HTTP (Hipertext Transfer Protocol)

HTTP es el protocolo para comunicación entre servidores y navegadores de web. Este es un protocolo sin estado (stateless), cuyas conexiones usan el protocolo TCP\IP para transferir datos los cuales codifica usando MIME (Multiporpose Internet Mail Extensions). El Protocolo básico define una secuencia de cuatro pasos para cada petición de un cliente al servidor.

1.- Realizar la conexión
El cliente establece una conexión TCP con el servidor, en el puerto 80 por default.

2.- Hacer la Petición
EL cliente envía un mensaje al servidor, solicitando una pagina en el url especificado. La forma general de esta petición es:

MetodoHTTP URLRelativo VersionHTTP

Donde MetodoHTTP puede ser: GET, POST o HEAD

Opcionalmente a la línea del método, una petición del cliente puede incluir otra información. Finalmente una línea en blanco termina la petición

3.- La Respuesta
La respuesta del servidor al cliente empieza con un código de respuesta. A continuación se muestra algunas cadenas que puede ser enviadas como códigos de respuesta

"http\1.0 200 OK\r\n"
"http\1.0 403 forbidden\r\n"
"http\1.0 404 Not found\r\n"
"http\1.0 500 Internal Server Error\r\n”

El código de respuesta va seguido por información de cabecera MIME, luego una línea en blanco y luego el documento solicitado o un mensaje de error

4.- Cerrando la conexión
Ya sea el cliente el servidor o ambos cierran la conexión. Por tanto cada petición usa una conexión de red

Ejemplo de una petición de un cliente y la respuesta del servidor

Petición

ENCABEZADO DE PETICIÓN
GET /index.html HTTP/1.0
Accept: text/html
Accept: text/plain
User-Agent: Lynx/2.4 libwww/2.1.4
[una línea en blanco conteniendo solo CRLF]

Respuesta

ENCABEZADO DE RESPUESTA
HTTP/1.0 200 OK
Server : NCSA/1.4.2
Mime-version:1.0
Content-type:text/html
Content-length:107
[una línea en blanco conteniendo solo CRLF]
< html >
< head >
< title >
A sample HTML file
< /title >
</head >
< body >
The rest of the document goes here
</body >
< /html >

CGI (Common GateWay Interface)
CGI, se usa para generar paginas web dinámicamente, el navegador invoca un programa en el servidor que crea una nueva pagina. El uso mas común de CGI es el procesar la entrada del usuario en formas HTML

A continuación se muestra como ejemplo una forma para captar el nombre y la dirección de e-mail de una persona

```
< html >
< head >
< title > forma ejemplo </title >
< /head >
< body >
< form method=GET action = "/ cgi-bin/ register.pl" >
< PRE>
        Nombre: < input name = “username” type = ”text” size = 40 >
        e-mail: < input name = “email” type = ”text”  size = 40 >
< /PRE >
< input type = "submit" value = ”Envia” >
< /form >
< /body >
</html >
```

El navegador lee los datos que se codifican de una forma simple. Por ejemplo los datos de la figura anterior se codifica como la siguiente cadena de consulta

username=Elliote + Rusty + Harold&email=elharo%40sunsite %2eunc%2eedu

Hay dos formas de enviar una cadena de consulta al servidor: GET y POST. Si la forma usa GET el navegador agrega la cadena de consulta al url y lo envía al servidor. Si usa post envía la cadena de consulta en un flujo de salida.

Como la forma anterior usa GET el navegador enviara el siguiente comando al servidor

GET /cgi-bin/register.pl?username=Elliotte+Rusty+Harold&email=elharo%40 sunsite%2eunc%2eedu HTTP/1.0

Con POST el navegador envía los encabezados usuales y los termina con una línea en blanco, y entonces envía la cadena de consulta. La cadena de consulta se le pasa al programa CGI en la entrada estándar. Como se ve en el siguiente ejemplo.

```
POST /cgi-bin/register.pl HTTP/1.0
Content-type:application/x-www-form-urlencoded
Content-length:65

username=Elliotte+Rusty+Harold&email=elharo%40sunsite%2eunc%2eedu
```

SERVLETS

Los servlets son el equivalente de los applets, pero del lado del servidor al cuál hacen extensible.

El servlet API es fruto de la evolución del sistema de programación CGI.

Un servidor se comunica con los servlet invocando métodos en la interfaz de java.servlet.Servlet: init(), service(), y destroy().

El ciclo de vida de un servlet es el siguiente:

1.- El servidor carga los byte codes del servlet requerido.
2.- El servidor instancia el objeto servlet
3.- El servidor llama el método init() del servlet
4.- El servidor construye un objeto petición que implementa la interfaz ServletRequest a partir de los datos en la petición del cliente
5.- El servidor construye un objeto respuesta que implementa la interfaz ServletResponse.
6.- El servidor invoca el método service (request, response) del Servlet.
7.- El método service procesa la petición, llamando los métodos en el argumento response para enviar información al cliente
8.- Mientras haya más peticiones, va al paso 4
9.- Cuando el servidor no necesita mas al servlet invoca al método destroy del servlet.

```
import javax.servlet.*;
import java.io.*;

public class HolaServlet extends GenericServlet  {
        static String hola="Hola, mundo!\r\n";

        public void service (ServletRequest request,ServletResponse response)
                throws ServletException, IOException {
                response.setContentLength (hola.length());
                response.setContentType("text/plain");
                PrintStream rs=new PrintStream(response.getOutputStream());
                rs.println(hola);
        }
}
```

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class ServletSimple extends HttpServlet
{
        public void doPost(HttpServletRequest req, HttpServletResponse res) throws
            ServletException,   IOException {
                PrintWriter out = res.getWriter();
                res.setContentType("text/plain");
                String cadena = req.getParameter("TEXTO");
                out.println ("Datos capturados : "+ cadena);
        }
}
```

```
import javax.servlet.*;
import java.util.*;
import java.io.*;
public class FormTestServlet extends GenericServlet {
        public String getServletInfo ( ) {
                return "Servlet para prueba de formas ";
        }
        public void service ( ServletRequest request, ServletResponse response)
                                throws ServletException, IOException {
                response.setContentType ("text/plain");
                PrintStream out=new PrintStream(response.getOutputStream());
                out.println("Los parámetros de la solicitud son como sigue:”);
                Enumeration params=request.getParameterNames();
                if(params !=null){
                        while(params.hasMoreElements()){
                                String param=(String)params.nextElement();
                                String [ ]
                                values=request.getParameterValues(param);
                                out.print(param + "=");
                                for (int i=0; i<values.length; i++)
                                        out.print(values[i]+"  ");
```

```
                    out.println("  ");
                }//while
            }//if
        }//service
}
```