usuario tiene su objeto `Session`, pero todos los usuarios comparten el mismo objeto `Application`.**

d) Las anteriores respuestas no son correctas.
(Explicación 57, página 103)

3.3. Diciembre de 2002

1. (51) Los orígenes de Internet se sitúan en:

   a) Las BBS.

   b) **(✓) ARPANET**.

   c) MILNET.

   d) FidoNet.
   (Explicación 6, página 66)

2. (52) Al decir TCP/IP, estamos hablando de:

   a) **(✓) Un conjunto de protocolos**.

   b) El sistema de direcciones de máquinas en Internet.

   c) Un protocolo para transmitir páginas web.

   d) Un sistema para conectarse a Internet con un módem.
   (Explicación 7, página 67)

3. (53) La estandarización de la Web es tarea de:

   a) Microsoft y sus compañías asociadas.

   b) Netscape y sus compañías asociadas

   c) **(✓) W3C**.

   d) IEEE.
   (Explicación 8, página 68)

4. (54) De las siguientes tecnologías, ¿cuál no se emplea para programar un cliente web?:

   a) HTML.

   b) **(✓) SSI**.

   c) JavaScript.

   d) VRML.
   (Explicación 13, página 71)

5. (55) Una Intranet es:

   a) Una red global que conecta millones de ordenadores por todo el mundo.

   b) **(✓) Una red privada basada en TCP/IP**.

   c) Una red pública basada en TCP/IP.

d) Las anteriores respuestas no son correctas.
(Explicación 12, página 71)

6. (56) La etiqueta `<OL>` define:

   a) Listas no ordenadas.

   b) (✓) **Listas ordenadas**.

   c) Elementos en una lista.

   d) Esta etiqueta no existe.
   (Explicación 26, página 77)

7. (57) ¿Cómo se puede abrir un enlace en una ventana nueva "siempre"?:

   a) Hace falta emplear código de JavaScript.

   b) (✓) `<a href="url" target="_blank">`.

   c) `<a href="url" target="blank">`.

   d) `<a href="url" target="_top">`.
   (Explicación 27, página 78)

8. (58) Para alinear a la izquierda el contenido de una celda de una tabla se emplea:

   a) `<td alignleft>`.

   b) (✓) `<td align="left">`.

   c) `<td valign="left">`.

   d) Las anteriores respuestas no son correctas.
   (Explicación 28, página 79)

9. (59) Para colocar en un formulario un área de texto multilínea, se emplea la etiqueta:

   a) `<input type="text">`.

   b) `<input type="textarea">`.

   c) `<input type="multiline">`.

   d) (✓) **Las anteriores respuestas no son correctas**.
   (Explicación 29, página 81)

10. (60) El esqueleto básico de una página HTML es:

    a) `<HTML>`, `<HEAD>`, `<BODY>`, `</BODY>`, `</HTML>`.

    b) `<HTML>`, `<HEAD>`, `<BODY>`, `</HEAD>`, `</BODY>`.

    c) `<HTML>`, `<HEAD>`, `<BODY>`, `</BODY>`, `</HEAD>`.

d) (✓) **Las anteriores respuestas no son correctas**.
(Explicación 15, página 72)

11. (61) En JavaScript, a la hora de concatenar cadenas, el mejor operador es:

    a) (✓) "+".

    b) "&".

    c) ".".

    d) No existe un operador para concatenar cadenas, se tienen que emplear funciones del objeto `String`.
    (Explicación 34, página 85)

12. (62) En JavaScript, los índices para los caracteres de una cadena empiezan en:

    a) -1.

    b) (✓) **0**.

    c) 1.

    d) 2, porque en la posición 1 se almacena la longitud de la cadena.
    (Explicación 35, página 86)

13. (63) Cuál es la forma correcta de hacer referencia un fichero externo que contiene código de JavaScript:

    a) (✓) `<script src="fichero.js">`.

    b) `<script href="fichero.js">`.

    c) `<script link="fichero.js">`.

    d) Las anteriores respuestas no son correctas.
    (Explicación 36, página 86)

14. (64) En JavaScript, ¿cómo se abre una ventana nueva del navegador?:

    a) (✓) `window.open("pagina.html", "nueva")`.

    b) `window.new("pagina.html", "nueva")`.

    c) `document.open("pagina.html", "nueva")`.

    d) `document.new("pagina.html", "nueva")`.
    (Explicación 37, página 87)

15. (65) Para obtener la información que un cliente web envía al servidor web, un programa CGI:

    a) Consulta la variable de entorno `REQUEST`.

    b) Consulta la variable de entorno `CONTENT`.

c) Consulta la variable de entorno `QUERY`.

d) **(✓) Las anteriores respuestas no son correctas**.
(Explicación 43, página 91)

16. (66) En SSI, para insertar el tamaño de un archivo en una página se emplea:

a) `<!-- #size file="archivo" -->`.

b) `<!-- #exec file="archivo" -->`.

c) `<!-- #include size="archivo" -->`.

d) **(✓) Las anteriores respuestas no son correctas**.
(Explicación 46, página 94)

17. (67) En VBScript, respecto a la declaración de variables:

a) Siempre hay que declarar las variables.

b) Se puede forzar la declaración con `Option Implicit`.

c) **(✓) Se declaran con la palabra reservada** `Dim`.

d) La b) y la c) son correctas.
(Explicación 39, página 89)

18. (68) ¿Qué hora mostrará la instrucción `<% Response.Write Time %>` en una página ASP?

a) **(✓) La hora en el servidor**.

b) La hora en el cliente.

c) Depende de la directiva `<@ RUNAT="Server|Client" @>`.

d) Depende del valor de `Request.Time`.
(Explicación 58, página 103)

19. (69) Si deseamos desde una página ASP redirigir al cliente a una nueva URL, usaremos:

a) `Session.Redirect`.

b) `Request.Redirect`.

c) `Server.Redirect`.

d) **(✓)** `Response.Redirect`.
(Explicación 59, página 104)

20. (70) Respecto al `Global.asa`, señala cuál de las siguientes afirmaciones es falsa:

a) En él se declaran los eventos de inicio y finalización de los objetos `Application` y `Session`.

b) **(✓) Es un fichero obligatorio en cualquier aplicación web realizada con ASP**.

c) No es accesible desde el cliente.

d) El código incluido en él no puede volcar datos en la página devuelta.
(Explicación 51, página 99)

21. (71) Para acceder a las variables de entorno desde una página ASP se emplea:

a) **(✓)** `Request.ServerVariables`.

b) `Request.GetVariables`.

c) `Application.ServerVariables`.

d) `Application.GetVariables`.
(Explicación 60, página 104)

22. (72) En ASP, ¿cómo se obtienen los datos enviados desde un formulario mediante el método `GET`?:

a) **(✓) Mediante** `Request.QueryString`.

b) Mediante `Request.Form`.

c) Mediante `Request.GetForm`.

d) Las anteriores respuestas no son correctas.
(Explicación 61, página 104)

23. (73) Cuál de las siguientes afirmaciones no es cierta:

a) JSP y servlet son tecnologías basadas en Java.

b) **(✓) Tanto JSP como servlet son tecnología de scripting (permiten integrar código de cliente y de servidor)**.

c) JSP es una tecnología con más prestaciones que servlets.

d) Tanto JSP como servlet permiten utilizar componentes de software (JavaBeans).
(Explicación 68, página 109)

24. (74) Cuál de las siguientes sintaxis no se emplea en los comentarios de código de JSP:

a) **(✓)** `' comentario`.

b) `// comentario`.

c) `/* comentario */`.

d) `<%-- comentario --%>`.
(Explicación 69, página 110)

25. (75) De las siguientes afirmaciones sobre Java, cuál es cierta:

    a) Ha sido desarrollado por W3C.

    b) Sólo se soporta en las siguiente plataformas: Windows, Linux y MacOS.

    c) Es un lenguaje basado en objetos.

    d) **(✓) Es un lenguaje fuertemente tipado**.
    (Explicación 63, página 105)

3.4. Examen extra

1. (76) ¿Qué significa ASP?:
   a) Active Standard Pages.
   b) **(✓) Active Server Pages**.
   c) Active Software Pages.
   d) Las anteriores respuestas no son correctas.
   (Explicación 55, página 102)

2. (77) Los orígenes de Internet se sitúan en:
   a) Las BBS.
   b) **(✓) ARPANET**.
   c) MILNET.
   d) FidoNet.
   (Explicación 6, página 66)

3. (78) El protocolo HTTP fue inventado por:
   a) **(✓) Tim Berners-Lee**.
   b) Marc Andresseen.
   c) Linus Torvalds.
   d) Steve Jobs.
   (Explicación 2, página 63)

4. (79) Si deseamos desde una página ASP redirigir al cliente a una nueva URL, usaremos:
   a) `Session.Redirect`.
   b) `Request.Redirect`.
   c) `Server.Redirect`.
   d) **(✓)** `Response.Redirect`.
   (Explicación 59, página 104)

5. (80) De las siguientes tecnologías, ¿cuál no se emplea para programar un servidor web?:
   a) Servlets.
   b) **(✓) Applets**.
   c) SSI.

d) ColdFusion.
(Explicación 11, página 70)

6. (81) En SSI, para insertar el tamaño de un archivo en una página se emplea:

   a) `<!-- #size file="archivo" -->`.

   b) `<!-- #exec file="archivo" -->`.

   c) `<!-- #include size="archivo" -->`.

   d) **(✓) Las anteriores respuestas no son correctas**.
   (Explicación 46, página 94)

7. (82) Respecto a XHTML, señala cuál de las siguientes afirmaciones es falsa:

   a) Las etiquetas y atributos tienen que escribirse siempre en minúsculas.

   b) El valor de los atributos tiene que ir entre comillas.

   c) No se admiten atributos sin valor.

   d) **(✓) No existen etiquetas vacías**.
   (Explicación 14, página 72)

8. (83) Para emplear variables globales a todos los usuarios, en ASP se utiliza:

   a) El objeto `Session`.

   b) **(✓) El objeto** `Application`.

   c) El objeto `Server`.

   d) La a) y la b) son correctas.
   (Explicación 54, página 101)

9. (84) La etiqueta `<P> ... </P>` se emplea para:

   a) **(✓) Marcar párrafos de texto**.

   b) Destacar el texto con un formato especial.

   c) Aumentar la sangría del texto.

   d) Las anteriores respuestas no son correctas.
   (Explicación 16, página 73)

10. (85) Si deseamos finalizar o abortar la sesión de un usuario en JSP se emplea:

    a) **(✓)** `session.invalidate`.

    b) `session.abandon`.

    c) `server.invalidate`.

    d) `response.abandon`.
    (Explicación 66, página 109)

11. (86) Para insertar un salto de línea en una página HTML se emplea la etiqueta:

    a) (✓) `<BR>`.

    b) ` `.

    c) `<BK>`.

    d) Las anteriores respuestas no son correctas.
    (Explicación 17, página 73)

12. (87) Para acceder a las variables de entorno desde una página ASP se emplea:

    a) (✓) `Request.ServerVariables`.

    b) `Request.GetVariables`.

    c) `Application.ServerVariables`.

    d) `Application.GetVariables`.
    (Explicación 60, página 104)

13. (88) En JavaScript, si queremos mostrar un mensaje de alerta, usaremos:

    a) (✓) `window.alert("mensaje")`.

    b) `window.alert = "mensaje"`.

    c) `document.alert("mensaje")`.

    d) `windows.alert("mensaje")`.
    (Explicación 31, página 83)

14. (89) Para alinear a la izquierda el contenido de una celda de una tabla se emplea:

    a) `<td alignleft>`.

    b) (✓) `<td align="left">`.

    c) `<td valign="left">`.

    d) Las anteriores respuestas no son correctas.
    (Explicación 28, página 79)

15. (90) Un programa CGI:

    a) Sólo puede recibir información por la entrada estándar.

    b) Se tiene que programar en un lenguaje compilado.

    c) Mantiene automáticamente el estado (sesión) entre una conexión y otra.

    d) (✓) **Las anteriores respuestas no son correctas**.
    (Explicación 40, página 90)

16. (91) Una Intranet es:

a) Una red global que conecta millones de ordenadores por todo el mundo.

b) **(✓) Una red privada basada en TCP/IP**.

c) Una red pública basada en TCP/IP.

d) Las anteriores respuestas no son correctas.
(Explicación 12, página 71)

17. (92) En la siguiente petición, "`pag.asp?id=123`", ¿cómo obtenemos el valor de "`id`" en la página ASP "`pag.asp`"?:

a) `Request.Form("id")`.

b) `Request.QueryString("id")`.

c) `Request("id")`.

d) **(✓) La b) y la c) son correctas**.
(Explicación 47, página 96)

18. (93) Cuál de los siguientes enlaces es correcto:

a) **(✓)** `<a href="http://www.ua.es/indice.html">UA</a>`.

b) `<a url="http://www.ua.es/indice.html">UA</a>`.

c) `<a name="http://www.ua.es/indice.html">UA</a>`.

d) Las anteriores respuestas no son correctas.
(Explicación 25, página 76)

19. (94) Cuando un servicio web recibe una petición sobre una página JSP:

a) Se interpreta el código de la página JSP y se devuelve una respuesta.

b) Se ejecuta directamente el servlet correspondiente a la página JSP.

c) Se busca la página JSP en la caché y se interpreta su código.

d) **(✓) Las anteriores respuestas no son correctas**.
(Explicación 64, página 106)

20. (95) Cuando nos referimos a Internet y a la Web:

a) **(✓) Internet incluye a la Web**.

b) La Web incluye a Internet.

c) Nos estamos refiriendo a lo mismo.

d) Las anteriores respuestas no son correctas.
(Explicación 3, página 64)

21. (96) ¿Cómo se indicaría en una página web que la función de un enlace es enviar un correo electrónico? En la dirección del enlace se escribiría:

a) La dirección de correo:
`xxx@alu.ua.es`

b) **(✓) La palabra `mailto:` seguida de la dirección de correo:**
`mailto:xxx@alu.ua.es`

c) La dirección de correo seguida del texto que se quiere enviar:
`xxx@alu.ua.es "Este es el correo que te envío"`

d) Todas las respuestas son correctas.
(Explicación 20, página 74)

22. (97) Respecto a la petición "`file:///localhost/p.asp`" ejecutada en un ordenador que posee el servidor Microsoft Personal Web Server:

a) El código ASP de la página se ejecuta si el servidor web está iniciado.

b) **(✓) El código ASP de la página no se ejecuta**.

c) El código ASP de la página no se ejecuta si el servidor web no está iniciado.

d) La a) y la c) son correctas.
(Explicación 48, página 97)

23. (98) En JavaScript, las cadenas literales (constantes) se escriben:

a) Como secuencias de caracteres encerrados entre comillas dobles (`" ... "`).

b) Como secuencias de caracteres encerrados entre llaves (`{ ... }`).

c) Como secuencias de caracteres encerrados entre comillas simples (`' ... '`).

d) **(✓) La a) y la c) son correctas**.
(Explicación 32, página 84)

24. (99) Cuál de las siguientes sintaxis no se emplea en los comentarios de código de JSP:

a) **(✓)** `' comentario`.

b) `// comentario`.

c) `/* comentario */`.

d) `<%-- comentario --%>`.
(Explicación 69, página 110)

25. (100) En SSI, para insertar el valor de una variable de entorno en una página se emplea:

a) **(✓)** `<!-- #echo var="variable" -->`.

b) `<!-- #print var="variable" -->`.

c) `<!-- #getenv var="variable" -->`.

d) Las anteriores respuestas no son correctas.
(Explicación 45, página 94)

Capítulo 4

Explicaciones

En este capítulo se presenta las explicaciones de las preguntas presentadas en los capítulos anteriores. Las explicaciones aparecen agrupadas por temas: Internet, Arquitecturas cliente/servidor, Aplicaciones web, **HTML**, *JavaScript*, *VBScript*, **CGI**, **SSI**, **ASP**, *Java* y **JSP**.

