Bài 2: Cơ bản Servlet

Mục tiêu bài học

- - Servlet là gì?
- - Servlet Scope Object
- - Servlet Request
- - Servlet Response
- - Section Tracking

Đặc điểm của công nghệ web tĩnh

- Chỉ chứa các nội dung cố định,
- Khó cập nhật nội dung, khó nâng cấp mở rộng nên chỉ thích hợp với những doanh nghiệp nhỏ,
- Không thân thiện với người dùng, người dùng muốn cập nhật, thêm mới thông tin thì phải hiểu biết ngôn ngữ lập trình HTML
- Khả năng tương tác yếu

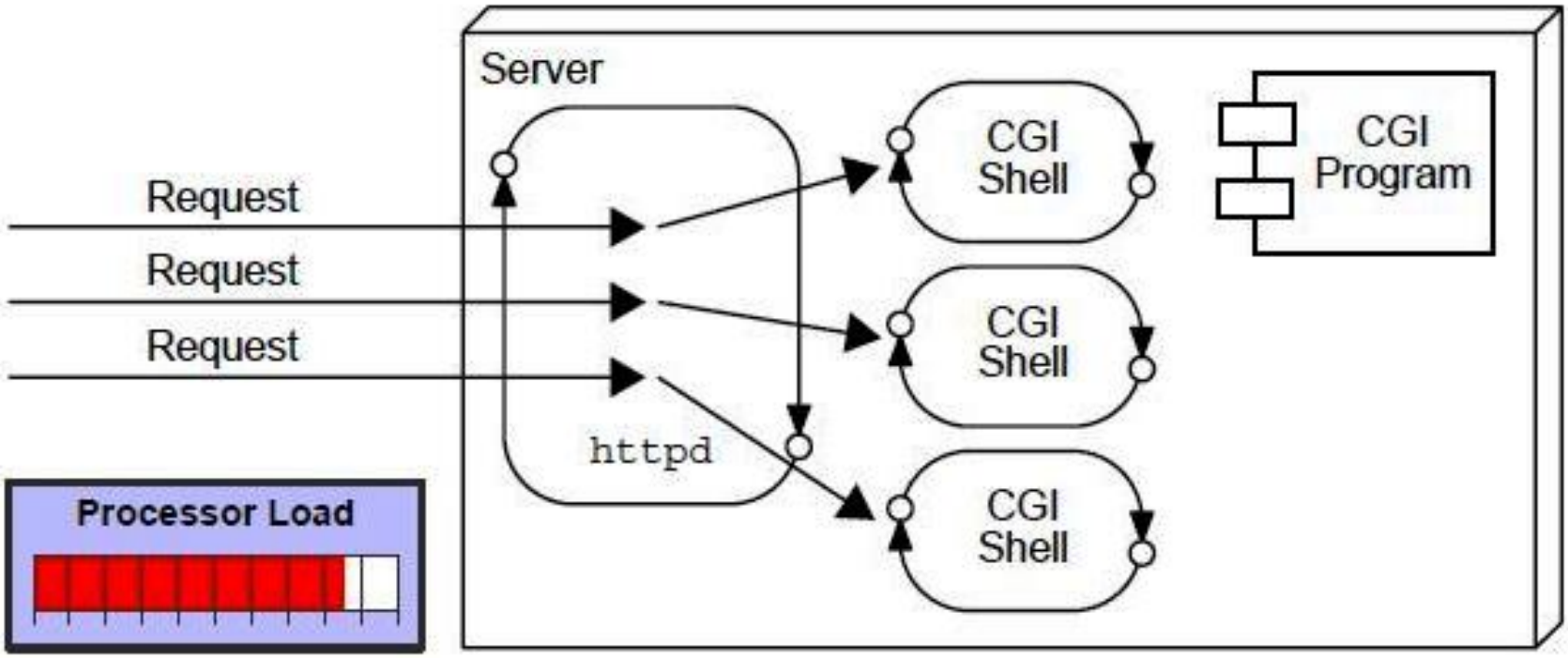
Sự tiến hóa của công nghệ web

- Do đó phát sinh nhu cầu sử dụng trang web như một ứng dụng:
    - Kết nối đến CSDL, do đó web có thể lưu trữ lượng thông tin lớn
    - Nhận yêu cầu từ phía client, xử lý và tính toán trên server
    - Nhiều người có thể cùng lúc cập nhật, thêm mới dữ liệu cho web mà không cần hiểu biết ngôn ngữ lập trình
- Có rất nhiều công nghệ để cho phép ta đưa ứng dụng vào chạy trong môi trường web như: CGI, ASP, ISAPI, JSP, Servlet… trong số đó có JSP/Servlet là công nghệ của Java

CGI là gì?

