JEE - Servlets

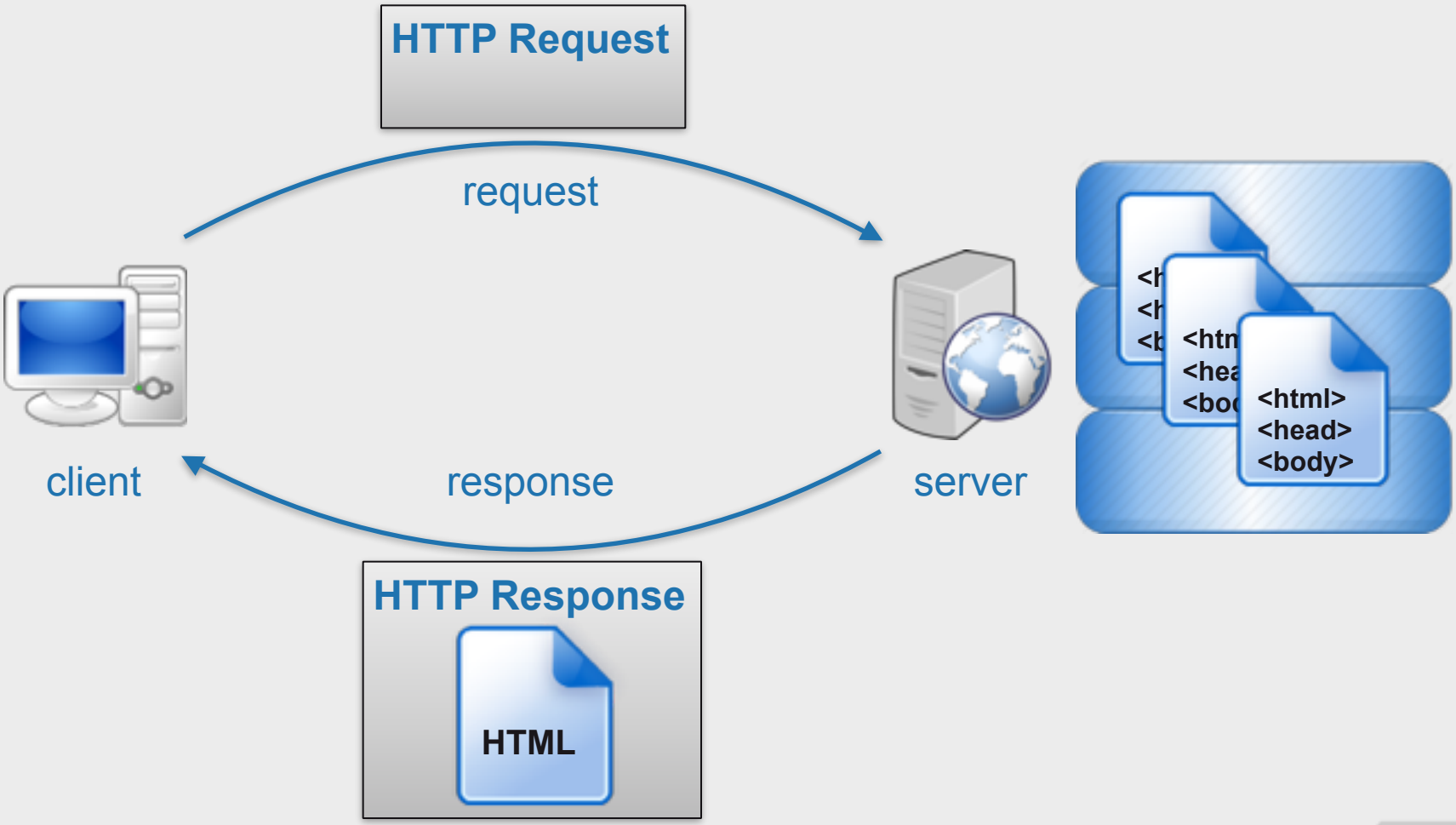
Servlets

- Request-response model
- Servlet Introduction
- Web Container
- Handling GET and POST
- Servlet Lifecycle
- Servlet API
- Examples

