SERVLETS I

Java Server-side technology

Objectives

- What is CGI?
- What is a servlet?
- What is a servlet container?
- FirstServlet
- The relationship between a servlet container and the Servlet API
- Understanding the HTTP Basics

The Web Tier

- In the Java world, servlets are the cornerstone of web component technology.
- Java servlets support the creation of a wide range of dynamic web content. From online catalogs, to chat applications, to stock trading sites, to complex portals, Java servlets can do almost anything that you can do with HTTP and a markup language.
- understanding servlet is the key to understanding all the web components in the Java EE platform.

What is servlet?

- A servlet is a Java class used to extend the capabilities of servers that host applications accessed by means of a request-response programming model.
- Servlets are the central processing unit of a Java web application.
- Servlet is the controller(MVC) responsible for most of the processing required by a java web application.
- Servlets are Java programs that run on a Web server. Java servlets can be used to process client requests or produce dynamic Web pages.
- The web application and its constituent components are managed and executed inside the web container
- In java web application this container is called a servlet container. This container provides additional features to the web application.(ex: security, low level details)
- Servlet technology is primarily designed for use with the HTTP protocol of the Web.
- A solid understanding of the Java servlet model and Java servlet API in Java EE is fundamental to understanding all the other web components in the platform.

PROTOCOL

- Client and server: The client is the machine requesting some information, and the server is the machine that provides that information.
- PROTOCOL:

The information data that flows from the information provider (the server) to the information requester (the client) is bound by a definite rule that governs the marshaling (package) of the information to be transmitted by the server and the unmarshaling (unpackaging) of the information to be translated or read by the client. This rule is called the protocol.

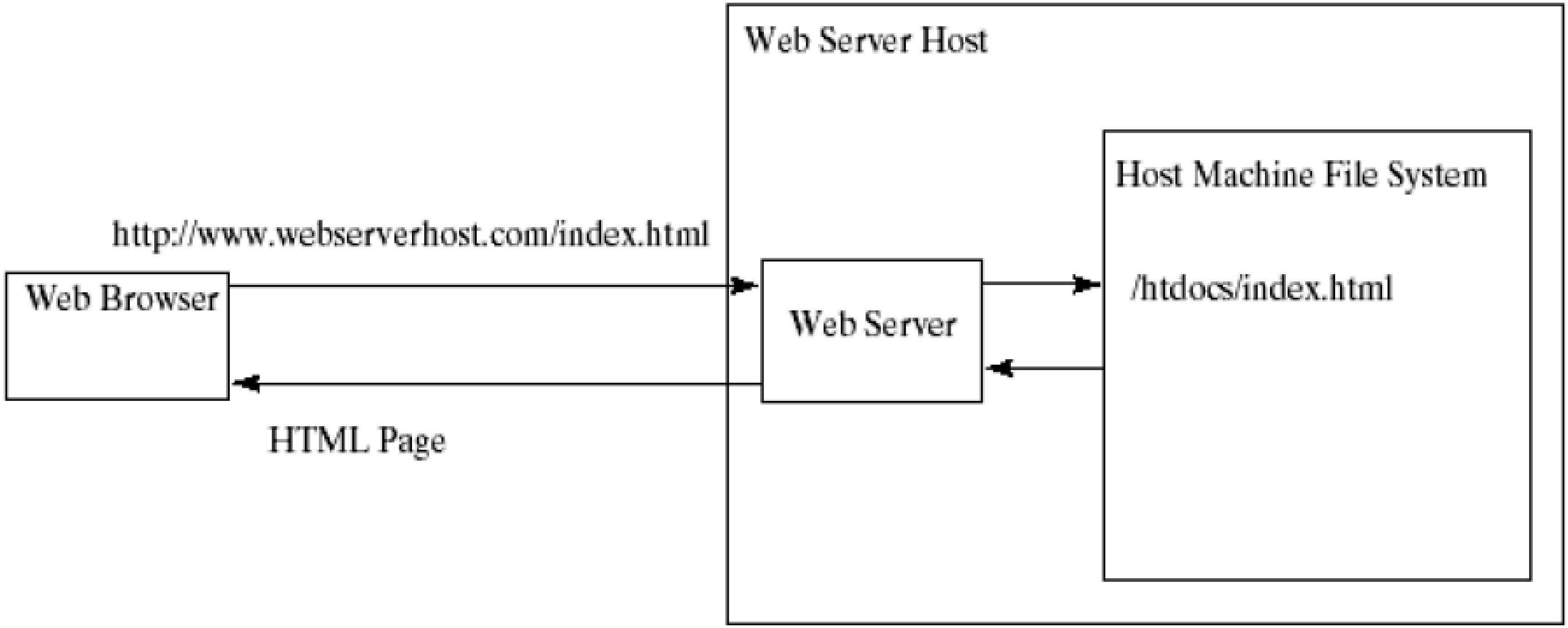
OR

A **protocol** is a standard used to define a method of exchanging data over a computer network. Each protocol has its own method of how data is formatted when sent and what to do with it once received.

HTTP Request And Response

- The web browser (the client), the web server (the server), and the web application all converse with each other through the Hypertext Transfer Protocol (HTTP).
- The clients send HTTP requests to the web servers, and the web servers return the requested data in the form of HTTP responses.
- The HTTP clients and the HTTP servers are the building blocks that lay the foundation of the World Wide Web.
- HTTP is the lingua franca(common language) of the Web*.*
- HTTP is a request-response stateless protocol, the corollary of which is that, from the web server's view, any request is the first request from the web browser.