- Là một chuẩn để viết ứng dụng web, được lập trình bằng C, C++ hoặc Perl.
- CGI cho phép máy chủ web gọi chương trình bên ngoài và chuyển thông tin yêu cầu tới một chương trình bên ngoài khác để xử lý.
- Với mỗi yêu cầu, chương trình CGI sẽ khởi tạo một tiến trình mới

CGI là gì?


Server
CGI
Shell
CGI
Program
Request
Request
Request
CGI
Shell
httpd
CGI
Shell
Processor Load

CGI là gì?

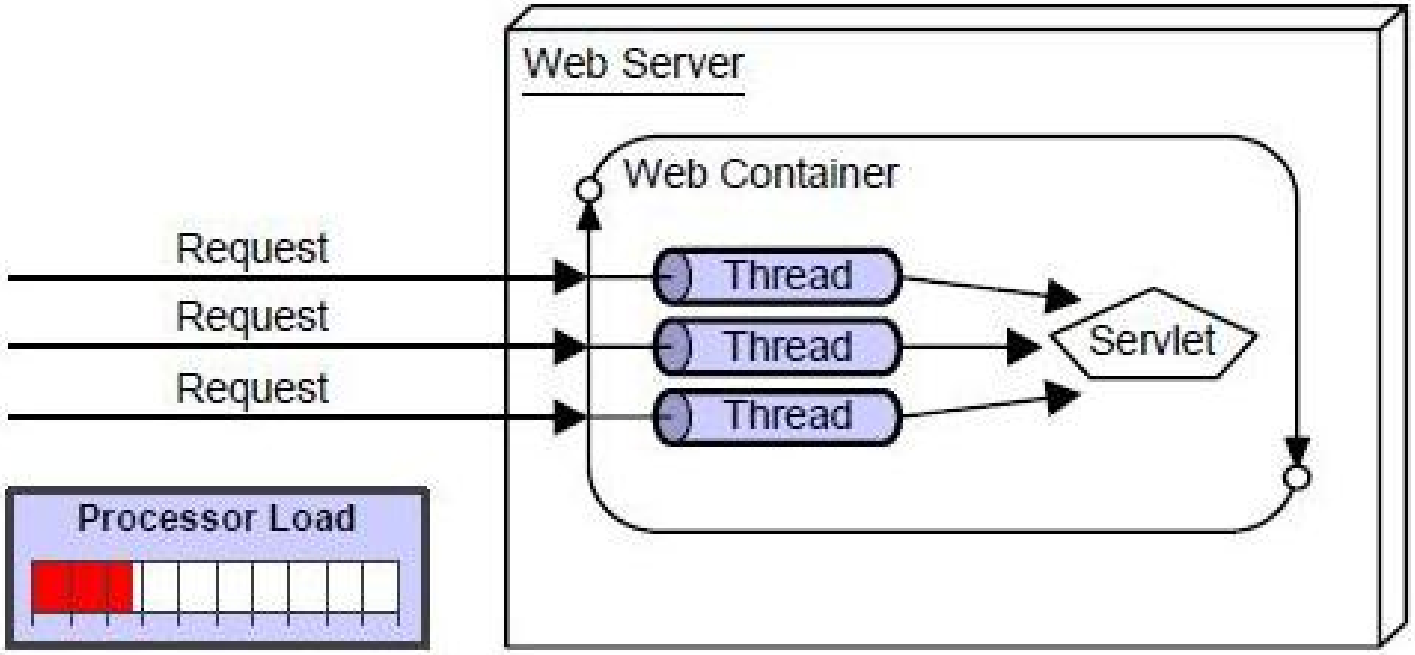
- **Nhược điểm của CGI**

  1. Nếu số client tăng thì thời gian trả lời yêu cầu từ client tăng lên
  2. Với mỗi yêu cầu, nó khởi tạo một tiến trình trong khi máy chủ Web bị hạn chế về tài nguyên

Servlet là gì?