Cada explicación contiene una referencia a la correspondiente pregunta. También puede contener referencias a otras preguntas y explicaciones relacionadas.

4.1. Internet

4.1.1. Explicación 1

(Pregunta 6, página 11)

Domain Name System (**DNS**) es el sistema que se emplea en Internet para traducir un nombre de dominio (por ejemplo, `www.ua.es`) en dirección *Internet Protocol* (**IP**) (`193.170.202.20`).

Fue inventado en 1983 en la Universidad de Wisconsin. Antes de su uso, cada ordenador debía de poseer un fichero (llamado `hosts`) que almacenaba la lista de todos los ordenadores conectados a Internet. Para acceder a un determinado ordenador, bastaba con consultar este archivo para conocer su dirección IP. Por ejemplo:

```
127.0.0.1       localhost
192.168.0.1     servidor
192.168.0.2     impresora
```

Conforme el número de ordenadores conectados a Internet fue creciendo, el tamaño de este fichero también creció y su mantenimiento era cada vez más costoso. Para resolver este problema, se desarrollo el sistema **DNS**.

DNS es una base de datos distribuida con una estructura jerárquica. En esta base de datos se encuentran los nombres de dominio y las direcciones **IP** de todos los ordenadores conectados a Internet. Para emplear este sistema únicamente hace falta conocer la dirección **IP** de un servidor de **DNS**.

Todavía hoy existe el archivo `hosts` en muchos sistemas operativos, aunque ya prácticamente no se emplea. Por ejemplo, en Microsoft Windows XP, este archivo se encuentra en la siguiente ruta:
`\WINDOWS\system32\drivers\etc`

4.1.2. Explicación 2

(Pregunta 10, página 12) (Pregunta 78, página 29)

El protocolo *HyperText Transfer Protocol* (**HTTP**) fue inventado por Tim Berners-Lee.

Tim Berners-Lee, graduado en Física por la Universidad de Oxford, propuso en 1989 la creación de un sistema electrónico para compartir información a través de una red de ordenadores mientras trabajaba en el *Conseil Européenne pour le Recherche Nucléaire* (**CERN**). El objetivo de este sistema era resolver los problemas de comunicación, mantenimiento y actualización de información que existían en dicha institución. Partiendo de esta propuesta desarrolló a principios de los años noventa la Web. A él se deben los tres elementos que fueron clave en el nacimiento de la Web: *HyperText Markup Language* (**HTML**), como lenguaje para crear los contenidos de la Web; **HTTP**, como protocolo de comunicación entre los ordenadores de la Web;

y *Universal Resource Locator* (**URL**) como medio de localización (direccionamiento) de los distintos recursos en Internet. El 25 de octubre de 2002 le fue entregado el Premio Príncipe de Asturias de Investigación Científica y Técnica, junto con Lawrence Roberts, Robert Kahn y Vinton Cerf, en representación de las "miles de personas y muchas instituciones que han hecho posible este avance de nuestro tiempo". En la actualidad, es director del *World Wide Web Consortium* (**W3C**) y trabaja en el Laboratorio de Ciencias Informáticas del famoso *Massachusetts Institute of Technology* (**MIT**).

Marc Andresseen, del *National Center for Supercomputing Applications* (**NCSA**) de la Universidad de Illinois, desarrolló junto con Eric Bina Mosaic for X en 1992, un navegador gráfico de la Web para Unix. En 1993, él y sus compañeros desarrollaron sendas versiones de Mosaic para los sistemas operativos Apple MacOS y Microsoft Windows. En diciembre de 1993, Marc Andresseen y sus compañeros abandonaron **NCSA** para crear su propia empresa, Mosaic Communications Corporation, que más tarde pasó a llamarse Netscape Communications Corporation.

Linus Torvalds es el creador de Linux, un sistema operativo basado en Unix de libre distribución que está disponible para distintas plataformas de hardware (Intel, Motorola, Sun Microsystems, etc.). La versión 0.01 de Linux se publicó en septiembre de 1991. Distribuido mediante la licencia de *GNU is Not Unix* (**GNU**), el código fuente de este sistema operativo es gratuito, lo que permite su copia, estudio y modificación. En la actualidad, Linus trabaja en Transmeta Corporation, fabricante del procesador Crusoe.

Steve Jobs es el actual jefe ejecutivo de Apple. En 1976 fundó junto con Steve Wozniak la empresa Apple Computer Corporation. Ambos revolucionaron el mundo de la informática personal, primero con los modelos Apple I y II a finales de los años 70 y más tarde con el modelo Macintosh en 1984. En 1985 dimitió de Apple. En 1986 fundó Pixar, creadora de las películas de animación "Toy Story" y "Bichos". En 1989 fundó NeXT Corporation, con la idea de fabricar un nuevo tipo de ordenador personal con su propio sistema operativo. En 1993 NeXT Corporation dejó de fabricar ordenadores, aunque continuó con el desarrollo de su sistema operativo. Finalmente, en 1996 Apple compró NeXT Corporation y Steve Jobs volvió a Apple.

Preguntas relacionadas: 29

Explicaciones relacionadas: 4

4.1.3. Explicación 3

(Pregunta 26, página 17) (Pregunta 95, página 32)

Internet es un conjunto de diferentes redes conectadas entre sí, que equivale a una red global que conecta millones de ordenadores por todo el mundo. El éxito de Internet se debe en gran medida al uso, por todos los ordenadores que se conectan a ella, de un protocolo común: TCP/IP. Dentro de esta red se ejecutan una serie de servicios:

web, correo electrónico, transmisión de ficheros, etc. Al conjunto de ordenadores de Internet que ofrecen el servicio web se les denomina "la Web".

La *World Wide Web* (**WWW**), conocida de forma abreviada como la Web, es una red de ordenadores que ofrecen una serie de documentos (las páginas web) que contienen texto, imágenes y otros recursos multimedia. El protocolo de comunicación es **HTTP**, el lenguaje de creación de documentos es **HTML** y el sistema de direccionamiento y localización de los recursos es **URL**.

Por tanto, Internet incluye a la Web, pero no al revés.

4.1.4. Explicación 4

(Pregunta 29, página 17)

El protocolo **HTTP** fue desarrollado por Tim Berners-Lee.

Lawrence (Larry) Roberts es considerado uno de los "padres de Internet". En 1996 abandonó el **MIT** para incorporarse a *Advanced Research Projects Agency* (**ARPA**), donde dirigió el desarrollo de ARPANET, que posteriormente se convirtió en la actual Internet. En 1976 también participó en el desarrollo de X25, una red de conmutación de paquetes.

Marc Andresseen, del **NCSA** de la Universidad de Illinois, desarrolló junto con Eric Bina Mosaic for X en 1992, un navegador gráfico de la Web para Unix. En 1993, él y sus compañeros desarrollaron sendas versiones de Mosaic para los sistemas operativos Apple Macintosh y Microsoft Windows. En diciembre de 1993, Marc Andresseen y sus compañeros abandonaron **NCSA** para crear su propia empresa, Mosaic Communications Corporation, que más tarde pasó a llamarse Netscape Communications Corporation.

Vinton Cerf es otro de los "padres de Internet". Estuvo implicado desde los primeros años en el desarrollo de ARPANET. En 1973, se unió al proyecto de Robert Kahn de interconexión de redes. Su mayor contribución ha sido el desarrollo, junto a Robert Kahn, de *Transmission Control Protocol/Internet Protocol* (**TCP/IP**), el protocolo que gobierna las comunicaciones en Internet y que permite conectar distintas redes independientes entre sí.

Larry Roberts y Vinton Cerf recibieron el 25 de octubre de 2002 el Premio Príncipe de Asturias de Investigación Científica y Técnica, junto con Tim Berners-Lee y Robert Kahn.

Preguntas relacionadas: 10
Explicaciones relacionadas: 2

4.1.5. Explicación 5

(Pregunta 35, página 18)

El origen de Internet se sitúa en ARPANET, un proyecto de la agencia **ARPA** del Departamento de Defensa de los Estados Unidos de Norteamérica. Inaugurada a finales 1969, inicialmente conectaba entre sí distintos nodos establecidos en universidades

de los Estados Unidos. Los primeros cuatro nodos que se establecieron conectaban la Universidad de California en Los Ángeles, el Stanford Research Institute, la Universidad de California en Santa Bárbara y la Universidad de Utah.

ARCNET es el acrónimo de *Attached Resource Computer Network*, uno de los tipos de redes locales más antiguo, simple y barato. ARCNET fue desarrollado por la empresa DATAPOINT CORPORATION en 1977. Emplea una topología *token ring*, con velocidades de 2.5 Mbps, sobre distintos medios físicos (par trenzado, cable coaxial, etc.) y hasta 255 ordenadores. Una especificación posterior, ARCNET Plus, soporta hasta 20 Mbps.

ETHERNET es el tipo de red local más empleado en la actualidad. Desarrollado por XEROX CORPORATION en cooperación con DEC e INTEL en 1976. ETHERNET emplea una topología en bus o en estrella (cuando se emplea un concentrador o *hub*). La versión básica permite velocidades de hasta 10 Mbps, pero versiones posteriores permiten velocidades superiores: Fast Ethernet (100 Mbps) y Gigabit Ethernet (1 Gbps).

En 1983, ARPANET se dividió en dos redes independientes: MILNET (formada por 45 nodos de carácter militar) y ARPANET (68 nodos de carácter civil).

4.1.6. Explicación 6

(Pregunta 51, página 23) (Pregunta 77, página 29)

Internet se originó a partir de una red anterior llamada ARPANET (un proyecto de la agencia **ARPA** del Departamento de Defensa de los Estados Unidos de Norteamérica), la cual fue desarrollada por universidades estadounidenses. Fue inaugurada a finales de 1969 y en un principio conectó cuatro nodos situados en la Universidad de California en Los Ángeles, el Stanford Research Institute, la Universidad de California en Santa Bárbara y la Universidad de Utah. Con el tiempo esta red creció y aumentó en usuarios y entidades conectadas por lo que hubo de separar las redes que aglutinaban los nodos civiles de los militares. En 1983, ARPANET se dividió en dos redes independientes: MILNET (formada por 45 nodos de carácter militar) y ARPANET (68 nodos de carácter civil). Posteriormente, la red civil ARPANET paso a llamarse Internet.

Paralelamente a esta red, durante los 80 surgieron servicios telemáticos (ya sea como iniciativas particulares, empresariales o de organismos públicos) llamadas *Bulletin Board System* (**BBS**) basadas en la *Red de Telefonía Básica* (**RTB**) y módems. Estas redes tuvieron mucho éxito durante un tiempo y poco a poco fueron sustituidas por Internet conforme los usuarios comenzaron a utilizar ésta última. Las **BBS** eran servicios locales de información que requerían que el usuario se conectase realizando una llamada telefónica (vía módem) al teléfono de la sede de la **BBS**, esto significaba que los usuarios tenían que llamar a un número y pagar la tarifa vigente (si querían acceder a una **BBS** del extranjero tenían que llamar con tarifa internacional). Con el fin de abaratar costes y ampliar el radio de acción surgieron las agrupaciones de **BBS**

que permitían a un usuario comunicarse e intercambiar información con otros usuarios de otras **BBS** asociadas independientemente de su ubicación geográfica (aunque con un considerable retardo) con solo una llamada local a la **BBS** de su entorno. Las **BBS** asociadas realizaban una sincronización e intercambio de mensajes y ficheros regularmente, con frecuencias de una o varias veces al día, según el servicio y el presupuesto. Una de las redes de **BBS** más famosas y con más aceptación fue FidoNet, que fue creada tras la unión de varias redes anteriores y con el objetivo de hacer frente a la incipiente Internet. En la Figura 4.1 se puede ver la página principal del sitio web de FidoNet.

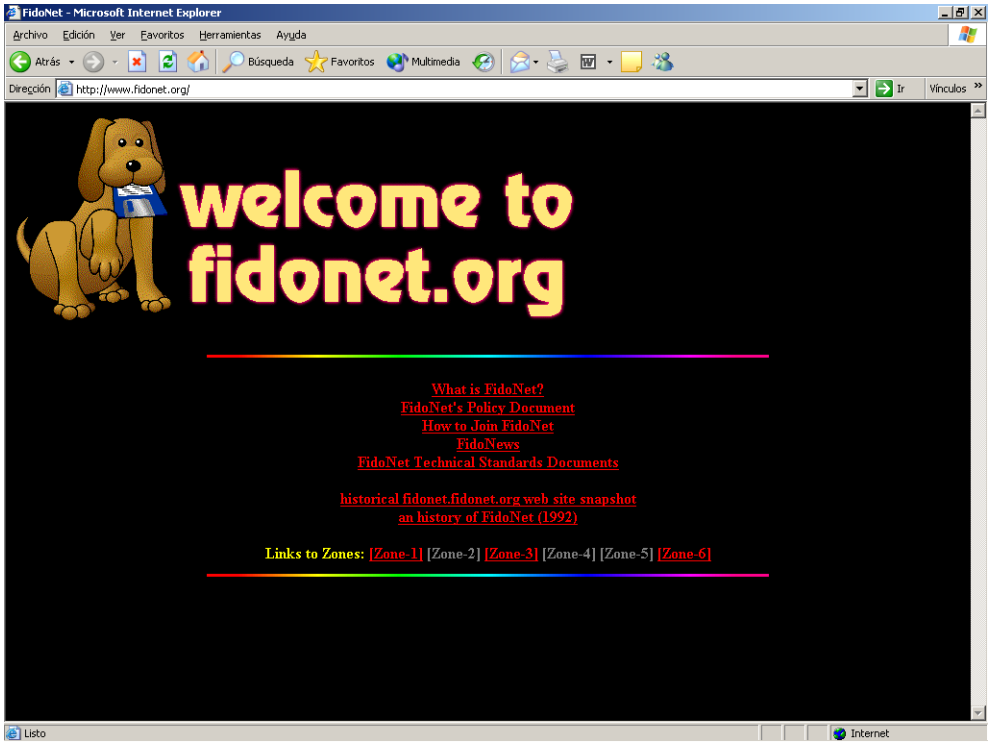


Figura 4.1: Página principal del sitio web de FidoNet

4.1.7. Explicación 7

(Pregunta 52, página 23)

TCP/IP es un conjunto de protocolos que permiten que todo ordenador conectado a una red pueda establecer comunicación con otros e intercambiar información.

Estos protocolos trabajan a distintos niveles de la red y tienen una tarea o especialización determinada. Existe un protocolo, el **HTTP** que esta diseñado para transmitir páginas web y sus objetos asociados (imágenes, documentos, etc.) por Internet. **TCP/IP** también integra otro protocolo que permite localizar un ordenador o servidor en la red por medio de un sistema de direcciones de Internet. Esto requiere que toda máquina que forme parte de la red, posea una dirección, llamada dirección **IP** única que le permite participar en la red (buscar servidores y ser encontrado por otras máquinas). Además, todo ordenador, que se conecte permanentemente o de forma temporal (con un modem, por ejemplo) a Internet necesita poder trabajar con el conjunto de protocolos **TCP/IP** para poderse comunicar con el resto de los ordenadores de la red. En la Figura 4.2 se muestra el modelo de referencia de **TCP/IP** comparado con el modelo de referencia *Open System Interconnection* (**OSI**).

