Giới thiệu về Servlet

[**I.Tổng quan:**](#_heading=h.30j0zll) **3**

[1.Khái niệm:](#_heading=h.1fob9te) 3

[2.Kiến trúc Servlet:](#_heading=h.3znysh7) 4

[3.Nhiệm vụ của Servlet:](#_heading=h.2et92p0) 4

[**II.So sánh Servlet và JSP:**](#_heading=h.tyjcwt) **5**

[1.Sơ đồ quan hệ xử lý request/response giữa Servlet và JSP:](#_heading=h.3dy6vkm) 5

[2.Ví dụ về 1 trang JSP:](#_heading=h.1t3h5sf) 6

[3.Phân biệt:](#_heading=h.4d34og8) 6

[**III.Vòng đời của Servlet:**](#_heading=h.2s8eyo1) **7**

[1.Tổng quan:](#_heading=h.17dp8vu) 7

[2.Phương thức init():](#_heading=h.3rdcrjn) 8

[3.Phương thức service():](#_heading=h.26in1rg) 9

[4.Phương thức destroy():](#_heading=h.lnxbz9) 9

[**IV.Lớp HttpServlet:**](#_heading=h.35nkun2) **10**

[1.Tổng quan:](#_heading=h.1ksv4uv) 10

[2.Các phương thức của lớp HttpServlet:](#_heading=h.44sinio) 11

[3.Phân biệt Get và Post:](#_heading=h.2jxsxqh) 11

[**V.Các phương thức của lớp HttpServletRequest/Response:**](#_heading=h.z337ya) **12**

