

SERVLET VÀ XỬ LÝ FORM ĐƠN GIẢN

□ Phần 1: Servlet và Xử lý form cơ bản

- ❖ Ánh xạ URL với servlet
- ❖ Các phương thức dịch vụ
- ❖ Xử lý tham số form đơn giản

□ Phần 2: Servlet và Xử lý form nâng cao

- ❖ Đọc các tham số nhiều giá trị
- ❖ Tìm hiểu vòng đời của servlet và cơ chế đa luồng

① SERVLET & XỬ LÝ FORM ĐƠN GIẢN

- ❑ Với công nghệ lập trình web của Sun Microsystems thì Servlet và JSP là 2 thành phần quan trọng. Trong đó
 - ❖ Servlet tiếp nhận yêu cầu, xử lý nghiệp vụ, tương tác CSDL...
 - ❖ JSP sinh giao diện (HTML, CSS, JS...) để phản hồi người sử dụng
- ❑ Ở góc độ kỹ thuật thì
 - ❖ Servlet là một lớp kế thừa từ HttpServlet được Sun cung cấp sẵn
 - ❖ Override phương thức dịch vụ (service(), doGet(), doPost()) để xử lý yêu cầu từ người dùng. Trong đó
 - doGet(): Xử lý GET request
 - doPost(): Xử lý POST request (submit form với method là POST)
 - service(): Xử lý cả 2 loại request (**khi override phương thức này thì doGet() và doPost() bị vô hiệu hóa**).

VÍ DỤ VỀ SERVLET

Không
phân biệt
POST hay GET

```
@WebServlet("/poly/servlet")
public class MyServlet extends HttpServlet{
    public void service(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException , IOException {
        String method = req.getMethod();
        System.out.println(method);
    }
}
```

Có phân biệt
POST và GET

```
@WebServlet("/poly/servlet")
public class MyServlet extends HttpServlet{
    public void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException , IOException {
        System.out.println("GET");
    }
    public void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException , IOException {
        System.out.println("POST");
    }
}
```

```
<form action="/hello/poly/servlet" method="POST">
    <button>OK</button>
</form>
```

- ❑ Annotation @WebServlet được sử dụng để ánh xạ một hoặc nhiều URL với một lớp servlet.
- ❑ Thuộc tính urlPatterns là mảng các URL được ánh xạ
 - ❖ @WebServlet(urlPatterns="/user/index") hoặc @WebServlet("/user/index")
 - ❖ @WebServlet({"/home/index", "/"})
- ❑ Cú pháp của URL
 - ❖ Phải bắt đầu bởi dấu / hoặc * ("*.php", "/user/*") và phải là **duy nhất** trong ứng dụng web
 - ❖ * được sử dụng để đại diện cho một nhóm ký tự bất kỳ (* *không cho phép xuất hiện ở giữa URL*)
- ❑ Khi một request có URL phù hợp với servlet thì phương thức dịch vụ sẽ thực thi

VÍ DỤ: TỔ CHỨC MỘT SERVLET

```
@WebServlet({"crud/index", "crud/edit/*", "crud/create"})
public class CrudServlet extends HttpServlet{
    public void service(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException , IOException {
        String uri = req.getRequestURI();
        if(uri.contains("index")) {
            this.index(req, resp);
        }
        else if(uri.contains("edit")) {
            this.edit(req, resp);
        }
        else if(uri.contains("create")) {
            this.create(req, resp);
        }
        req.getRequestDispatcher("/views/crud/index.jsp").forward(req, resp);
    }
}
```

http://.../crud/index
http://.../crud/edit/2020
http://.../crud/edit/TeoNV
http://.../crud/edit/...
http://.../crud/create



❑ Các phương thức dịch vụ của HttpServlet có cú pháp giống nhau chỉ khác tên

❖ **doGet**(HttpServletRequest, HttpServletResponse)

❖ **doPost**(HttpServletRequest, HttpServletResponse)

❖ **service**(HttpServletRequest, HttpServletResponse)

Trong đó

➤ **HttpServletRequest**: là đối tượng chứa dữ liệu gửi từ trình duyệt (tham số, cookie, header) và một số thông số hệ thống khác (ip, tên server...)