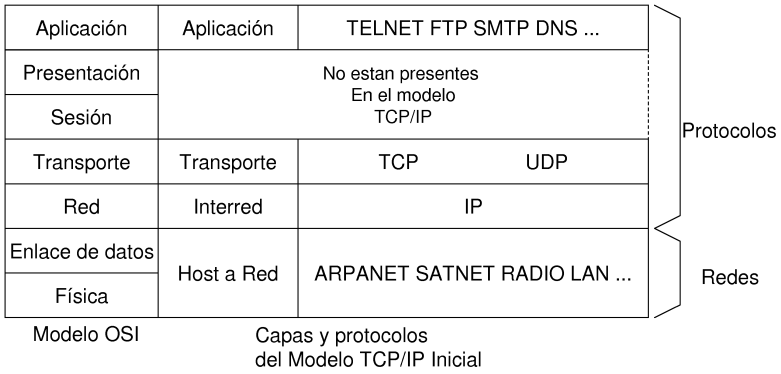


Figura 4.2: Modelo de referencia TCP/IP

4.1.8. Explicación 8

(Pregunta 53, página 23)

El **W3C** es el organismo (sin ánimo de lucro) destinado a analizar, estudiar, aprobar y recomendar protocolos, sistemas, software, hardware y en definitiva los estándares que funcionan por la red Internet. Algunos de estos estándares son *designaciones de iure* (previos a su implantación) y otros son *de facto* (aprobados tras un periodo de coexistencia de varias normas, en el cual se establece cual es la más aceptada o la más apropiada).

4.2. Arquitecturas cliente/servidor

4.2.1. Explicación 9

(Pregunta 1, página 11)

En las arquitecturas cliente/servidor se distingue por un lado la parte cliente (también llamada *front-end*) que interactúa con el usuario (hace de interfaz entre el usuario y el resto de la aplicación) y la parte servidor (o *back-end*) que interactúa con los recursos compartidos (bases de datos, impresoras, módems) e implementa las reglas de negocio (*business rules*). Las denominaciones *display-end* y *process-end* no tienen ningún sentido dentro de las arquitecturas cliente/servidor.

4.3. Aplicaciones web

4.3.1. Explicación 10

(Pregunta 5, página 11) (Pregunta 42, página 19)

De las cuatro tecnologías, **HTML** es la única que se emplea para programar un cliente web (Figura 4.3).

HTML es el lenguaje de marcas que permite definir el formato y contenido de una página web y que permite crear el hipertexto. **HTML** únicamente permite crear páginas estáticas. Este lenguaje fue desarrollado por Tim Berners-Lee a finales de los años ochenta y principio de los noventa.

Common Gateway Interface (**CGI**) es un estándar que define cómo un servidor web puede utilizar programas externos. En concreto, este estándar define cómo los servidores web pasan los datos enviados por el cliente (el navegador) a los programas **CGI** y cómo estos programas tienen que devolver el resultado al servidor web para que lo reenvíe al cliente. Un **CGI** normalmente se programa en *C*, *C++* o *Perl*, aunque puede emplearse cualquier lenguaje de programación de propósito general.

Active Server Pages (**ASP**) es una tecnología de secuencias de comandos del servidor web que permite crear páginas web dinámicas en el servidor. **ASP** es una tecnología propietaria de Microsoft. Cuando un servidor web recibe una petición de una página **ASP**, procesa las secuencias de comando del servidor que contiene la página para generar la página web que se envía al cliente. Una página **ASP** puede contener **HTML** y secuencias de comandos del servidor. Las páginas **ASP** suelen tener la extensión `.asp`.

Por último, ColdFusion es otra tecnología de secuencias de comandos del servidor muy similar a **ASP**. ColdFusion fue inicialmente desarrollado por la empresa allaire, aunque en la actualidad pertenece a Macromedia. Al igual que **ASP**, ColdFusion permite crear páginas web dinámicas en el servidor. Una página ColdFusion también puede contener **HTML** y secuencias de comandos del servidor. ColdFusion emplea el

lenguaje de etiquetas *ColdFusion Markup Language* (**CFML**). Las páginas ColdFusion suelen tener la extensión `.cfm`.

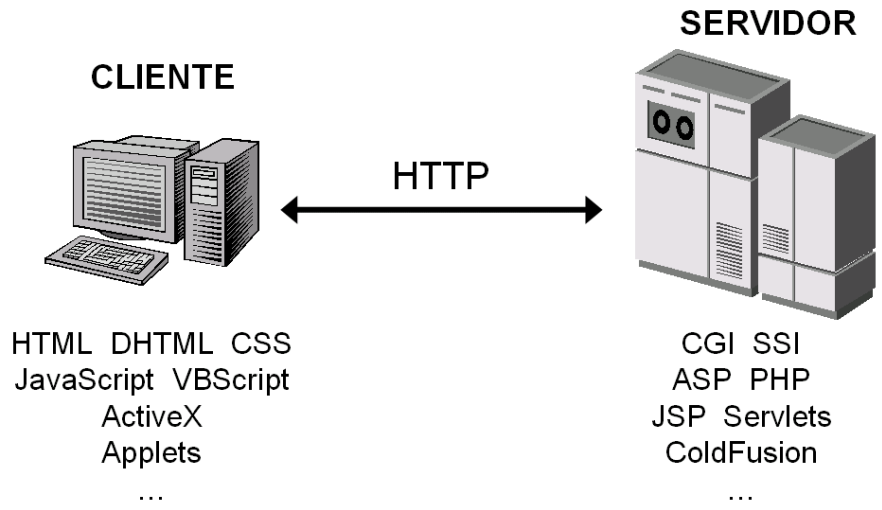


Figura 4.3: Tecnologías empleadas en el cliente y el servidor web

Preguntas relacionadas: 9
Explicaciones relacionadas: 11

4.3.2. Explicación 11

(Pregunta 9, página 12) (Pregunta 80, página 29)

La única tecnología que no se emplea para programar un servidor web son los applets (Figura 4.3).

Los servlets son programas desarrollados con el lenguaje *Java* que se ejecutan en el servidor web cuando éste recibe una petición que los invoca. Para ejecutar un servlet es necesario que el servidor disponga de una *Máquina Virtual Java* (**MVJ**). Las principales ventajas que ofrece esta tecnología son: está basada en *Java*, que es un lenguaje orientado a objetos muy potente, ofrece una extensa *Application Program Interface* (**API**) y es multiplataforma (a condición de que el sistema disponga de la correspondiente **MVJ**).

Los applets son pequeños programas desarrollados con el lenguaje *Java* que se incluyen como un elemento más dentro de una página web y que son ejecutados por el navegador (el cliente). Los applets permiten ampliar enormemente las posibilidades de programación de los navegadores, ya que se programan con un lenguaje orientado

a objetos muy potente. El único inconveniente es que el navegador debe de disponer de una **MVJ** para poder ejecutar un applet.

Server Side Include (**SSI**) es una tecnología que permite incluir comandos en una página web y que son ejecutados por el servidor web antes de transmitir la página al cliente. Por ejemplo, **SSI** permite incluir en una página web el contenido de un fichero, la fecha de la última modificación de un fichero o el resultado producido por un programa al ejecutarse. El formato de un comando **SSI** es:

```
<!-- #comando atributo="valor" -->
```

El comando más sencillo es `#include`, que permite insertar en una página web el contenido de otro fichero. Desgraciadamente, no existe un estándar sobre **SSI**, por lo que cada servidor web puede definir los comandos **SSI** de diferente forma. Sin embargo, algunos comandos, como `#include` o `#exec`, se han convertido en estándar de facto y su comportamiento es idéntico (o muy similar) en la mayoría de los servidores web. Las páginas que contienen comandos **SSI** suelen tener la extensión `.shtml`.

Por último, ColdFusion es una tecnología de secuencias de comandos del servidor que permite crear páginas web dinámicas en el servidor. ColdFusion fue inicialmente desarrollado por la empresa ALLAIRE, aunque en la actualidad pertenece a MACROMEDIA. Cuando un servidor web recibe una petición de una página, ColdFusion procesa las secuencias de comando del servidor que contiene la página para generar la página web que se envía al cliente. Una página ColdFusion puede contener **HTML** y secuencias de comandos del servidor. ColdFusion emplea el lenguaje de etiquetas **CFML**. Las páginas ColdFusion suelen tener la extensión `.cfm`.

4.3.3. Explicación 12

(Pregunta 14, página 13) (Pregunta 55, página 23) (Pregunta 91, página 31)

Una intranet es una red de ordenadores privada basada en los protocolos que gobiernan Internet (**TCP/IP**) que pertenece a una organización y que es accesible únicamente por los miembros o empleados de la misma. El servicio web de una intranet tiene la misma funcionalidad y aspecto que cualquier otro de Internet con la diferencia que su acceso es privado y esta bajo identificación del usuario.

Relacionado con el anterior término, llamamos extranet a una intranet que esta abierta parcial o totalmente a accesos externos. Si bien una intranet está protegida tras un cortafuegos (*firewall*) y sólo personal autorizado puede acceder, una extranet permite varios niveles de accesibilidad para personal externo que no pertenece a la organización, siempre y cuando tengan una cuenta que les de acceso.

4.3.4. Explicación 13

(Pregunta 54, página 23)

La única tecnología que no se emplea para programar un cliente web es **SSI**.

HTML es el lenguaje de diseño de páginas web que interpretan los clientes web. *JavaScript* es el lenguaje de programación asociado al **HTML** y su ámbito es el de una

página web, por tanto es una tecnología de cliente también. *Virtual Reality Modeling Language* (**VRML**) es un lenguaje de diseño o modelado de escenas tridimensionales que pretenden simular espacios reales (lenguaje de modelado de escenas 3D de realidad virtual). Este lenguaje esta basado en la misma filosofía que el **HTML**, funciona con un conjunto de etiquetas y modificadores que recibe un navegador (un cliente) y este interpreta las etiquetas para mostrar al usuario en pantalla la escena 3D diseñada. Incorpora un motor de movimiento que permite al usuario moverse por la escena como si estuviese en su interior. Es, por tanto, una tecnología de cliente porque todo ocurre y se lleva a cabo en el ordenador del cliente que accede al mundo 3D. **SSI** son comandos que se incluyen en una página **HTML** y que son ejecutados por el servidor web antes de transmitir la página al cliente. Por tanto, se trata de tecnología de servidor.

Preguntas relacionadas: 5 9 42
Explicaciones relacionadas: 10 11

4.4. HTML

4.4.1. Explicación 14

(Pregunta 2, página 11) (Pregunta 82, página 30)

En enero de 2000 aparece *eXtensible HyperText Markup Language* (**XHTML**) 1.0, el futuro sustituto de **HTML**. Como dice el propio estándar, se trata de "una reformulación de HTML en XML 1.0". **XHTML** es el lenguaje **HTML** escrito según las normas que impone *Extensible Markup Language* (**XML**). Por tanto, es una aplicación concreta de **XML** y no deben confundirse entre sí. Las principales diferencias entre **HTML** y **XHTML** 1.0 son:

- Las etiquetas y atributos tienen que escribirse en minúsculas. Por ejemplo, `<IMG SRC="logo.gif">` es correcto en **HTML**, pero no en **XHTML** (debería ser `<img src="logo.gif">`).
- Los valores de los atributos tienen que ir entre comillas. Por ejemplo, `<font face=arial>` es correcto en **HTML**, pero no en **XHTML** (debería ser `<font face="arial">`).
- No se admiten atributos sin valor. Por ejemplo, `<hr noshade>` es correcto en **HTML**, pero no en **XHTML** (debería ser `<hr noshade="noshade">`).
- Todas las etiquetas tienen que aparecer por parejas (inicio y fin) o como etiquetas vacías. Por ejemplo, la etiqueta `<img src="logo.gif">` es correcta en **HTML**, pero no en **XHTML** (debería ser `<img src="logo.gif"></img>` o como etiqueta vacía `<img src="logo.gif"/>`).

4.4.2. Explicación 15

(Pregunta 19, página 14) (Pregunta 28, página 17) (Pregunta 60, página 24)

Una página web es un documento **HTML** que se estructura en dos bloques: una cabecera y un cuerpo. A su vez cada bloque tiene una estructura determinada. Esta estructura se indica con unas etiquetas que delimitan cada uno de los bloques de la siguiente forma:

```
<HTML> <!-- Abrimos un documento HTML -->
    <HEAD> <!-- Abrimos la cabecera -->
    ... Aquí va el contenido de la cabecera ...
    </HEAD> <!-- Cerramos la cabecera -->
    <BODY> <!-- Abrimos el cuerpo de la página web -->
    ... Aquí se ubica el cuerpo de la página ...
    </BODY> <!-- Cerramos el cuerpo -->
</HTML> <!-- Cerramos la definición del documento -->
```

De forma resumida, la estructura básica de una página web es: `<Html><Head> ... </Head><Body> ... </Body></Html>`.

Como en **HTML** no se distinguen mayúsculas y minúsculas (las etiquetas `<html>` y `<HTML>` representan la misma etiqueta), también son válidas las etiquetas:

- `<html><head> ... </head><body> ... </body></html>`.
- `<HTML><HEAD> ... </HEAD><BODY> ... </BODY></HTML>`.

4.4.3. Explicación 16

(Pregunta 20, página 14) (Pregunta 84, página 30)

La etiqueta `<P> ... </P>` se emplea para marcar párrafos de texto. Esta etiqueta posee el atributo `ALIGN="left | center | right | justify"` que permite modificar el alineamiento del texto.

La recomendación de **W3C** sobre **HTML** 4.01 indica que la presentación visual de un párrafo depende del navegador (agente de usuario) que se emplee. Normalmente, el texto en un párrafo está alineado a la izquierda y se deja un espacio en blanco antes y después del párrafo.

4.4.4. Explicación 17

(Pregunta 30, página 17) (Pregunta 86, página 30)

Para insertar un salto de línea en una página **HTML** se emplea la etiqueta `<BR>`. ` ` (también se puede usar ` ` or ` `) es una entidad que representa un espacio en blanco que no se puede "romper" (*non-breaking space*), es decir, un espacio en blanco en el que no se puede producir un salto de línea y `<BK>` no existe por ahora en el estándar de **HTML**.

4.4.5. Explicación 18

(Pregunta 32, página 18)
Para crear tablas se emplean las etiquetas `<tr>`, `<td>` y `<th>`:

- `<tr> ... </tr>` (*table row*): Define una fila de una tabla que contiene celdas.
- `<td> ... </td>` (*table data cell*): Define una celda de datos de una tabla.
- `<th> ... </th>` (*table header cell*): Define una celda de encabezamiento de una tabla. Este tipo de celdas suelen visualizarse con una apariencia distinta a la de las celdas de datos (por ejemplo, con el contenido en negrita).

Del resto de etiquetas, `<table>` sí que se emplea para crear una tabla (define una tabla), pero `<tt>` se emplea para especificar texto con formato teletipo (texto monoespaciado) y `<tf>` no existe por ahora en el estándar de **HTML**.
Preguntas relacionadas: 43
Explicaciones relacionadas: 23

4.4.6. Explicación 19

(Pregunta 36, página 18)
Para insertar una imagen se emplea la etiqueta `<IMG>` sin etiqueta de cierre. Por tanto, `<IMG> ... </IMG>` no es una etiqueta correcta. La etiqueta `<A> ... </A>` se emplea para crear enlaces y la etiqueta `<IMAGE> ... </IMAGE>` no existe por ahora en el estándar de **HTML**.

