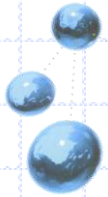




CC68J APLICACIONES EMPRESARIALES CON JEE



PROTOCOLO HTTP

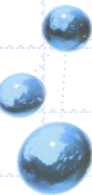
Entendiendo como se realiza la comunicación

Profesores:

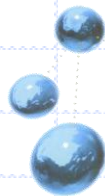
■ Andrés Farías

Objetivos: aprender a...

1. ¿Qué es y para qué sirve el protocolo HTTP?
2. Estándares sobre los que se basa el protocolo.
3. Como se utiliza el protocolo
4. Su sintaxis.

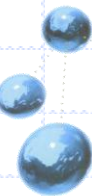


INTRODUCCIÓN



Estándares del Web

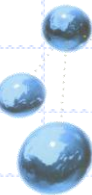
- El Web está basado en estándares de comunicación y de documentos.
 - ✓ Muchos tipos de navegador sobre diferentes plataformas,
 - ✓ Existen muchas implementaciones de servidores web.
- El Web es abierto respecto a los tipos de recursos que pueden ser publicados y compartidos en él.
 - ✓ Páginas web (archivos de texto), imágenes, sonido, video, etc.
 - ✓ Si alguien inventa un nuevo formato los archivos en dicho formato pueden ser publicadas inmediatamente en el Web.
 - ✓ Los navegadores están diseñados para acomodar la nueva funcionalidad de presentación en forma de aplicaciones colaboradoras y conectores (plug-ins).



Los estándares para el web

Sus componentes

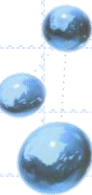
- El Web está basado en tres componentes tecnológicos de carácter estándar básicos:
 - ✓ URLs como identificadores generales de recursos.
 - ✓ HTTP como protocolo para acceder a recursos que utilizan URLs.
 - ✓ HTML para definir la estructura de los documentos.



Los estándares para el web

HTTP y URL's

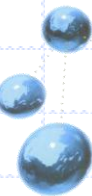
- HTTP/CGI fue el modelo precursor de desarrollo de aplicaciones basadas en Web.
- HTTP – **HYPERTEXT TRANSFER PROTOCOL** provee una semántica simple similar a RPC sobre sockets.
- HTTP puede acceder recursos basados en URL's.
- Principalmente estos recursos son páginas HTML y objetos MIME (**MULTIPURPOSE INTERNET MAIL EXTENSION**).
- Un URL – **UNIVERSAL RESOURCE LOCATOR** es un esquema de nombre universal para identificar todos los recursos Web.



El protocolo HTTP

Overview del protocolo HTTP

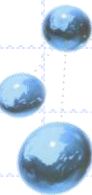
- Cliente (browser) – Página HTML – Tag especial <FORM>
 - ✓ Envía el requerimiento HTTP al servidor enviando los parámetros y el programa a ejecutar.
- El Servidor HTTP recibe el requerimiento y pasa los parámetros y el requerimiento a otro programa usando el protocolo CGI – Common Gateway Interface
- El servidor CGI puede estar escrito en cualquier lenguaje que pueda leer la entrada/salida estándar y variables de ambiente.
- El servidor HTTP lanza el programa CGI por cada requerimiento.
- El programa se ejecuta y retorna los resultados en formato HTML/HTTP hacia el servidor.
- El servidor a su vez retorna el resultado en el mismo formato hacia el browser.
- Clientes y Servidor HTTP usan MIME para la representación de datos para describir el contenido de los mensajes.