Download

- https://dl.dropboxusercontent.com/u/15156388/Servlets.zip

Request-response model



Key elements of request-response stream

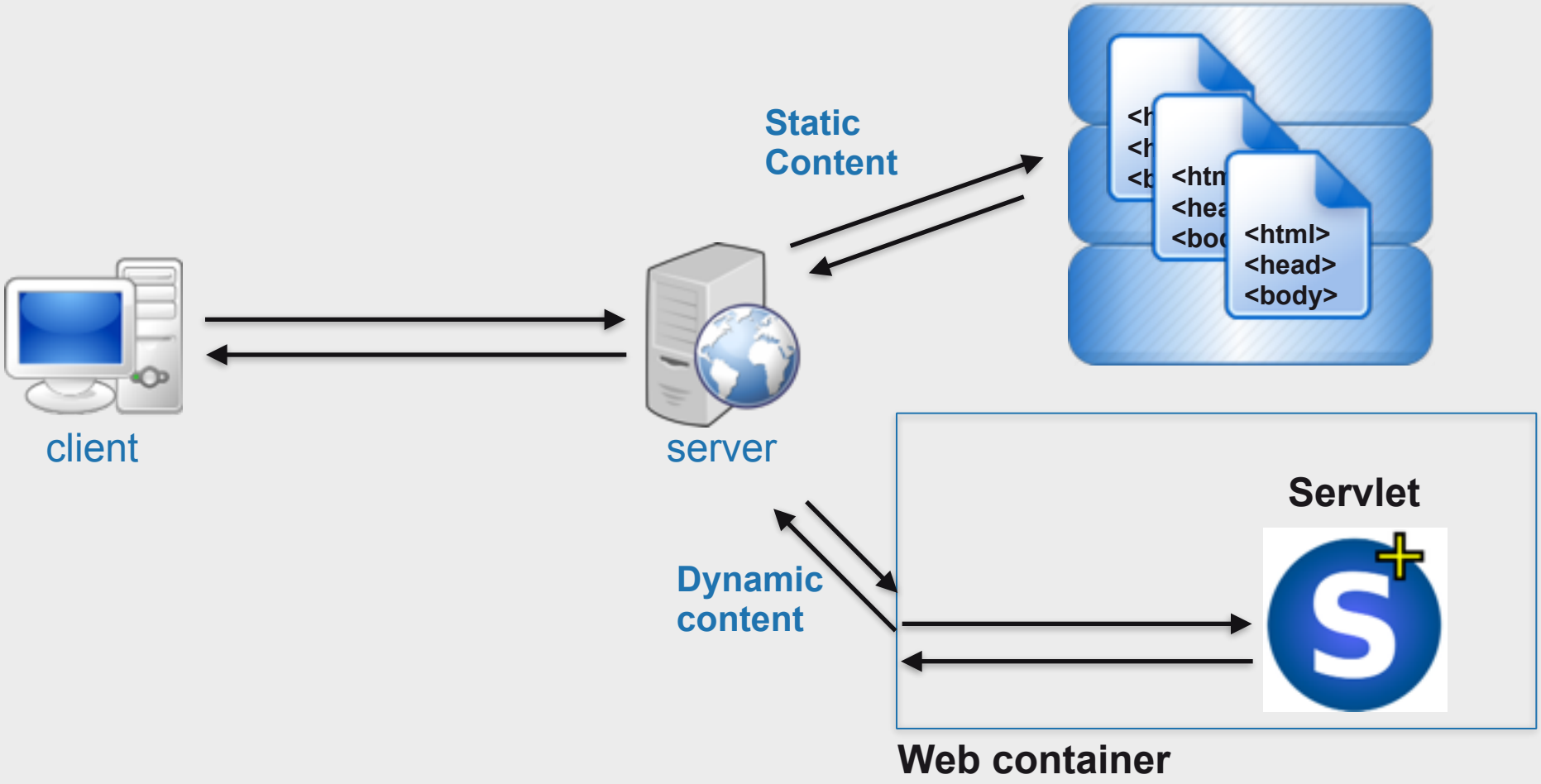
HTTP Request

- HTTP method
  - action to be performed
- The page to access
  - a URL
- Form parameters
  - variables

HTTP Response

- Status code
  - e.g. error 404
- Content type
  - text, picture, html, etc
- Content
  - actual content

Dynamic Content



What Servlets can do

- Create and return an entire HTML Web page containing dynamic content based on the nature of the client request
- Create a portion of an HTML Web page (an HTML fragment) that can be embedded in an existing HTML page
- Communicate with other server resources, including databases and Java-based applications
- Handle connections with multiple clients, accepting input from and broadcasting results to the multiple clients. A servlet can be a multi-player game server, for example.