- Servlet là các đối tượng Java, mở rộng chức năng của một HTTP server, do đó được viết bằng ngôn ngữ Java
- Là những chương trình độc lập platform và chạy phía server
- Cơ chế hoạt động theo mô hình CGI mở rộng
- Chương trình servlet:
  - Thường extends class HttpServlet. Không có method main
  - Phải được dịch ra ở dạng byte-code và khai báo với web server

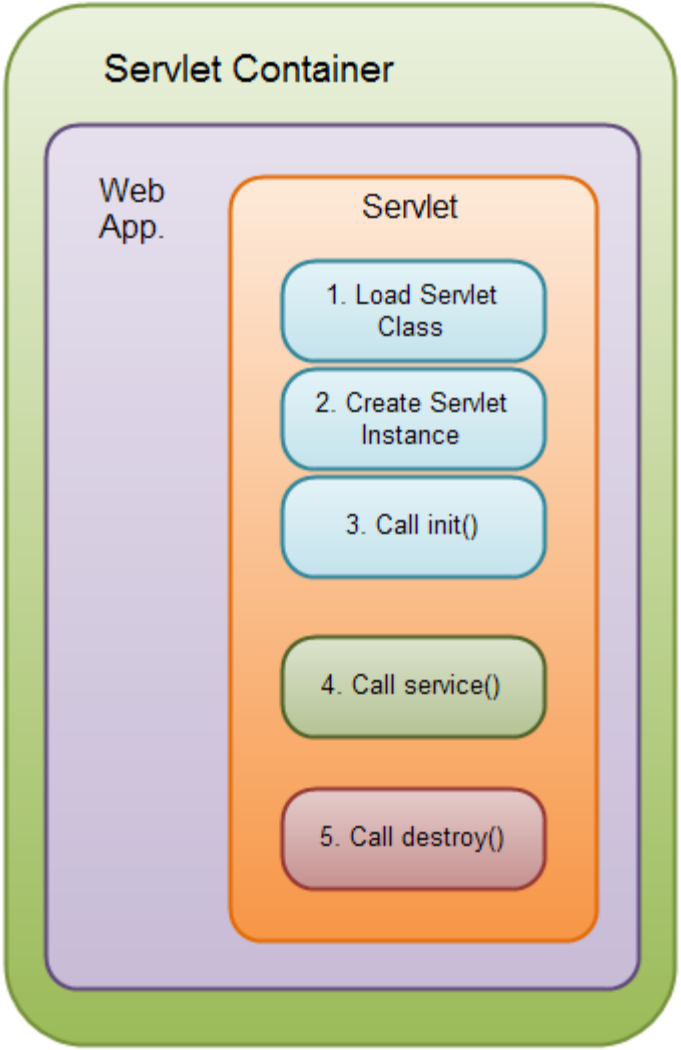
Servlet là gì?



Servlet là gì?

- Servlet có nhiều ưu điểm so với CGI.
  - **Hiệu suất xử lý** tốt hơn(better performance): vì nó tạo ra một thread cho một yêu cầu chứ không phải tiến trình.
  - **Khả chuyển**: bởi vì servlet được phát triển từ ngôn ngữ Java
  - **Mạnh**(Robust): Servlet được quản lý bởi JVM, JVM chủ động quản lý bộ nhớ và thu thập rác.
  - **An toàn**(Secure): bởi vì Java là ngôn ngữ an toàn

