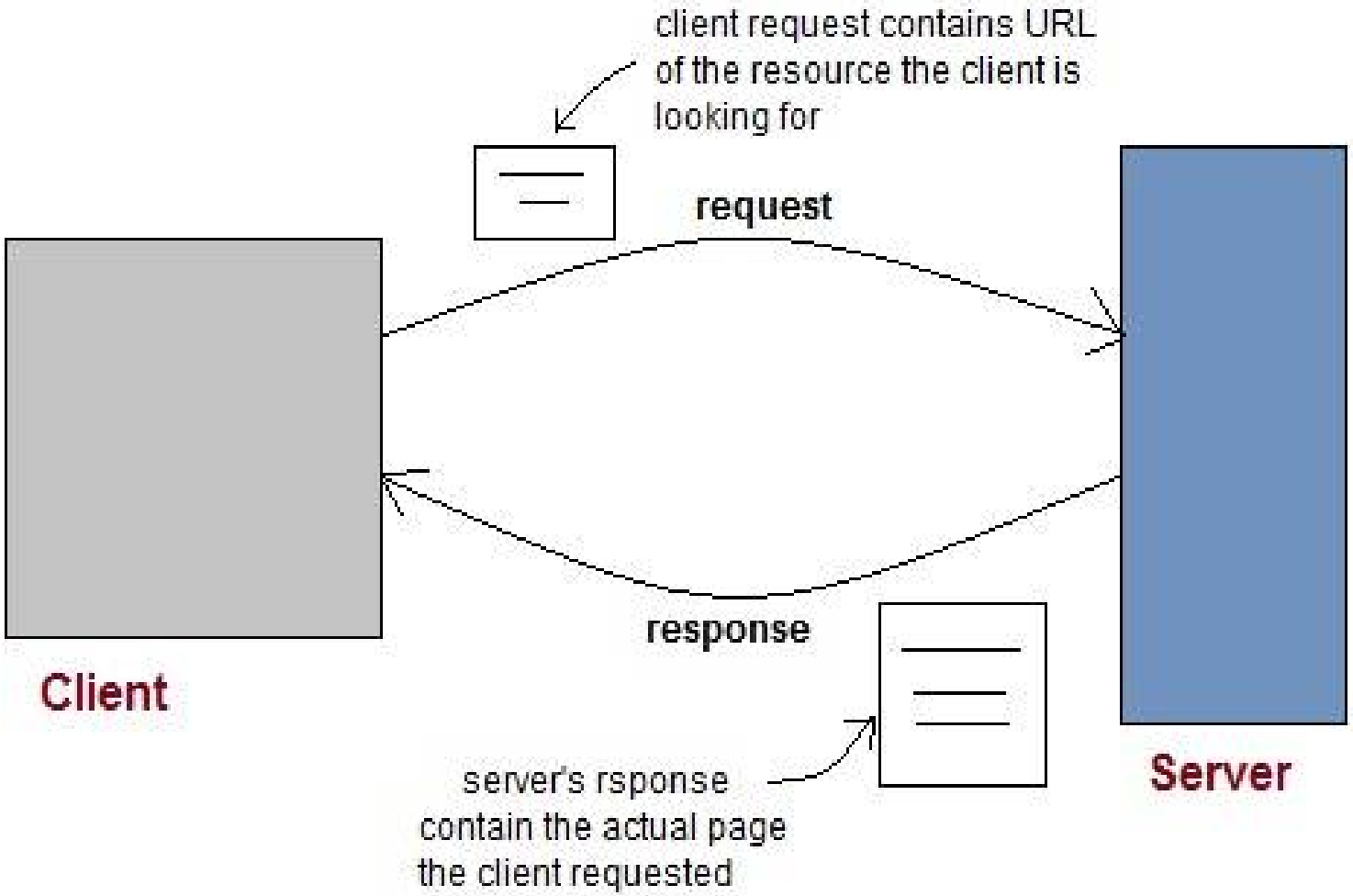
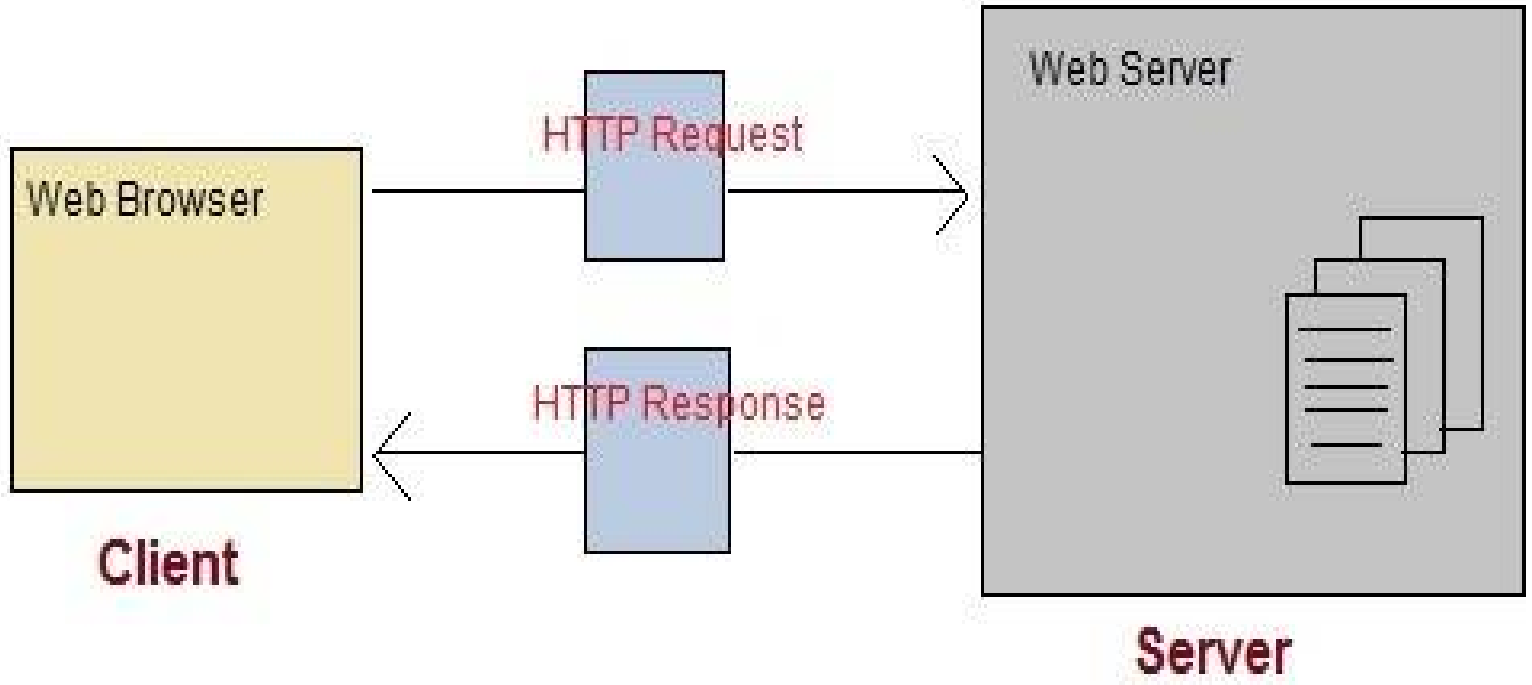
Servlets