What Servlets can do

- Open a new connection from the server to an applet on the browser and keep the connection open, allowing many data transfers on the single connection
- Filter data by MIME type for special processing, such as image conversion and server-side includes (SSI)
- Provide customized processing to any of the server's standard routines. For example, a servlet can modify how a user is authenticated.

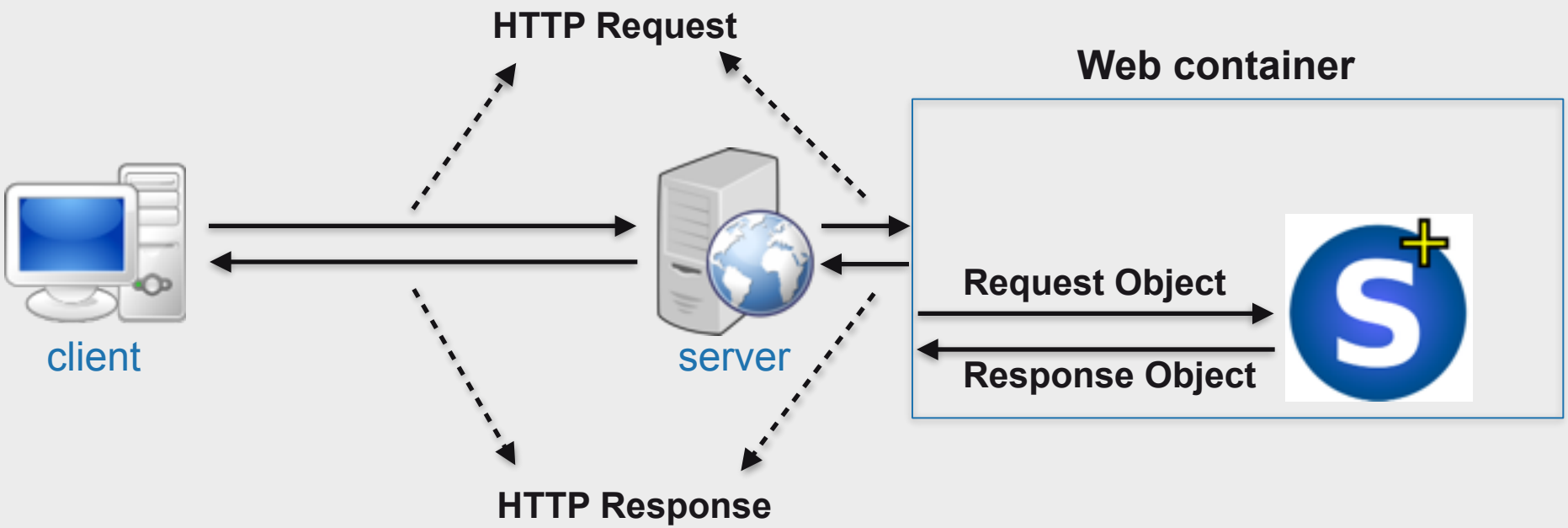
Why use Servlets ?

- Java
  - portable and popular ... it's the heart of the HTTP part of Java EE
- power
  - You can use all Java APIs,
  - Powerful mechanism of annotations,
  - Integration across multiple profiles "Java EE" lightweight Web Profile, big profile "Enterprise" for applications in clusters, etc.
- Efficace
  - Highly scalable

Why use Servlets ?