➤ **HttpServletResponse**: là đối tượng chứa dữ liệu phản hồi về client

❑ Căn cứ vào cặp (URL, Method) để xác định phương thức dịch vụ nào được thực hiện (chú ý: service() không quan tâm Method)

❑ doPost()

❖ `<form action=".../poly" method="post">`
 `<button>Submit</button>`

❖ `</form>`

❑ doGet()

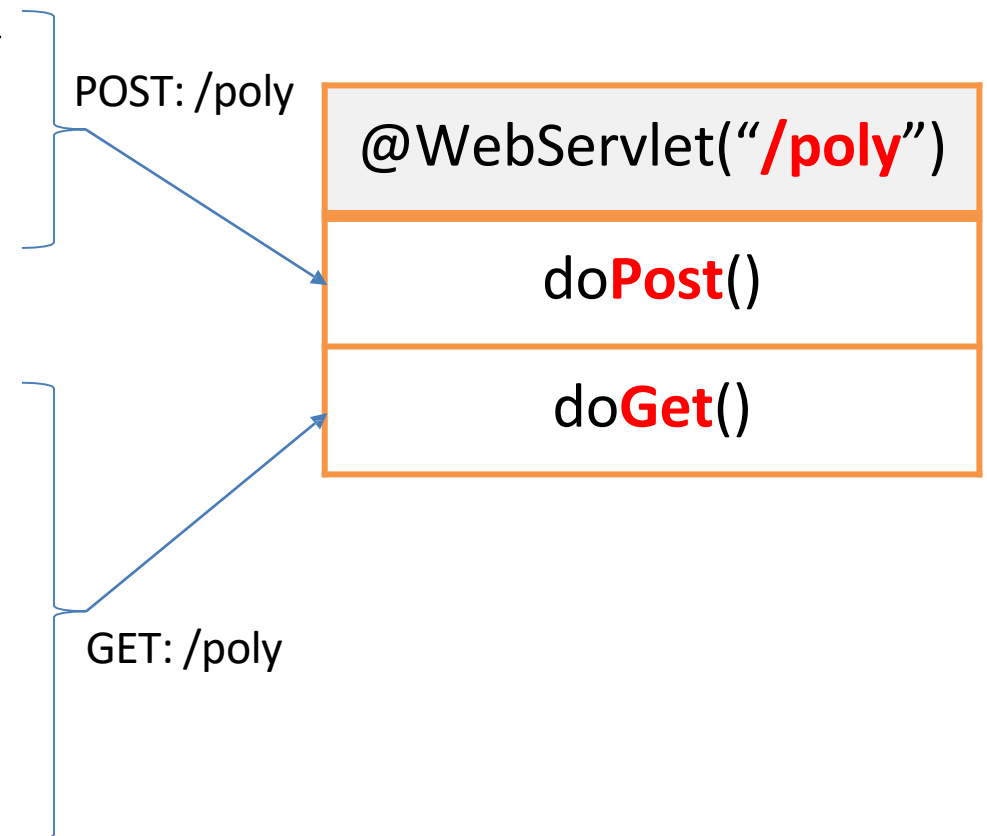
❖ Gõ vào ô địa chỉ của trình duyệt

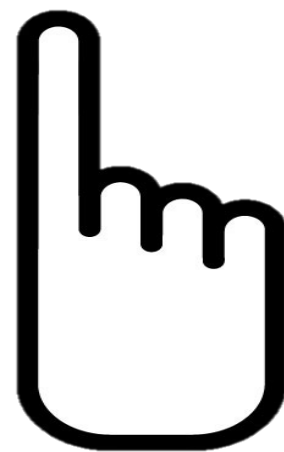
❖ `Liên kết`

❖ `<form action=".../poly" method="get">`
 `<button>Submit</button>`

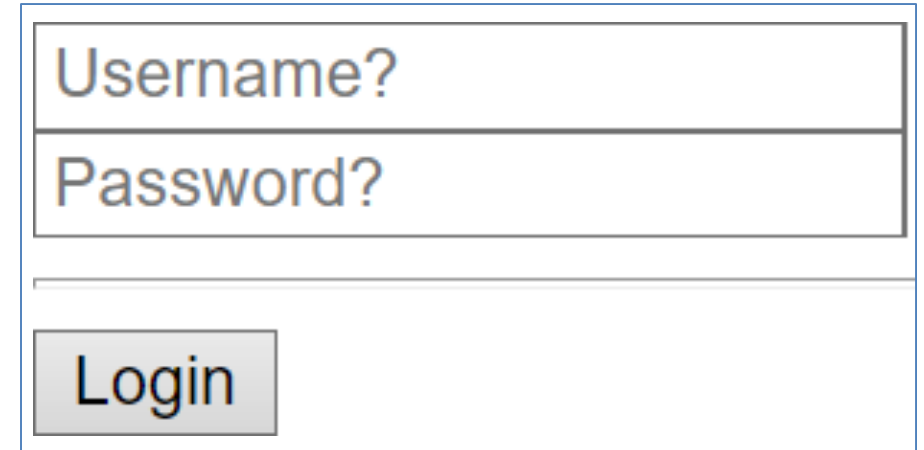
❖ `</form>`

❖ ...





- ❑ Form được sử dụng để lấy dữ liệu từ người sử dụng hoặc hiển thị dữ liệu cho người sử dụng.
- ❑ Thông thường để làm việc với form cần thực hiện 2 request
 - ❖ GET request (link) để hiển thị form (gọi doGet())
 - ❖ POST request (submit form) để xử lý dữ liệu nhập vào form (gọi doPost())

