3.1. Actividades de Reflexión inicial.

Señor aprendiz, a continuación, responda las siguientes preguntas:

1. ¿Cuál es la arquitectura de una aplicación Web con Java?. Explique cada de ellas

* El navegador web o el cliente: Lado del cliente , envía peticiones al servidor solicitando una respuesta
* El servidor de aplicaciones web: recibe las peticiones, las procesa y retorna una respuesta.
* El servidor de la base de datos.: Procesa datos de la aplicación, mediante la gestión de su propio motor.

2. Explique el proceso de una petición HTTP.

Se establece una petición con el servidor.

El cliente solicita una petición mediante el protocolo http y el servidor realiza un proceso en el cual el usuario vera la solicitud en formato HTML .El cual es interpretado mediante los navegadores web.

3. ¿Qué es un Servlet?.

Los servlet hace mas fácil construir una aplicación mas fácil atraves de la tecnología java. El problema de los servlet es que tienen un ciclo de vida.

4. ¿Cuáles son las funciones de un Servlet?

Son pedazos de código que son ejecutados según las necesidades del cliente por lo cual el cliente recibirá una respuesta en formato html.

5. ¿Cuáles son los métodos HTTP y Procesamiento con los Servlets?

Metodos http :

[**GET**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET)

El método GET  solicita una representación de un recurso específico. Las peticiones que usan el método GET sólo deben recuperar datos.

[**HEAD**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/HEAD)

El método HEAD pide una respuesta idéntica a la de una petición GET, pero sin el cuerpo de la respuesta.

[**POST**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST)

El método POST se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor.

[**PUT**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PUT)

El modo PUT reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición.

[**DELETE**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/DELETE)

El método DELETE borra un recurso en específico.

[**CONNECT**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/CONNECT)

El método CONNECT establece un túnel hacia el servidor identificado por el recurso.

[**OPTIONS**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/OPTIONS)

El método OPTIONS es utilizado para describir las opciones de comunicación para el recurso de destino.

[**TRACE**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/TRACE)

El método TRACE  realiza una prueba de bucle de retorno de mensaje a lo largo de la ruta al recurso de destino.

[**PATCH**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PATCH)

El método PATCH  es utilizado para aplicar modificaciones parciales a un recurso.

Los servlets son encargados de procesar solicitudes entre clients.

6. ¿Cuál es el ciclo de vida de un Servlet?

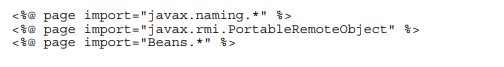
Escribir , compilar y desplegar.

7. ¿Cuál es la estructura de una página JSP?

Es una estructura básica html con código java con extensión .jsp lo cual le dice al servidor que deberá ser interpretada

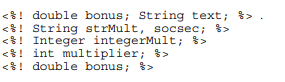
8. ¿Cuáles son las directivas JSP?

Las directivas son instrucciones dadas al motor jsp para que incluya ciertos paquetes y clases están encerradas entre <%@ %>



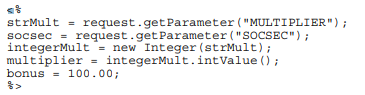
9. ¿Para qué nos sirven las declaraciones JSP?

Permite configurar variables para usarlas en el archivo jsp y se encierran <%! y %>



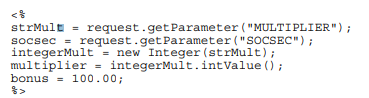
10. ¿Qué son los Scriplets?

Nos permiten embeber segmentos de código java dentro de una página JSP. El código se ejecutara cuando la página será consultada y puede usar las declaraciones de la misma van encerrados <% y %>



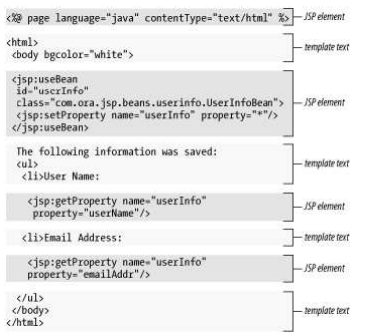
11. ¿Qué son las variables predefinidas?