- **Security**
  - Strong typing,
  - Efficient memory management
- **Strong integration with the server**
  - Strong exchanges between the server and servlets through “context” variable: configuration, resource sharing etc.
- **Scalability & Flexibility**
  - Theoretically the Servlet model is not for HTTP,
  - Powerful mechanisms of "filters" and "chaining" of treatments. More details later.

Web container



Web Container

- Communication support
- Lifecycle management
- Multithreading support
- Security
- JSP support

Servlet API

- javax.servlet
  - Support generic, protocol-independent servlets
  - Servlet (interface)
  - GenericServlet (class)
    - service()
  - ServletRequest and ServletResponse
    - Provide access to generic server requests and responses

Servlet API

- javax.servlet.http
  - Extended to add HTTP-specific functionality
  - HttpServlet (extends GenericServlet)
    - doGet()
    - doPost()
  - HttpServletRequest and HttpServletResponse
    - Provide access to HTTP requests and responses

User-defined Servlets

- For general servlets, define the service() method.
- For HTTP servlets, override one or more of the following methods as needed:
  - doGet()
  - doPost()
  - doPut()
  - doDelete()
- Have no `main()` method

doGet() and doPost()

```
protected void doGet(
                HttpServletRequest req,
                HttpServletResponse resp)

protected void doPost(
                HttpServletRequest req,
                HttpServletResponse resp)
```

doGet() and doPost()

- These methods have two parameters:
- The HttpServletRequest object
  - Contains the client's request
    - Header of the HTTP request
    - HTTP parameters (form data or passed with the URL parameters)
    - Other data (cookies, URL, relative path, etc.)
- The HttpServletResponse object
  - Encapsulates the data back to the client
    - HTTP response header (content type, cookies, etc.)
    - Response body (as OutputStream)

Compare GET vs. POST

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	maximum URL length is 2048 characters	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure because data sent as part of the URL	POST is a little safer because no browser history and no web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