HTTP and static HTML



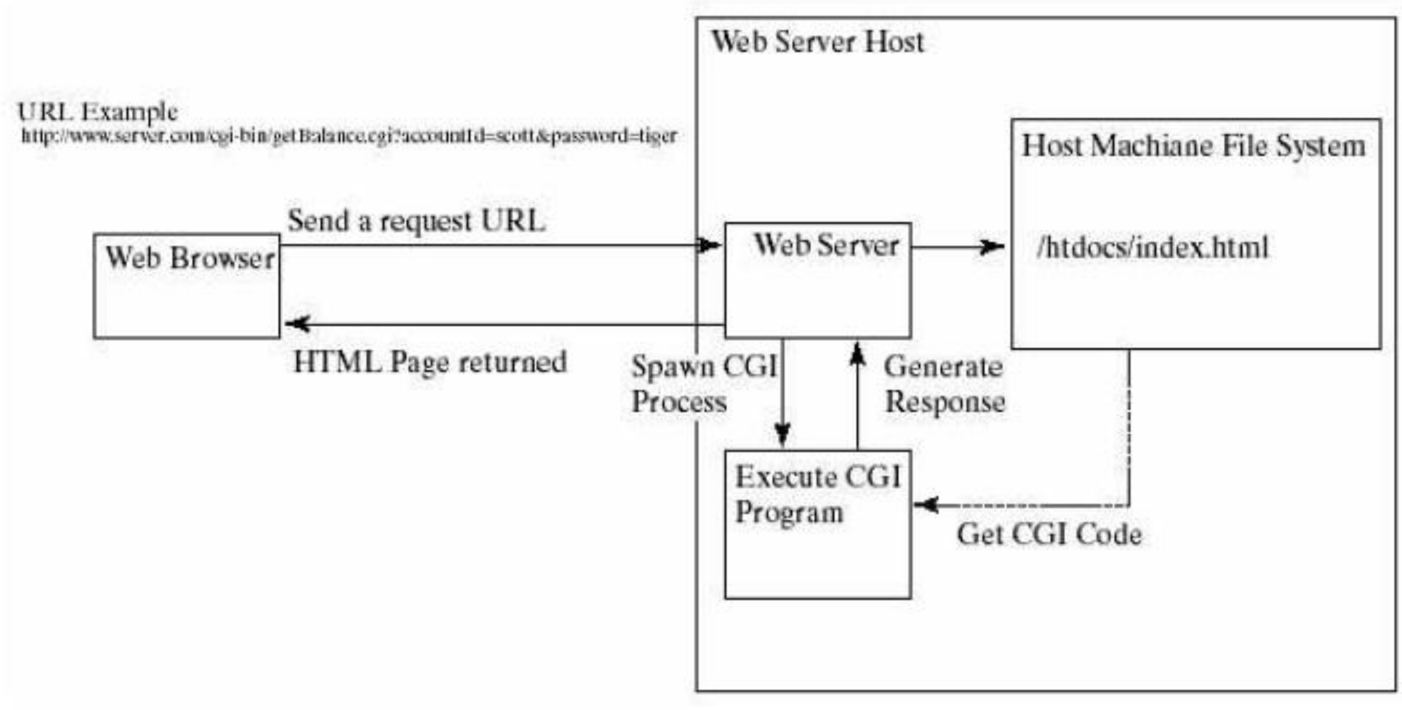
Server Responsibility

- Every server that provides services to remote clients has two main responsibilities. The first is to handle client requests; the second is to create a response to be sent back.
- The first task involves socket programming to extract information from request messages, and implementing client-server protocols, such as FTP and HTTP.
- The second task, creating the response, varies from service to service. It can be as simple as just locating a file and send it back to the client. For a business web application the task involve: retrieving data from the database, applying business rules, and presenting the output in the formats desired by different clients.
- In the early days server is just a single program that takes care of all the different tasks: managing the network, implementing protocols, locating data, applying business logics and replying.
- However, this is not practical for a business web application.
- There is a better design: divide the code into two executable parts—one that handles the network and one that provides the application logic—and let the two executables have a standard interface between them.
- This kind of separation makes it possible to modify the code in the application logic without affecting the network module, as long as the rules of the interface are followed.

Common Gate Way Interface(CGI)

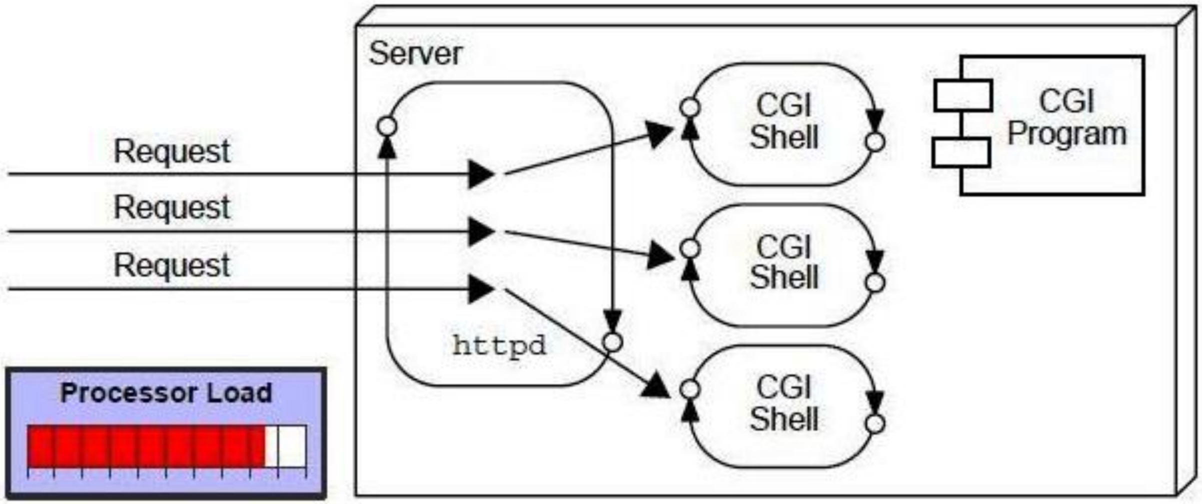
- Traditionally, people have implemented this design for HTTP servers using Common Gateway Interface (CGI). On one side of this interface is the main web server, and on the other side are the CGI scripts.
- The web server acts as the network communications module dealing with low level details and the CGI scripts act as data processing modules and deliver the output. They follow the rules of the “common gateway interface” to pass data between them.
- The Common Gateway Interface, or CGI, was proposed to generate dynamic Web contents. The interface provides a standard framework for Web servers to interact with external programs, known as the CGI programs.
- Perl is one of the earliest popular language for writing CGI
- Servlet is an improved version of CGI script from early days.