Un scriptlet puede usar las siguientes variables predefinidas: session, request, response, out, e in. Este ejemplo usa la variable predefinida request, que es un objeto HttpServletRequest. De igual forma, response es un objeto HttpServletResponse, out es un objeto PrintWriter, e in es un objeto BufferedReader. Las variables predefinidas se usan en los scriptlets de la misma forma que se usan en los servelts, excepto que no las declaramos.



12. ¿Qué son las etiquetas específicas en JSP?

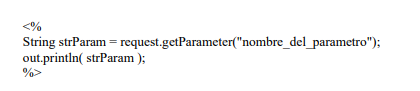
Ayuda a encontrar partes de código en un archivo jsp mediante etiquetas previamente asignadas:



13. ¿Cuáles son los objetos implícitos?. Explique cada uno de ellos.

Son objetos ya establecidos que no necesitan ser instanciados deben ser utilizados dentro del código java

Algunos : Page, config, request, response, out, session, application, pageContext, exception



14. ¿Qué son los EJBs y cuáles son sus beneficios?

Los EJBs son encargados de encapsular parte de la aplicación en el servidor, para luego ser invocados por clientes remotos permitiendo el acceso a servicios proporcionados por la aplicación.

* Los desarrolladores pueden concentrarse en solventar la lógica del negocio
* Clientes pequeños
* Desarrollo rápido

15. ¿Cuáles son los componentes de EJBs?

Session y entity

|  |  |
| --- | --- |
| EJB de Sesión | EJB de Entidad |
| Transitorio | Persistente |
| Puede ser usado por un sólo cliente. | Puede ser usado por muchos clientes. |
| No tiene identidad | Tiene una identidad (como una clave primaria) |
|  |  |
|  |  |
|  |  |
|  |  |

16. ¿Cuál es la diferencia entre EJB y los Servelts?

Los EJBs no puedan aceptar peticiones http.

Los servlets aceptan peticiones http y no pueden manejar transacciones distribuidas.

Los EJBs pueden ser llamados desde cualquier cliente basado en java

17. ¿Cómo se manejan los formularios en JSP?.

Usando Jsp , los datos se almacenan en un objeto request que es enviado desde el navegador hasta el contenedor jsp , la petición es procesada y se envía mediante un objeto response devuelta al navegador .

18. ¿Qué son los REQUEST HEADERS?

Transmiten información del estado de un solicitud web procesada , mediante métodos ,path y protocolos .

19. Enumere y explique los métodos de la clase HttpServletRequest.

* getParameter: devuelve el valor de un parámetro nombrado.
* getParameterValues: Para obtener valores multiples de un parametro
* getParameterNames: Devuelve nombres de los parametros.
* getQueryString: Devuelve un string .
* getReader:Devuelve BufferedReader de linea de datos
* getInputStream: Devuelve un ServletInputStream utilizado para leer la línea de datos.

20. Explique qué son los códigos de estado HTTP.

Muestran con un código cual fue el estado de la petición , depediendo el numero se conocera dicho estado . por ejemplo 200 es que la petición fue exitosa.

21. ¿Qué APIS Servlets, se usan con los códigos de estado?

[flushBuffer](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/ServletResponse.html#flushBuffer()), [getBufferSize](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/ServletResponse.html#getBufferSize()), [getCharacterEncoding](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/ServletResponse.html#getCharacterEncoding()), [getContentType](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/ServletResponse.html#getContentType()), [getLocale](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/ServletResponse.html#getLocale()), [getOutputStream](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/ServletResponse.html#getOutputStream()), [getWriter](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/ServletResponse.html#getWriter()), [isCommitted](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/ServletResponse.html#isCommitted()), [reset](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/ServletResponse.html#reset()), [resetBuffer](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/ServletResponse.html#resetBuffer()), [setBufferSize](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/ServletResponse.html#setBufferSize(int)), [setCharacterEncoding](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/ServletResponse.html#setCharacterEncoding(java.lang.String)), [setContentLength](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/ServletResponse.html#setContentLength(int)), [setContentType](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/ServletResponse.html#setContentType(java.lang.String)), [setLocale](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/ServletResponse.html#setLocale(java.util.Locale))