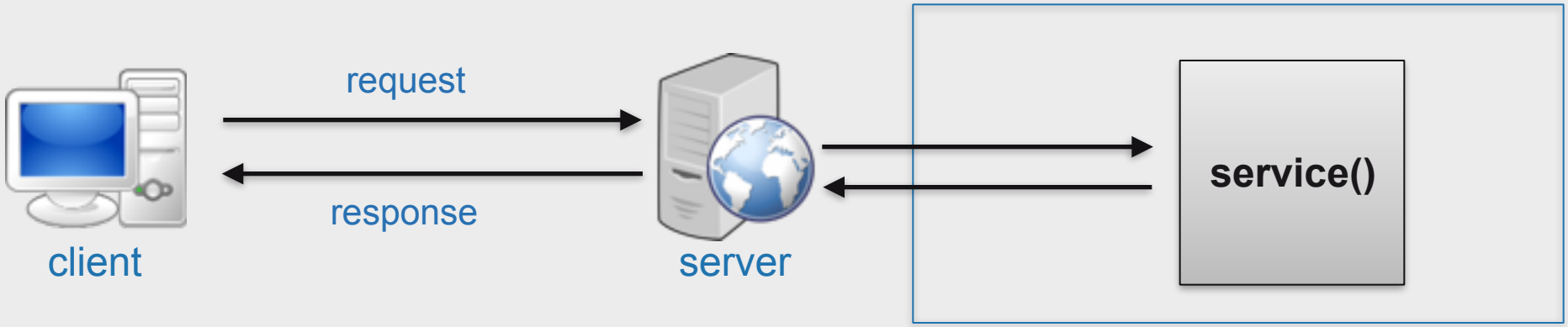
Hello World

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet {
        public void doGet(HttpServletRequest req, HttpServletResponse res)
                throws ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("<html>");
        out.println("<head><title>hello world</title></head>");
        out.println("<body>");
        out.println("<big>hello world</big>");
        out.println("</body></html>");
        }
public void doPost(HttpServletRequest req, HttpServletResponse res)
                throws ServletException, IOException {
        doGet(req, res);
        }
}
```

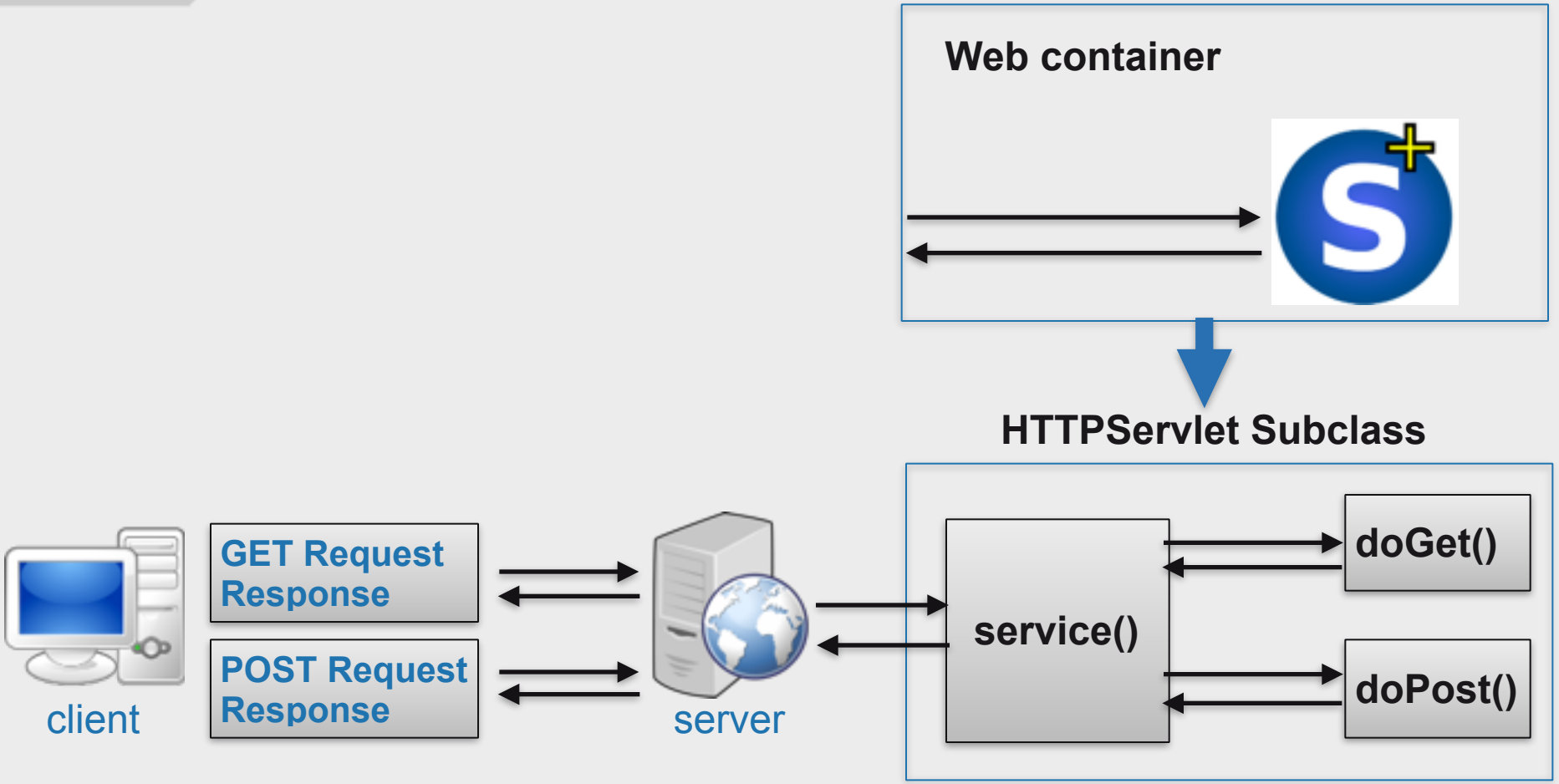
Using JAVA in Servlets- Date Servlet

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

public class DateServlet extends HttpServlet {
        public void doGet(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {

        PrintWriter out = Res response.getWriter();
        out.println("<HTML>");
        out.println("The time is: " + new java.util.Date());
        out.println("</HTML>");
        }
}
```

GenericServlet



Handling GET and POST Request



Servlet execution process (1/2)