[1.Các phương thức của lớp HttpServletRequest:](#_heading=h.3j2qqm3) 12

[2.Các phương thức của lớp HttpServletResponse:](#_heading=h.1y810tw) 13

[**VI.Forward và Redirect:**](#_heading=h.4i7ojhp) **13**

[1.Forward:](#_heading=h.2xcytpi) 13

[2.Redirect:](#_heading=h.1ci93xb) 14

[3.Phân biệt Forward và Redirect:](#_heading=h.3whwml4) 15

[**VII.Session và Cookie:**](#_heading=h.2bn6wsx) **17**

[1.Session:](#_heading=h.qsh70q) 17

[a.Tổng quan:](#_heading=h.3as4poj) 17

[b.Hoạt động của session:](#_heading=h.1pxezwc) 17

[c.Ứng dụng và tiện ích:](#_heading=h.49x2ik5) 18

[2.Cookie:](#_heading=h.2p2csry) 19

[a.Tổng quan:](#_heading=h.147n2zr) 19

[b.Hoạt động của Cookie:](#_heading=h.3o7alnk) 20

[c.Ứng dụng:](#_heading=h.23ckvvd) 20

[d.Ưu điểm của Cookie:](#_heading=h.ihv636) 21

[e.Nhược điểm của Cookie:](#_heading=h.32hioqz) 21

[3.Phân biệt Session và Cookie:](#_heading=h.1hmsyys) 21

[**VIII.RequestDispatcher:**](#_heading=h.41mghml) **22**

[1.Định nghĩa:](#_heading=h.2grqrue) 22

[2.Tạo một đối tượng RequestDispatcher:](#_heading=h.vx1227) 23

[3.Sử dụng RequestDispatcher:](#_heading=h.3fwokq0) 23

[a.Các tập tin cần được tạo ra:](#_heading=h.1v1yuxt) 24

[Index.html:](#_heading=h.4f1mdlm) 25

[Validate.java:](#_heading=h.2u6wntf) 25

[Welcome.java:](#_heading=h.19c6y18) 25

[Web.xml:](#_heading=h.3tbugp1) 26

[b.Các màn thể hiện các bước:](#_heading=h.28h4qwu) 26

[**IX.Session tracking:**](#_heading=h.nmf14n) **28**

[1.Định nghĩa:](#_heading=h.37m2jsg) 28

[2.Các cách session tracking:](#_heading=h.1mrcu09) 29

[a.Sử dụng URL Rewriting:](#_heading=h.46r0co2) 29

[Tổng quan:](#_heading=h.2lwamvv) 29

[Ưu điểm:](#_heading=h.111kx3o) 29

[Nhược điểm:](#_heading=h.3l18frh) 30

[b.Sử dụng Hidden form fields:](#_heading=h.206ipza) 30

[Tổng quan:](#_heading=h.4k668n3) 30

[Ví dụ:](#_heading=h.2zbgiuw) 30

[Ưu điểm:](#_heading=h.1egqt2p) 32

[Nhược điểm:](#_heading=h.3ygebqi) 32

[c.Sử dụng HttpSession:](#_heading=h.2dlolyb) 32

[Tổng quan:](#_heading=h.sqyw64) 32

[Ví dụ:](#_heading=h.3cqmetx) 32

[**X.Web filter:**](#_heading=h.1rvwp1q) **36**

[1.Định nghĩa:](#_heading=h.4bvk7pj) 36

[2.Chức năng của Filter:](#_heading=h.2r0uhxc) 37

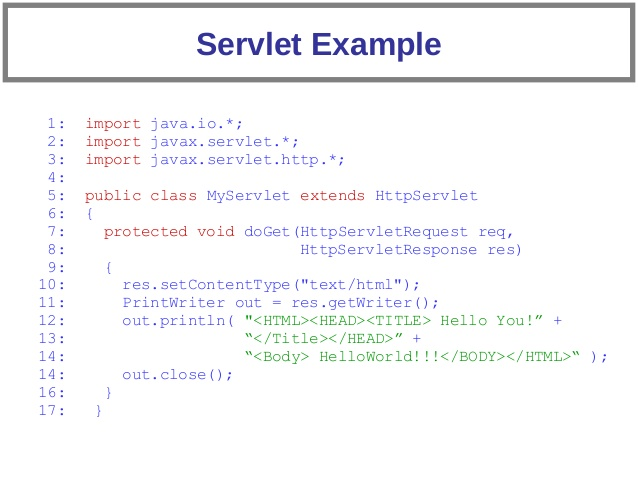
[3.Khai báo và triển khai một filter:](#_heading=h.1664s55) 38

[a.Khai báo:](#_heading=h.3q5sasy) 38

[b.Triển khai 1 filter sử dụng web.xml:](#_heading=h.25b2l0r) 40

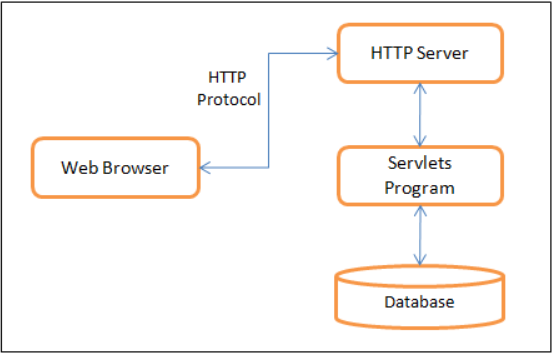
# I.Tổng quan:

## 1.Khái niệm:

* Chủ đề của Servlet khá lớn, và tùy thuộc vào ngữ cảnh sử dụng mà Servlet được định nghĩa theo những cách khác nhau như:
* Servlet là một công nghệ được sử dụng để tạo ra ứng dụng web.
* Servlet là một API cung cấp các interface và lớp bao gồm các tài liệu.
* Servlet là một thành phần web được triển khai trên máy chủ để tạo ra trang web động.
* Servlet có thể được xác định như một đối tượng, nhưng chúng ta không thể khởi tạo đối tượng Servlet trực tiếp như các đối tượng Java thông thường mà nó sẽ được thực hiện bởi Web container.
* Ảnh minh họa:
* 

## 2.Kiến trúc Servlet:

Sơ đồ dưới đây cho thấy vị trí của Servlets trong một Ứng dụng Web.

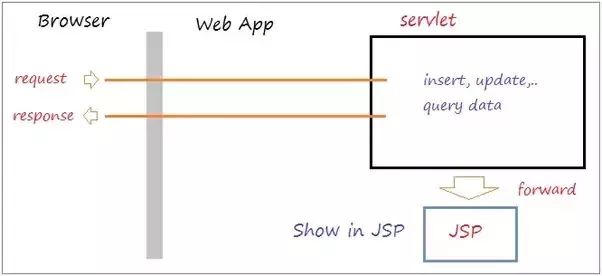


## 3.Nhiệm vụ của Servlet:

* Đọc dữ liệu rõ ràng do khách hàng (trình duyệt) gửi. Điều này bao gồm một mẫu HTML trên một trang Web hoặc nó cũng có thể đến từ một applet hoặc một chương trình khách hàng HTTP tùy chỉnh.
* Đọc dữ liệu yêu cầu HTTP ẩn được gửi bởi khách hàng (trình duyệt). Điều này bao gồm các cookie, loại phương tiện truyền thông và các chương trình nén mà trình duyệt hiểu được, v.v.
* Xử lý dữ liệu và tạo ra các kết quả. Quá trình này có thể yêu cầ u nói chuyện với một cơ sở dữ liệu, thực hiện một cuộc gọi RMI hoặc CORBA, gọi một dịch vụ Web, hoặc tính trực tiếp phản hồi.
* Gửi dữ liệu rõ ràng (tức là tài liệu) tới khách hàng (trình duyệt). Tài liệu này có thể được gửi bằng nhiều định dạng, bao gồm văn bản (HTML hoặc XML), nhị phân (hình ảnh GIF), Excel, v.v ...
* Gửi phản hồi HTTP ẩn cho khách hàng (trình duyệt). Điều này bao gồm nói với trình duyệt hoặc các trình khách khác loại tài liệu đang được trả về (ví dụ, HTML), thiết lập cookie và các tham số bộ nhớ đệm, và các tác vụ khác.