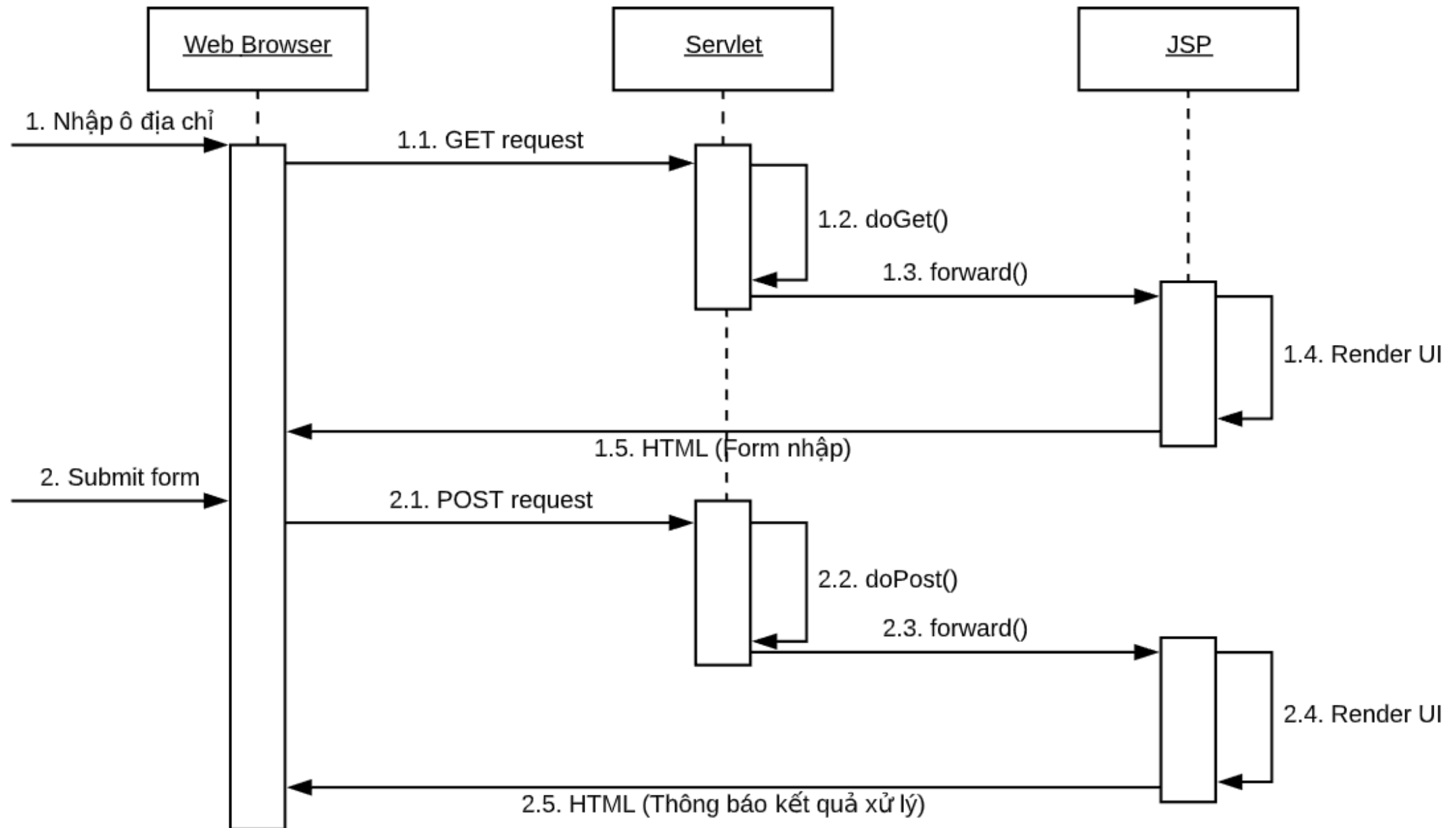


Username?

Password?

Login

SEQUENCE DIAGRAM



TỔ CHỨC SERVLET XỬ LÝ FORM

GET: http://.../login/form

```
@WebServlet({" /login/form", " /login/check"})
public class LoginServlet extends HttpServlet{
    //GET: /login/form
    public void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException , IOException {
        req.getRequestDispatcher("/views/login.jsp").forward(req, resp);
    }
    // POST: /login/check
    public void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException , IOException {
        // TODO: check existence of user here
        req.getRequestDispatcher("/views/login.jsp").forward(req, resp);
    }
}
```

POST: http://.../login/check

```
<form action=".../login/check" method="post">
    <button>Login</button>
</form>
```

- ❑ POST và GET là 2 phương pháp truyền dữ liệu từ trình duyệt đến server.
- ❑ Khi submit form với
 - ❖ Method="GET" thì dữ liệu form được thu thập và tạo thành chuỗi truy vấn (ghép vào sau dấu ? của url) để gửi đến server
 - ❖ Method="POST" thì trình duyệt sẽ tạo một kênh riêng để truyền dữ liệu đến server