Vòng đời của servlet



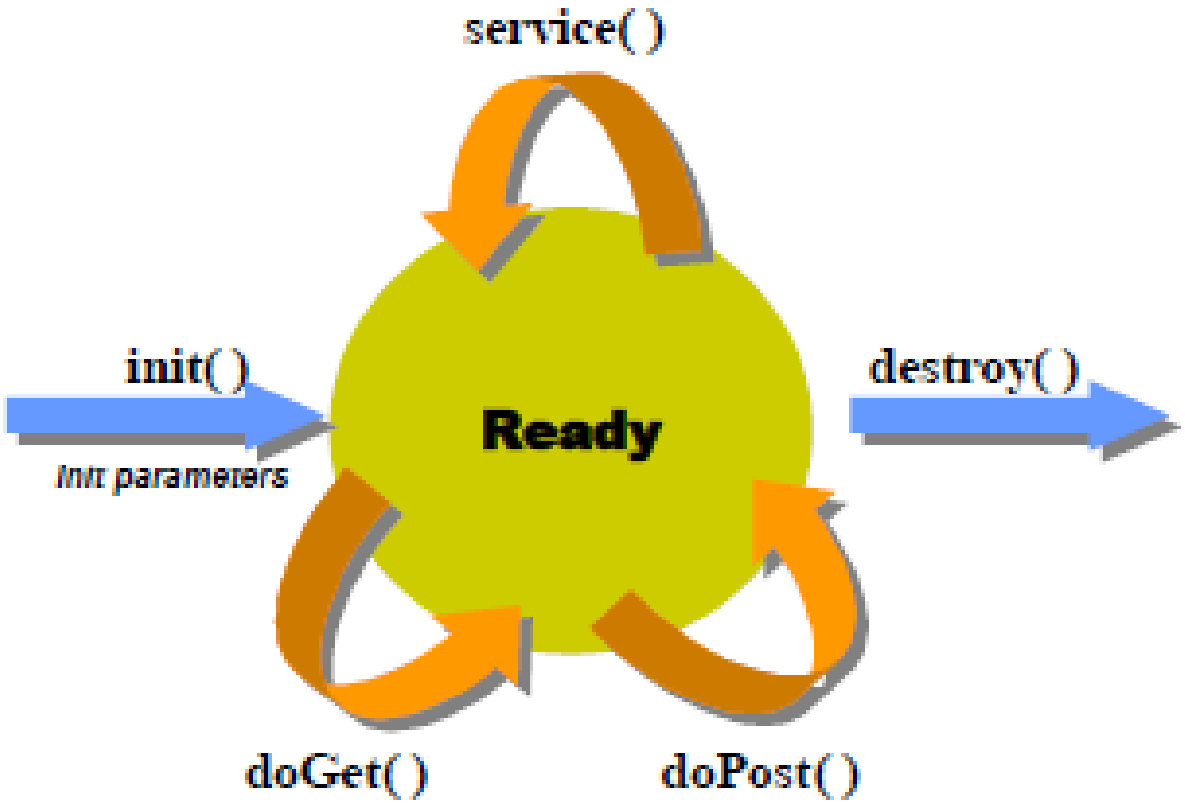
Vòng đời của Servlet

- Có 5 bước:
  - Tải Servlet Class vào bộ nhớ.
  - Tạo đối tượng Servlet.
  - Gọi method servlets init()
  - Gọi method servlets service().
  - Gọi method servlets destroy()

Vòng đời của Servlet

- **Bước 1, 2 và 3** được thực thi một lần duy nhất, khi mà servlet được nạp lần đầu. Mặc định các servlet không được tải lên cho tới khi nó nhận một đòi hỏi đầu tiên từ người dùng. Bạn có thể buộc ServletContainer (Bộ chứa các servlet) tải các servlet khi nó khởi động.
- **Bước 4** được thực thi nhiều lần, mỗi khi có đòi hỏi từ phía người dùng tới servlet.
- **Bước 5** nó được thực thi khi bộ chứa servlet (Servlet Container) trút bỏ (unloaded) servlet

Các phương thức trong vòng đời Servlet



Các phương thức trong vòng đời Servlet

- init()
  - Được gọi MỘT lần khi servlet được tạo thể hiện hóa lần đầu tiên
  - Thực hiện các khởi tạo trong phương thức này
    - Ví dụ: tạo 1 kết nối CSDL
- destroy()
  - Được gọi trước khi hủy 1 servlet instance
  - Thực hiện thao tác dọn dẹp
    - Ví dụ: đóng kết nối CSDL đã mở

Các phương thức trong vòng đời Servlet

```
public class CatalogServlet extends HttpServlet {
  private BookDB bookDB;

  // Perform any one-time operation for the servlet,
  // like getting database connection object.

  // Note: In this example, database connection object is assumed
  // to be created via other means (via life cycle event mechanism)
  // and saved in ServletContext object. This is to share a same
  // database connection object among multiple servlets.
  public void init() throws ServletException {
    bookDB = (BookDB)getServletContext().
                     getAttribute("bookDB");
    if (bookDB == null) throw new
      UnavailableException("Couldn't get database.");
  }
  ...
}
```

int() đọc tham số cấu hình

```
public void init(ServletConfig config) throws
   ServletException {
      super.init(config);
      String driver = getInitParameter("driver");
      String fURL = getInitParameter("url");
      try {
        openDBConnection(driver, fURL);
      } catch (SQLException e) {
         e.printStackTrace();
      } catch (ClassNotFoundException e){
         e.printStackTrace();
      }
}
```