4.4.7. Explicación 20

(Pregunta 38, página 19) (Pregunta 96, página 32)
Mediante el protocolo `mailto:` se puede enviar un correo electrónico cuando el usuario pulse sobre un enlace. Realmente no se envía un correo electrónico automáticamente: se abre el programa de correo electrónico que tenga configurado el usuario por defecto y se crea un nuevo mensaje, pero no se envía hasta que el usuario no lo confirme.
La sintaxis de este protocolo es:
`mailto:dir1[,dir2,...][?opcion1[&opcion2&...]]`
donde `dir1, dir2, ...` son direcciones de correo electrónico y `opcion1, opcion2, ...` es una lista de parámetros opcionales. Los principales valores que se pueden emplear aparecen en el Cuadro 4.1.
Estas opciones se pueden combinar en un único enlace. Sólo hay que tener en cuenta que se tienen que separar con el símbolo "`&`" y que los valores de las opciones se tienen que escribir empleando "codificación URL" (*URL encoding*). En esta codificación, una serie de caracteres especiales, como por ejemplo "`&`", "`%`", "`$`" o "`ñ`", se

Opción de envío	Acción
`cc` (*carbon copy*)	Indica el envío de copias del mensaje a otros destinatarios
`bcc` (*blind carbon copy*)	Indica el envío de copias del mensaje a otros destinatarios ocultos
`subject`	Define el título del mensaje
`body`	Especifica el texto del mensaje

Cuadro 4.1: Opciones de envío con el protocolo mailto:

codifican usando el símbolo "`%`" seguido de dos dígitos que expresan, en hexadecimal, su código *American Standard Code for Information Interchange* (**ASCII**). Por ejemplo, la cadena "`&%$ñ`" se codificaría como "`%26%25%24%F1`".

4.4.8. Explicación 21

(Pregunta 39, página 19)

Ninguna de las respuestas es correcta. Los formatos de imágenes que aceptan la mayoría de los navegadores son *Graphics Interchange Format* (**GIF**) y *Joint Photographic Experts Group* (**JPG**). *Portable Network Graphics* (**PNG**) es otro formato gráfico destinado para la Web que cada vez aceptan más navegadores.

Bit-map (**BMP**) es un formato gráfico de mapa de bits estándar en los sistemas operativos Microsoft Windows. *Tagged Image File Format* (**TIFF**) es un formato gráfico de mapa de bits muy popular tanto en entorno Microsoft Windows como Apple Macintosh. Por último, *Moving Picture Experts Group* (**MPG**) es un formato de vídeo digital muy empleado por el nivel de compresión que obtiene.

4.4.9. Explicación 22

(Pregunta 41, página 19)

Si en una tabla se modifica el color de fondo de la tabla y el color de fondo de una celda, el color de fondo de la celda domina sobre el color de fondo de la tabla.

Cuando en una tabla se modifica el color de distintos elementos (tabla, fila o celda) domina siempre el último color modificado, tal como se indica a continuación:

- Color de la tabla
  - Color de la fila
    - Color de la celda

El siguiente ejemplo muestra una tabla con distintos colores de fondo para la tabla, la fila y la celda. En la Figura 4.4 vemos como se muestra esta página en un navegador.

```
<HTML>
<HEAD>
<TITLE>Color de fondo de una tabla</TITLE>
</HEAD>
<BODY>
<CENTER>
<TABLE BORDER="1" BGCOLOR="Red">
  <TR>
    <TD>El color de fondo es rojo</TD>
    <TD>Se muestra el color de fondo de la tabla</TD>
  </TR>
  <TR BGCOLOR="Green">
    <TD>El color de fondo es verde</TD>
    <TD>Se muestra el color de fondo de la fila</TD>
  </TR>
  <TR BGCOLOR="Green">
    <TD BGCOLOR="Blue">El color de fondo es azul</TD>
    <TD BGCOLOR="Blue">Se muestra el color de fondo de
    la celda</TD>
  </TR>
</TABLE>
</CENTER>
</BODY>
</HTML>
```

4.4.10. Explicación 23

(Pregunta 43, página 20)

La etiqueta `<TD> ... </TD>` define una celda de una tabla.

Para definir una fila de una tabla se emplea la etiqueta `<TR> ... </TR>` y para definir una celda que es encabezado de una tabla se emplea `<TH> ... </TH>`.

Preguntas relacionadas: 32

Explicaciones relacionadas: 18

4.4.11. Explicación 24

(Pregunta 45, página 20)

Para cambiar el color de fondo de una página **HTML** se emplea el atributo `BGCOLOR` de la etiqueta `<BODY> ... </BODY>`.

El atributo `COLOR` no existe en la etiqueta `<BODY> ... </BODY>` y la etiqueta `<HTML> ... </HTML>` no posee ni el atributo `COLOR` ni el atributo `BGCOLOR`.

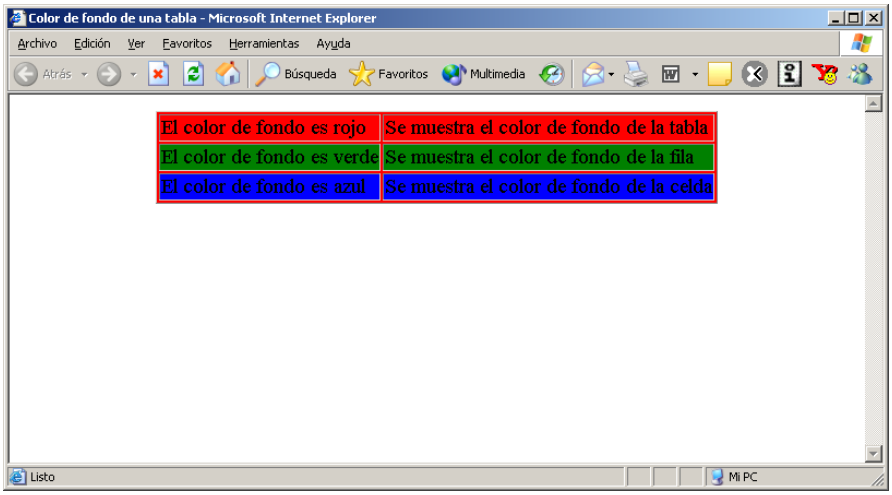


Figura 4.4: Ejemplo de tabla con color de fondo

4.4.12. Explicación 25

(Pregunta 48, página 21) (Pregunta 93, página 32)

El enlace correcto es:

```
<a href="http://www.ua.es/indice.html">UA</a>
```

`<a url="http://www.ua.es/indice.html">UA</a>` no es correcto porque `url` no es un atributo de la etiqueta `<a> ... </a>`.

El enlace `<a name="http://www.ua.es/indice.html">UA</a>` no es correcto porque el atributo `name` se emplea para definir un destino de un enlace y no un enlace en sí mismo: el valor del atributo `name` es un nombre de destino (un lugar dentro de un documento **HTML**) y no una **URL**.

4.4.13. Explicación 26

(Pregunta 56, página 23)

La etiqueta `<OL>` permite definir una lista ordenada (*ordered list*). La etiqueta `<UL>` define listas no ordenadas (*unordered list*). Para definir un elemento en ambos tipos de lista se emplea la etiqueta `<LI>` (*list item*). Por ejemplo, el siguiente código define una lista ordenada:

```
<OL>
  <LI>Elemento 1</LI>
  <LI>Elemento 2</LI>
```

```
  <LI>...</LI>
  <LI>Elemento N</LI>
</OL>
```

En la Figura 4.5, se puede observar como se muestra una lista ordenada y una lista no ordenada en un navegador:

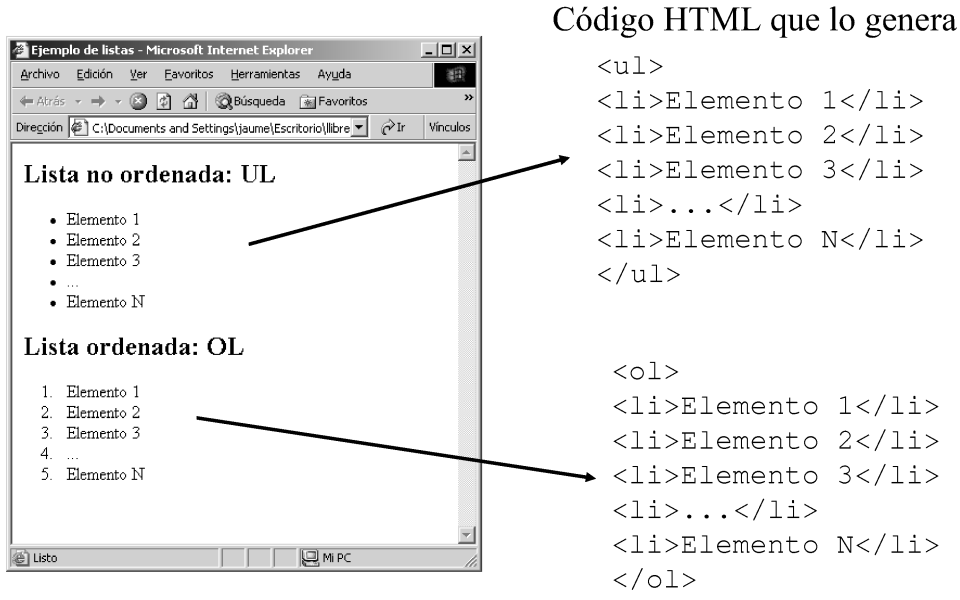


Figura 4.5: Ejemplo de lista ordenada y no ordenada

4.4.14. Explicación 27

(Pregunta 57, página 24)

Para provocar que se abra una ventana nueva del navegador al pulsar sobre un enlace de una página, hay que añadir a la etiqueta `<a> ... </a>` que genera el enlace el atributo `target` y asignarle el valor `_blank`. Por ejemplo:

```
<a href="url" target="_blank"> ... </a>
```

Así, sin necesidad de usar *JavaScript*, conseguimos que este enlace abra en una ventana nueva la **URL** de destino que especifiquemos.

El atributo `target` de la etiqueta `<a> ... </a>` permite especificar el lugar donde se cargará el destino del enlace. Así, es posible determinar que un enlace se cargue en un marco determinado de la página, simplemente poniendo como valor del atributo el nombre del marco.

Existen cuatro valores predefinidos para el atributo `target`. Estos son:

- `_blank`: abre el enlace en una ventana nueva.
- `_top`: abre el enlace en la misma ventana, en el nivel superior de la jerarquía de ventanas (marcos). De este modo, se pueden eliminar los marcos de una ventana.
- `_parent`: abre el enlace en la misma ventana pero en el marco inmediatamente superior en la jerarquía de marcos del documento donde está contenido el enlace.
- `_self`: abre el destino del enlace en la misma ventana (marco). Este es el valor por defecto en caso de no especificar el atributo `target`.

4.4.15. Explicación 28

(Pregunta 58, página 24) (Pregunta 89, página 31)

Para alinear a la izquierda el contenido de una celda de una tabla en **HTML** se emplea la etiqueta `<TD> ... </TD>`, que es la que define las celdas de la tabla y se complementa con el atributo `align`, que tiene que tomar el valor `left`. Por ejemplo:

```
<td align="left"> ... </td>
```

Con el atributo `align` se puede alinear todo el contenido de la celda: texto, imágenes y otros objetos que contenga. El valor por defecto que toma este atributo es `left`. Este atributo puede tomar los siguientes valores:

- `LEFT`: ajusta el contenido a la izquierda de la celda. Ejemplo:
  `<td align="LEFT"> ... </td>`
- `CENTER`: ajusta el contenido al centro de la celda. Ejemplo:
  `<td align="CENTER"> ... </td>`
- `RIGHT`: alinea el contenido a la derecha de la celda. Ejemplo:
  `<td align="RIGHT"> ... </td>`

El contenido de una celda también se puede alinear verticalmente por medio del atributo `valign`. Este atributo puede tomar los siguientes valores:

- `TOP`: alineación a la parte superior de la celda. Ejemplo:
  `<td valign="TOP"> ... </td>`

- `MIDDLE`: alineación al centro de la celda. Ejemplo:
  `<td valign="MIDDLE"> ... </td>`
- `BOTTOM`: alineación a la parte inferior de la celda. Ejemplo:
  `<td valign="BOTTOM"> ... </td>`
- `BASELINE`: alineación a la línea base. Ejemplo:
  `<td valign="BASELINE"> ... </td>`.

Los atributos `align` y `valign` también son aplicables a otros elementos **HTML** como párrafos (`<P>`), divisiones (`<DIV>`), imágenes (`<IMG>`), etc.

El siguiente ejemplo muestra una tabla con distinto alineamiento horizontal y vertical. En la Figura 4.6 vemos como se muestra esta página en una navegador.

```
<HTML>
<BODY>
<BR><BR>
<CENTER>
<TABLE BORDER="1">
<TR ALIGN="CENTER">
  <TD>Varias<BR>líneas<BR>de texto</TD>
  <TD>Alineamiento CENTER</TD>
  <TD>Alineamiento CENTER</TD>
  <TD>Alineamiento CENTER</TD>
</TR>
<TR ALIGN="RIGHT" VALIGN="TOP">
  <TD>Varias<BR>líneas<BR>de texto</TD>
  <TD>Alineamiento RIGHT y TOP</TD>
  <TD>Alineamiento RIGHT y TOP</TD>
  <TD>Alineamiento RIGHT y TOP</TD>
</TR>
<TR>
  <TD>Varias<BR>líneas<BR>de texto</TD>
  <TD ALIGN="LEFT">Alineamiento LEFT</TD>
  <TD ALIGN="CENTER">Alineamiento CENTER</TD>
  <TD ALIGN="RIGHT">Alineamiento RIGHT</TD>
</TR>
<TR>
  <TD>Varias<BR>líneas<BR>de texto</TD>
  <TD VALIGN="TOP">Alineamiento TOP</TD>
  <TD VALIGN="MIDDLE">Alineamiento MIDDLE</TD>
  <TD VALIGN="BOTTOM">Alineamiento BOTTOM</TD>
</TR>
<TR VALIGN="MIDDLE">
```

```
  <TD>Varias<BR>líneas<BR>de texto</TD>
  <TD VALIGN="TOP">Alineamiento TOP</TD>
  <TD VALIGN="MIDDLE">Alineamiento MIDDLE</TD>
  <TD VALIGN="BOTTOM">Alineamiento BOTTOM</TD>
</TR>
</TABLE>
</CENTER>
</BODY>
</HTML>
```

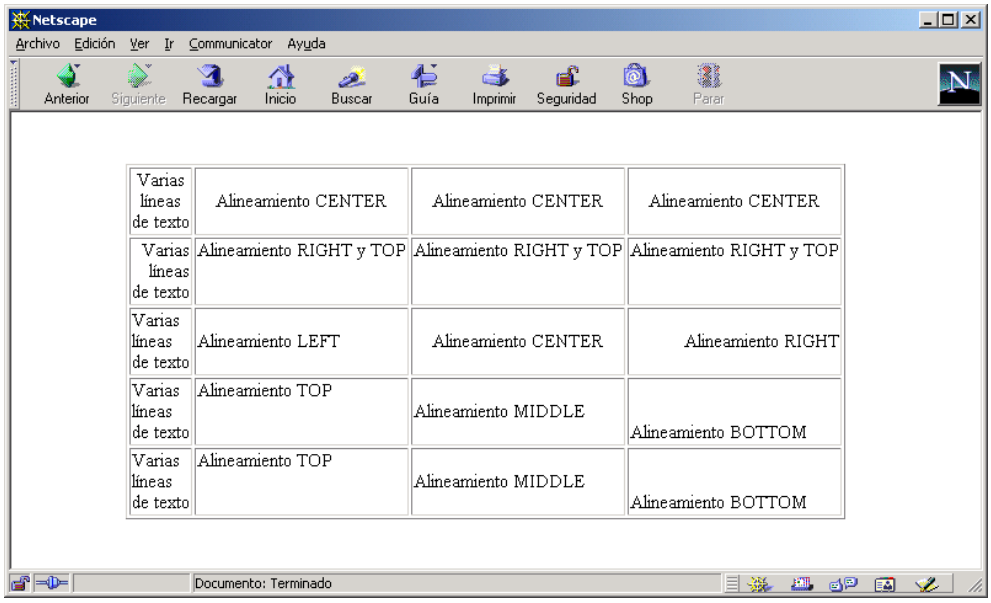


Figura 4.6: Alineamiento del contenido de una tabla

4.4.16. Explicación 29

(Pregunta 59, página 24)

Si deseamos ubicar en un formulario **HTML** un cuadro de texto multilínea debemos hacerlo con la etiqueta:

```
<TEXTAREA NAME="..." COLS="..." ROWS="..." ...>
...
</TEXTAREA>
```

Esta etiqueta define un control de texto editable multilínea cuyo tamaño se define por el valor del atributo `COLS` para las columnas y el valor de `ROWS` para las líneas de texto. Los principales atributos de esta etiqueta son:

- `COLS`: define el número de columnas (anchura) que tiene el control.
- `ROWS`: especifica el número de filas o líneas horizontales (altura) del cuadro de texto.
- `WRAP`: especifica si las líneas más largas que la anchura del área de texto se tienen que cortar para que se visualicen sin tener que realizar un desplazamiento horizontal. Los posibles valores de este atributo son:
  - `OFF`: las líneas no se dividen automáticamente.
  - `HARD`: las líneas se dividen automáticamente, y los saltos de línea se conservan al enviar el formulario.
  - `SOFT`: las líneas se dividen automáticamente, pero los saltos de línea no se conservan al enviar el formulario.