PMCA502L: Thilagavathi M, AP(Sr.), SCORE

Introduction to Web



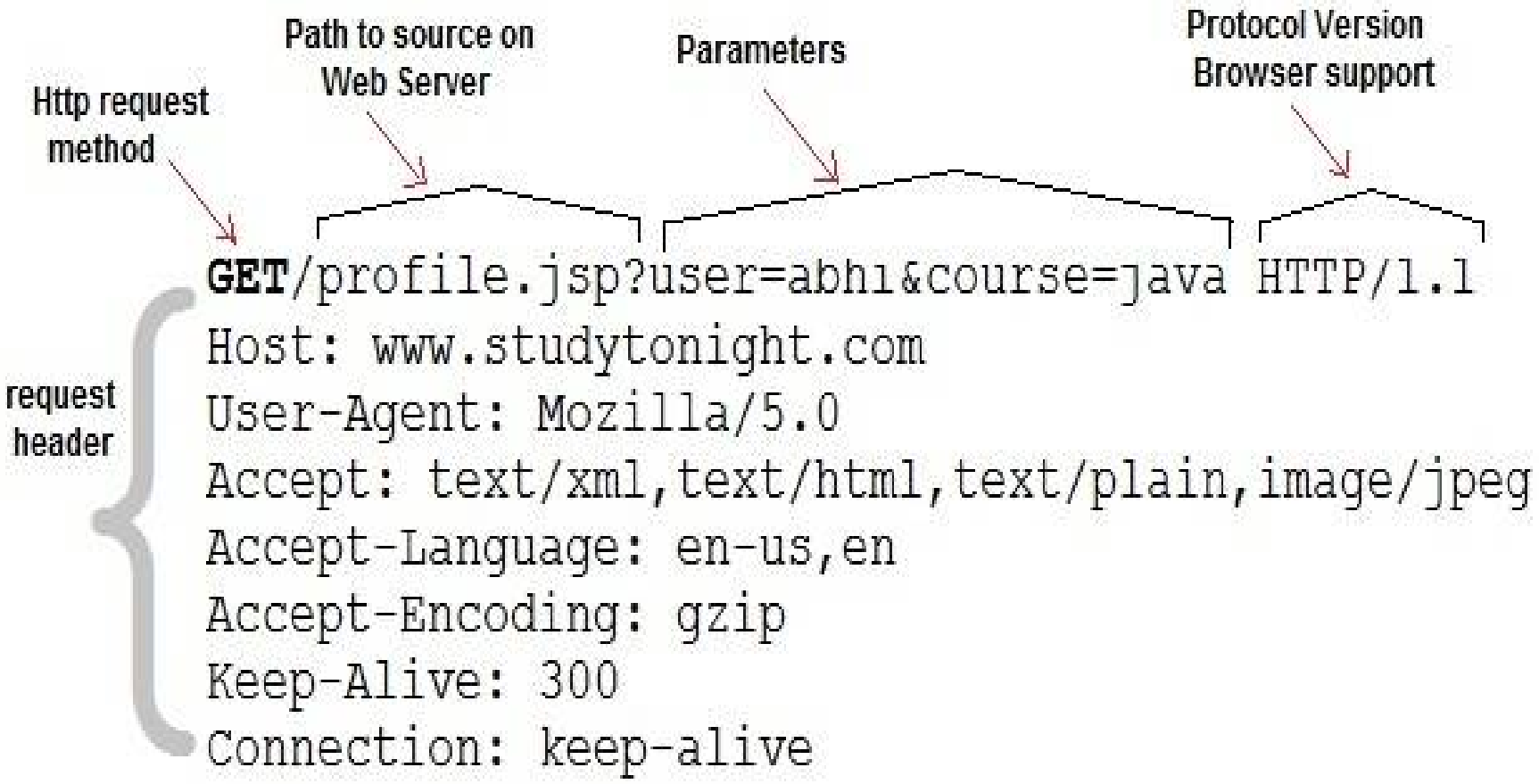
HTTP


Web Server
HTTP Request
Web Browser
HTTP Response
Client
Server

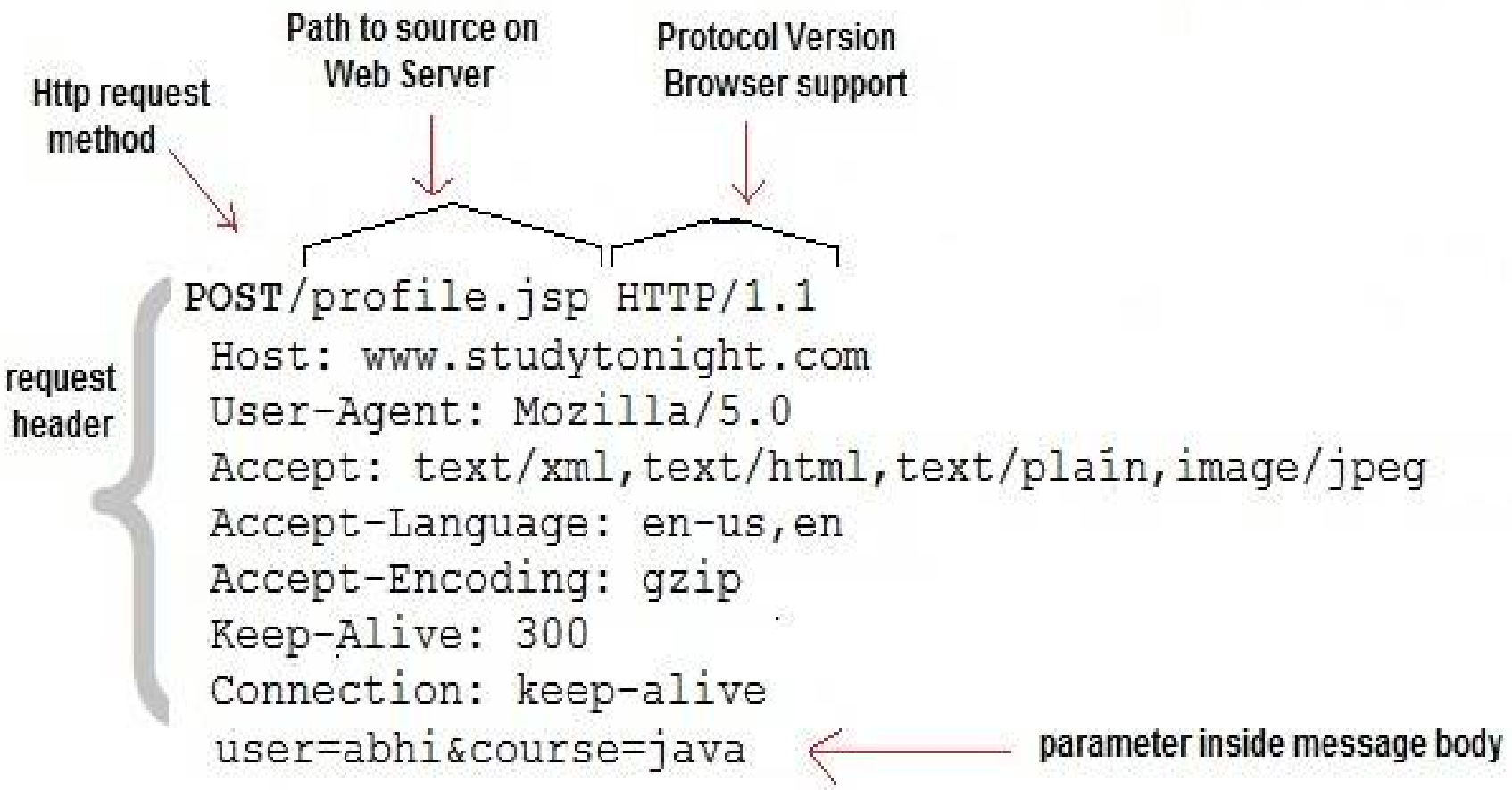
HTTP Important Methods

- GET
- POST
- OPTIONS
- DELETE