Thiết lập các tham số trong web.xml

```
<web-app>
    <servlet>
        <servlet-name>chart</servlet-name>
        <servlet-class>ChartServlet</servlet-class>
        <init-param>
            <param-name>driver</param-name>
            <param-value>
              COM.cloudscape.core.RmiJdbcDriver
            </param-value>
        </init-param>

        <init-param>
            <param-name>url</param-name>
            <param-value>
              jdbc:cloudscape:rmi:CloudscapeDB
            </param-value>
        </init-param>
    </servlet>
</web-app>
```

destroy()

```
public class CatalogServlet extends HttpServlet {
  private BookDB bookDB;

  public void init() throws ServletException {
    bookDB = (BookDB)getServletContext().
                     getAttribute("bookDB");
    if (bookDB == null) throw new
      UnavailableException("Couldn't get database.");
  }
  public void destroy() {
         bookDB = null;
  }
  ...
}
```

service()

- Method service() của servlet được gọi mỗi khi có request từ client đến. Tùy vào các tình huống cụ thể nó gọi đến một trong các method doGet(..), doPost(...). Tại các servlet của bạn, bạn phải ghi đè và xử lý tại các method này

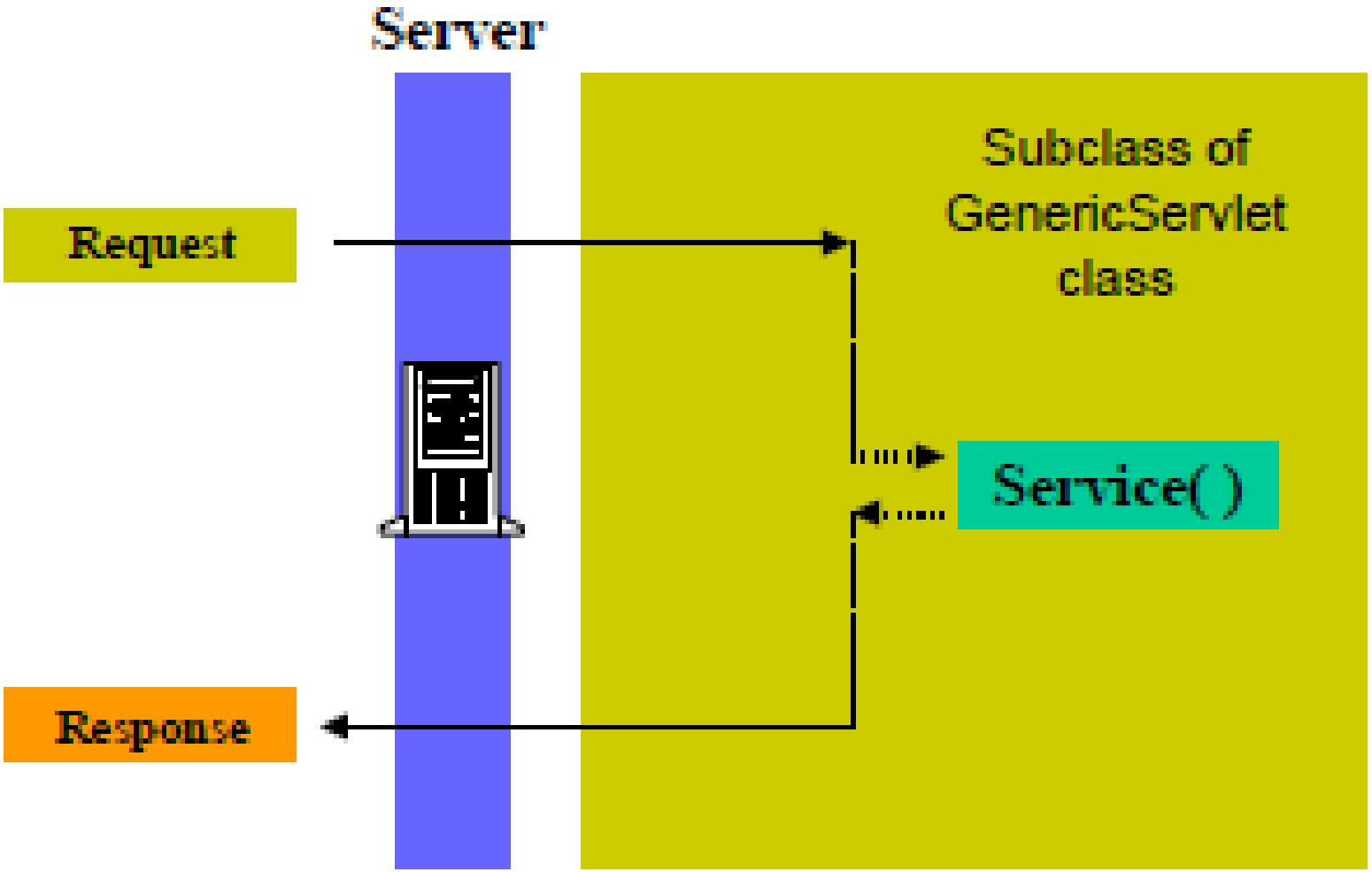
service()