Si se quiere que el área de texto muestre un texto por defecto, se puede incluir entre las etiquetas de inicio y fin. El siguiente ejemplo muestra dos áreas de texto de distinto tamaño, una de ellas con un texto por defecto. En la Figura 4.7 se puede ver esta página visualizada en un navegador. Se pueden observar las barras de desplazamiento vertical y horizontal.

```
<HTML>
<BODY>
<FORM>
Área 1:
<TEXTAREA ROWS="2" COLS="40">Texto por defecto</TEXTAREA>
<BR>
Área 2:
<TEXTAREA ROWS="4" COLS="20"></TEXTAREA>
</FORM>
</BODY>
</HTML>
```

Por otro lado, y a modo de breve resumen, en un formulario de una página web se pueden emplear los siguiente controles:

- `<input type="...">`: crea distintos tipos de controles en función del valor del atributo `TYPE`:
  - `text`, crea un cuadro de texto de una sola línea.

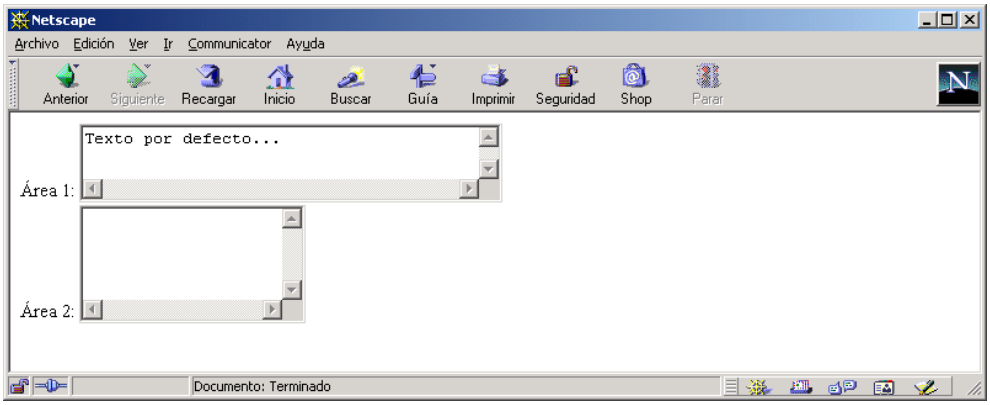


Figura 4.7: Áreas de texto de distinto tamaño

- `hidden`, crea un campo oculto en el formulario.
- `password`, crea un cuadro de texto velado, donde lo que escribe el usuario queda oculto.
- `radio`, crea un botón de tipo radio (campo excluyente).
- `checkbox`, crea un botón de comprobación (campo de verificación).
- `button`, crea un botón sin una acción específica.
- `submit`, crea un botón que envía el formulario al ser pulsado.
- `reset`, crea un botón que borra el formulario al ser pulsado.

- `<select> ... </select>`: crea una lista desplegable o multilínea.
- `<textarea> ... </textarea>`: crea un control de texto multilínea.

4.5. JavaScript

4.5.1. Explicación 30

(Pregunta 15, página 13)

JavaScript es un lenguaje basado en objetos, pero no es un lenguaje orientado a objetos "puro", ya que carece de conceptos clave en la orientación a objetos como son el concepto de clave, herencia u ocultación.

4.5.2. Explicación 31

(Pregunta 21, página 14) (Pregunta 88, página 31)

En *JavaScript*, para mostrar un mensaje de alerta se emplea el método `alert` del objeto `window`. Por ejemplo:

```
window.alert("mensaje")
```

Realmente, el objeto `window` no pertenece al lenguaje *JavaScript*, sino al *Document Object Model* (**DOM**) que refleja la estructura de la página web. Otros métodos que posee el objeto `window` para mostrar ventanas son `confirm` y `alert`.

Por ejemplo, el siguiente código genera las ventanas que se muestran en la Figura 4.8, 4.9 y 4.10 y que se han obtenido al ejecutar el código en el navegador Netscape Navigator 4.78.

```
<script language="JavaScript">
  window.alert("Es una prueba de alert");
  window.confirm("Esto es una prueba de confirm");
  window.prompt("Esto es una prueba de prompt");
</script>
```

Figura 4.8: Ventana de alerta

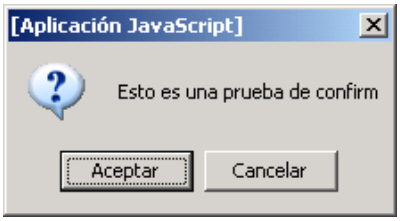


Figura 4.9: Ventana de confirmación

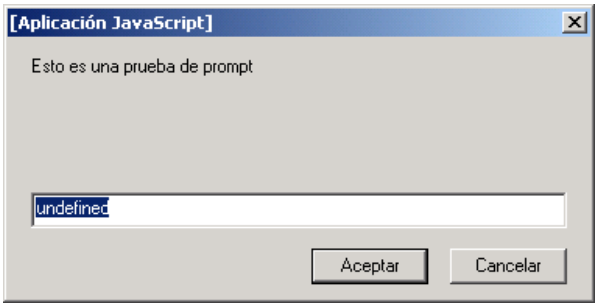


Figura 4.10: Ventana de solicitud de datos

4.5.3. Explicación 32

(Pregunta 33, página 18) (Pregunta 98, página 33)

En *JavaScript*, las cadenas literales (constantes) se escriben como secuencias de caracteres encerrados entre comillas dobles (`" ... "`) o entre comillas simples (`' ... '`). Una cadena tiene que estar delimitada por comillas (dobles o simples) del mismo tipo. El uso de los dos tipos de comillas tiene su sentido, ya que *JavaScript* se usa conjuntamente con el lenguaje **HTML**, y como éste usa las comillas dobles en su sintaxis, tiene que haber algún método de distinguir las cadenas de *JavaScript* de las cadenas de **HTML**.

Una cadena puede contener cero o más caracteres (cualquier tipo de combinación de números, letras, espacios y símbolos). Además de los caracteres normales, en una cadena también se pueden incluir caracteres especiales que permiten generar saltos de línea o caracteres a partir de su código **ASCII**. El Cuadro 4.2 muestra los caracteres especiales que se pueden emplear dentro de una cadena.

4.5.4. Explicación 33

(Pregunta 49, página 21)

En *JavaScript*,para comprobar que la variable "`i`" es igual a "`5`" se escribe la sentencia condicional `if (i == 5)`.

El operador de comparación es `==` y la sintaxis de la sentencia condicional simple es `if (condición)`.

4.5.5. Explicación 34

(Pregunta 61, página 24)

El operador para concatenar cadenas en *JavaScript* es el símbolo de la suma (+). Por ejemplo:

Carácter	Significado
\b	Retroceso (*backspace*)
\f	Salto de página (*form feed*)
\n	Salto de línea (*new line*)
\r	Retorno de carro (*carriage return*)
\t	Tabulador
\'	Apóstrofe o comilla simple
\"	Comilla doble
\\	Barra invertida (*backslash*)
\XXX	El carácter de la codificación Latin-1 especificado por los tres dígitos octales entre 0 y 377.
\xXX	El carácter de la codificación Latin-1 especificado por los dos dígitos hexadecimales entre 00 y FF.
\uXXXX	El carácter Unicode especificado por los cuatro dígitos hexadecimales entre 0000 y FFFF.

Cuadro 4.2: Caracteres especiales

```
cadena = "Hola" + " Mundo";
```

En cambio para concatenar cadenas en *VBScript*, el operador utilizado es el *ampersand* (&). Un ejemplo:

```
cadena = "Hola" & " Mundo"
```

Con la tecnología PHP, las cadenas se concatenan usando el operador punto (.). Por ejemplo:

```
$cadena = "Hola" . " Mundo";
```

4.5.6. Explicación 35

(Pregunta 62, página 25)

En *JavaScript*, los índices de los vectores (*arrays*) y de las cadenas de caracteres comienzan por el 0, al igual que en *C*, *C++* y *Java*.

En *VBScript*, los índices también comienzan por el 0.

En *Java Server Pages* (**JSP**), al estar basado en el lenguaje *Java*, los índices de los vectores también comienzan por el 0.

4.5.7. Explicación 36

(Pregunta 63, página 25)

Para incluir código *JavaScript* a partir de un fichero externo en una página web se debe utilizar la siguiente instrucción:

```
<SCRIPT LANGUAGE="JavaScript" SRC="URL"></SCRIPT>
```

donde `URL` puede ser una ruta relativa o absoluta, o puede ser una ruta local o de Internet (con `http://`).

El atributo `LANGUAGE` es opcional, y en caso de omitirse, el valor por defecto que se adopta es *JavaScript*.

Mediante esta instrucción podemos encapsular funciones y código *JavaScript* para ser reutilizado de forma sencilla en varias páginas web. Además al estar el código centralizado en un solo fichero su mantenimiento se simplifica.

En el caso de querer incluir una declaración de estilos *CSS* ubicados en un fichero externo, debemos utilizar la etiqueta `<LINK>`, cuya sintaxis es:

```
<LINK REL="stylesheet"
      HREF="ruta/fichero.extension"
      TYPE="TEXT/CSS">
```

Los atributos que necesita son (entre otros):

- `REL`: especifica la relación que existe entre el objeto enlazado y la página que lo enlaza.
- `HREF`: para indicarle la ruta y el nombre del fichero que contiene los estilos *Cascading Style Sheets* (**CSS**).
- `TYPE`: le indicamos el tipo *Multipurpose Internet Mail Extensions* (**MIME**) el objeto que enlazamos.

4.5.8. Explicación 37

(Pregunta 64, página 25)

Para abrir una nueva ventana del navegador con *JavaScript* debemos declararla con la siguiente instrucción:

```
var w = window.open("url", "nombreVentana", "parametros");
```

Aunque de forma simplificada sería válida la siguiente instrucción:

```
window.open("url", "nombreVentana");
```

En ambos casos la función `open` del objeto `window` tiene la siguiente definición:

```
window.open(url, nombre, parametros);
```

Siendo:

- `url`: la ruta y nombre del fichero que queremos cargar en la nueva ventana del navegador.
- `nombre`: el nombre que le damos a la nueva ventana para poder ser referenciada desde otros lugares del navegador con el fin de interactuar con ella.
- `parametros`: definición de la nueva ventana, se controlan aspectos como coordenadas, dimensiones, barras de desplazamiento, etc. Estos parámetros se especifican mediante parejas de la forma `nombre=valor` y cuando hay varios se separan por comas. Los parámetros principales que se pueden emplear son:
  - `channelmode = yes | no | 1 | 0`. Indica si en la nueva entana aparecera la banda de canales. Por defecto esta a 'no'.
  - `directories = yes | no | 1 | 0`. Especifica si aparecen los botones de añadir directorio. Por defecto esta a 'yes'.
  - `fullscreen = yes | no | 1 | 0`. Indica si la ventana se mostrará a pantalla completa ocultando todos los controles, menus, etc. del navegador. El valor por defecto es 'no'.
  - `height = numero`. Con este parámetro indicamos la altura en pixels de la ventana.
  - `left = numero`. Indicamos la posición izquierda de la nueva ventana, en pixels.
  - `location = yes | no | 1 | 0`. Especifica si se muestra o no la barra de dirección. Por defecto esta a 'yes'.
  - `menubar = yes | no | 1 | 0`. Indicamos si vamos a mostrar o no la barra de menu. Por defecto esta a 'yes'.
  - `resizable = yes | no | 1 | 0`. Con este parámetro le decimos si la ventana será modificable (redimensionable) o no. Por defecto está a 'yes'.
  - `scrollbars = yes | no | 1 | 0`. Nos permite indicar si queremos barras de desplazamiento horizontales o verticales. Por defecto está a 'yes'.
  - `status = yes | no | 1 | 0`. Indica si se mostrará o no la barra de estado en la parte inferior de la ventana. Por defecto está a 'yes'.
  - `titlebar = yes | no | 1 | 0`. Permite mostrar u ocultar la barra de titulo de la ventana. Por defecto esta a 'yes'.

- `toolbar = yes | no | 1 | 0`. Indica si mostramos o no la botonera de herramientas del navegador. Por defecto esta a 'yes'.
- `top = numero`. Le indicamos con este parámetro la posición vertical de la nueva ventana, en pixles.
- `width = numero`. Ajusta el ancho de la ventana en pixels.

4.6. VBScript

4.6.1. Explicación 38

(Pregunta 24, página 15)

En *VBScript*, se pueden emplear tanto `+` como `&` para concatenar cadenas. Sin embargo, el mejor operador es `&` porque se emplea exclusivamente para concatenar cadenas, mientras que `+` también se emplea para sumar dos números por lo que se pueden generar situaciones ambiguas donde no se pueda determinar que operación se está realizando.

El subtipo de un operando determina el comportamiento del operador `+`:

- Si los dos operandos son numéricos, se realiza la suma.
- Si los dos operandos son cadenas, se realiza la concatenación.
- Si uno de los operandos es numérico y el otro es una cadena, se realiza la suma.

Por ejemplo, `1 + "2"` o `"1" + 2` devuelve `3`, mientras que `"1" + "2"` devuelve `12`.

4.6.2. Explicación 39

(Pregunta 67, página 25)

En *VBScript* no es necesario declarar las variables que se usan, a no ser que al principio del programa o *script* se incluya la declaración `Option Explicit`.

En ese caso todas las variables se deben declarar antes de ser usadas por primera vez. Para declarar una variable, se debe utilizar la palabra reservada `Dim`. Por ejemplo:

```
Dim sMiVariable
```

Las variables que se utilicen en todo caso, se pueden declarar en cualquier parte del código y su ámbito corresponde al lugar donde se hayan declarado. Además, tampoco es necesario indicar el tipo de las variables. De hecho es posible cambiar el tipo de una variable en tiempo de ejecución sin necesidad de declararla de nuevo.

4.7. CGI

4.7.1. Explicación 40

(Pregunta 12, página 13) (Pregunta 90, página 31)

Todas las afirmaciones son falsas.

Un programa **CGI** puede recibir información desde un servidor web de cuatro formas distintas:

- A través de la línea de comandos (*command line*).
- A través de la URL (`QUERY_STRING`).
- A través de la entrada estándar (`stdin`).
- A través de la información de ruta (`PATH_INFO`).

Un programa **CGI** se puede programar con un lenguaje compilado (*C*, *C++*, *Pascal*, etc.) o con un lenguaje interpretado (*Perl*, *shell* de Unix, etc.).

Finalmente, la tecnología **CGI** no mantiene automáticamente el estado (sesión) entre una conexión y otra. El programador tiene que desarrollar sus propios métodos (empleo de *cookies*, identificador de la sesión en la **URL**, etc.) para mantener la sesión entre distintas conexiones.