- HEAD

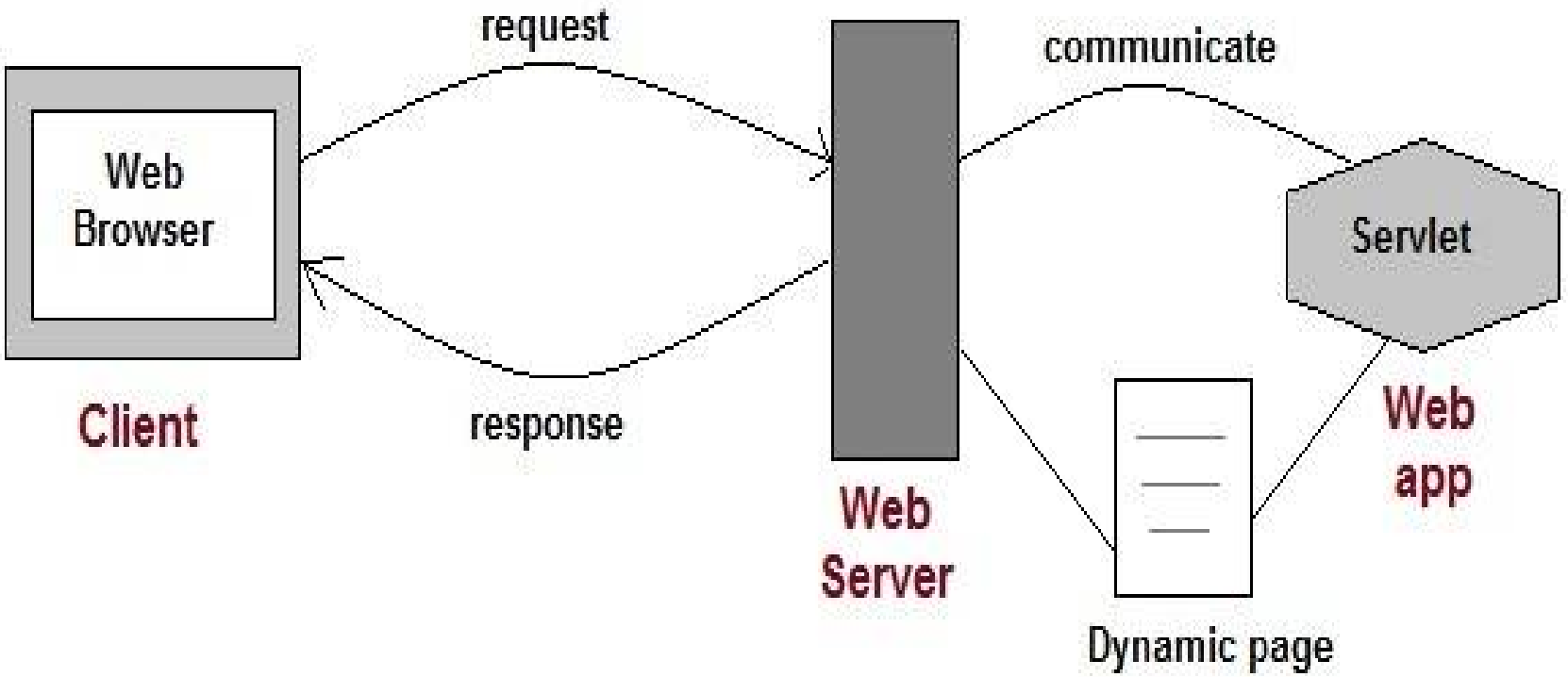
Anatomy of an HTTP GET request



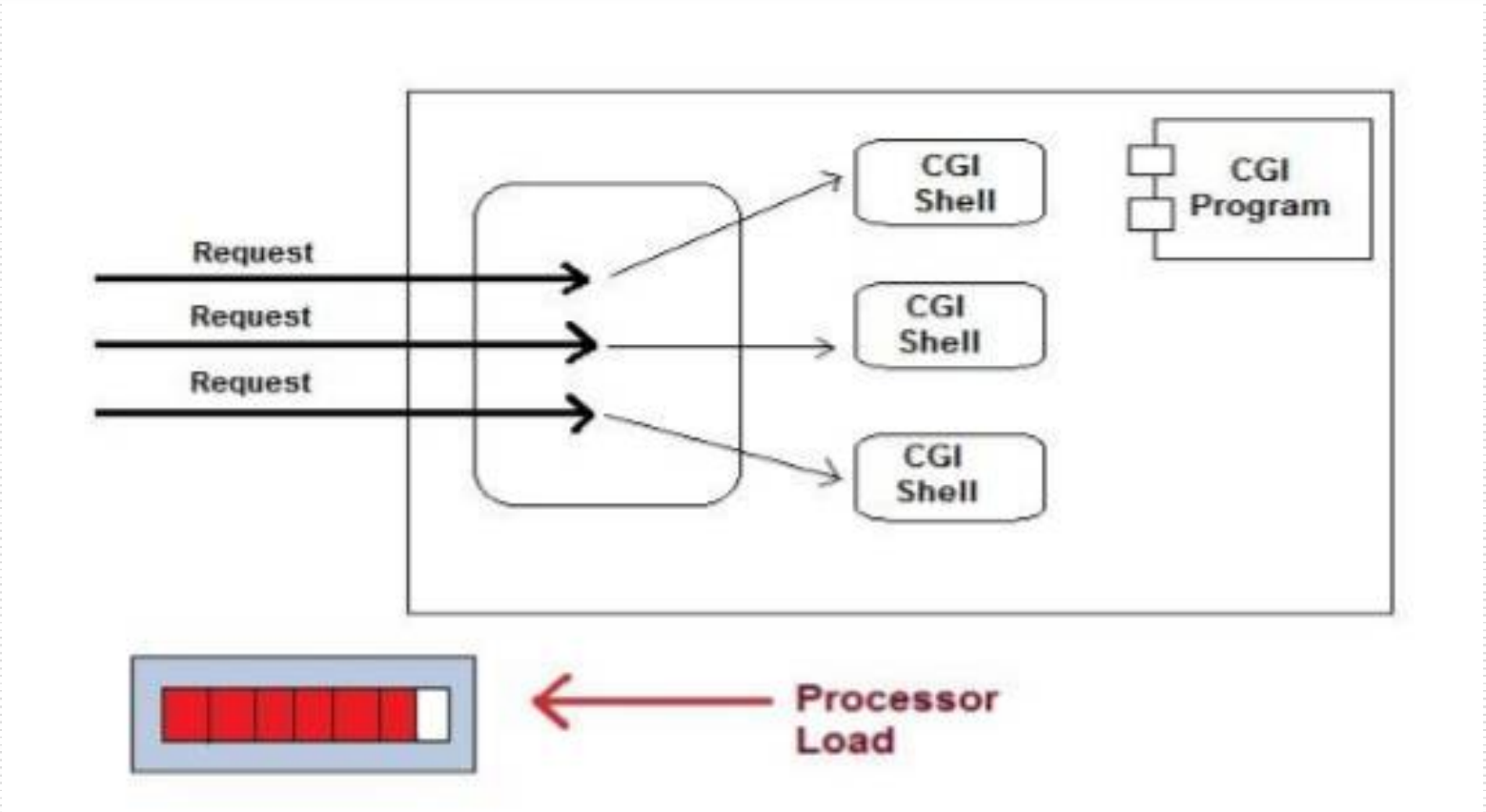
Anatomy of an HTTP POST request



Servlet

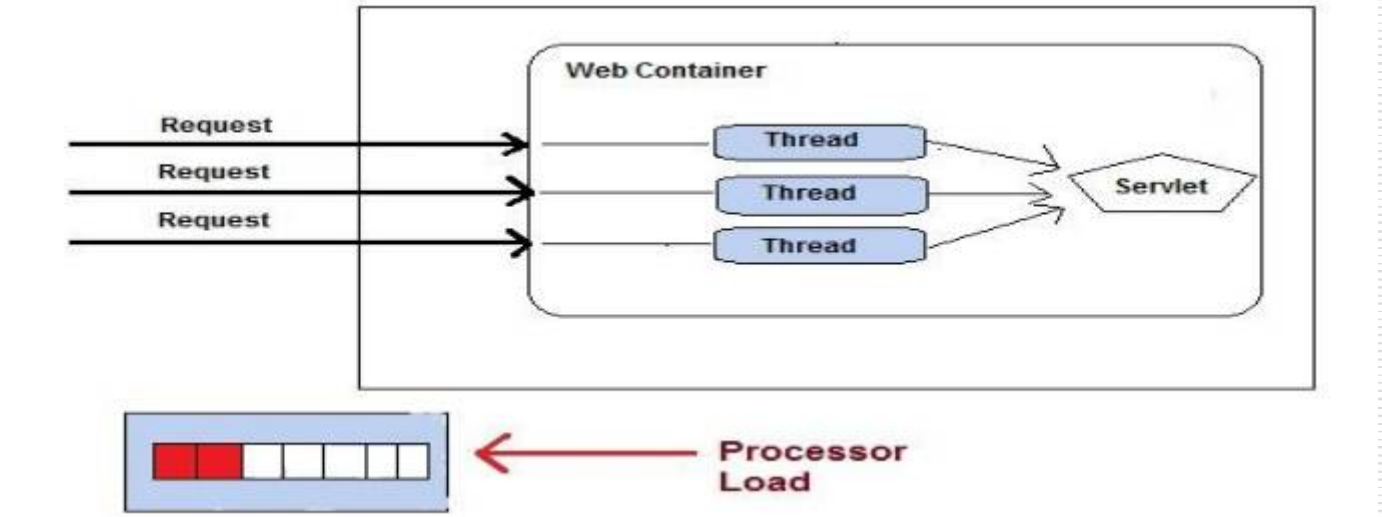

request
communicate
Web
Browser
Servlet
Client
response
Web