How Does CGI Work?



http://www.webserverhost.com/cgi-bin/getBalance.cgi?accountId=scott+smith&password=tiger

Disadvantages of CGI

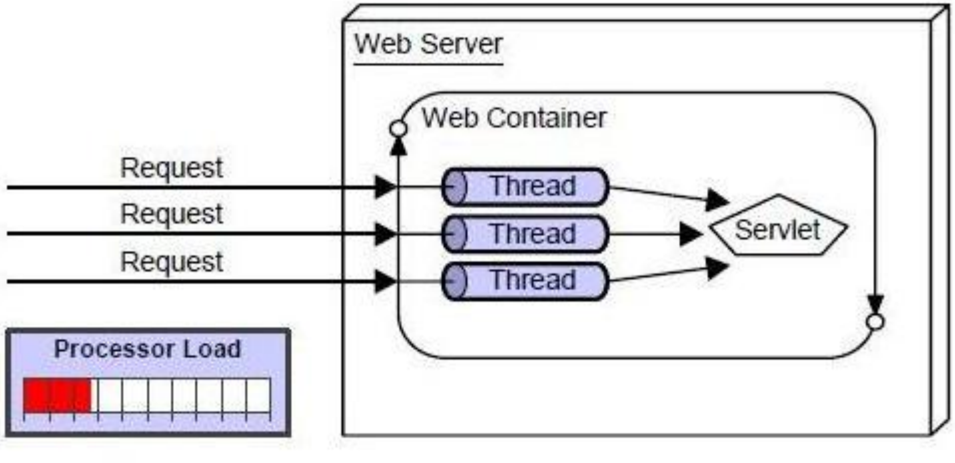


1. Not scalable: If number of requests increase, it takes more time for sending response.

2. Heavy Load: For each request, a new process is spawn

3. Platform dependent language used e.g. C, C++, perl.

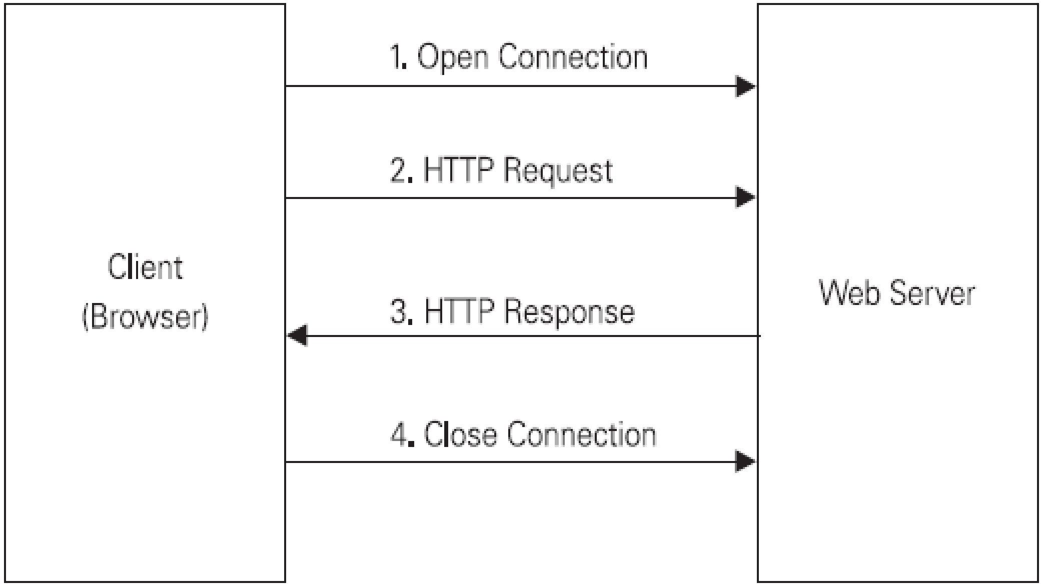
Advantage of Servlet



Better performance: Creates a thread for each request not process. Thread is light weight
Portability: because it uses java language.
Robust: Servlets are managed by JVM so no worry about memory leak, security and etc.

HTTP Protocol

Hypertext Transfer Protocol is a request-response-based stateless protocol.



HTTP request

- The two most commonly used HTTP requests, also known as methods, are GET and POST.
- The Web browser issues a request using a URL or an HTML form to trigger the Web server to execute a CGI program.
- When issuing a CGI request directly from a URL, the GET method is used. This URL is known as a query string.

Query String

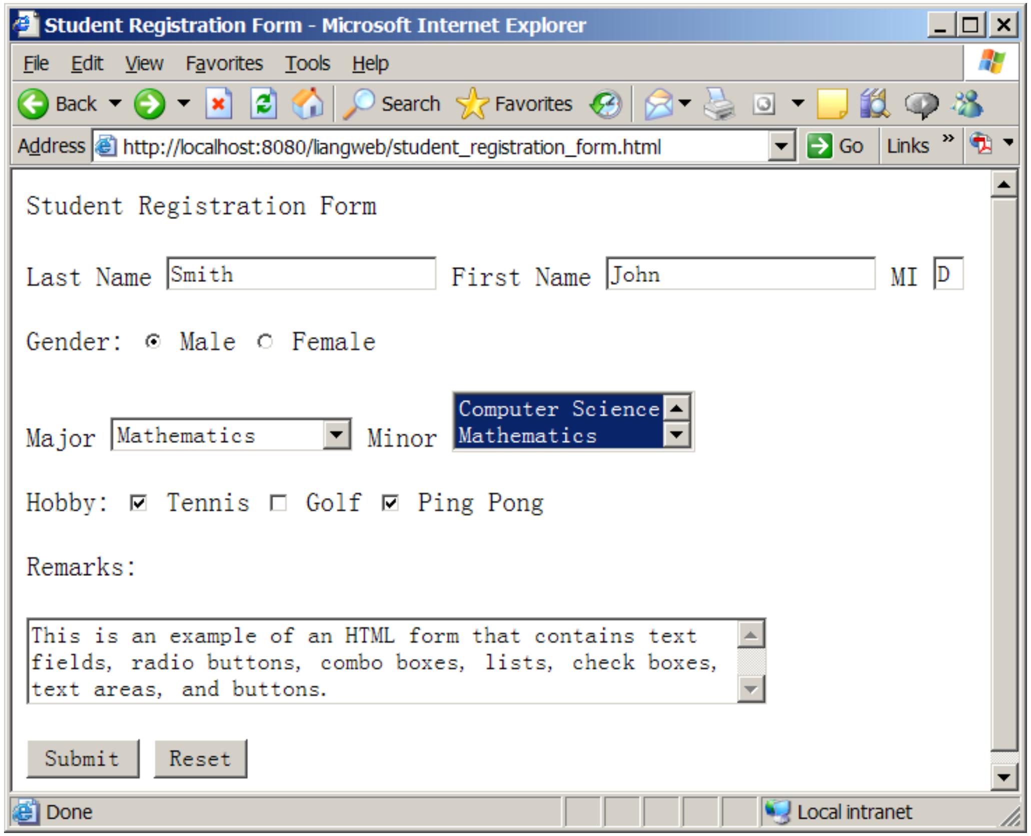
The URL query string consists of the location of the CGI program, parameters and their values.