- ❑ Ưu và nhược điểm của POST và GET
 - ❖ GET nhanh hơn POST nhưng có một số hạn chế như:
 - Dữ liệu nhỏ
 - Chỉ cho phép text (không upload được)
 - Phơi bày dữ liệu tế nhị (password và trường ẩn)
 - ❖ POST khắc phục được những hạn chế của GET

```
<form action="/my-servlet" method="POST">  
  <input name="fullname" placeholder="Fullname?"> <br>  
  <input name="age" placeholder="Age?"> <br>  
  <input name="salary" placeholder="Salary?">  
  <button>Submit</button>  
</form>
```

Fullname?
Age?
Salary?
Submit

```
String fullname = req.getParameter("fullname");  
String age = req.getParameter("age");  
String salary = req.getParameter("salary");
```


❑ Xử lý một số trường hợp đặc biệt với `getParameter(name)`

❖ `<select name="cbo_">`

➤ Lấy giá trị mục chọn

❖ `<input type="radio" name="rdo_">`

➤ Lấy giá trị radio được chọn

❖ `<input type="checkbox" name="chk_">`

➤ Lấy giá trị khi có check, null khi không check

❖ `<button type="submit" name="btn_">`

➤ Lấy giá trị nút bị click, null khi không click

United States ▾

☐ Male ☒ Female

☒ Single?