app
Web
Server
Dynamic page

CGI


Request
Request
Request
CGI Shell
CGI Shell
CGI Shell
CGI Program
Processor Load

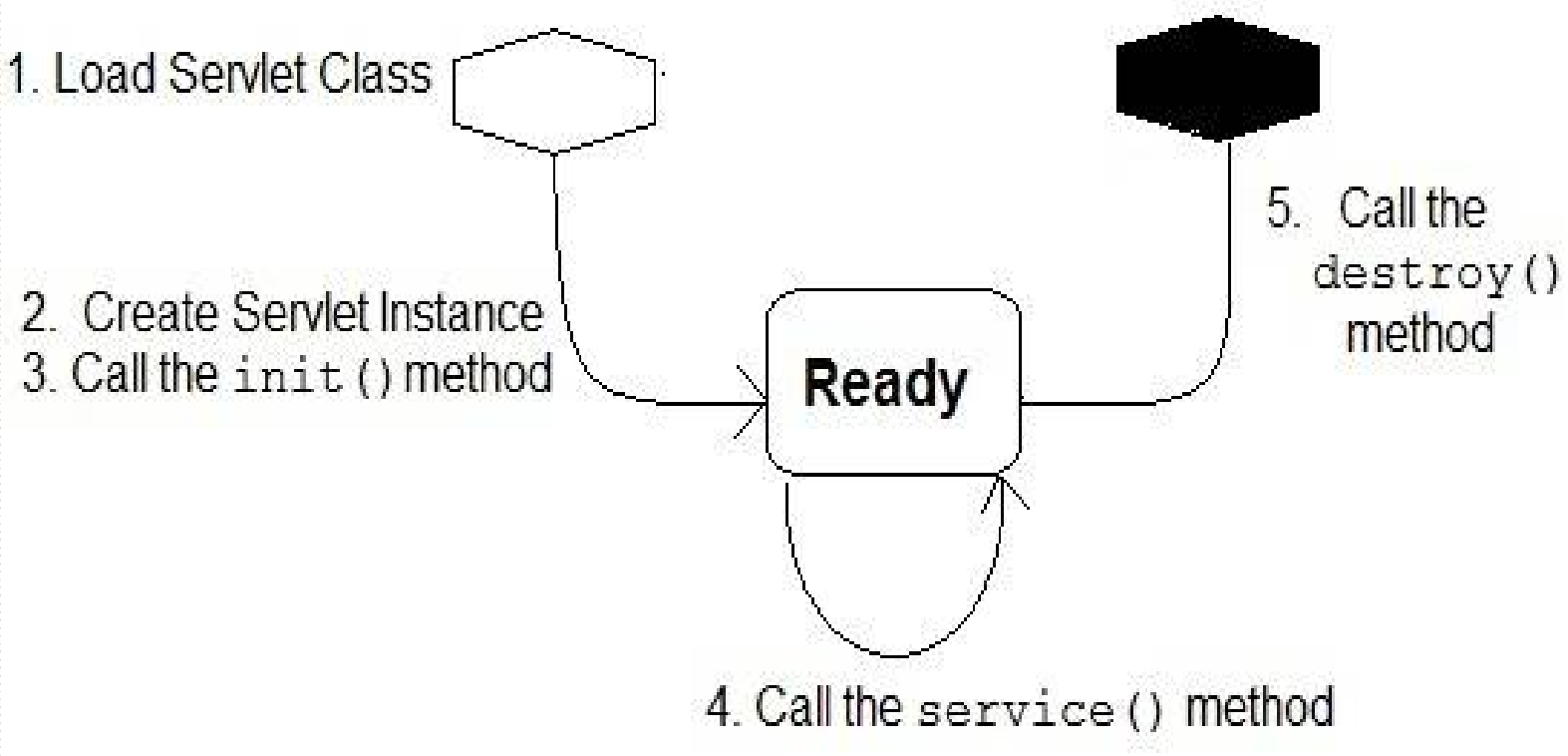
Advantages of Servlets over Traditional CGI

- Less response time because each request run in a separate thread.