Client

- Makes a request to a servlet

Web Server

- Receives the request
- Identifies the request as a servlet request
- Passes the request to the servlet container

Servlet execution process (2/2)

Servlet Container

- Locates the servlet
- Passes the request to the servlet

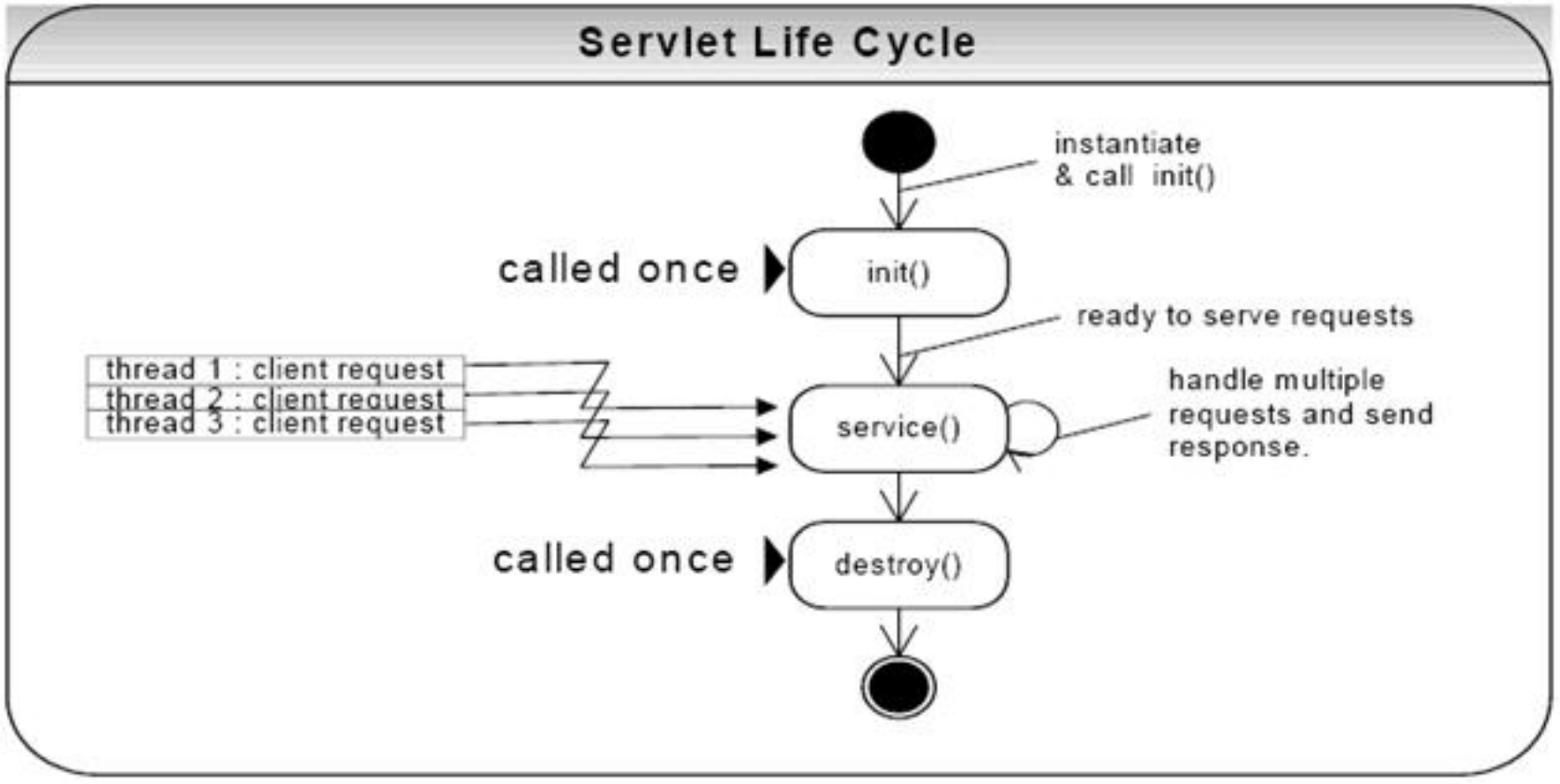
Servlet

- Executes in the current thread
- The servlet can store/retrieve objects from the container
- Output is sent back to the requesting browser via the web server
- Servlet continues to be available in the servlet container