22. ¿Cuáles son los códigos de estado HTTP, más comunes?

|  |  |
| --- | --- |
| static int | [**SC\_ACCEPTED**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_ACCEPTED)            Status code (202) indicating that a request was accepted for processing, but was not completed. |
| static int | [**SC\_BAD\_GATEWAY**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_BAD_GATEWAY)            Status code (502) indicating that the HTTP server received an invalid response from a server it consulted when acting as a proxy or gateway. |
| static int | [**SC\_BAD\_REQUEST**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_BAD_REQUEST)            Status code (400) indicating the request sent by the client was syntactically incorrect. |
| static int | [**SC\_CONFLICT**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_CONFLICT)            Status code (409) indicating that the request could not be completed due to a conflict with the current state of the resource. |
| static int | [**SC\_CONTINUE**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_CONTINUE)            Status code (100) indicating the client can continue. |
| static int | [**SC\_CREATED**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_CREATED)            Status code (201) indicating the request succeeded and created a new resource on the server. |
| static int | [**SC\_EXPECTATION\_FAILED**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_EXPECTATION_FAILED)            Status code (417) indicating that the server could not meet the expectation given in the Expect request header. |
| static int | [**SC\_FORBIDDEN**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_FORBIDDEN)            Status code (403) indicating the server understood the request but refused to fulfill it. |
| static int | [**SC\_FOUND**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_FOUND)            Status code (302) indicating that the resource reside temporarily under a different URI. |
| static int | [**SC\_GATEWAY\_TIMEOUT**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_GATEWAY_TIMEOUT)            Status code (504) indicating that the server did not receive a timely response from the upstream server while acting as a gateway or proxy. |
| static int | [**SC\_GONE**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_GONE)            Status code (410) indicating that the resource is no longer available at the server and no forwarding address is known. |
| static int | [**SC\_HTTP\_VERSION\_NOT\_SUPPORTED**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_HTTP_VERSION_NOT_SUPPORTED)            Status code (505) indicating that the server does not support or refuses to support the HTTP protocol version that was used in the request message. |
| static int | [**SC\_INTERNAL\_SERVER\_ERROR**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_INTERNAL_SERVER_ERROR)            Status code (500) indicating an error inside the HTTP server which prevented it from fulfilling the request. |
| static int | [**SC\_LENGTH\_REQUIRED**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_LENGTH_REQUIRED)            Status code (411) indicating that the request cannot be handled without a defined *Content-Length*. |
| static int | [**SC\_METHOD\_NOT\_ALLOWED**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_METHOD_NOT_ALLOWED)            Status code (405) indicating that the method specified in the *Request-Line* is not allowed for the resource identified by the *Request-URI*. |
| static int | [**SC\_MOVED\_PERMANENTLY**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_MOVED_PERMANENTLY)            Status code (301) indicating that the resource has permanently moved to a new location, and that future references should use a new URI with their requests. |
| static int | [**SC\_MOVED\_TEMPORARILY**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_MOVED_TEMPORARILY)            Status code (302) indicating that the resource has temporarily moved to another location, but that future references should still use the original URI to access the resource. |
| static int | [**SC\_MULTIPLE\_CHOICES**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_MULTIPLE_CHOICES)            Status code (300) indicating that the requested resource corresponds to any one of a set of representations, each with its own specific location. |
| static int | [**SC\_NO\_CONTENT**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_NO_CONTENT)            Status code (204) indicating that the request succeeded but that there was no new information to return. |
| static int | [**SC\_NON\_AUTHORITATIVE\_INFORMATION**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_NON_AUTHORITATIVE_INFORMATION)            Status code (203) indicating that the meta information presented by the client did not originate from the server. |
| static int | [**SC\_NOT\_ACCEPTABLE**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_NOT_ACCEPTABLE)            Status code (406) indicating that the resource identified by the request is only capable of generating response entities which have content characteristics not acceptable according to the accept headers sent in the request. |