http://www.webserverhost.com/cgi-bin/getBalance.cgi?accountId=scott+smith&password=tiger

The ? symbol separates the program from the parameters. The parameter name and value are associated using the = symbol. The parameter pairs are separated using the & symbol. The + symbol denotes a space character.

Issuing a request from an HTML form

HTML forms enable you to submit data to the Web server in a convenient form. The form can contain text fields, text area, check boxes, combo boxes, lists, radio buttons, and buttons.


Student Registration Form - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/liangweb/student_registration_form.html
Go Links »
Done
Local intranet

Student Registration Form

Last Name Smith First Name John MI D

Gender: ● Male ○ Female

Major Mathematics Minor Computer Science Mathematics

Hobby: ☑ Tennis □ Golf ☑ Ping Pong

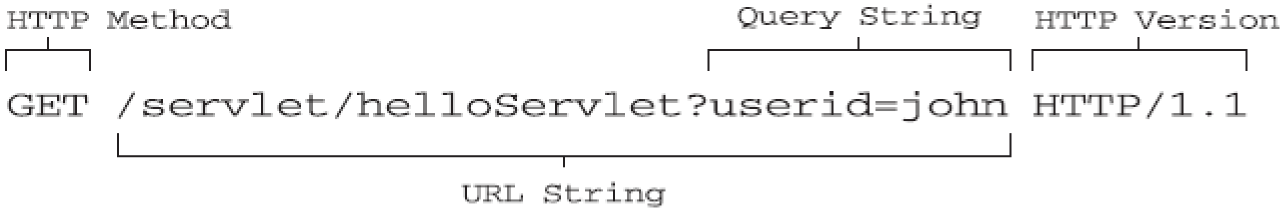
Remarks:

This is an example of an HTML form that contains text fields, radio buttons, combo boxes, lists, check boxes, text areas, and buttons.

Submit Reset

The structure of an HTTP request

- The initial line for an HTTP request has three parts, separated by spaces
  - A method name
  - The local path of the requested resource (URI)
  - The version of HTTP being used
- Example:
- GET /servlet/helloServlet?userid=john HTTP/1.1



- Other HTTP methods: HEAD, POST,PUT, OPTIONS, DELETE, TRACE, and CONNECT

The structure of an HTTP response

- The initial line of an HTTP response is called the status line. It has three parts, separated by spaces: the HTTP version, a response status code that tells the result of the request, and an English phrase describing the status code.
- Example:

```
HTTP/1.1 404 Not Found
HTTP/1.1 500 Internal Error
```

- A typical HTTP response looks like this:

```
HTTP/1.1 200 OK
Date: Tue, 01 Sep 2004 23:59:59 GMT
Content-Type: text/html
Content-Length: 52
<html>
 <body>
    <h1>Hello, John!</h1>
 </body>
</html>
```

GET vs POST

- The GET and POST methods both send requests to the Web server. The POST method always triggers the execution of the corresponding CGI program.
- The GET method may not cause the CGI program to be executed, if the previous

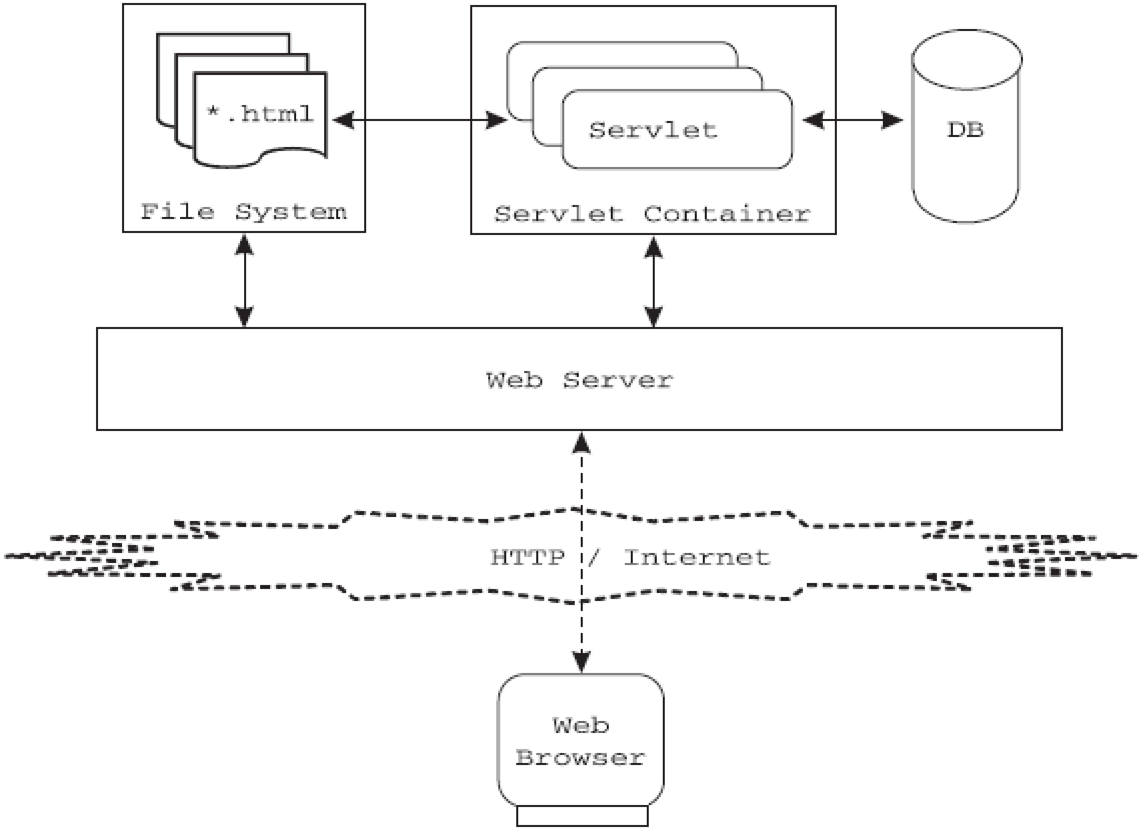
  same request is cached in the Web browser.
- Web browsers often cache Web pages so that the same request can be quickly responded to without contacting the Web server.
- The browser checks the request sent through the GET method as a URL query string. If the results for the exact same URL are cached on a disk, then the previous Web pages for the URL may be displayed.
- To ensure that a new Web page is always displayed, use the POST method.
- For example, use a POST method if the request will actually update the database.
- If your request is not time sensitive, such as finding the address of a student in the database, use the GET method to speed up performance.