El protocolo HTTP

Principales características

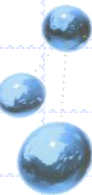
- Varias versiones:
 - ✓ HTTP/0.9
 - ✓ HTTP/1.0
 - ✓ HTTP/1.1
- HTTP es un protocolo del estilo RPC sobre TCP/IP
- HTTP es stateless
 - ✓ Protocolo Request / Response
 - ✓ Cliente -> Servidor (Request)
 - ✓ Servidor -> Cliente (Response)
- HTTP Data representation
 - ✓ Pasa datos auto-descritos
 - ✓ El cliente informa al servidor que representación de datos puede entender.
 - ✓ Cliente y Servidor negocian los tipos de datos



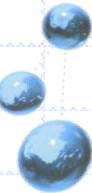
El protocolo HTTP

El protocolo de transporte y las transacciones

- En el Internet, las comunicaciones HTTP generalmente toman lugar sobre conexiones TCP.
 - ✓ El puerto por omisión es el 80, pero también pueden ser usados otros puertos.
 - ✓ Sin embargo, HTTP puede ser implementado sobre cualquier otro protocolo o red.
 - ✓ HTTP solo requiere un protocolo de transporte confiable.
- Una transacción HTTP es dividida en cuatro pasos:
 - ✓ El navegador abre una conexión
 - ✓ El navegador envía una solicitud al servidor
 - ✓ El servidor envía una respuesta al navegador
 - ✓ La conexión es cerrada

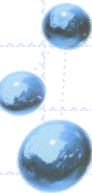
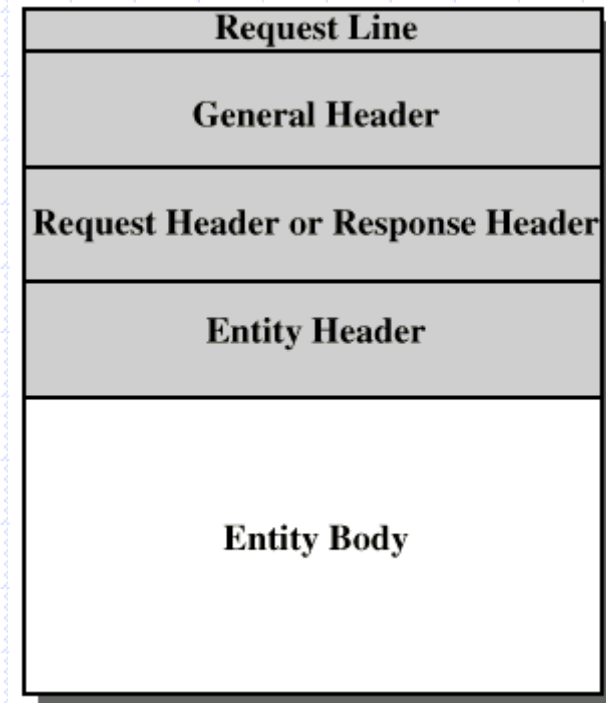


EL PROTOCOLO HTTP



Solicitud HTTP

- Línea de encabezado: método (o comando), recurso destino, y versión HTML.
- Campos de encabezado: información adicional sobre la solicitud y el cliente.
- Cuerpo de entidad: a veces utilizado para transferir información.

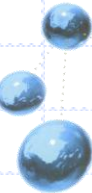


Solicitud HTTP

Un pequeño ejemplo

```
<method><resource identifier><HTTP version><crLf>  
[<Header> : <value>]<crLf>  
    ...  
[<Header> : <value>]<crLf>  
    blank line    <crLf>  
[Entity body]
```

```
GET /path/file.html HTTP/1.0  
Accept: text/html  
Accept: audio/x  
User-agent: MacWeb
```



Los mensajes HTTP

Los mensajes

■ Requests

- ✓ Request line
- ✓ General header
- ✓ Request header
- ✓ Entity header
- ✓ Entity body

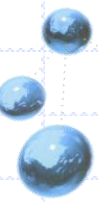
■ Responses

- ✓ Response line
- ✓ General header
- ✓ Response header
- ✓ Entity header
- ✓ Entity body

Los métodos

- GET: Recuperar el URL especificado.
- HEAD: Recuperar sólo el encabezado.
- POST: Enviar datos al URL especificado.
- PUT*: Almacenar datos.
- PATCH*: Almacenar diferencias.
- COPY*: Copiar recursos
- MOVE*: Mover recurso
- DELETE*: Borrar el recurso.
- LINK*: Establecer liga entre recursos.
- UNLINK*: Eliminar liga entre recursos.
- TRACE*: Notificar recepción de datos.
- OPTIONS*: Solicitar información sobre opciones de comunicación
- WRAPPED*: Empaca la solicitud