| static int | [**SC\_NOT\_FOUND**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_NOT_FOUND)            Status code (404) indicating that the requested resource is not available. |
| static int | [**SC\_NOT\_IMPLEMENTED**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_NOT_IMPLEMENTED)            Status code (501) indicating the HTTP server does not support the functionality needed to fulfill the request. |
| static int | [**SC\_NOT\_MODIFIED**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_NOT_MODIFIED)            Status code (304) indicating that a conditional GET operation found that the resource was available and not modified. |
| static int | [**SC\_OK**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_OK)            Status code (200) indicating the request succeeded normally. |
| static int | [**SC\_PARTIAL\_CONTENT**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_PARTIAL_CONTENT)            Status code (206) indicating that the server has fulfilled the partial GET request for the resource. |
| static int | [**SC\_PAYMENT\_REQUIRED**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_PAYMENT_REQUIRED)            Status code (402) reserved for future use. |
| static int | [**SC\_PRECONDITION\_FAILED**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_PRECONDITION_FAILED)            Status code (412) indicating that the precondition given in one or more of the request-header fields evaluated to false when it was tested on the server. |
| static int | [**SC\_PROXY\_AUTHENTICATION\_REQUIRED**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_PROXY_AUTHENTICATION_REQUIRED)            Status code (407) indicating that the client *MUST* first authenticate itself with the proxy. |
| static int | [**SC\_REQUEST\_ENTITY\_TOO\_LARGE**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_REQUEST_ENTITY_TOO_LARGE)            Status code (413) indicating that the server is refusing to process the request because the request entity is larger than the server is willing or able to process. |
| static int | [**SC\_REQUEST\_TIMEOUT**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_REQUEST_TIMEOUT)            Status code (408) indicating that the client did not produce a request within the time that the server was prepared to wait. |
| static int | [**SC\_REQUEST\_URI\_TOO\_LONG**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_REQUEST_URI_TOO_LONG)            Status code (414) indicating that the server is refusing to service the request because the *Request-URI* is longer than the server is willing to interpret. |
| static int | [**SC\_REQUESTED\_RANGE\_NOT\_SATISFIABLE**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_REQUESTED_RANGE_NOT_SATISFIABLE)            Status code (416) indicating that the server cannot serve the requested byte range. |
| static int | [**SC\_RESET\_CONTENT**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_RESET_CONTENT)            Status code (205) indicating that the agent *SHOULD* reset the document view which caused the request to be sent. |
| static int | [**SC\_SEE\_OTHER**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_SEE_OTHER)            Status code (303) indicating that the response to the request can be found under a different URI. |
| static int | [**SC\_SERVICE\_UNAVAILABLE**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_SERVICE_UNAVAILABLE)            Status code (503) indicating that the HTTP server is temporarily overloaded, and unable to handle the request. |
| static int | [**SC\_SWITCHING\_PROTOCOLS**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_SWITCHING_PROTOCOLS)            Status code (101) indicating the server is switching protocols according to Upgrade header. |
| static int | [**SC\_TEMPORARY\_REDIRECT**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_TEMPORARY_REDIRECT)            Status code (307) indicating that the requested resource resides temporarily under a different URI. |
| static int | [**SC\_UNAUTHORIZED**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_UNAUTHORIZED)            Status code (401) indicating that the request requires HTTP authentication. |
| static int | [**SC\_UNSUPPORTED\_MEDIA\_TYPE**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_UNSUPPORTED_MEDIA_TYPE)            Status code (415) indicating that the server is refusing to service the request because the entity of the request is in a format not supported by the requested resource for the requested method. |
| static int | [**SC\_USE\_PROXY**](https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html#SC_USE_PROXY)            Status code (305) indicating that the requested resource *MUST* be accessed through the proxy given by the *Location* field. |

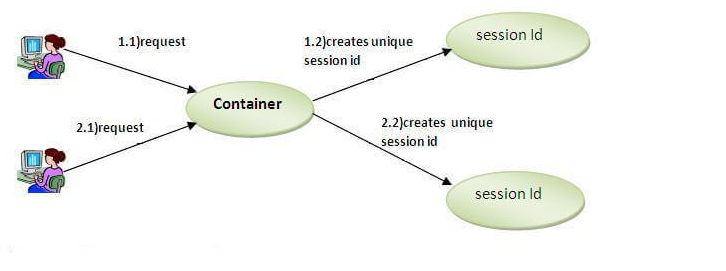
23. ¿Cuáles son los métodos más comunes para establecer cabeceros de respuesta?