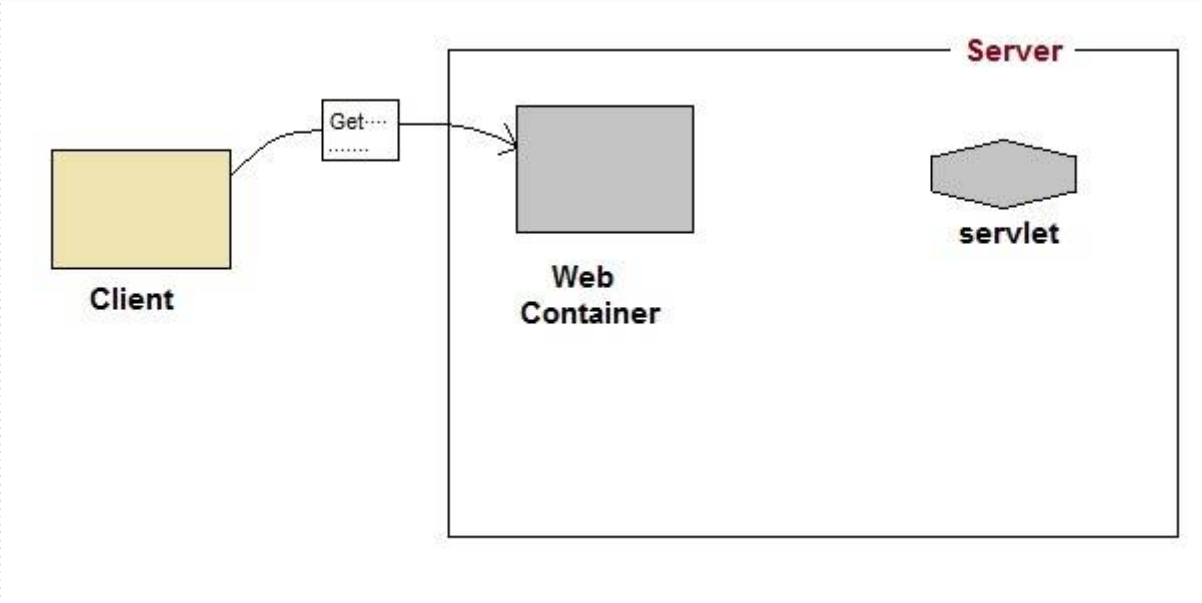
- Servlet are scalable.
- Servlet are robust and object oriented.
- Servlet are platform independent.

Servlet Life Cylce



How does a Servlet Application Work?

1) User sends request for a servlet by clicking a link that has URL to a servlet.

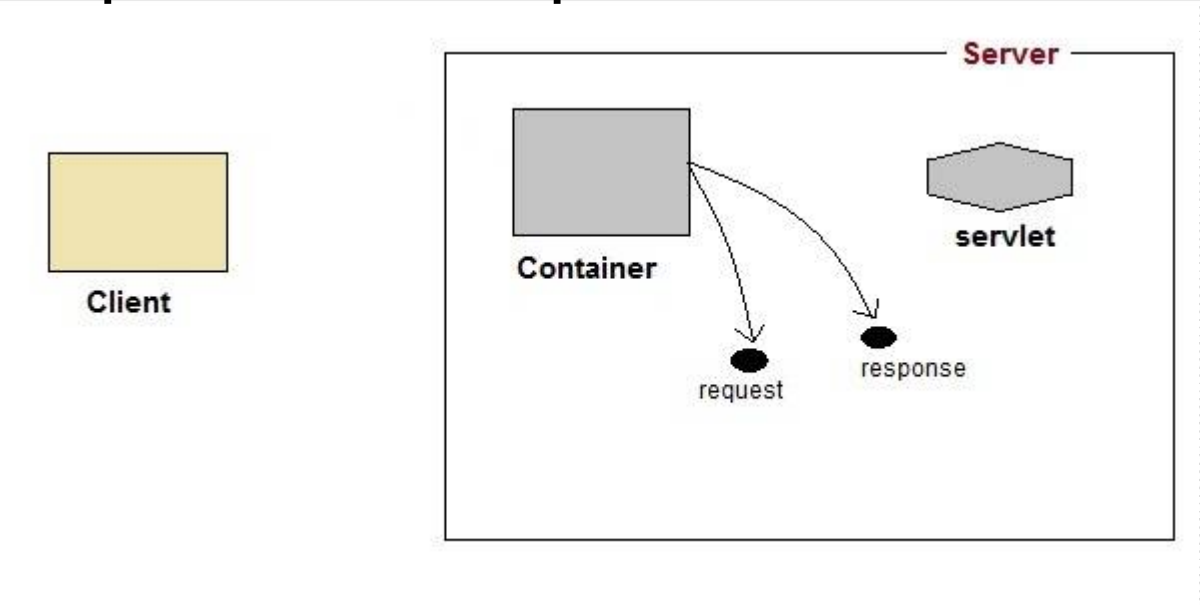


How does a Servlet Application Work?

2) The container finds the servlet using deployment descriptor and creates two objects