FirstServlet

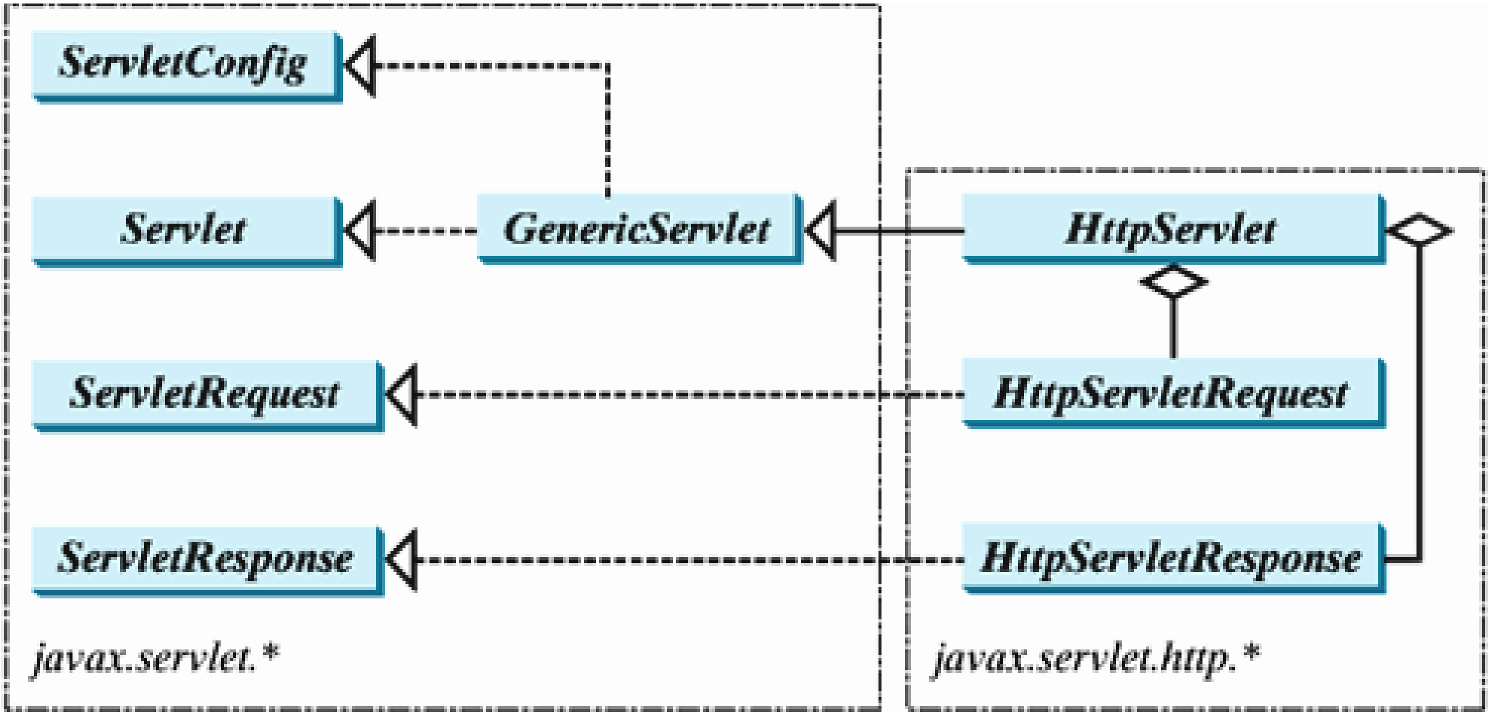
```
import javax.servlet.*;
import javax.servlet.http.*;

public class FirstServlet extends HttpServlet {
  /** Handle the HTTP <code>GET</code> method.
   * @param request servlet request
   * @param response servlet response
   */
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, java.io.IOException {
    response.setContentType("text/html");
    java.io.PrintWriter out = response.getWriter();
    // output your page here
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Servlet</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("Hello, Java Servlets");
    out.println("</body>");
    out.println("</html>");
    out.close();
  }
}
```