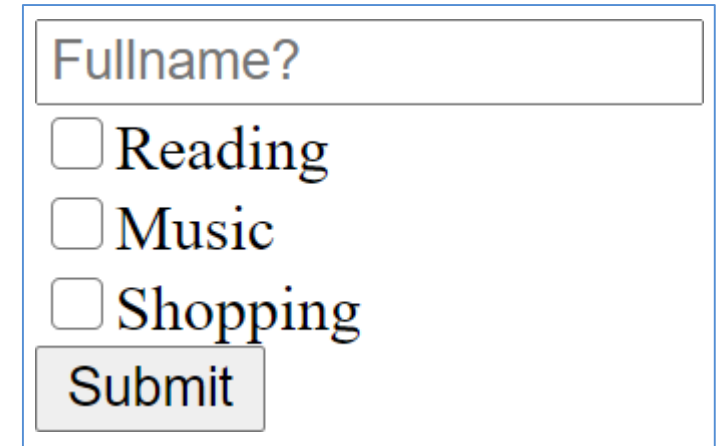
Create Update Delete

❑ Filter sau giúp xử lý form với tiếng Việt (sẽ được học sau)

```
@WebFilter("/*")
public class Utf8Filter implements Filter {
    @Override
    public void destroy() {}
    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {
        request.setCharacterEncoding("utf-8");
        response.setCharacterEncoding("utf-8");
        chain.doFilter(request, response);
    }
    @Override
    public void init(FilterConfig filterConfig) throws ServletException {}
}
```

② SERVLET VÀ XỬ LÝ FORM NÂNG CAO

- ❑ Hai thành phần của form sau đây khi submit thì trình duyệt sẽ gửi đi nhiều giá trị
 - ❖ ListBox (<select multiple>)
 - ❖ Nhiều control cùng tên (nhiều checkbox cùng tên)
- ❑ Nếu sử dụng `getParameter(name)` chỉ lấy được giá trị đầu tiên
- ❑ Để đọc tất cả giá trị hãy sử dụng `getParameterValues(name)`



Fullname?

☐ Reading


☐ Music

☐ Shopping

Submit

THAM SỐ NHIỀU GIÁ TRỊ

```
<form action="/my-servlet" method="POST">  
  <input name="fullname" placeholder="Fullname?"> <br>  
  <input name="hobbies" type="checkbox" value="RE">Reading <br>  
  <input name="hobbies" type="checkbox" value="MS">Music <br>  
  <input name="hobbies" type="checkbox" value="SP">Shopping <br>  
  <button>Submit</button>  
</form>
```



Fullname?

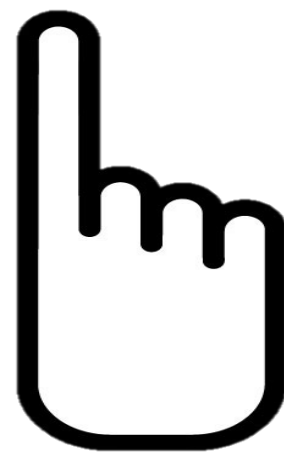
☐ Reading

☐ Music

☐ Shopping

Submit

```
String fullname = req.getParameter("fullname");  
String[] hobbies = req.getParameterValues("hobbies");
```



HTTPSERVLETREQUEST API

Phương thức	Mô tả
getParameter (String): String	Đọc giá trị của 1 tham số
getParameterValues(String): String[]	Đọc các giá trị của 1 tham số
getParameterNames(): Enumeration<String>	Lấy tên của tất cả tham số
getParameterMap(): Map<String, String[]>	Lấy Map chứa tất cả tham số
getContextPath (): String	Đường dẫn ứng dụng
getRequestURI () : String	Lấy đường dẫn URI hiện tại
getRequestURL() : String	Lấy đường dẫn URL hiện tại
getMethod () : String	Lấy tên phương thức web

```
System.out.println(req.getContextPath());  
System.out.println(req.getRequestURI());  
System.out.println(req.getRequestURL());  
System.out.println(req.getMethod());
```

```
Enumeration<String> names = req.getParameterNames();  
while(names.hasMoreElements()) {  
    String name = names.nextElement();  
    String value = req.getParameter(name);  
    System.out.printf("+ %s=%s\r\n", name, value);  
}
```


❑ PATH INFORMATION:

- ❖ `getRequestURL()` => `http://localhost:8080/fpoly/x/y/1/2/3`
- ❖ `getRequestURI()` => `/fpoly/x/y/1/2/3`
- ❖ `getContextPath()` => `/fpoly`
- ❖ `getPathInfo()` => `/1/2/3`

❑ PARAMETER INFORMATION:

- ❖ `getQueryString()` => `a=AA&b=BB`
- ❖ `getParameter("a")` => `AA`
- ❖ `getParameterNames()` => `[a, b]`
- ❖ `getParameterMap()` => `{a=[AA], b=[BB]}`

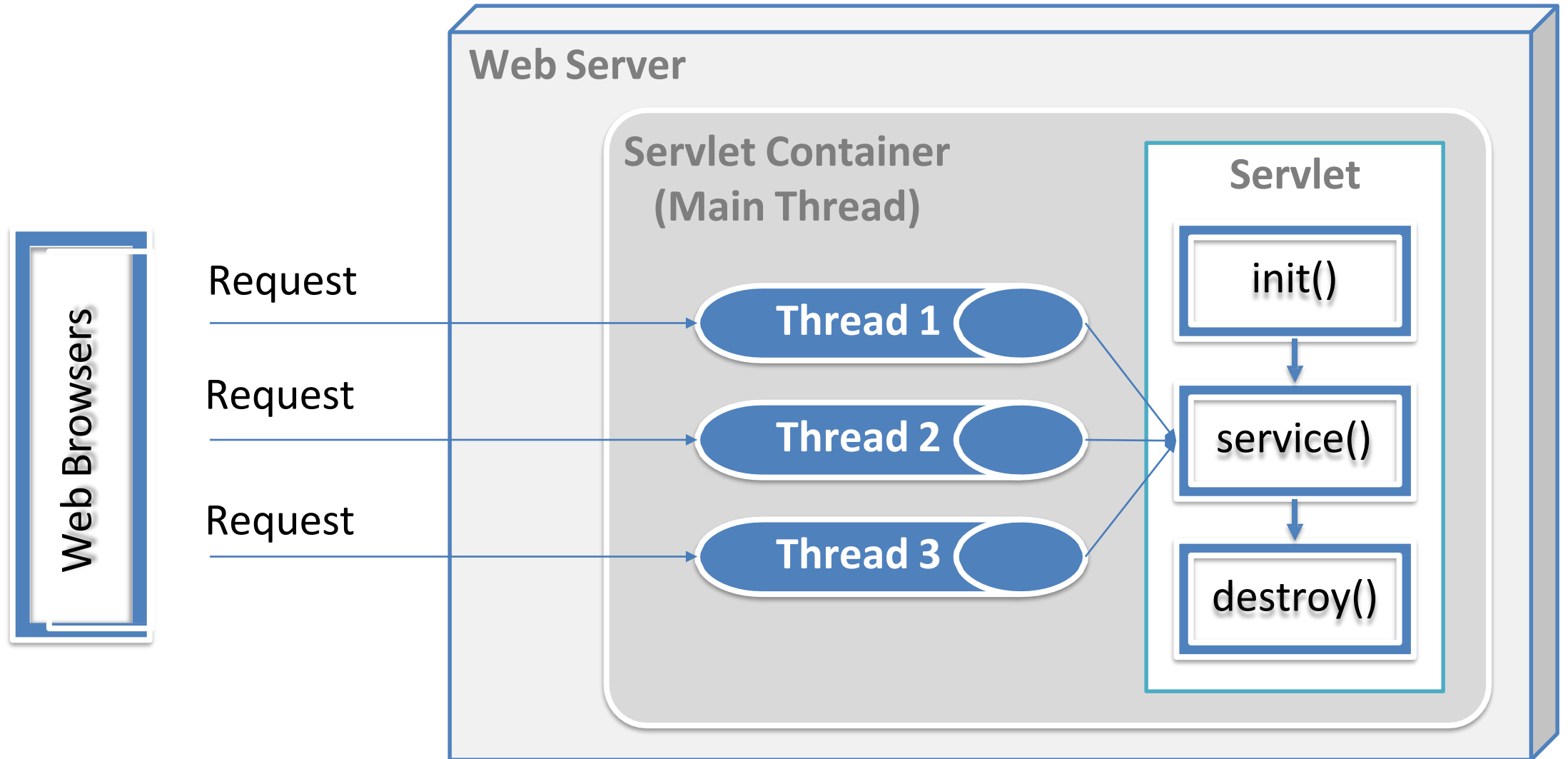
❑ HOST INFORMATION:

- ❖ `getScheme()` => `http`
- ❖ `getProtocol()` => `HTTP/1.1`
- ❖ `getServerName()` => `localhost`
- ❖ `getServerPort()` => `8080`

`http://localhost:8080/fpoly/x/y/1/2/3?a=AA&b=BB`

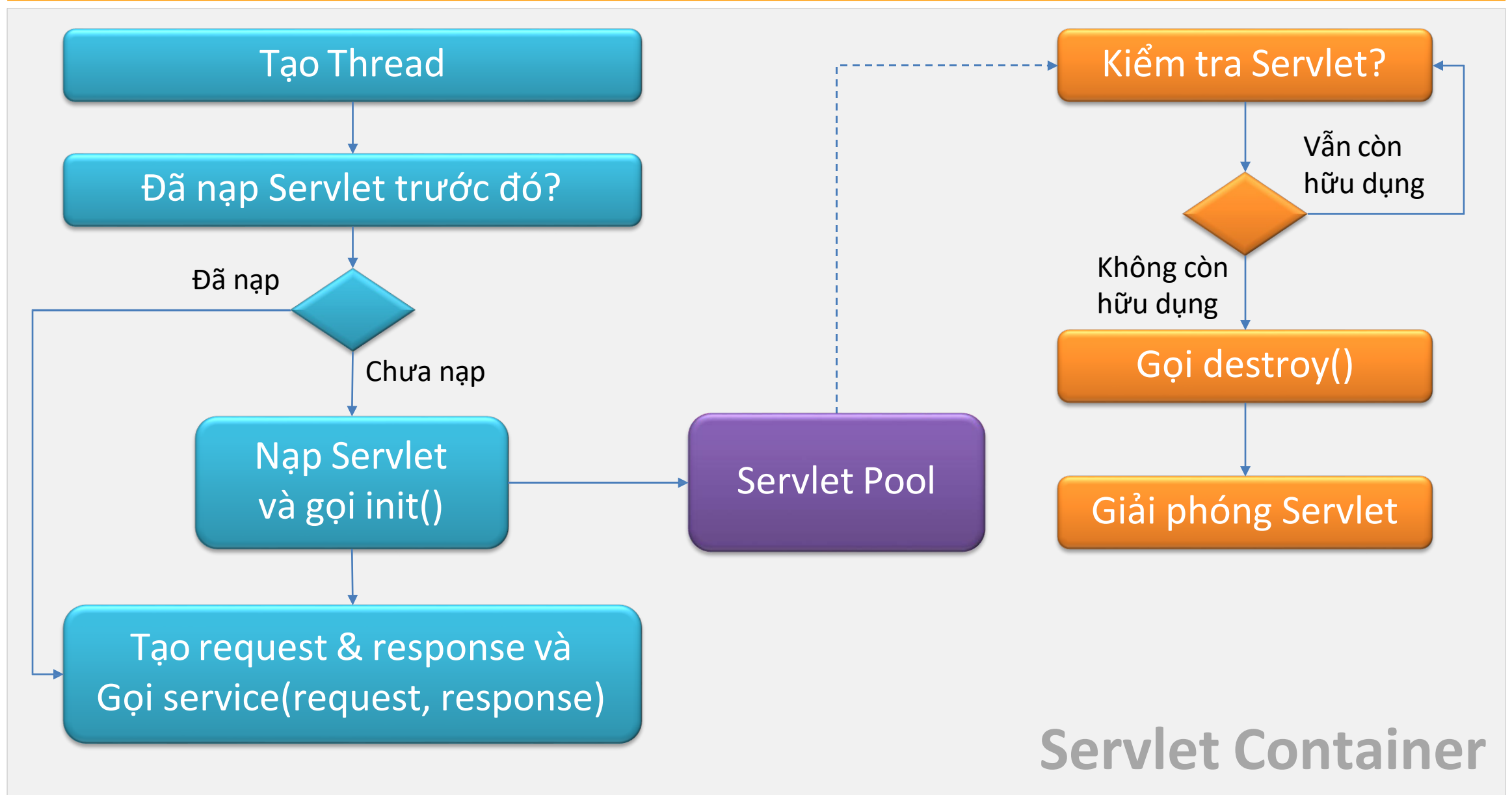
```
@WebServlet("/x/y/*")  
public class MyServlet extends HttpServlet{...}
```