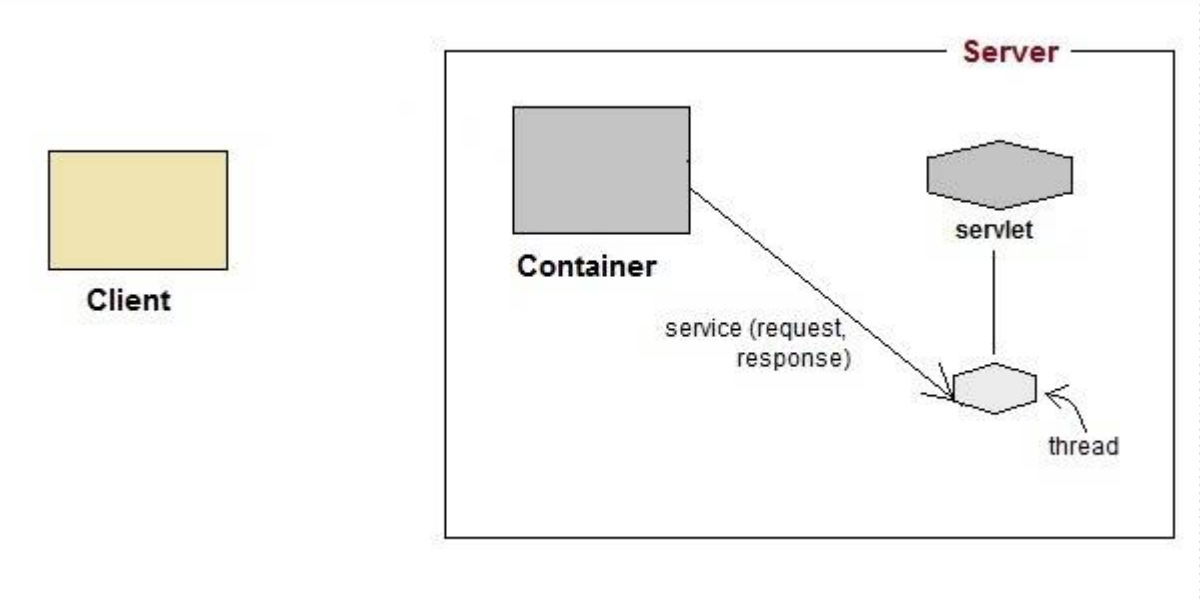
HttpServletRequest

HttpServletResponse



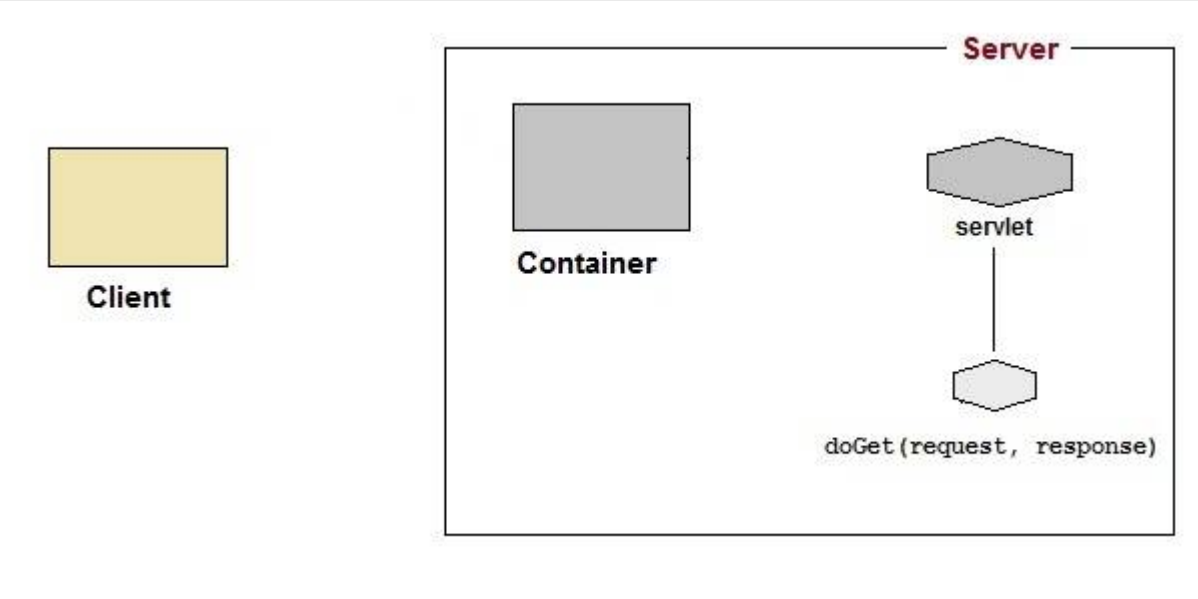
How does a Servlet Application Work?

3) The container creates or allocates a thread for that request and calls the servlet's service() method and passes the request, response objects as argument



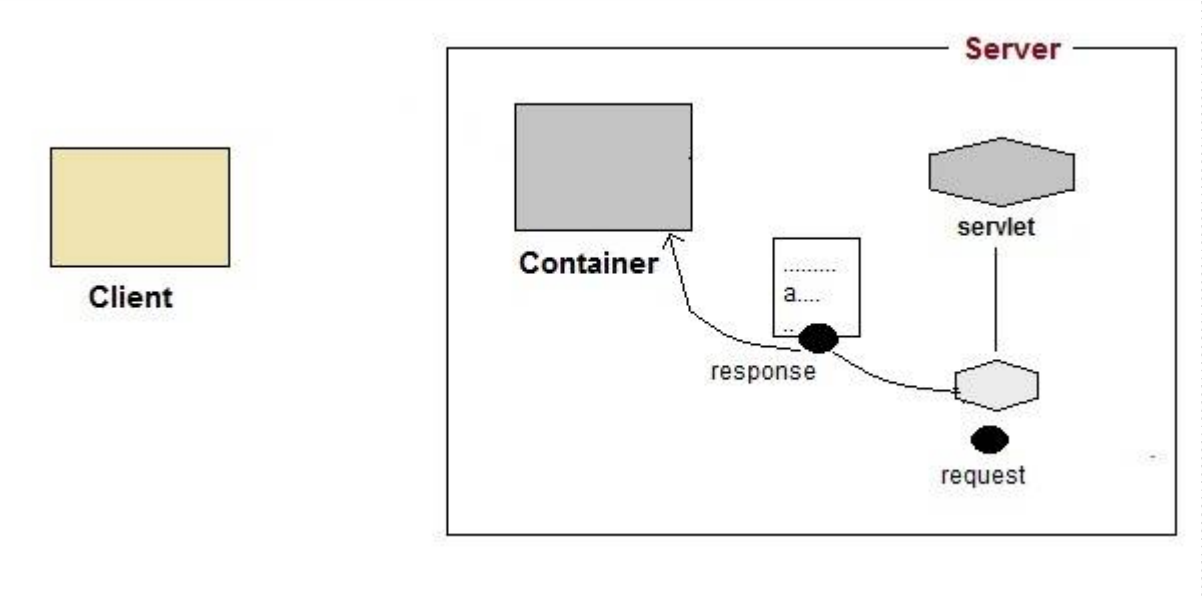
How does a Servlet Application Work?

4) The service method decide which servlet method doGet() or doPost() to call based on HTTP Request Method(Get,Post) sent by the client.



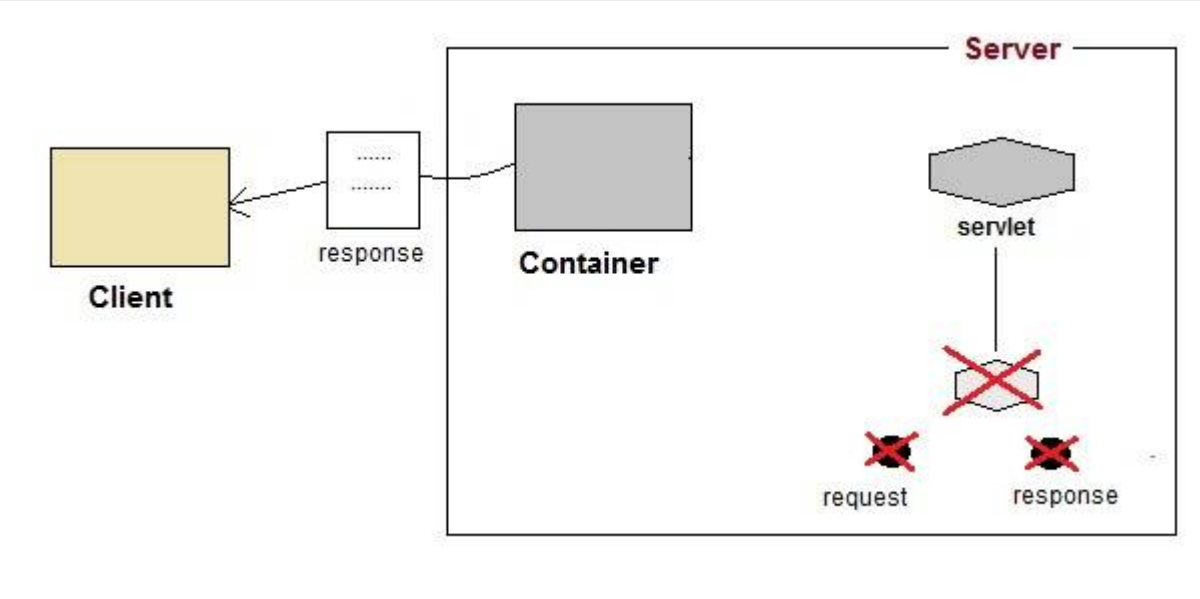
How does a Servlet Application Work?

5) The servlet uses response object to write the response back to the client



How does a Servlet Application Work?

6) After the service() method is completed the thread dies. The request and response object are ready for garbage collection



Servlet Program Essentials

Servlet API

- Servlet API consists of two important packages
  - javax.servlet
  - javax.servlet.http

javax.servlet

- Some Important Classes and Interface of javax.servlet

Interface	Description
Servlet	Declares life cycle methods for a servlet.
ServletConfig	Allows servlets to get initialization parameters.
ServletContext	Enables servlets to log events and access information about their environment.
ServletRequest	Used to read data from a client request.
ServletResponse	Used to write data to a client response.
SingleThreadModel	Indicates that the servlet is thread safe.

javax.servlet

Class	Description
GenericServlet	Implements the Servlet and ServletConfig interfaces.
ServletInputStream	Provides an input stream for reading requests from a client.
ServletOutputStream	Provides an output stream for writing responses to a client.
ServletException	Indicates a servlet error occurred.
UnavailableException	Indicates a servlet is unavailable.