Servlet Container

- Provides web server with servlet support
  - Execute and manage servlets
  - e.g., Tomcat
- Must conform to the following lifecycle contract
  - Create and initialize the servlet
  - Handle zero or more service calls from clients
  - Destroy the servlet and then garbage collect it
- Servlet Container types
  - Standalone container
  - Add-on container
  - Embeddable container

Servlet Lifecycle



Loading Servlet

- The init() method executes only one time during the lifetime of the servlet.
- It is not repeated regardless of how many clients access the servlet
- It can be overridden to manage servlet-wide resources.
  - for example is initializing a database connection
- May be called ...
  - When the server starts
  - When the servlet is first requested, just before the service() method is invoked
  - At the request of the server administrator

Unloading Servlet

- The destroy() method executes only one time
  - when the server stops and unloads the servlet
- destroy()
  - Resources acquired should be freed up
  - A chance to write out its unsaved cached info
  - Last step before being garbage collected
- It can be overridden, typically to manage servlet-wide resources.
  - for example, closing a database connection

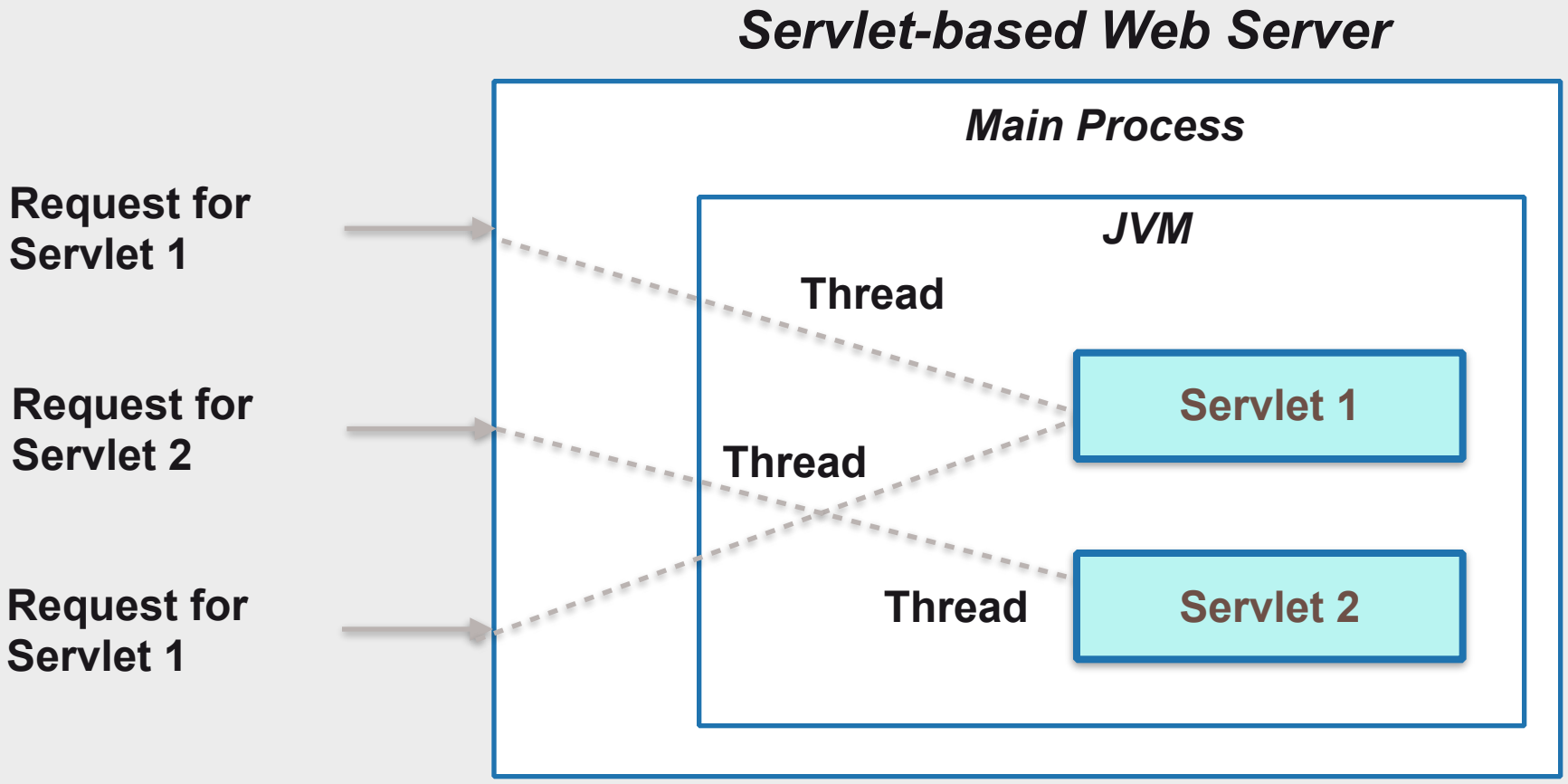
service()