SERVLET CONTAINER

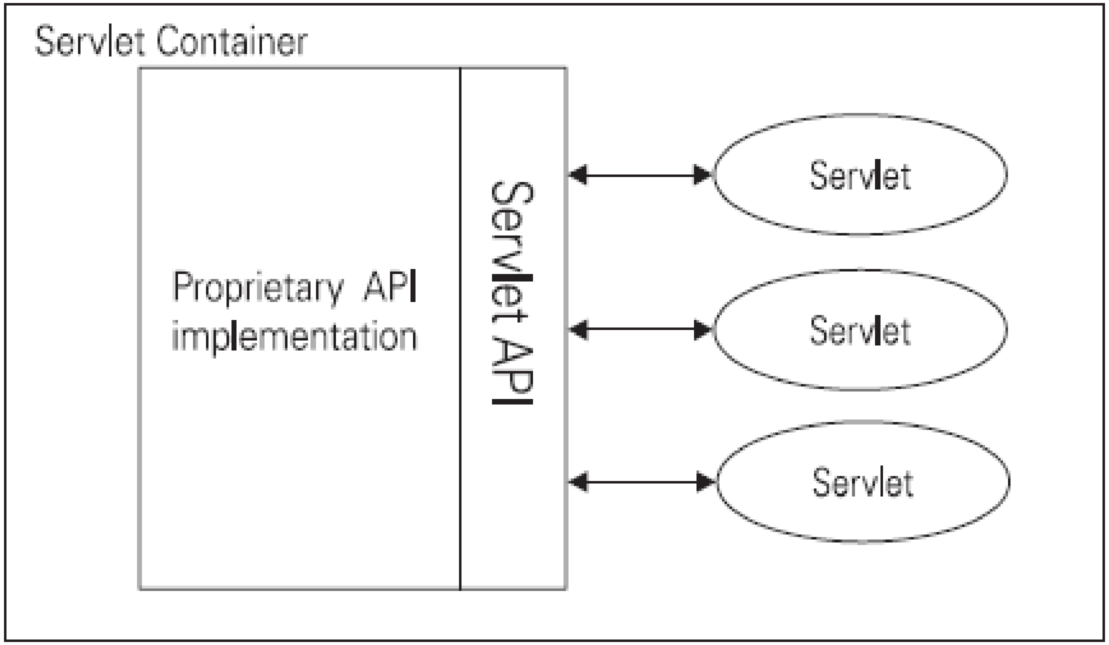
A web server uses a separate module to load and run servlets



Servlet Application Programming Interfaces, or the Servlet API

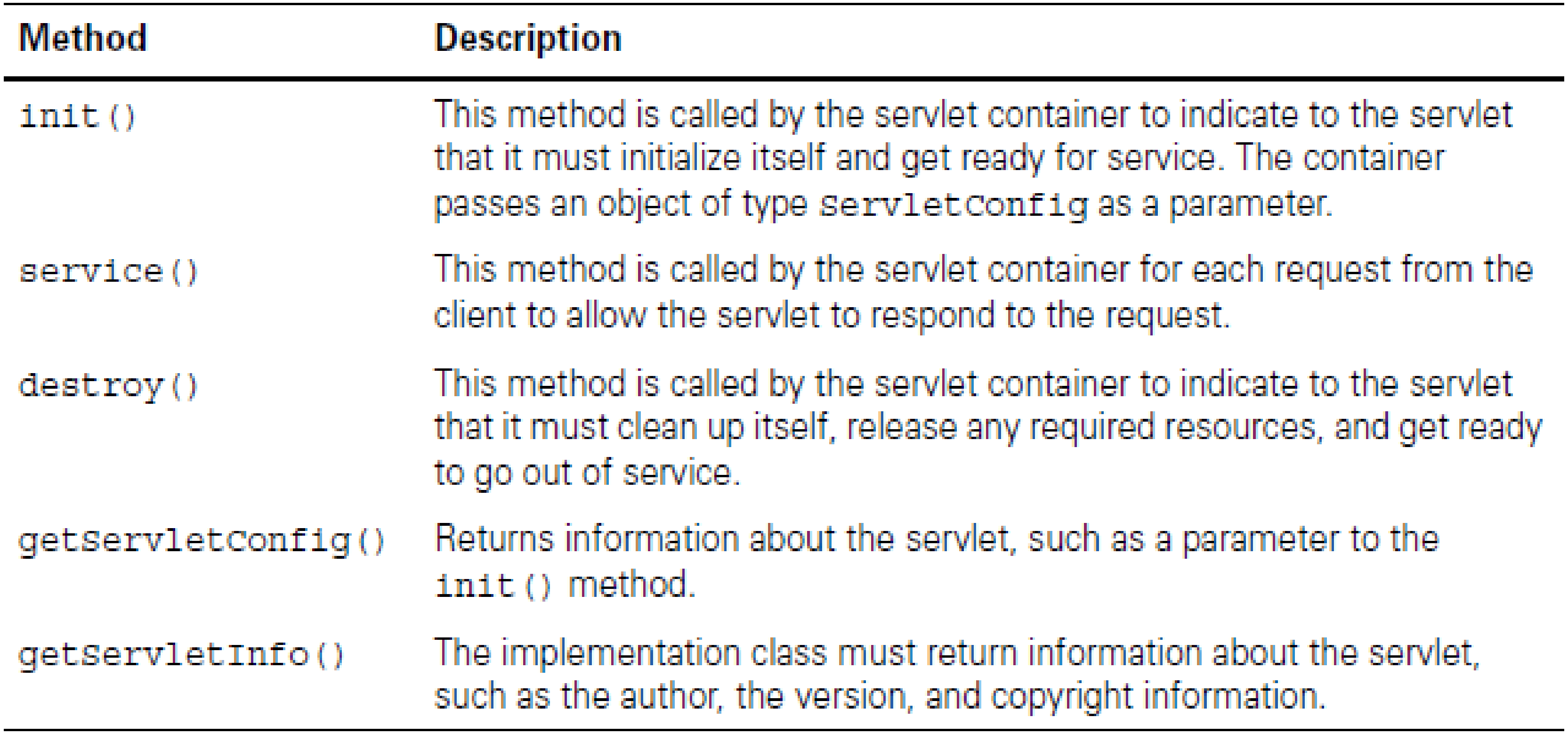


THE RELATIONSHIP BETWEEN A SERVLET CONTAINER AND THE SERVLET API



The javax.servlet.Servlet interface

- Every servlet class must directly or indirectly implement this interface. It has five methods, as shown in the table below



Method	Description
`init()`	This method is called by the servlet container to indicate to the servlet that it must initialize itself and get ready for service. The container passes an object of type `ServletConfig` as a parameter.
`service()`	This method is called by the servlet container for each request from the client to allow the servlet to respond to the request.
`destroy()`	This method is called by the servlet container to indicate to the servlet that it must clean up itself, release any required resources, and get ready to go out of service.
`getServletConfig()`	Returns information about the servlet, such as a parameter to the `init()` method.
`getServletInfo()`	The implementation class must return information about the servlet, such as the author, the version, and copyright information.