- service() trong javax.servlet.GenericServlet
  - Phương thức Abstract
- service() trong lớp javax.servlet.http.HttpServlet
  - Phương thức cụ thể (đã cài đặt)
  - gọi tới (dispatch) doGet(), doPost()
  - KHÔNG override phương thức này!
- doGet(), doPost(), doXxx() trong javax.servlet.http.HttpServlet
  - Xử lý các HTTP GET, POST requests
  - Lập trình viên override những phương thức này trong servlet của mình để có xử lý phù hợp

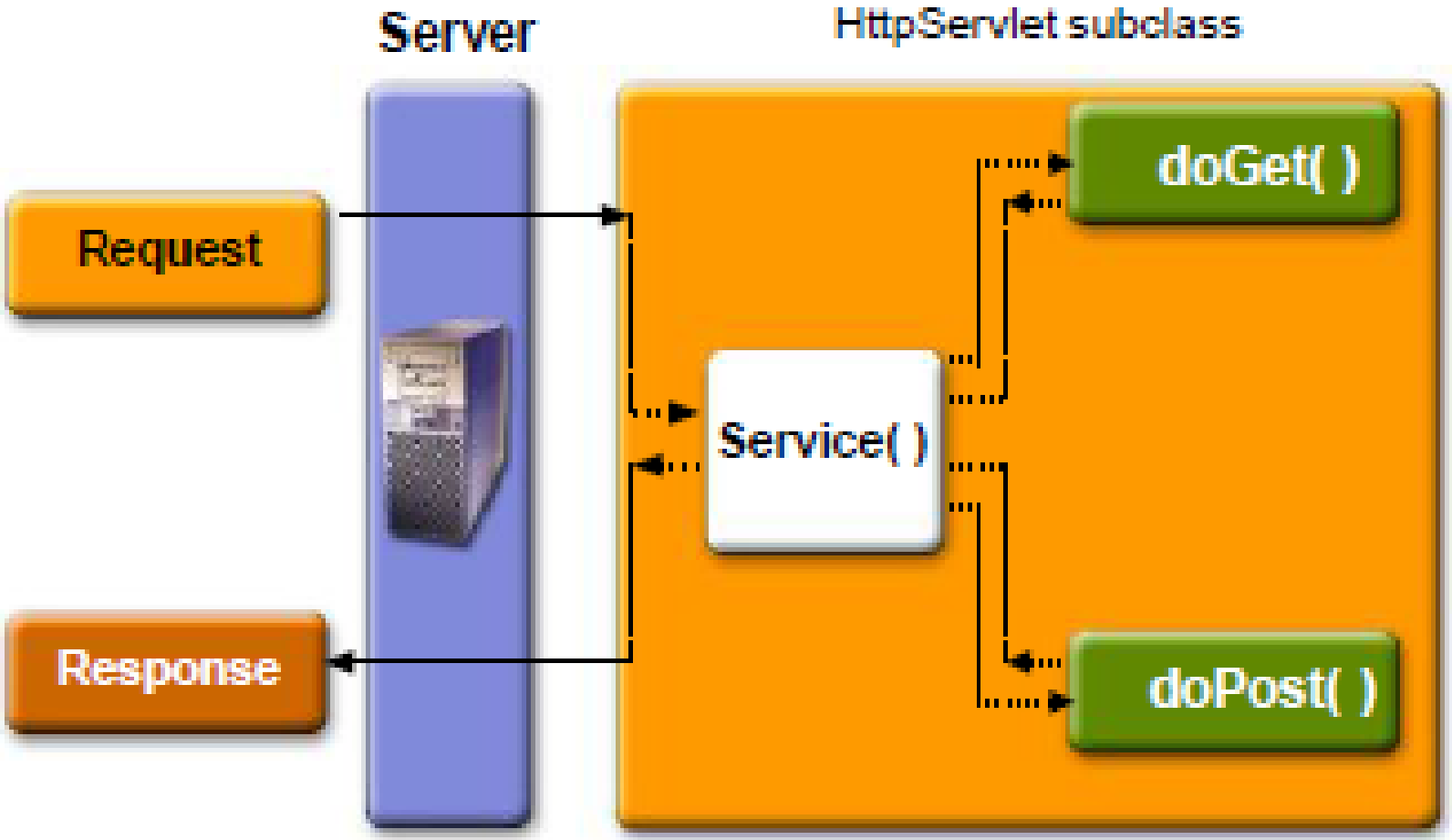
service()