4.7.2. Explicación 41

(Pregunta 23, página 15)

Existen dos métodos para obtener la información que un cliente web envía con una **URL** al servidor web:

- Mediante la variable de entorno `QUERY_STRING`: a través de esta variable se reciben los datos que se añaden al final de la **URL**, separados del nombre del programa **CGI** mediante un signo de interrogación (`?`). También se emplea esta variable cuando se envía un formulario con el método `GET`.
- Mediante la variable de entorno `PATH_INFO`: a través de esta variable se reciben los datos que se añaden al final de la **URL**, separados del nombre del programa **CGI** mediante una barra inclinada (`/`).

4.7.3. Explicación 42

(Pregunta 40, página 19)

Cuando un cliente web envía la información de un formulario al servidor, el navegador codifica automáticamente la entrada del usuario. Los datos introducidos en un formulario se envían al programa **CGI** con el siguiente formato:

```
control1=valor1&control2=valor2&...&controln=valorn
```

Por tanto,las distintas parejas `campo=valor` se separan por `&`.

El signo "+" se emplea para sustituir el espacio en blanco en la entrada del usuario, mientras que el signo "`%`" se emplea para indicar la codificación en hexadecimal de un carácter especial. Por ejemplo, si en un formulario existe un cuadro de texto cuyo nombre es `nomusuario` y el usuario escribe *Sergio Luján Mora*, el navegador envía la cadena:

```
nombre=Sergio+Luj%E1n+Mora
```

donde los espacios en blanco y los caracteres acentuados aparecen codificados.

4.7.4. Explicación 43

(Pregunta 65, página 25)

Un programa **CGI** puede obtener la información que un cliente web le envía de cuatro formas diferentes:

- Por medio de la línea de comandos, usando una petición de consulta del tipo `ISINDEX`.
- A través de la **URL**, accediendo a `QUERY_STRING`, una variable de entorno que el servidor web instancia en la transferencia e invocación del **CGI**. Este es el sistema usado cuando el cliente envía los datos por un formulario con el método `GET`.
- Obteniendo los datos por la entrada estándar (`stdin`). Este es el caso cuando el cliente transmite la información por medio de un formulario con el método `POST`.
- Utilizando la información de ruta del *script*, ésta se puede obtener accediendo a la variable de entorno `PATH_INFO`.

Las variables de entorno `REQUEST`, `CONTENT` y `QUERY` no existen en la plataforma **CGI** y por tanto no son métodos válidos para acceder a la información que envía el cliente. La plataforma de comunicación **CGI** ofrece al programador un conjunto de variables de entorno con el fin de facilitar el acceso a información útil que el cliente envía o que se genere en la transferencia (ya sea la petición del cliente o la respuesta del servidor). Algunas de estas variables son:

- `ALL_HTTP`: Obtiene el valor de todas las cabeceras enviadas por el cliente.
- `ALL_RAW`: Obtiene todas las cabeceras en formato crudo tal cual las envía el cliente.

- `APPL_PHYSICAL_PATH`: Da acceso al directorio físico del 'root' del entorno de aplicación del sitio web.
- `AUTH_PASSWORD`: Permite acceder al valor de la contraseña aportada por el usuario en el dialogo de autentificación (solo si el método de autentificación es el básico).
- `AUTH_TYPE`: Obtiene el método de autentificación que el servidor utiliza cuando se intenta acceder a un recurso protegido.
- `AUTH_USER`: Nombre del usuario autentificado en el acceso al servidor.
- `CONTENT_LENGTH`: Número de bytes enviados con la petición a la entrada estándar del programa **CGI** (enviados desde el cliente con el método `POST`).
- `CONTENT_TYPE`: Tipo **MIME** de los datos enviados por el cliente mediante el método `POST`.
- `GATEWAY_INTERFACE`: Nombre y versión de la plataforma **CGI** que usa el servidor. Formato: CGI/Versión.
- `HEADER_<HeaderName>`: Obtiene el valor almacenado en la cabecera de nombre <HeaderName>. Es util para obtener cabeceras personalizadas del cliente.
- `HTTP_<HeaderName>`: Obtiene el valor almacenado en la cabecera de nombre <HeaderName>. Es util para obtener cabeceras personalizadas del cliente. Se diferencia del anterior en la forma de tratar los nombres de la cabecera.
- `HTTP_ACCEPT`: Tipos de datos que acepta el cliente. Formato: Lista de pares Tipo/Subtipo.
- `HTTP_REFERER`: Contiene la dirección desde donde el cliente ha accedido al servidor.
- `HTTP_USER_AGENT`: Nombre y versión del software que utiliza el cliente.
- `HTTPS`: Devuelve 'ON' si la petición vino de un canal seguro (por ejemplo SSL), devuelve 'OFF' en caso contrario.
- `HTTPS_KEYSIZE`: Número de bits de la clave de conexión SSL.
- `HTTPS_SECRETKEYSIZE`: Número de bits del certificado privado del servidor en unas conexión segura.
- `HTTPS_SERVER_ISSUER`: Proveedor del certificado seguro.
- `HTTPS_SERVER_SUBJECT`: Sujeto del cerfiticado seguro.

- `LOCAL_ADDR`: Retorna la dirección del servidor desde donde ha venido la respuesta.
- `LOGON_USER`: Obtiene el nombre de la cuenta que se ha conectado desde la web.
- `PATH_INFO`: Ruta del programa **CGI** invocado (porción de la **URL** a partir del nombre de servidor hasta el comienzo de los parámetros).
- `PATH_TRANSLATED`: La ruta física del **CGI** invocado.
- `QUERY_STRING`: Datos de la consulta enviada por el cliente con el método `GET` (porción de la **URL** a partir del nombre del script, a partir de la '?').
- `REMOTE_ADDR`: Dirección **IP** del cliente que ha efectuado la petición.
- `REMOTE_HOST`: Nombre de dominio del cliente (si es posible resolverla).
- `REMOTE_PORT`: El número del puerto de la conexión del cliente.
- `REMOTE_USER`: Nombre del usuario remoto que ha efectuado la petición.
- `REQUEST_METHOD`: Método utilizado en la petición. Puede tomar los valores: `GET`, `POST`, `HEAD`, etc.
- `SCRIPT_NAME`: Ruta virtual del programa **CGI** que se ha invocado.
- `SERVER_NAME`: Nombre del servidor (Nombre de dominio o si no lo posee, dirección **IP**).
- `SERVER_PORT`: Número de puerto donde el servidor ha recibido la petición **HTTP**.
- `SERVER_PORT_SECURE`: Obtiene un 1 si el puerto es un puerto seguro, 0 en otro caso.
- `SERVER_PROTOCOL`: Nombre y versión del protocolo que usa el servidor. Formato: Protocolo/Versión.
- `SERVER_SOFTWARE`: Nombre y versión del software del servicio web del servidor. Formato: Nombre/Versión.

4.8. SSI

4.8.1. Explicación 44

(Pregunta 17, página 13)

En **SSI**, para insertar en una página la fecha de la última modificación de un archivo se puede emplear:

```
<!-- #flastmod file="archivo" -->
```
y
```
<!-- #flastmod virtual="archivo" -->
```

Los atributos `virtual` y `file` indican el tipo de ruta de acceso que se emplea para localizar el archivo. El atributo `virtual` permite indicar una ruta de acceso que empieza en un directorio virtual. El atributo `file` indica una ruta de acceso relativa; las rutas relativas empiezan en el directorio que contiene el archivo que posee el comando **SSI**.

4.8.2. Explicación 45

(Pregunta 44, página 20) (Pregunta 100, página 33)

En **SSI**, para insertar el valor de una variable de entorno en una página se emplea:
```
<!-- #echo var="variable" -->
```

Algunos ejemplos del comando `#echo` (en cursiva aparece el comando **SSI** y a continuación el resultado que produce):

```
La fecha y hora de hoy es <!-- #echo var="DATE_LOCAL" -->
La fecha y hora de hoy es Thursday, 23-Aug-2001 12:57:24 EDT

Has llegado aquí desde la página <!-- #echo var="HTTP_REFERER" -->
Has llegado aquí desde la página http://www.ua.es/index.html

La hora en el meridiano de Greenwich es <!-- #echo var="DATE_GMT"
-->
La hora en el meridiano de Greenwich es Thursday, 23-Aug-2001 16:57:24
GMT

Tu IP es <!-- #echo var="REMOTE_ADDR" -->
Tu IP es 193.145.234.59

El nombre de esta página es <!-- #echo var="DOCUMENT_NAME" -->
El nombre de esta página es ssi-prueba.shtml

Tu navegador y sistema operativo es <!-- #echo var="HTTP_USER_AGENT"
-->
Tu navegador y sistema operativo es Mozilla/4.78 [en] (Windows NT 5.0;
U)
```

Por último, los comandos `#print` y `#getenv` no existen en **SSI**.

4.8.3. Explicación 46

(Pregunta 66, página 25) (Pregunta 81, página 29)

Con **SSI**, para insertar la información del tamaño de un fichero en una página web se utiliza:

```
<!-- #fsize parametro="valor" -->
```

donde `parametro` puede tomar los valores `file` o `virtual` y `valor` contendrá la ruta y el nombre del archivo. Esta instrucción mostrará el tamaño del fichero en Kilobytes o en la unidad que se haya especificado en la configuración de **SSI**.

La diferencia que existe al usar `file` o `virtual` radica en que `file` se usa para indicar rutas relativas a la carpeta actual donde está ubicado el archivo que contiene la instrucción, y `virtual` se usa para especificar rutas completas a partir del directorio principal (*root*) del sitio web.

En cuanto al resto de directivas:

- La directiva `#size` no existe en **SSI**.
- La directiva `#exec` se utiliza para invocar la ejecución de programas o comandos en el sistema operativo del servidor e insertar el resultado en la respuesta al cliente. Su sintaxis es:

  ```
  <!-- #exec parametro="valor" -->
  ```

  donde `parametro` es el tipo de programa que se desea invocar: `cgi`, `cmd`, `exe` y `script` (no acepta `file` como atributo). Y el valor se refiere a la ruta al programa o comando a ejecutar.

- La directiva `#include` es utilizada para insertar el contenido de otros ficheros en la página **HTML** del cliente. Su forma es:

  ```
  <!-- #include parametro="valor" -->
  ```

  donde `parametro` toma los valores `file` para rutas relativas a la carpeta actual y el valor `virtual` para especificar rutas absolutas a partir del directorio principal (*root*) del sitio web. El valor el atributo se refiere a la ruta y nombre de fichero que se desea volcar en la página **HTML**.

4.9. ASP

4.9.1. Explicación 47

(Pregunta 3, página 11) (Pregunta 92, página 31)

En una página **ASP**, para recuperar los valores de las variables de la cadena de consulta **HTTP** (en la **URL**) se emplea la colección `QueryString` del objeto `Request`.

La cadena de consulta **HTTP** se especifica mediante los valores que aparecen a continuación del signo de interrogación (`?`). Existen varias situaciones distintas en las que pueden haber datos en una cadena de consulta. Por ejemplo:

- En un enlace:
  `<A HREF="cliente.asp?id=123">José Pérez Pérez</A>`
  genera una variable llamada `id` con el valor `123`.
- En un formulario:
  `<FORM ACTION="familia.asp?id=123&orden=asc">`
  genera dos variables, una llamada `id` con el valor `123` y otra llamada `orden` con el valor `asc`.
- Al enviar un formulario mediante el método `GET`.
- Cuando un usuario escribe una consulta directamente en la barra de dirección de un navegador de Internet.

Además de `QueryString`, el objeto `Request` también posee otras colecciones que permiten acceder al resto de datos que transmite el navegador al servidor web durante una petición **HTTP**:

- `Form`: los valores de los controles de un formulario enviados en el cuerpo de una petición **HTTP** mediante el método `POST`.
- `Cookies`: los valores de las *cookies* enviadas junto con una petición **HTTP**.
- `ClientCertificate`: los valores de un certificado de seguridad X.509 que el navegador envía al servidor en una petición **HTTP**.
- `ServerVariables`: las variables de entorno del servidor.

Se puede acceder directamente a todas las variables si se emplea `Request` sin indicar el nombre de la colección. En tal caso, el servidor web busca en las colecciones del objeto `Request` en el siguiente orden:

1. `QueryString`.
2. `Form`.

3. `Cookies`.
4. `ClientCertificate`.
5. `ServerVariables`.

Si existe una variable con el mismo nombre en más de una colección, el objeto `Request` devuelve la primera instancia que encuentra. Para evitar ambigüedades y errores, se aconseja siempre emplear el nombre de la colección.

4.9.2. Explicación 48

(Pregunta 7, página 12) (Pregunta 97, página 32)

El código de la página **ASP** no se ejecuta, ya que el servidor web no procesa la página. La razón es que el protocolo de comunicación empleado es `file`, lo que significa que la página se está abriendo localmente, sin pasar por el servidor web. La página nunca va a ser ejecutada, independientemente de que el servidor web esté o no esté iniciado.

Si no se puede ejecutar una página **ASP** en un equipo, se tienen que realizar las siguientes comprobaciones:

- Comprobar que el servidor web (Microsoft Personal Web Server o Microsoft Internet Information Server) está correctamente instalado, bien configurado e iniciado.
- Comprobar que la **URL** está correctamente escrita. Tiene que aparecer una de las siguientes dos posibilidades:
  - `http://localhost/ruta/pagina.asp`
  - `http://nombreEquipo/ruta/pagina.asp`
- Comprobar que el archivo correspondiente a la página se encuentra en el lugar indicado por la **URL**.
- Comprobar que la página **ASP** no está abierta por otro programa. A veces, un programa abre un archivo en modo exclusivo e impide que otros programas puedan acceder a él.

4.9.3. Explicación 49

(Pregunta 13, página 13)

La propiedad `Session.Timeout` configura el tiempo de vida de una sesión en **ASP**.

La propiedad `Timeout` del objeto `Session`, un objeto integrado de **ASP**, especifica el periodo de tiempo de espera, en minutos, asignado al objeto `Session`. La sesión termina si el usuario no solicita o actualiza una página durante el tiempo de espera

establecido en esta propiedad. Al terminarse una sesión, se destruyen todos los objetos almacenados en el objeto `Session` y se liberan sus recursos.

El tiempo de espera predeterminado de una sesión es de 20 minutos. Este tiempo se puede modificar a través de la herramienta de administración de Microsoft Internet Information Server, tal como se aprecia en la Figura 4.11.

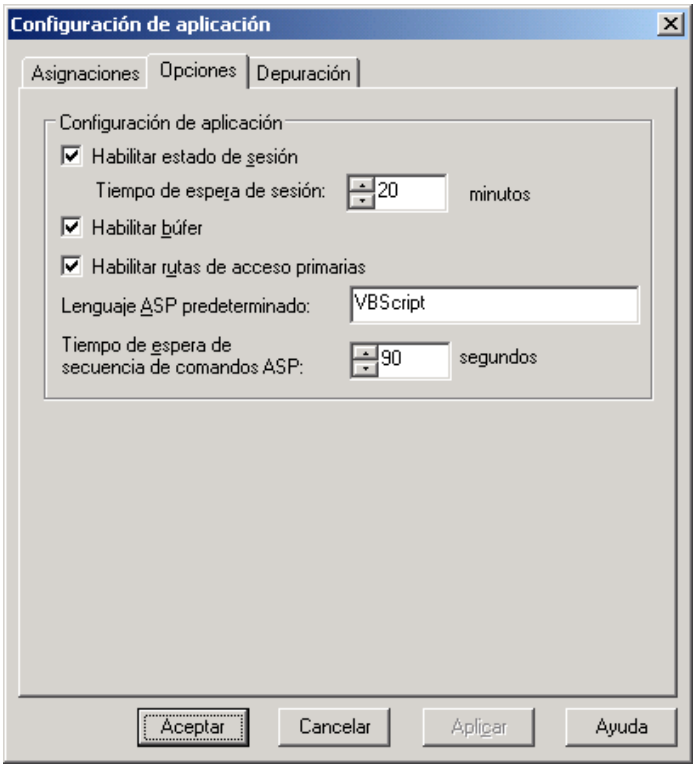


Figura 4.11: Configuración de parámetros de ASP en Microsoft Internet Information Server

Por último, también se puede terminar una sesión en cualquier momento con el método `Abandon` del objeto `Session`.