Servlet Methods

- /**Invoked for every servlet constructed*/
public void init(ServletConfig p0) throws ServletException;

- /**Invoked to respond to incoming requests*/
public void service(ServletRequest p0, ServletResponse p1)
  throws ServletException, IOException;

- /**Invoked to release resource by the servlet*/
public void destroy();

- /**Return information about the servlet*/
public String getServletInfo();

- /**Return configuration objects of the servlet*/
public ServletConfig getServletConfig();

Servlet Life-Cycle Methods

- 1. The init method is called when the servlet is first created, and is not called again as long as the servlet is not destroyed.
- 2. The service method is invoked each time the server receives a request for the servlet. The server spawns a new thread and invokes service.
- 3. The destroy method is invoked once all threads within the servlet's service method have exited or after a timeout period has passed. This method releases resources for the servlet.

The javax.servlet.GenericServlet class

- The javax.servlet.GenericServlet class defines a generic, protocol independent servlet. It implements javax.servlet.Servlet and javax.servlet.ServletConfig. ServletConfig is an interface that defines four methods (getInitParameter, getInitParameterNames, getServletContext, and getServletName) for obtaining information from a Web server during initialization.
- All the methods in Servlet and ServletConfig are implemented in GenericServlet except the service() method. It also adds a few methods to support logging.
- We can extend this class and implement the service() method to write any kind of servlet.

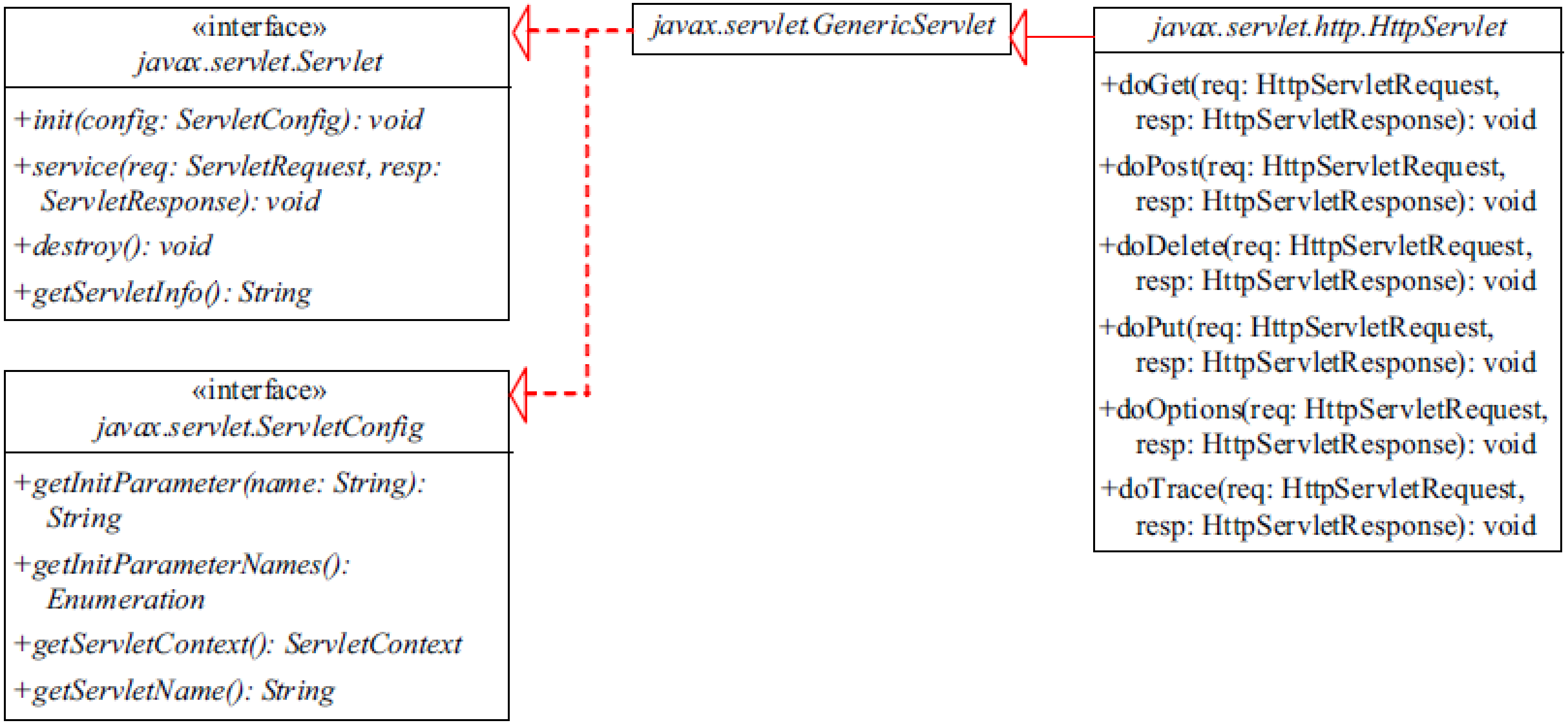
The javax.servlet.ServletRequest interface

- The ServletResponse interface provides a generic way of sending requests. It defines methods that assist in sending a proper requests to the client.

The javax.servlet.ServletResponse interface

- The ServletResponse interface provides a generic way of sending responses. It
- defines methods that assist in sending a proper response to the client.

HttpServlet Extends GenericServlet



«interface»
javax.servlet.Servlet

+init(config: ServletConfig): void

+service(req: ServletRequest, resp: ServletResponse): void

+destroy(): void

+getServletInfo(): String

«interface»
javax.servlet.ServletConfig

+getInitParameter(name: String): String

+getInitParameterNames(): Enumeration

+getServletContext(): ServletContext

+getServletName(): String

javax.servlet.GenericServlet

javax.servlet.http.HttpServlet

+doGet(req: HttpServletRequest, resp: HttpServletResponse): void

+doPost(req: HttpServletRequest, resp: HttpServletResponse): void

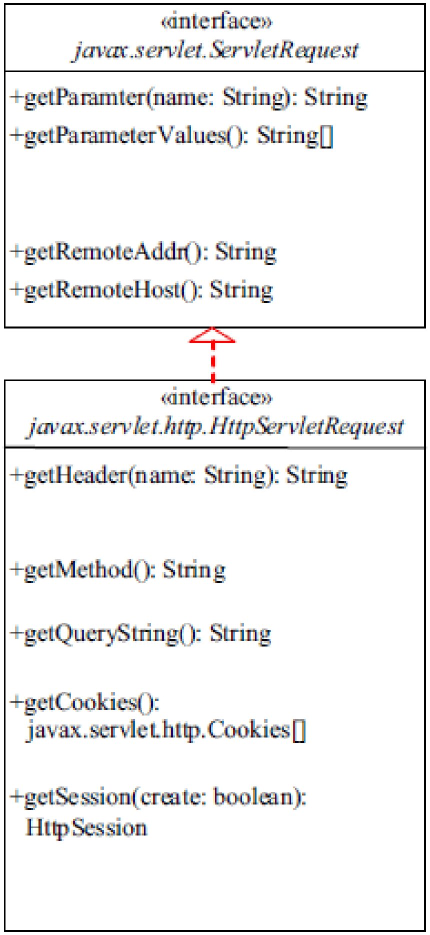
+doDelete(req: HttpServletRequest, resp: HttpServletResponse): void