ĐA LƯỜNG VÀ VÒNG ĐỜI SERVLET



- ❑ Công nghệ servlet áp dụng cơ chế xử lý đa tiểu trình (MultiThreading) để xử lý đồng thời các request.
- ❑ Khi có request đến, Servlet Container tạo một Thread mới, kiểm tra sự tồn tại của servlet trong bộ nhớ.
 - ❖ Nếu servlet đã tồn tại thì sẽ tạo các đối tượng request, response và gọi phương thức service(request, response)
 - ❖ Ngược lại, nạp servlet từ đĩa vào bộ nhớ, gọi phương thức init() trước khi gọi service()
- ❑ Sau khi phục vụ request xong, Servlet không bị giải phóng ngay mà lưu trữ lại trong bộ nhớ để sẵn sàng phục vụ cho request khác.
- ❑ Vì một lý do nào đó (thiếu tài nguyên, lâu quá không có request...) thì Servlet Container sẽ giải phòng Servlet. Phương thức destroy() sẽ được gọi trước khi giải phóng servlet.

VÒNG ĐỜI CỦA SERVLET



```
@WebServlet("/servlet/life/cycle")
public class LifeCycleServlet extends HttpServlet{
    @Override
    public void init() throws ServletException {
        System.out.println("init() in T" + Thread.currentThread().getId());
    }
    public void service(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException , IOException {
        System.out.println("service() in T" + Thread.currentThread().getId());
    }
    @Override
    public void destroy() {
        System.out.println("destroy() in T" + Thread.currentThread().getId());
    }
}
```

PHÂN TÍCH KẾT QUẢ THỰC HIỆN

❑ Gọi Servlet lần đầu tiên

❖ init() **T25**

❖ service() **T25**

❑ Gọi Servlet lần 2

❖ service() **T22**

❑ Gọi Servlet lần 3

❖ service() **T21**

❑ Khi Servlet bị giải phóng

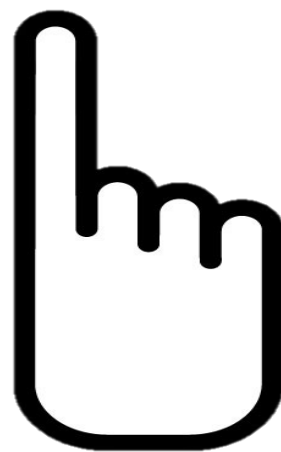
❖ destroy() **T16**

Vì Servlet chưa được nạp vào nên Servlet Container nạp vào và gọi init() trước khi gọi service(). ***Cả 2 phương thức init() và service() đều được thực hiện trong cùng 1 Thread***

Không gọi init() vì Servlet đang tồn tại trong Servlet Pool. ***Phương thức service() được thực hiện trong một Thread khác nhau***

Destroy() được thực hiện trong ***Thread chính của Servlet Container***

- ❑ Các đối tượng request và response được Servlet Container tạo ra với mục đích
 - ❖ Request chứa dữ liệu được gửi đến từ client (parameter, cookie, header) và một số thông số hệ thống cung cấp cho phương service
 - ❖ Response để chứa dữ liệu (HTML, Cookies, Headers) phản hồi về client.
- ❑ Vì một servlet chỉ tồn tại một instance duy nhất để phục vụ cho tất cả mọi request nên cần chú ý về việc xung đột dữ liệu của các field. Nếu không có mục đích chia sẻ thì sử dụng biến local.



- ✓ @WebServlet({u1, u2...})
- ✓ Phân biệt doPost(), doGet(), service()
- ✓ Các phương thức
 - ✓ Req.getMethod()
 - ✓ Req.getRequestURI()
 - ✓ Req.getParameter()
 - ✓ Req.getParameterValues()
 - ✓ Req.setAttribute()
 - ✓ Req.getRequestDispatcher().forward()
- ✓ Cơ chế xử lý đa luồng và vòng đời của servlet