Servlet Interface

- Methods

void init(ServetConfig config) throws ServletException

void service(ServletRequest req, ServletResponse resp)
throws ServletException, IOException

void destroy()

String getServletInfo()

Servlet Interface

Method	Description
void destroy( )	Called when the servlet is unloaded.
ServletConfig getServletConfig( )	Returns a **ServletConfig** object that contains any initialization parameters.
String getServletInfo( )	Returns a string describing the servlet.
void init(ServletConfig *sc*) throws ServletException	Called when the servlet is initialized. Initialization parameters for the servlet can be obtained from *sc*. An **UnavailableException** should be thrown if the servlet cannot be initialized.
void service(ServletRequest *req*, ServletResponse *res*) throws ServletException, IOException	Called to process a request from a client. The request from the client can be read from *req*. The response to the client can be written to *res*. An exception is generated if a servlet or IO problem occurs.

ServletConfig Interface

Method	Description
ServletContext getServletContext( )	Returns the context for this servlet.
String getInitParameter(String *param*)	Returns the value of the initialization parameter named *param*.
Enumeration getInitParameterNames( )	Returns an enumeration of all initialization parameter names.
String getServletName( )	Returns the name of the invoking servlet.

ServletContext Interface

Method	Description
String getServerInfo( )	Returns information about the server.
void log(String *s*)	Writes *s* to the servlet log.
void log(String *s*, Throwable *e*)	Write *s* and the stack trace for *e* to the servlet log.

ServletRequest Interface

- int getContentLength();
- String getContentType();
- ServletInputStream getInputStream();
- BufferedReader getReader();
- String getParameter(String pname);
- String[] getParameterValues(String pname);
- Enumeration getParameterNames();

ServletRequest Interface

int getContentLength( )	Returns the size of the request. The value –1 is returned if the size is unavailable.
String getContentType( )	Returns the type of the request. A **null** value is returned if the type cannot be determined.
ServletInputStream getInputStream( ) throws IOException	Returns a **ServletInputStream** that can be used to read binary data from the request. An **IllegalStateException** is thrown if **getReader( )** has already been invoked for this request.

ServletRequest Interface

String getParameter(String *pname*)	Returns the value of the parameter named *pname*.
Enumeration getParameterNames( )	Returns an enumeration of the parameter names for this request.
String[ ] getParameterValues(String *name*)	Returns an array containing values associated with the parameter specified by *name*.
String getProtocol( )	Returns a description of the protocol.
BufferedReader getReader( ) throws IOException	Returns a buffered reader that can be used to read text from the request. An **IllegalStateException** is thrown if **getInputStream( )** has already been invoked for this request.

ServletRequest Interface

String getRemoteAddr( )	Returns the string equivalent of the client IP address.
String getRemoteHost( )	Returns the string equivalent of the client host name.
String getScheme( )	Returns the transmission scheme of the URL used for the request (for example, "http", "ftp").
String getServerName( )	Returns the name of the server.
int getServerPort( )	Returns the port number.

ServletResponse Interface

Method	Description
ServletOutputStream getOutputStream( ) throws IOException	Returns a **ServletOutputStream** that can be used to write binary data to the response. An **IllegalStateException** is thrown if **getWriter( )** has already been invoked for this request.
PrintWriter getWriter( ) throws IOException	Returns a **PrintWriter** that can be used to write character data to the response. An **IllegalStateException** is thrown if **getOutputStream( )** has already been invoked for this request.
void setContentLength(int *size*)	Sets the content length for the response to *size*.
void setContentType(String *type*)	Sets the content type for the response to *type*.

GenericServlet

- Provides implementations of the basic life cycle methods for a servlet.
- Implements Servlet and ServletConfig interfaces.
- Additional methods

  void log(String s)

  void log(String s, Throwable e)

javax.servlet.http

Class	Description
HttpServlet	Provides methods to handle HTTP requests and responses.

Interface	Description
HttpServletRequest	Enables servlets to read data from an HTTP request.
HttpServletResponse	Enables servlets to write data to an HTTP response.
HttpSession	Allows session data to be read and written.

HTTPServlet

- The HttpServlet class is an abstract class that extends the GenericServlet class and implements a Serializable interface.

HttpServlet Class

Method	Description
void doDelete(HttpServletRequest *req*, HttpServletResponse *res*) throws IOException, ServletException	Performs an HTTP DELETE.
void doGet(HttpServletRequest *req*, HttpServletResponse *res*) throws IOException, ServletException	Performs an HTTP GET.
void doHead(HttpServletRequest *req*, HttpServletResponse *res*) throws IOException, ServletException	Performs an HTTP HEAD.
void doOptions(HttpServletRequest *req*, HttpServletResponse *res*) throws IOException, ServletException	Performs an HTTP OPTIONS.

HttpServlet Class

void doPost(HttpServletRequest *req*, HttpServletResponse *res*) throws IOException, ServletException	Performs an HTTP POST.
long getLastModified(HttpServletRequest *req*)	Returns the time (in milliseconds since midnight, January 1, 1970, GMT) when the requested resource was last modified.
void service(HttpServletRequest *req*, HttpServletResponse *res*) throws IOException, ServletException	Called by the server when an HTTP request arrives for this servlet. The arguments provide access to the HTTP request and response, respectively.

HttpServletRequest Interface

Method	Description
String getHeader(String *field*)	Returns the value of the header field named *field*.
Enumeration getHeaderNames( )	Returns an enumeration of the header names.
int getIntHeader(String *field*)	Returns the **int** equivalent of the header field named *field*.
Cookie[ ] getCookies( )	Returns an array of the cookies in this request.
String getMethod( )	Returns the HTTP method for this request.

HttpServletRequest Interface

String getQueryString( )	Returns any query string in the URL.
String getRemoteUser( )	Returns the name of the user who issued this request.
String getRequestedSessionId( )	Returns the ID of the session.
StringBuffer getRequestURL( )	Returns the URL.
String getServletPath( )	Returns that part of the URL that identifies the servlet.
HttpSession getSession( )	Returns the session for this request. If a session does not exist, one is created and then returned.
HttpSession getSession(boolean *new*)	If *new* is **true** and no session exists, creates and returns a session for this request. Otherwise, returns the existing session for this request.
boolean isRequestedSessionIdFromCookie( )	Returns **true** if a cookie contains the session ID. Otherwise, returns **false**.
boolean isRequestedSessionIdFromURL( )	Returns **true** if the URL contains the session ID. Otherwise, returns **false**.
boolean isRequestedSessionIdValid( )	Returns **true** if the requested session ID is valid in the current session context.

HttpServletResponse Interface

Method	Description
void addCookie(Cookie *cookie*)	Adds *cookie* to the HTTP response.
boolean containsHeader(String *field*)	Returns true if the HTTP response header contains a field named *field*.
void sendError(int *c*) throws IOException	Sends the error code *c* to the client.
void sendError(int *c*, String *s*) throws IOException	Sends the error code *c* and message *s* to the client.
void sendRedirect(String *url*) throws IOException	Redirects the client to *url*.

HttpServletResponse Interface

Method	Description
void setHeader(String *field*, String *value*)	Adds *field* to the header with value equal to *value*.
void setIntHeader(String *field*, int *value*)	Adds *field* to the header with value equal to *value*.
void setStatus(int *code*)	Sets the status code for this response to *code*.