+doPut(req: HttpServletRequest, resp: HttpServletResponse): void

+doOptions(req: HttpServletRequest, resp: HttpServletResponse): void

+doTrace(req: HttpServletRequest, resp: HttpServletResponse): void

HttpServletRequest is a subinterface of ServletRequest



«interface»
javax.servlet.ServletRequest

+getParamter(name: String): String

+getParameterValues(): String[]

Returns the value of a request parameter as a String, or null if the parameter does not exist. Request parameters are extra information sent with the request. For HTTP servlets, parameters are contained in the query string or posted from data. Only use this method when you are sure that the parameter has only one value. If it has more than one value, use getParameterValues.

+getRemoteAddr(): String

Returns the Internet Protocol (IP) address of the client that sent the request.

+getRemoteHost(): String

Returns the fully qualified name of the client that sent the request, or the IP address of the client if the name cannot be determined.

«interface»
javax.servlet.http.HttpServletRequest

+getHeader(name: String): String

Returns the value of the specified request header as a String. If the request did not include a header of the specified name, this method returns null. Since the header name is case-insensitive, you can use this method with any request header.

+getMethod(): String

Returns the name of the HTTP method with which this request was made; for example, GET, POST, DELETE, PUT, OPTIONS, or TRACE.

+getQueryString(): String

Returns the query string that is contained in the request URL after the path. This method returns null if the URL does not have a query string.

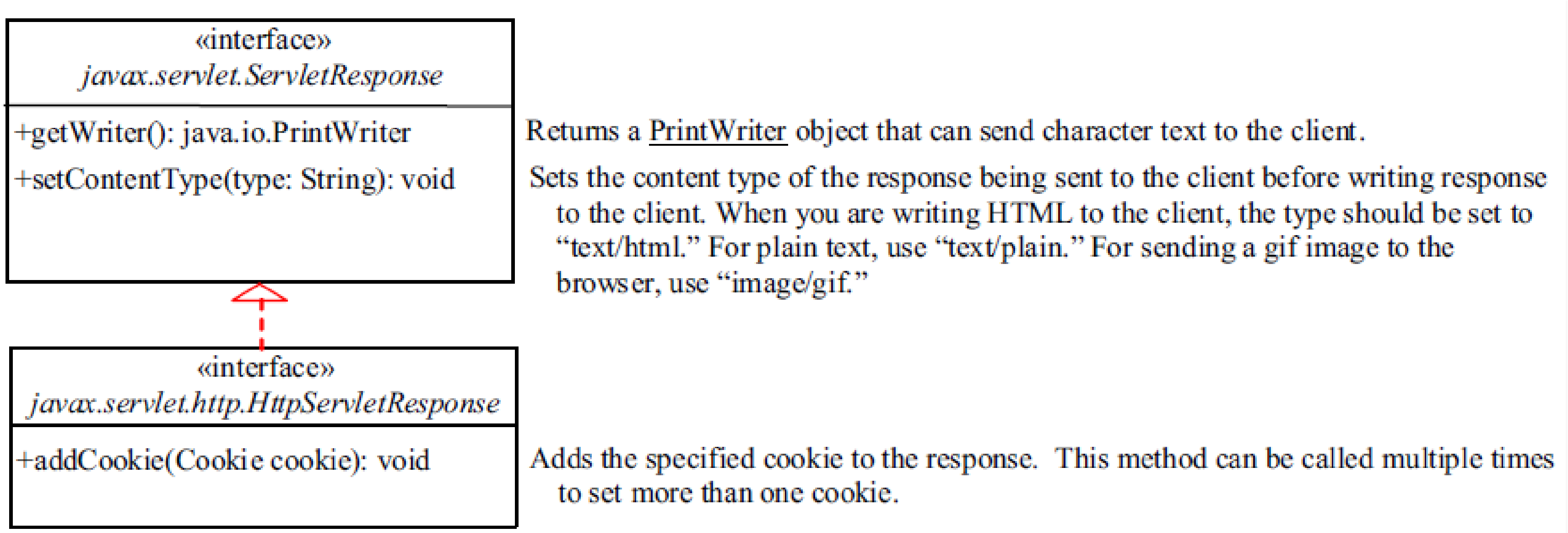
+getCookies(): javax.servlet.http.Cookies[]

Returns an array containing all of the Cookie objects the client sent with the request. This method returns null if no cookies were sent. Using cookies is introduced in Section 26.8.2, "Session Tracking Using Cookies."

+getSession(create: boolean): HttpSession

getSession(true) returns the current session associated with this request. If the request does not have a session, it creates one. getSession(false) returns the current session associated with the request. If the request does not have a session, it returns null. The getSession method is used in session tracking, which is introduced in Section 26.8.3, "Session Tracking Using the Servlet API."

HttpServletResponse is a subinterface of ServletResponse



«interface»
javax.servlet.ServletResponse

+getWriter(): java.io.PrintWriter

+setContentType(type: String): void

«interface»
javax.servlet.http.HttpServletResponse

+addCookie(Cookie cookie): void

Returns a PrintWriter object that can send character text to the client.

Sets the content type of the response being sent to the client before writing response to the client. When you are writing HTML to the client, the type should be set to "text/html." For plain text, use "text/plain." For sending a gif image to the browser, use "image/gif."

Adds the specified cookie to the response. This method can be called multiple times to set more than one cookie.