- The service() method is invoked for each client request.
- In HttpServlet, the default service function invokes the do function corresponding to the method of the HTTP request.
  - If the form information is provided by GET, the doGet() method is invoked
  - If the form information is provided by POST, the doPost() method is invoked
  - When a client invokes a servlet directly by the servlet's URL, the client uses GET

Servlet Instance Persistence

- Servlets persist between requests as object instances
- When servlet is loaded, the server creates a single instance
- The single instance handles every request made of the servlet
- Improves performance in three ways
  - Keeps memory footprint small
  - Eliminates the object creation overhead
  - Enables persistence
    - May have already loaded required resources

Servlet Thread Model



Servlets API details

- Main functions of a Servlet:
  - Treat HTTP parameters received in the request (GET or POST)
    - HttpServletRequest.getParameter (String)
  - Retrieve a configuration parameter of the web application (in the web.xml descriptor)
    - ServletConfig.getInitParameter ()
  - Recover a part of the HTTP header
    - HttpServletRequest.getHeader (String)

Servlets API details

- Specify the response type in the header
  - HttpServletResponse.setHeader (<name>, <value>) / HttpServletResponse.setContentType (String)
- Retrieve a Writer to write the response
  - HttpServletResponse.getWriter ()
- or ... if the response is binary
  - HttpServletResponse.getOutputStream ()
- Redirect the request to another URL
  - HttpServletResponse.sendRedirect ()

Getting Request information

- The Request object has specific methods to retrieve information provided by the client:
  - getParameterNames()
    - returns a string array containing the parameters
  - getParameter()
    - when provided a parameter name, returns its corresponding value
  - getParameterValues()

Returning Response information

- The Response object creates a response to return to the requesting client.
  - Response header
  - Response body
- The Response object also has the getWriter() method to return a PrintWriter object.
  - Use the print() and println() methods

Example: Parameters

```
<html>
<body>

<form method="GET" action="HelloServlet">
Enter your name :
<input type="text" name="name">
<input type="submit" value="OK">
</form>

</body>
</html>
```

Example: Parameters

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloServlet extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType("text/html");
        ServletOutputStream out = response.getOutputStream();
        String nom = request.getParameter("name");
        out.println("<html><head>");
        out.println("\t<title>Hello Servlet</title>");
        out.println("</head><body>");
        out.println("\t<h1>Hello, " + nom + "</h1>");
        out.println("</body></html>");
        }
}
```

Sending a zip through Servlet

Example from http://www.kodejava.org/examples/590.html

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
        try {
        String path = getServletContext().getRealPath("data");
        File directory = new File(path);
        String[] files = directory.list();
        if (files != null && files.length > 0) {
              byte[] zip = zipFiles(directory, files);
              ServletOutputStream sos = response.getOutputStream();
              response.setContentType("application/zip");
              response.setHeader("Content-Disposition", "attachment;filename=\"DATA.ZIP\"");
              sos.write(zip); sos.flush();
}
        } catch (Exception e) {
              e.printStackTrace();
        }
}
```