- Phương thức service() nhận các requests và responses tổng quát:
  - `service(ServletRequest request, ServletResponse response)`
- doGet() và doPost() nhận các HTTP requests và responses:
  - `doGet(HttpServletRequest request, HttpServletResponse response)`
  - `doPost(HttpServletRequest request, HttpServletResponse response)`

service()


Server
Subclass of GenericServlet class
Request
Service( )
Response

doGet() và doPost()



doGet() và doPost()

- Trích xuất các thông tin gửi từ client (HTTP parameter) từ HTTP request
- Thiết lập/truy cập các thuộc tính của các Scope objects
- Thực hiện các xử lý nghiệp vụ (business logic) hoặc truy cập CSDL
- Tùy chọn forward request tới các Web components khác (Servlet hoặc JSP)
- Sinh HTTP response và trả về cho client

Ví dụ doGet()

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

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

    // Just send back a simple HTTP response
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<title>First Servlet</title>");
    out.println("<big>Hello J2EE Programmers! </big>");
  }
}
```

Mô hình Servlet Request & Response

- Nhiệm vụ của Servlet:
  - Nhận client request (Hầu hết ở dạng HTTP request)
  - Trích xuất 1 số thông tin từ request
  - Xử lý nghiệp vụ (truy vấn DB, gọi EJBs,…) hoặc sinh nội dung động
  - Tạo và gửi response cho client (hầu hết ở dạng HTTP response) hoặc forward request cho servlet khác

Servlet Request

- Request là gì?
  - Thông tin được gửi từ client tới 1 server
    - Ai tạo ra request
    - Dữ liệu gì được user nhập vào và gửi đi
    - HTTP headers
- Response là gì?
  - Thông tin được gửi đến client từ 1 server
    - Dữ liệu Text (html, thuần text) hoặc dữ liệu binary (image)
    - HTTP headers, cookies, ...

Servlet Request

- HTTP request bao gồm
  - header
  - Phương thức
    - Get: Thông tin nhập vào trong form được truyền như 1 phần của URL
    - Post: Thông tin nhập vào trong form được truyền trong nội dung thông điệp (message body)
    - Put
    - Header
  - Dữ liệu trong request (request data)

Servlet Request

- Các client requests thông dụng nhất
  - HTTP GET & HTTP POST
- GET requests:
  - Thông tin người dùng nhập vào đính kèm trong URL dưới dạng 1 query string
  - Chỉ gửi được lượng dữ liệu giới hạn
    - .../servlet/ViewCourse?FirstName=Sang&LastName=Shin
- POST requests:
  - Thông tin người dùng nhập vào được gửi dưới dạng dữ liệu (không đính kèm vào URL)
  - Gửi được lượng dữ liệu bất kỳ

Servlet Request

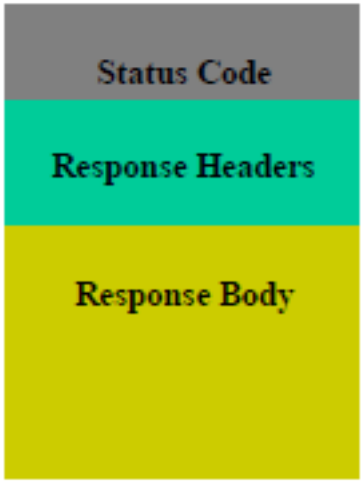
- Tất cả các servlet requests đều thực thi giao diện ServletRequest định nghĩa các phương thức truy cập tới:
  - Các tham số (parameters) gửi từ clients
  - Object-valued attributes
  - Client và server
  - Input stream
  - Thông tin về giao thức (Protocol information)
  - Content type
  - request có được tạo trên 1 kênh truyền secure không

Servlet Response

- Tất cả các các servlet responses thực thi giao diện ServletResponse
  - Lấy 1 output stream
  - Chỉ định content type
  - Có thiết lập buffer đầu ra không
  - Thiết lập localization information
- HttpServletResponse kế thừa giao diện ServletResponse
  - Mã trạng thái HTTP trả về (HTTP response status code)