4.9.4. Explicación 50

(Pregunta 16, página 13)

Desde una página **ASP**, para enviar información al cliente se emplea el objeto `Response`.

Entre otras propiedades y métodos, este objeto posee el método `BinaryWrite` para escribir datos binarios, el método `Write` para escribir una cadena de texto en el resultado **HTTP** que se envía al cliente y el método `Redirect` para redirigir al cliente a una **URL** nueva.

4.9.5. Explicación 51

(Pregunta 18, página 14) (Pregunta 70, página 26)

El fichero `Global.asa` no es un fichero obligatorio en cualquier aplicación web realizada con **ASP**.

`Global.asa` es un archivo opcional en el que se definen secuencias de comandos de eventos y que permite declarar objetos con alcance de sesión o de aplicación. En él se declaran los eventos de inicio y finalización de los objetos `Application` y `Session`. La estructura típica de este fichero es:

```
<SCRIPT LANGUAGE="Lenguaje" RUNAT="Server">
Sub Application_OnStart
. . .
End Sub

Sub Application_OnEnd
. . .
End Sub

Sub Session_OnStart
. . .
End Sub

Sub Session_OnEnd
. . .
End Sub
</SCRIPT>
```

donde `Lenguaje` es cualquier lenguaje de programación (por ejemplo, *VBScript* o *JavaScript*) del que se disponga el correspondiente intérprete compatible con **ASP**.

El fichero `Global.asa` no es accesible desde el cliente. Si se solicita, el servidor web devuelve una página de error como la mostrada en la Figura 4.12.

Por último, el código incluido en el fichero `Global.asa` no puede escribir datos en la página devuelta al cliente.

4.9.6. Explicación 52

(Pregunta 22, página 14)

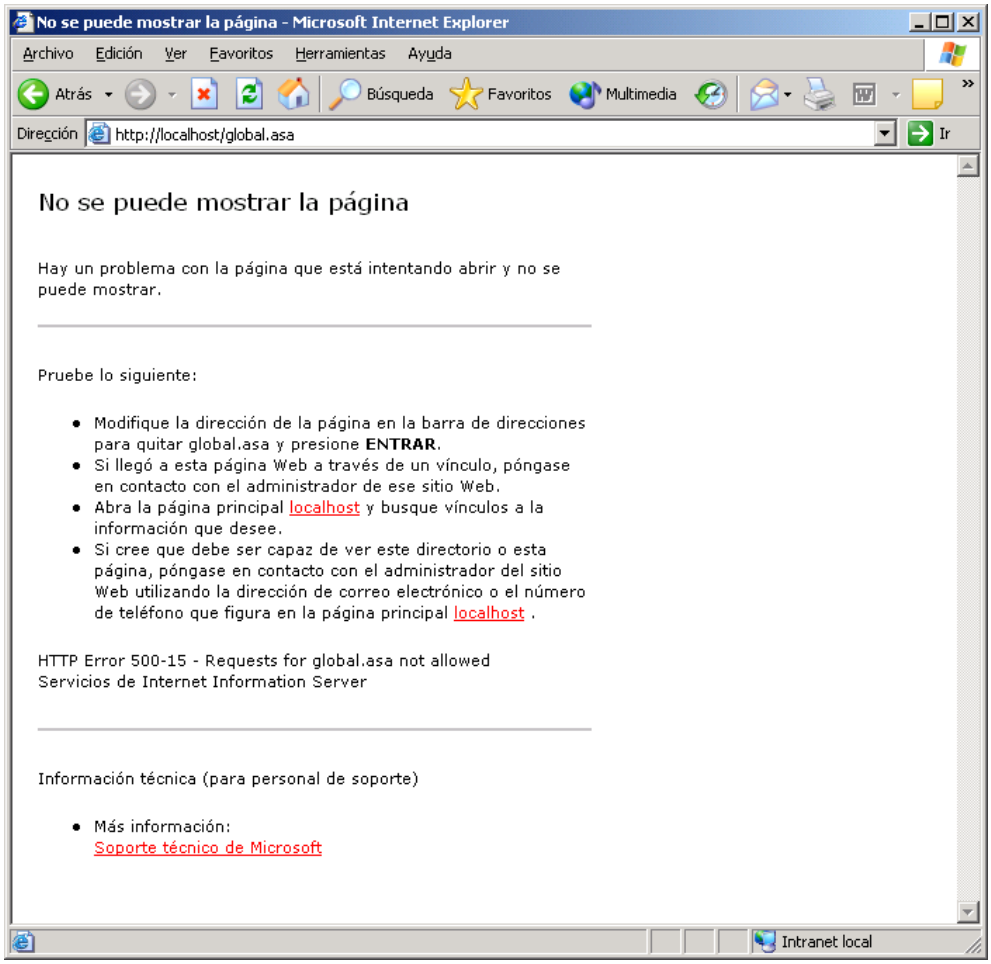


Figura 4.12: Página de error al solicitar el fichero Global.asa

Desde una página **ASP** sólo se puede acceder al sistema de archivos existente en el propio servidor web.

El sistema de archivos en el cliente es inaccesible desde una página **ASP**. Si fuese accesible, existiría un potencial agujero de seguridad, ya que cualquier servidor web al que se conectase un usuario podría acceder a sus archivos.

Una página **ASP** sólo puede acceder a un archivo que existe en el sistema de archivos del cliente si el usuario explícitamente envía el fichero al servidor web a través de un control de tipo `<input type="file">` en un formulario de una página web.

4.9.7. Explicación 53

(Pregunta 25, página 15)

En la instalación por defecto (estándar) de **ASP**, se puede emplear como lenguaje de programación tanto *VBScript* como *JavaScript*, ya que dispone de los intérpretes de ambos lenguajes.

El lenguaje principal de **ASP** es el lenguaje que se utiliza para procesar las instrucciones que se encuentren entre los delimitadores `<%` y `%>`. De forma predeterminada, el lenguaje principal de **ASP** es *VBScript*. El valor por defecto del lenguaje se puede establecer mediante la herramienta de administración de Microsoft Internet Information Server. En la Figura 4.11 se pude ver la ventana que permite definir el lenguaje de **ASP** predeterminado. También se puede emplear como lenguaje principal cualquier lenguaje de secuencia de comandos para el que se disponga de un intérprete.

En una página **ASP** se pueden escribir secuencias de comandos en distintos lenguajes de programación. Para ello se tiene que emplear la sintaxis:
`<SCRIPT LANGUAGE="Lenguaje" RUNAT="SERVER"> ... </SCRIPT>`
donde `Lenguaje` es el lenguaje de programación que se quiere emplear. El atributo `RUNAT` indica al servidor web que tiene que procesar el código. Sin este atributo, las secuencias de comando se enviarían al cliente y serían procesadas por su navegador.

Para establecer el lenguaje principal en una única página **ASP**, se tiene que añadir la directiva `<%@ LANGUAGE="Lenguaje" %>` al principio de la página, donde `Lenguaje` es el lenguaje principal de secuencia de comandos que se desea establecer en esa página concreta. Este valor sustituye al valor por defecto que se haya configurado para todas las páginas desde la herramienta de administración del servidor web.

4.9.8. Explicación 54

(Pregunta 31, página 17) (Pregunta 83, página 30)

El objeto `Application` permite emplear variables globales a todos los usuarios de una aplicación web. Por ello, se suele utilizar este objeto para compartir información entre todos los usuarios de una aplicación (en **ASP**, una aplicación es el conjunto de ficheros `.asp` de un directorio virtual y sus subdirectorios). Como al objeto

`Application` pueden acceder varios usuarios a la vez, los métodos `Lock` y `Unlock` de este objeto permiten garantizar un acceso exclusivo al objeto. El método `Lock` impide que otros clientes modifiquen las variables almacenadas en el objeto `Application`.

El objeto `Session` se suele emplear para almacenar información necesaria para una sesión de usuario determinada. Este objeto persiste (no se destruye) cuando el usuario pasa de una página a otra de la misma aplicación. Este objeto se crea automáticamente la primera vez que un usuario accede a una página de una aplicación y permanece en memoria hasta que se abandona la sesión (`Session.Abandon`) o hasta que el servidor web la destruye porque ha caducado (se ha superado el tiempo máximo de vida de la sesión establecido en `Session.Timeout`).

Por último, el objeto `Server` se emplea para acceder a ciertos métodos y propiedades relacionados con el servidor web.

En el Cuadro 4.3 se resumen las colecciones, eventos, métodos y propiedades de los principales objetos de **ASP**.

Objeto	Colecciones	Eventos	Métodos	Propiedades
Application	Contents StaticObjects	Application_OnStart Application_OnEnd	Lock UnLock	
Server			CreateObject HTMLEncode MapPath URLEncode	ScriptTimeout
Session	Contents StaticObjects	Session_OnStart Session_OnEnd	Abandon	CodePage LCID SessionId TimeOut

Cuadro 4.3: Colecciones, eventos, métodos y propiedades de los objetos Application, Server y Session

Preguntas relacionadas: 50
Explicaciones relacionadas: 57

4.9.9. Explicación 55

(Pregunta 37, página 18) (Pregunta 76, página 29)

ASP son las siglas de *Active Server Pages*. **ASP** es una tecnología de secuencias de comandos del servidor web que permite crear páginas web dinámicas en el servidor. **ASP** es una tecnología propietaria de MICROSOFT. Cuando un servidor web recibe una petición de una página **ASP**, procesa las secuencias de comando del servidor que contiene la página para generar el documento que se envía al cliente. Una página **ASP** puede contener **HTML** y secuencias de comandos del servidor. Las páginas **ASP** suelen tener la extensión `.asp`.

4.9.10. Explicación 56

(Pregunta 47, página 20)

Para finalizar la ejecución de una página **ASP** se emplea el método `Response.End`. Al llamar a este método, se envía el resultado que se haya producido hasta el momento y se detiene el procesamiento del resto de la página.

El método `Response.Clear` se emplea para borrar cualquier resultado **HTML** almacenado en el búfer de salida (sólo se borra el cuerpo de la respuesta y no cualquier encabezado que se haya emitido con el método `Response.AddHeader`). Si el búfer no está activado (por ejemplo, porque se ha desactivado con la instrucción `Response.Buffer`) se producirá un error en tiempo de ejecución.

Por último, `Response.Stop` no existe en **ASP**.

4.9.11. Explicación 57

(Pregunta 50, página 21)

En **ASP**, cada usuario tiene su objeto `Session`, pero todos los usuarios comparten el mismo objeto `Application`.

El servidor web crea automáticamente un objeto `Session` cuando un usuario que aún no tiene una sesión solicita una página de una aplicación. Del mismo modo, el servidor web destruye automáticamente el objeto `Session` de un usuario cuando caduca o se abandona la sesión.

En **ASP**, se crea un objeto `Application` por cada aplicación que posea el servidor web. Una aplicación se define como el conjunto de archivos `.asp` de un directorio virtual y sus subdirectorios.

Preguntas relacionadas: 31

Explicaciones relacionadas: 54

4.9.12. Explicación 58

(Pregunta 68, página 26)

La instrucción `<% Response.Write Time %>` mostrará en el cliente la hora del servidor donde la página **ASP** esté alojada, porque el código lo interpreta el servidor antes de elaborar una respuesta en formato **ASCII** o **HTML** y enviarla al cliente.

Ni `<@ RUNAT="Server|Client" @>` ni `Request.Time` existen, así que esas respuestas son falsas.

Respecto a RUNAT, es un atributo de la etiqueta <SCRIPT>. Así, en el caso de escribir:

```
<SCRIPT LANGUAJE="VBScript" RUNAT="Client">
  Response.Write Time
</SCRIPT>
```

el código no se interpretaría en el servidor, sino que se transmitiría al servidor, y entonces se generaría un error de cliente puesto que el navegador o bien no entiende *VBScript* (sólo Microsoft Internet Explorer es capaz de interpretar *VBScript*) o bien

no tiene acceso al objeto `Response` que no está en el modelo de objetos de documento de las páginas web, puesto que este objeto pertenece al conjunto de objetos accesibles desde una página **ASP** en el servidor.

4.9.13. Explicación 59

(Pregunta 69, página 26) (Pregunta 79, página 29)

Para redirigir al usuario automáticamente a otra **URL** desde **ASP** se debe invocar el método `Redirect` del objeto `Response`, de esta forma:

```
Response.Redirect URL
```

donde **URL** contiene la dirección de Internet hacia la cual el navegador del cliente sera redirigido.

Las otras respuestas posibles son erróneas porque no responden a ningún método de los objetos especificados:
`Session.Redirect`, `Request.Redirect` y `Server.Redirect`
no existen.

4.9.14. Explicación 60

(Pregunta 71, página 26) (Pregunta 87, página 30)

Para acceder a las variables de entorno desde **ASP** se debe invocar el método `ServerVariables` del objeto `Request`, de esta forma:

```
Request.ServerVariables("variableDeEntorno")
```

Siendo 'variableDeEntorno' el nombre de la variable de entorno de servidor a la que se desea acceder para consultar su valor.

Las otras respuestas posibles son erróneas porque no responden a ningún método de los objetos especificados.

Preguntas relacionadas: 65

Explicaciones relacionadas: 43

4.9.15. Explicación 61

(Pregunta 72, página 26)

Los datos enviados desde un formulario **HTML** con el método `GET` se pueden recuperar desde un script **ASP** utilizando la función `QueryString` del objeto `Request`. El acceso a cada variable individual se puede realizar de dos formas, mediante un número que representa el orden o mediante una cadena que represente el nombre de la variable. Ejemplos:

```
' Accedemos a la primera variable de la QueryString.
vMiVar = Request.QueryString(0)
' Accedemos a la variable 'pVariable'.
vMivar2 = Request.QueryString("pVariable")
```

En cambio, los datos enviados desde un formulario **HTML** con el método `POST` se deben recuperar con la función `Form` del mismo objeto `Request`. La forma de acceder a cada dato en concreto es la misma que en el caso anterior. Ejemplos:

```
' Accedemos a la segunda variable del formulario recibido
' por POST
vMiVar3 = Request.Form(1)
' Accedemos a la variable 'pVariable2'
vMivar4 = Request.Form("pVariable2")
```

La opción `Request.GetForm` no existe (por lo que provocaría un error) puesto que `GetForm` no es un método del objeto `Request`.

4.10. Java

4.10.1. Explicación 62

(Pregunta 4, página 11) (Pregunta 34, página 18)

Los orígenes de *Java* se sitúan en un proyecto iniciado por la empresa Sun Microsystems en 1991 cuyo objetivo era facilitar el desarrollo de software avanzado para dispositivos electrónicos de consumo (relojes, calculadoras, etc.). Por tanto, el propósito inicial no fue, ni mucho menos, el desarrollo de aplicaciones en Internet. Sin embargo, con el paso del tiempo sí que se han incluido numerosos recursos para programar en red en general y en especial en Internet.

Java es un lenguaje multiplataforma, ya que al compilarse un fichero fuente en *Java* no se genera código máquina de una plataforma concreta, sino código (*bytecodes*) para una máquina virtual (la **MVJ**).

Java es un lenguaje fuertemente tipado (*strongly typed*): cada variable, cada expresión, tiene un tipo que se conoce en tiempo de compilación.