* Disponibles en HTTP/1.1



Los mensajes HTTP

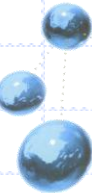
General & Request Header Fields

General Header Fields

- Cache control
- Connection
- Data
- Forwarded
- Keep alive
- MIME version
- Pragma
- Upgrade

Request Header Fields

- Accept
- Accept charset
- Accept encoding
- Accept language
- Authorization
- From
- Host
- If modified since
- Proxy authentication
- Range
- Referrer
- Unless
- User agent



El Response

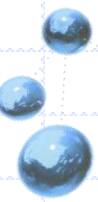
El mensaje de respuesta

¿Qué es?

- Línea de estado (status) seguida por uno o más respuestas y encabezados generales, seguidos por un cuerpo de entidad opcional.
- Status-Line =
HTTP-Version <SP> Status-Code
<SP> Reason-Phrase <CRLF>

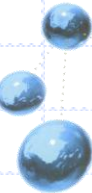
Códigos de status

- Informational (1xx)
- Successful (2xx)
 - ✓ 200: OK
- Redirection (3xx)
- Client error (4xx)
 - ✓ 401: No autorizado.
 - ✓ 403: Prohibido.
 - ✓ 404: No encontrado.
- Server error (5xx)
 - ✓ 500: Error interno
 - ✓ 503: No disponible
 - ✓ 504: Time-out
 - ✓ 505: Versión no soportada.



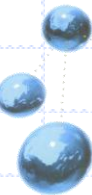
Response Header Fields

- Location
- Proxy authentication
- Public
- Retry after
- Server
- WWW-Authenticate



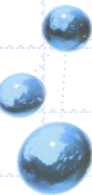
Entity Header Fields

- Allow
- Content encoding
- Content language
- Content length
- Content MD5
- Content range
- Content type
- Content version
- Derived from
- Expires
- Last modified
- Link
- Title
- Transfer encoding
- URL header
- Extension header



Entity Body

- Secuencia arbitraria de bytes.
- HTTP transfiere cualquier tipo de datos, incluyendo:
 - ✓ text
 - ✓ binary data
 - ✓ audio
 - ✓ images
 - ✓ video
- Interpretación del dato determinado por los campos del encabezado.
 - ✓ Content encoding, content type, transfer encoding



Ejemplos

Request / Response

Request

GET /path/file.html HTTP/1.0

Accept: text/html

Accept: audio/x

Host: 200.12.180.4

User-Agent: MacWeb

Response

HTTP/1.0 200 OK

Server: NCSA/1.3

Mime_version: 1.0

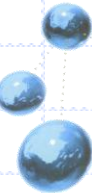
Content_type: text/html

Content_lenght: 2000

<HTML>

...

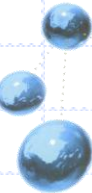
</HTML>



Tipos de peticiones

GET vs POST

- Cuando se envía una petición con el método **GET**, los parámetros son enviados como parte del URL.
- **GET** causa que los campos de entrada de la forma “name=valor” sean adicionados al final del URL despues de un “?”
 - ✓ Ejemplo:
`http://www.server.com/page.jsp?name1=value1&name2=value2&name3=value3`
- **GET** causa que el servidor Web parsee la URL e inserte el string después del “?” en una variable de ambiente llamada “query-string”
- **POST** adiciona el contenido en el body de un mensaje HTML
- El programa en el servidor lee los campos de entrada de la entrada estándar.
- Cual es mejor?
 - ✓ Muchas veces algunos S.O limitan el tamaños de las variables de ambiente a 256 – 1024 bytes
 - ✓ Lo más recomendado sería el método **POST** que no tiene limitaciones de tamaño.





CONSULTAS?