Example 1

```
<form name="Form1" method="post" action="PostParametersServlet">
        <table>
                <tr>
                <td><B>Employee</td>
                <td><input type=textbox name="e" size="25" value=""></td>
                </tr>
                <tr>
                <td><B>Phone</td>
                <td><input type=textbox name="p" size="25" value=""></td>
                </tr>
        </table>
<input type=submit value="Submit">
</body>
</html>
```

```
import java.io.*;
importjava.util.*;
importjavax.servlet.*;
public class PostParametersServlet extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
                          response) throws ServletException, IOException
  {
      PrintWriter pw = response.getWriter();
      Enumeration e = request.getParameterNames();
      while(e.hasMoreElements( ))    {
          String pname = (String)e.nextElement();
          pw.print(pname + " = ");
          String pvalue = request.getParameter(pname);
          pw.println(pvalue);
      }
      pw.close();
   }
}
```

Example

```
html>
    <head>
    </head>
    <body>
        <form name="Form1" action="ColorGetServlet">
                <B>Color:</B>
                <select name="color" size="1">
                    <option value="Red">Red</option>
                    <option value="Green">Green</option>
                    <option value="Blue">Blue</option>
                                    </select>
                <br><br>
                <input type=submit value="Submit">
          </form>
    </body>
</html>
```

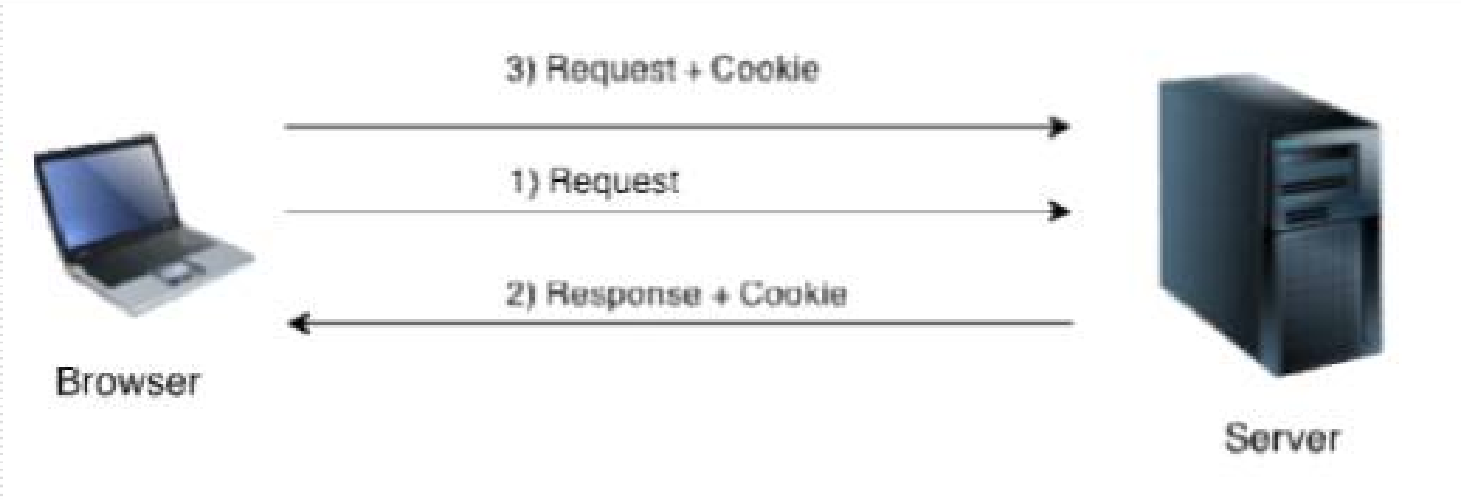
```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class ColorGetServlet extends HttpServlet {
        public void doGet(HttpServletRequest request,
                HttpServletResponse response)
                                throws ServletException, IOException
        {
                String color = request.getParameter("color");
                response.setContentType("text/html");
                PrintWriter pw = response.getWriter();
                pw.println("<B>The selected color is: ");
                pw.println(color);
                pw.close();
        }
}
```

Cookies

Cookies

- Cookies are text files stored on the client computer and they are kept for various information tracking purpose.
- A cookie is a small piece of information that persists between the multiple client requests.
- A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

How Cookie Works



Types of Cookie

- Non-persistent
- Persistent

Methods

- HttpServletRequest Interface
  - Cookies[] getCookies() method is provided to get the array of Cookies from request, since there is no point of adding Cookie to request, there are no methods to set or add cookie to request.
- HttpServletResponse Interface
  - addCookie(Cookie c) method is provided to attach cookie in response header, there are no getter methods for cookie.
- Cookie class has a single constructor that takes name and value because they are mandatory parameters for a cookie, all other parameters are optional.

Methods of Cookie Class

- getComment() setComment()
- getDomain() setDomain()
- getMaxAge() setMaxAge()
- getName()
- getPath() setPath()
- boolean getSecure(); setSecure()
- getValue() setValue()
- getVersion()
- isHttpOnly()

Sessions

- HTTP is a "stateless" protocol which means each time a client retrieves a Web page, the client opens a separate connection to the Web server and the server automatically does not keep any record of previous client request.
- Still there are three ways to maintain session between web client and web server:
  - Cookies
  - Hidden Form Fields
    - Example

    <input type="hidden" name="sessionid" value="12345">
  - URL Rewriting
    - Example

    http://www.vit.ac.in/admissions.htm;sessionid=12345

Sessions

- The HttpSession Object:
  - Apart from the above mentioned three ways, servlet provides HttpSession Interface which provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.
  - The servlet container uses this interface to create a session between an HTTP client and an HTTP server. The session persists for a specified time period, across more than one connection or page request from the user.
  - You would get HttpSession object by calling the public method getSession() of HttpServletRequest, as below:

    HttpSession session = request.getSession();

    HttpSession session = request.getSession(boolean);

HttpSession Interface

Method	Description
long getCreationTime( )	Returns the time (in milliseconds since midnight, January 1, 1970, GMT) when this session was created.
String getId( )	Returns the session ID.
long getLastAccessedTime( )	Returns the time (in milliseconds since midnight, January 1, 1970, GMT) when the client last made a request for this session.
void invalidate( )	Invalidates this session and removes it from the context.
boolean isNew( )	Returns true if the server created the session and it has not yet been accessed by the client.

Servlet with DB

```
<html>
    <head>
        <title>Servlet DB</title>
    </head>
    <body>
        <Form name="form1" method="post" action="NewServlet_DB">
            Enter ID: <Input Type="text" name="id"><BR>
            Enter Name: <Input Type="text" name="name"><BR>
            Enter Total: <Input Type="text" name="total"><BR>
            Enter CGPA: <Input Type="text" name="cgpa"><BR>
            <input type="submit" value="Submit">
        </Form>
    </body>
</html>
```

Servlets with DB

```
public class NewServlet_DB extends HttpServlet {
        Connection c=null;   Statement s=null;     ResultSet rs=null;
        protected void processRequest(HttpServletRequest request, HttpServletResponse
                response) throws ServletException, IOException {
            try{
                Class.forName("org.apache.derby.jdbc.ClientDriver");
                c=DriverManager.getConnection("jdbc:derby://localhost:1527/Login;
                                                create=true;user=test;password=test");
                System.out.println("Connection established");
                s=c.createStatement();
                String id=request.getParameter("id");
                String name=request.getParameter("name");
                int total = Integer.parseInt(request.getParameter("total"));
                double cgpa=Double.parseDouble(request.getParameter("cgpa"));
                int n=s.executeUpdate("INSERT INTO student
                            VALUES('"+id+"','"+name+"',"+total+","+cgpa+")");
                response.setContentType("text/html;charset=UTF-8");
                PrintWriter out = response.getWriter();
                if(n==1)
                    out.println("<h2>Inserted<h2>");
                rs.close(); s.close();   c.close();
            }
            catch(SQLException | ClassNotFoundException e){
              System.out.println(e);
            }
        }
```

```
@Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
            response) throws ServletException, IOException {
        processRequest(request, response);
    }

@Override
    protected void doPost(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOException
   {
        processRequest(request, response);
    }
}
```