Por último, en *Java* no hace falta liberar memoria cuando ya no se emplea (no existe un equivalente a la sentencia `free` en C o `delete` en *C++*). Ello es posible gracias a que incorpora un recolector de basura (*garbage collector*), que "sabe" cuando un objeto ya no se usa y se puede liberar la memoria que ocupa.

4.10.2. Explicación 63

(Pregunta 75, página 27)

El lenguaje de programación *Java*, aún estando muy ligado al mundo de Internet, fue desarrollado por una empresa que nada tiene que ver con las entidades normalizadoras o recomendadoras de la red (**W3C**, *Internet Society* (**ISOC**), etc.). En concreto, *Java* fue creado por Sun Microsystems.

Una de las mejoras o innovaciones que se deseaba aportar con la creación de este lenguaje era su característica de ser multiplataforma, es decir, que un mismo programa, con el mismo código fuente pudiese funcionar en cualquier plataforma y sistema operativo que soportase una **MVJ**.

En la actualidad *Java* está soportado en múltiples plataformas (Sun Microsystems, Intel, IBM y Apple) y en distintos sistemas operativos, como Microsoft Windows, Apple MacOS, y distintas variantes de Unix (Linux, IRIX, AIX, Solaris, etc.) entre otros. Para cada una de estas plataformas y sistemas operativos se ha desarrollado una máquina virtual o intérprete capaz de ejecutar programas *Java* compilados.

Otra de las características que aporta este lenguaje es que es orientado a objetos (que es distinto de basado en objetos). Todo en *Java* es un objeto (salvo algunas pocas excepciones, como los tipos básicos de datos). Por tanto, permite aprovechar las características de este paradigma de programación: encapsulación, reutilización, privacidad, sobrecarga, polimorfismo, herencia, etc. Además, por esto mismo, *Java* es un lenguaje fuertemente tipado que obliga al programador a declarar las variables antes de usarlas, definir su tipo de dato (entero, booleano, etc.), jugar con conversiones de tipo, etc.

4.11. JSP

4.11.1. Explicación 64

(Pregunta 8, página 12) (Pregunta 94, página 32)

La ejecución de una página **JSP** es un proceso compuesto de varios pasos (Figura 4.13). Al recibir una petición de una página **JSP**, primero se comprueba si existe el correspondiente servlet. Si no existe o si la página **JSP** y su correspondiente servlet no están actualizados (porque el código de la página **JSP** ha cambiado), se interpreta el **JSP** y se genera el código del servlet equivalente. A continuación, el código del servlet se compila y se generan los *bytecodes*. Se carga en la memoria del servidor los *bytecodes* del servlet y se ejecutan. La correspondiente respuesta se envía al cliente.

Todo este proceso se realiza la primera vez que se invoca una página **JSP**. Las siguientes veces, como el servlet ya existe, únicamente se tiene que comprobar si el servlet está cargado en la memoria del servidor. Si no está, entonces se carga el servlet en la memoria. A continuación se ejecuta y se envía la respuesta al cliente.

Los servidores poseen una memoria caché donde se almacenan los servlets que se ejecutan. Para gestionar esta memoria se emplean diversos algoritmos: LRU (*less recently used*), LFU (*less frequently used*), etc.

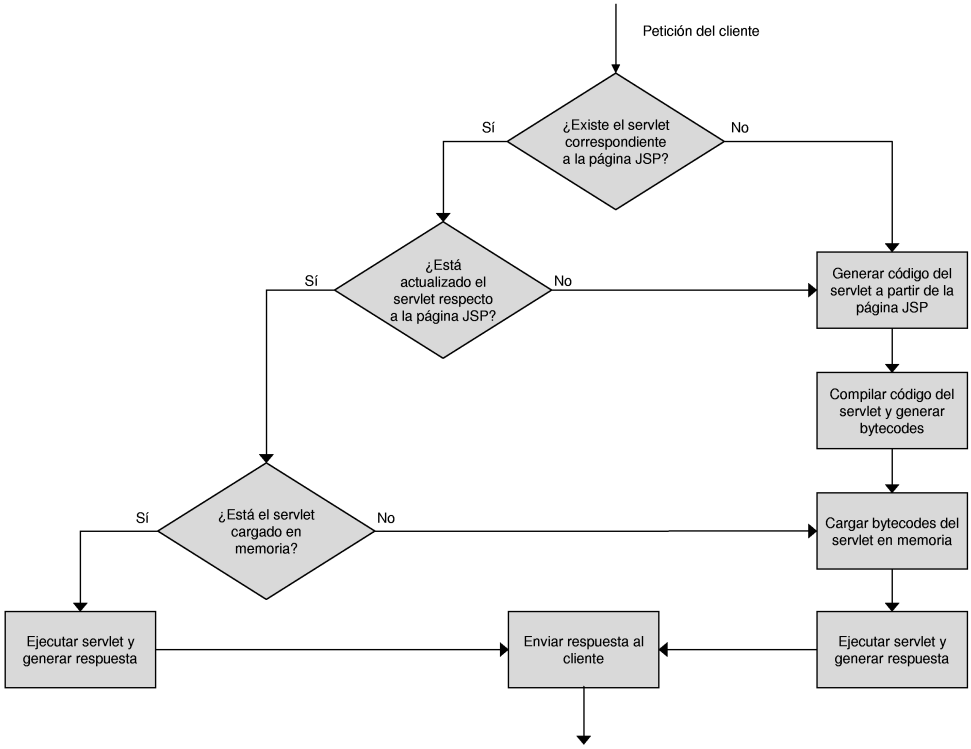


Figura 4.13: Ciclo de vida de una página JSP y su correspondiente servlet

4.11.2. Explicación 65

(Pregunta 11, página 12)

Una expresión es una forma rápida y cómoda de emitir volcados de datos en formato `String` a la página de respuesta, más concretamente al objeto `Response` que posteriormente sera enviado al cliente y que éste visualizará en forma de página web. Su sintaxis es:

```
<%= código %>
```

También se puede emplear la sintaxis basada en **XML**:

```
<jsp:expression>código</jsp:expression>
```

Las acciones en **JSP** son utilizadas para aportar prestaciones adicionales al **JSP**, como: acceso a componentes software, inclusión de contenido de ficheros, etc. Éstas únicamente se pueden expresar en formato **XML**. Se trata básicamente de funciones predefinidas que actúan sobre los objetos embebidos del sistema. Además, por medio de las directivas de definición de etiquetas nuevas, estas se invocan como si se tratase de acciones, usando la misma sintaxis. Algunos ejemplos de acciones en **JSP** son:

- `<jsp:useBean .... />`, para invocar y declarar un componente software y poder acceder a sus métodos y propiedades.
- `<jsp:include ... />`, para incluir el contenido de ficheros externos en el **JSP**.
- `<jsp:forward ... />`, para redirigir al cliente a otra **URL**.

El propósito de las directivas en **JSP** es el de poder configurar el motor **JSP** y el entorno de compilación y ejecución. Además, permiten definir etiquetas nuevas para añadir funcionalidades nuevas a las aplicaciones. Su sintaxis es:

```
<%@ directiva atributo="valor" %>
```

o en formato **XML**:

```
<jsp:directive.directiva atributo="valor" />
```

Hay tres conjuntos de directivas:

- `Page`, permite configurar parámetros del **JSP**.
- `Include`, inserta código procedente de otra **URL**, en nuestro **JSP**.
- `Taglib`, permite definir etiquetas personalizadas. Estas etiquetas se añaden al conjunto de acciones disponibles en el sistema y se invocan como tales.

Por tanto, la respuesta "Es necesario hacer una operación de conversión (cast) cuando en una expresión se devuelvan datos que no sean de tipo `String`" es la afirmación falsa y se explica completando la definición de expresión, donde añadimos que "es una forma rápida y cómoda de emitir volcados de datos en formato `String` a la página web, realizando si es necesario una conversión de tipo (*cast*) automática". Es decir, el interprete de **JSP** realiza automáticamente una conversión de tipo al volcar los datos al objeto `Response`, evitando al programador de realizar esta tarea.

4.11.3. Explicación 66

(Pregunta 27, página 17) (Pregunta 85, página 30)

Para cerrar una sesión de usuario con **JSP** debemos invocar el método `invalidate` del objeto `session` como a continuación se indica:

```
session.invalidate();
```

En el caso de `session.abandon`, esta es la opción de cierre de una sesión en la tecnología de **ASP**. Cualquier otra opción es incorrecta al no existir los métodos en los objetos que se ofrecen como posible respuesta: `server.invalidate` y `response.abandon`.

No obstante, la sesión se cierra automáticamente si el usuario que visita nuestro servidor permanece un determinado tiempo sin realizar ninguna interacción con el servidor. Este periodo, llamado *session timeout*, tiempo de espera de la sesión, se puede configurar desde código y desde la propia configuración del servidor web o del servidor de aplicaciones.

4.11.4. Explicación 67

(Pregunta 46, página 20)

Para redirigir al usuario automáticamente a otra **URL** desde **JSP** se debe invocar el método `sendRedirect` del objeto `response`. Por ejemplo:

```
response.sendRedirect(URLDestino);
```

El parámetro `URLDestino` es una cadena de caracteres que contiene la dirección hacia donde vamos a dirigir al cliente.

Por otro lado, `response.redirect` es la instrucción utilizada en **ASP** para redirigir al usuario a una nueva **URL** (pero sin los paréntesis). Las otras respuestas posibles (`session.redirect()` y `request.redirect()`) son erróneas porque no responden a ningún método de los objetos especificados.

4.11.5. Explicación 68

(Pregunta 73, página 27)

La empresa SUN MICROSYSTEMS desarrolló el lenguaje *Java* y las tecnologías de programación en Internet para el servidor basadas en este lenguaje, como Servlet y **JSP**. La primera propuesta de desarrollo desde el lado del servidor fue Servlet, que básicamente consisten en programas desarrollados en *Java* que incluyen unas determinadas bibliotecas de funciones que les permiten interactuar con el servidor web (acceso a objetos y datos de sesión, recuperación de datos de un formulario, etc). En otras palabras, son programas **CGI** realizados en *Java* con una version de la plataforma **CGI** mejorada.

Uno de los inconvenientes de esta tecnología consiste en que el desarrollo está orientado al lado del servidor y es difícil integrar y mantener el código de cliente generado, pues está embebido en las instrucciones del código de servidor, algo que dificulta el mantenimiento de la aplicación y su depuración, sobre todo en lo que respecta al código de cliente, al diseño de la web. Es decir, que las etiquetas **HTML** de la página de respuesta están entremezcladas con las instrucciones del programa *Java*, del Servlet.

A partir de esta tecnología, se desarrolló **JSP** que aportó mejoras sensibles en el desarrollo para Internet con *Java*. **JSP** tiene mejores prestaciones que Servlet, entre las cuales cabe destacar:

- Posibilidad de ampliar el conjunto de etiquetas **HTML** con etiquetas extendidas en **XML** que permiten una mejor reutilización y encapsulación del código.
- Programación de servidor con tecnología de *web scripting* que permite mezclar código de cliente y de servidor de forma sencilla, facilitando el desarrollo, depuración y el mantenimiento.

Además, al estar ambas tecnologías basadas en *Java*, otra ventaja adicional es que pueden disfrutar de todas las prestaciones del lenguaje, como la posibilidad de trabajar con componentes software de *Java*, llamados JavaBeans, que permiten llamadas a métodos de objetos remotos, serialización, encapsulación de aplicaciones y reutilización de componenetes software, orientación a objetos, posibilidades multiplataforma, etc.

4.11.6. Explicación 69

(Pregunta 74, página 27) (Pregunta 99, página 33)

Para comentar código con la tecnología **JSP** existen diversas formas. Los comentarios de una sola línea dentro de un *scriptlet* se indican con la doble barra de dividir (igual que con *Java*, *C* y *C++*). Por ejemplo:

```
// Esto es un comentario de una sola línea
```

Podemos realizar comentarios de más de una línea, para escribir párrafos, encabezados de funciones o incluso para eliminar un fragmento de código. Este tipo de comentarios se indican con `/*` para abrir el comentario y `*/` para cerrarlo, igual que con *Java*, *C* y *C++*. Por ejemplo:

```
/* Comentario de
   mas de una línea
   de código */
```

También tenemos otra forma de comentar código, también multilínea, pero en este caso la sintaxis es exclusiva de **JSP**. Usa `<%-` para abrir el bloque comentado y `-%>` para cerrarlo. Por ejemplo:

```
<%-- Comentario de
     mas de una línea
     de código --%>
```

Esta última forma es la más cómoda para comentar acciones, directivas y demás elementos exclusivos de la tecnología **JSP**.

En el caso de trabajar con **ASP**, los comentarios de código de servidor se realizan utilizando la palabra reservada `Rem` o su abreviatura, el apóstrofe (`'`). En ambos casos, el comentario abarca una sola línea. Por ejemplo:

```
REM Comentario en Visual Basic y ASP.
' Comentario utilizando la abreviatura.
```

Bibliografía recomendada

A continuación se incluye una serie de libros recomendados agrupados en una serie de categorías: General, HTML, Diseño web, JavaScript y Programación de servidor.

General

- Eduardo Parra Murga. *Diccionario de Internet*. Nóesis, Madrid, 1996
- Andrew S. Tanenbaum. *Redes de computadores*. Prentice-Hall, 1997
- Jesús Bobadilla Sancho. *Superutilidades para Webmasters*. Osborne McGraw-Hill, Madrid, 1999
- H. M. Deitel, P. J. Deitel, T. R. Nieto. *Internet and World Wide Web. How to program*. Prentice Hall, New Jersey, 2000
- Sergio Luján Mora. *Programación en Internet: Clientes Web*. Club Universitario, Alicante, 2001
- Sergio Luján Mora. *Programación de aplicaciones web: historia, principios básicos y cliente web*. Club Universitario, Alicante, 2002
- Chris Bates. *Web Programming: Building Internet Applications*. John Wiley & Sons, New York, 2002

HTML

- Sergio Ríos Aguilar. *Lenguajes HTML, Java y CGI. El diseño de páginas Web para Internet a su alcance*. Abeto Editorial, Madrid, 1996
- Ian S. Graham. *HTML Sourcebook: A Complete Guide to HTML 3.2 and HTML Extensions*. John Wiley & Sons, New York, 1997
- Molly Holzschlag. *Special Edition Using HTML 4*. Que, Macmillan Computer Publishing, 1999

Diseño web

- Sarah Horton Patrick J. Lynch. *Principios de diseño básicos para la creación de sitios web.* Ediciones G. Gili, México, 2000
- Jakob Nielsen. *Usabilidad: diseño de sitios Web.* Prentice Hall, Madrid, 2000
- Steve Krug. *No me hagas pensar: una aproximación a la usabilidad en la Web.* Prentice Hall, Madrid, 2001

JavaScript

- Danny Goodman. *Programación en JavaScript.* Vía@Internet, Anaya Multimedia, Madrid, 1996
- Oscar González Moreno. *Programación en JavaScript.* Guías Prácticas, Anaya Multimedia, Madrid, 1998
- José Manuel Alarcón. *Programación en JavaScript (actualizada hasta JavaScript 1.3 y JScript 5).* Guías Prácticas, Anaya Multimedia, Madrid, 2000

Programación de servidor

- Marty Hall. *Core Servlets and JavaServer Pages.* Prentice Hall PTR, 2000
- Sergio Luján Mora. *Programación de servidores web con CGI, SSI e IDC.* Club Universitario, Alicante, 2001
- Jorge Serrano Pérez. *Programación con ASP 3.* Anaya Multimedia, Madrid, 2001
- Damon Hougland, Aaron Tavistock. *Core JSP.* Prentice Hall PTR, 2001

Índice alfabético