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | **String encodeRedirectURL(String url)** |
|  |
| Encodes the specified URL for use in the sendRedirect method or, if encoding is not needed, returns the URL unchanged. |
| 2 | **String encodeURL(String url)** |
|  |
| Encodes the specified URL by including the session ID in it, or, if encoding is not needed, returns the URL unchanged. |
| 3 | **boolean containsHeader(String name)** |
|  |
| Returns a Boolean indicating whether the named response header has already been set. |
| 4 | **boolean isCommitted()** |
|  |
| Returns a Boolean indicating if the response has been committed. |
| 5 | **void addCookie(Cookie cookie)** |
|  |
| Adds the specified cookie to the response. |
| 6 | **void addDateHeader(String name, long date)** |
|  |
| Adds a response header with the given name and date-value. |
| 7 | **void addHeader(String name, String value)** |
|  |
| Adds a response header with the given name and value. |
| 8 | **void addIntHeader(String name, int value)** |
|  |
| Adds a response header with the given name and integer value. |
| 9 | **void flushBuffer()** |
|  |
| Forces any content in the buffer to be written to the client. |
| 10 | **void reset()** |
|  |
| Clears any data that exists in the buffer as well as the status code and headers. |
| 11 | **void resetBuffer()** |
|  |
| Clears the content of the underlying buffer in the response without clearing headers or status code. |
| 12 | **void sendError(int sc)** |
|  |
| Sends an error response to the client using the specified status code and clearing the buffer. |
| 13 | **void sendError(int sc, String msg)** |
|  |
| Sends an error response to the client using the specified status. |
| 14 | **void sendRedirect(String location)** |
|  |
| Sends a temporary redirect response to the client using the specified redirect location URL. |
| 15 | **void setBufferSize(int size)** |
|  |
| Sets the preferred buffer size for the body of the response. |
| 16 | **void setCharacterEncoding(String charset)** |
|  |
| Sets the character encoding (MIME charset) of the response being sent to the client, for example, to UTF-8. |
| 17 | **void setContentLength(int len)** |
|  |
| Sets the length of the content body in the response In HTTP servlets, this method sets the HTTP Content-Length header. |
| 18 | **void setContentType(String type)** |
|  |
| Sets the content type of the response being sent to the client, if the response has not been committed yet. |
| 19 | **void setDateHeader(String name, long date)** |
|  |
| Sets a response header with the given name and date-value. |
| 20 | **void setHeader(String name, String value)** |
|  |
| Sets a response header with the given name and value. |
| 21 | **void setIntHeader(String name, int value)** |
|  |
| Sets a response header with the given name and integer value |
| 22 | **void setLocale(Locale loc)** |
|  |
| Sets the locale of the response, if the response has not been committed yet. |
| 23 | **void setStatus(int sc)** |
|  |
| Sets the status code for this response |

24. ¿Cómo es el manejo de sesiones con Servlets.

Se contenedor el cual creado un id para identificar cada usuario , un objeto od http sesión puede ser usado para :

1. unir objetos
2. ver y manipular información acerca de la sesión tales como identificador de sesión , tiempo de creación y ultimo acceso tiempo de acceso



25. ¿Qué son los Cookies?

Informacion almacenada en el navegador en forma dinamica .

26. Explique como se maneja las sesiones con los Servlets

Se crea un objeto que contendrá la información de los usuarios conectados , nosotros mediante sus atributos podremos conocer la información del usuario , tiempo de conexión.