  - Cookies

Servlet Response

- Cấu trúc Response



Servlet Response

- Tại sao cần HTTP response status code?
  - Giúp trình duyệt forward đến 1 trang khác
  - Chỉ ra được có resource bị thiếu
  - Hướng dẫn browser sử dụng bản sao được cache của dữ liệu

Servlet Response

- public void setStatus(int statusCode)
  - Mã trạng thái được định nghĩa trong HttpServletResponse
  - Các mã trạng thái chia làm 5 nhóm:
    - 100-199 Informational
    - 200-299 Successful
    - 300-399 Redirection
    - 400-499 Incomplete
    - 500-599 Server Error
  - Mã trạng thái mặc định là 200 (OK)

Response Header dùng làm gì?

- Forward đến địa chỉ mới nào
- Sửa cookies
- Cung cấp thông tin thời gian chỉnh sửa page.
- Hướng dẫn trình duyệt load lại trang sau 1 khoảng thời gian nhất định
- Đưa ra kích thước file được sử dụng trong HTTP connections loại persistent
- Chỉ định loại document sinh ra & trả về client
- ...

Phương thức thiết lập Response Header

- setContentType
  - Thiết lập Content-Type header. Servlets gần như luôn sử dụng phương thức này.
- setContentLength
  - Thiết lập Content-Length header. Được sử dụng cho HTTP connections loại persistent .
- addCookie
  - Thêm 1 giá trị trong Set-Cookie header.
- sendRedirect
  - Thiết lập Location header và thay đổi mã trạng thái

Response Body

- Một servlet gần như luôn trả về 1 response body
- Response body có thể là một PrintWriter hoặc một ServletOutputStream
- PrintWriter
  - Sử dụng phương thức response.getWriter()
  - Cho output loại ký tự (character-based)
- ServletOutputStream
  - Sử dụng phương thức response.getOutputStream()
  - Cho dữ liệu dạng binary (ví dụ: image)

Session Tracking

- Vì phương thức HTTP là stateless (ko lưu các thông tin lịch sử), điều này ảnh hưởng sâu sắc đến lập trình ứng dụng web. Có 4 kỹ thuật để lưu dấu session:
  - URL rewritting – Thêm các tham số vào cuối URL
  - Hidden fields – Sử dụng các trường ẩn
  - Cookies – Sử dụng cookie để trao đổi dữ liệu giữa client và server
  - Session *objects* – Sử dụng các đối tượng có phạm vi (scope) là session để truyền thông tin.

Session Tracking

- **URL Rewritting:**
- VD: Http://localhost:8080/Demo/test?x=abc&y=xyz
- Phần sau ?x=abc&y=xyz là phần được thêm vào để truyền 2 parameter x và y
- Để lấy giá trị này ta dùng lệnh **request.getParameter("x")**, tương tự với y.

Session Tracking

- **Hidden fields:**
- <INPUT TYPE=HIDDEN Name=hField VALUE="abc”>
- Để lấy giá trị này ta dùng lệnh **request.getParameter(“hField”)**.

Session Tracking

- **Cookies:** được lưu bởi server và gửi về client cùng response. Request được gửi tới server cùng với cookie nhưng ko thay đổi giá trị của cookie. Giá trị của cookie được lưu trong bộ nhớ (ổ cứng) của client.

Session Tracking

- VD1: lưu cookie (sử dụng response)

```
Cookie c1 = new Cookie("userName", "Helen");
Cookie c2 = new Cookie("password", "Keppler");
c2.setMaxAge(300);
response.addCookie(c1);
response.addCookie(c2);
```

- VD2: đọc cookie

```
Cookie[] cookies = request.getCookies();
for (int i = 0; i < cookies.length; i++) {
   Cookie cookie = cookies[i];
   out.println("Name->" + cookie.getName() + " Value->" + cookie.getValue());
}
```

XIN CẢM ƠN!