# II.So sánh Servlet và JSP:

## 1.Sơ đồ quan hệ xử lý request/response giữa Servlet và JSP:



## 2.Ví dụ về 1 trang JSP:



## 3.Phân biệt:

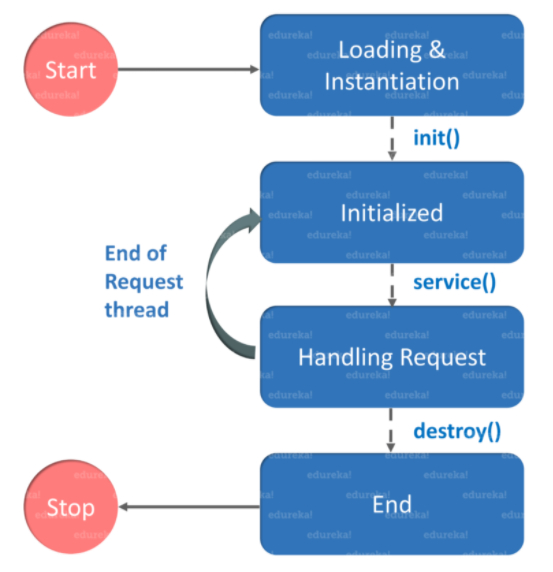
| **JSP** | **Servlet** |
| --- | --- |
| JSP thường được định hướng nhiều hơn để hiển thị thông tin. | Servlet được định hướng nhiều hơn để xử lý thông tin. |
| JSP hoạt động như một trình xem. | Servlet hoạt động như một controller. |
| JSP là một mã dựa trên HTML. | Servlet là một mã java. |
| JSP chậm hơn Servlets, vì bước đầu tiên trong vòng đời của JSP là chuyển đổi mã JSP sang Java và sau đó là biên dịch mã. | Servlet nhanh hơn so với JSP, vì chúng có thời gian phản hồi ngắn. |
| JSP là một ngôn ngữ kịch bản trang web, thường được sử dụng để tạo nội dung web động. | Servlets là các chương trình Java đã được biên dịch và cũng tạo ra nội dung web động. |

# III.Vòng đời của Servlet:

## 1.Tổng quan:

* Toàn bộ vòng đời của một servlet được quản lý bởi Servlet container sử dụng giao diện javax.servlet.Servlet để hiểu đối tượng Servlet và quản lý nó.
* Vòng đời của Servlet có thể được định nghĩa như là một tiến trình đầy đủ từ khi được tạo ra đến khi bị hủy theo các giai đoạn sau:
* Servlet được khởi tạo bởi gọi phương thức init().
* Servlet gọi phương thức service() để xử lý một yêu cầu từ Client.
* Servlet bị hủy bởi phương thức destroy().
* Cuối cùng, servlet trở thành rác và được thu thập bởi Garbage Collector của JVM.

***Hình minh họa vòng đời của 1 Servlet***



## 2.Phương thức init():



* Phương thức được thiết kế để chỉ được gọi một lần. Nó được gọi khi Servlet lần đầu được tạo, và không được gọi lại cho mỗi yêu cầu của người dùng.
* Thường thì, Servlet được tạo khi một người dùng lần đầu gọi một URL tương ứng với Servlet đó, nhưng chúng ta cũng có thể xác định rằng Servlet này được tải khi Server được khởi động lần đầu.
* Khi một người sử dụng gọi một Servlet, một đối tượng đơn của Servlet được tạo, với mỗi yêu cầu từ người dùng, là kết quả trong một thread mới, mà được thao tác tới doGet hoặc doPost một cách thích hợp. Phương thức init() tạo hoặc tải một vài dữ liệu mà sẽ được sử dụng qua vòng đời của Servlet đó.

## 3.Phương thức service():



* Phương thức service() là phương thức chính để thực hiện tác vụ thực sự. Nơi chứa servlet (ví dụ như Web Server) gọi phương thức service() để xử lý các yêu cầu từ Client (hoặc trình duyệt) và viết phản hồi đã được định dạng trở lại Client đó.
* Mỗi khi Server nhận một yêu cầu cho một Servlet, thì Server tạo một Thread mới và triệu hồi service(). Phương thức service() kiểm tra kiểu yêu cầu HTTP (Kiểu GET, POST, PUT, DELETE, .v.v.) và gọi các phương thức doGet, doPost, doPut, doDelete .v.v. tương ứng một cách thích hợp.
* Phương thức service() được gọi bởi Container và nó triệu hồi các phương thức doGet, doPost, doPut, doDelete, .v.v. tương ứng. Vì thế, bạn không phải làm gì cả với service() nhưng việc bạn ghi đè phương thức hoặc doGet() hoặc doPost tùy thuộc vào kiểu yêu cầu mà bạn nhận từ Client.

## 4.Phương thức destroy():



* Phương thức destroy() chỉ được gọi một lần ở giai đoạn cuối trong vòng đời Servlet. Phương thức này giúp servlet của bạn một cơ hội để đóng các kết nối tới Database, dừng thread, viết các danh sách cookie hoặc viết tính toán trên đĩa, và thực hiện các hoạt động cleanup khác.
* Sau khi phương thức destroy() được gọi, đối tượng servlet này được đánh dấu cho Garbage Collector.

# IV.Lớp HttpServlet:

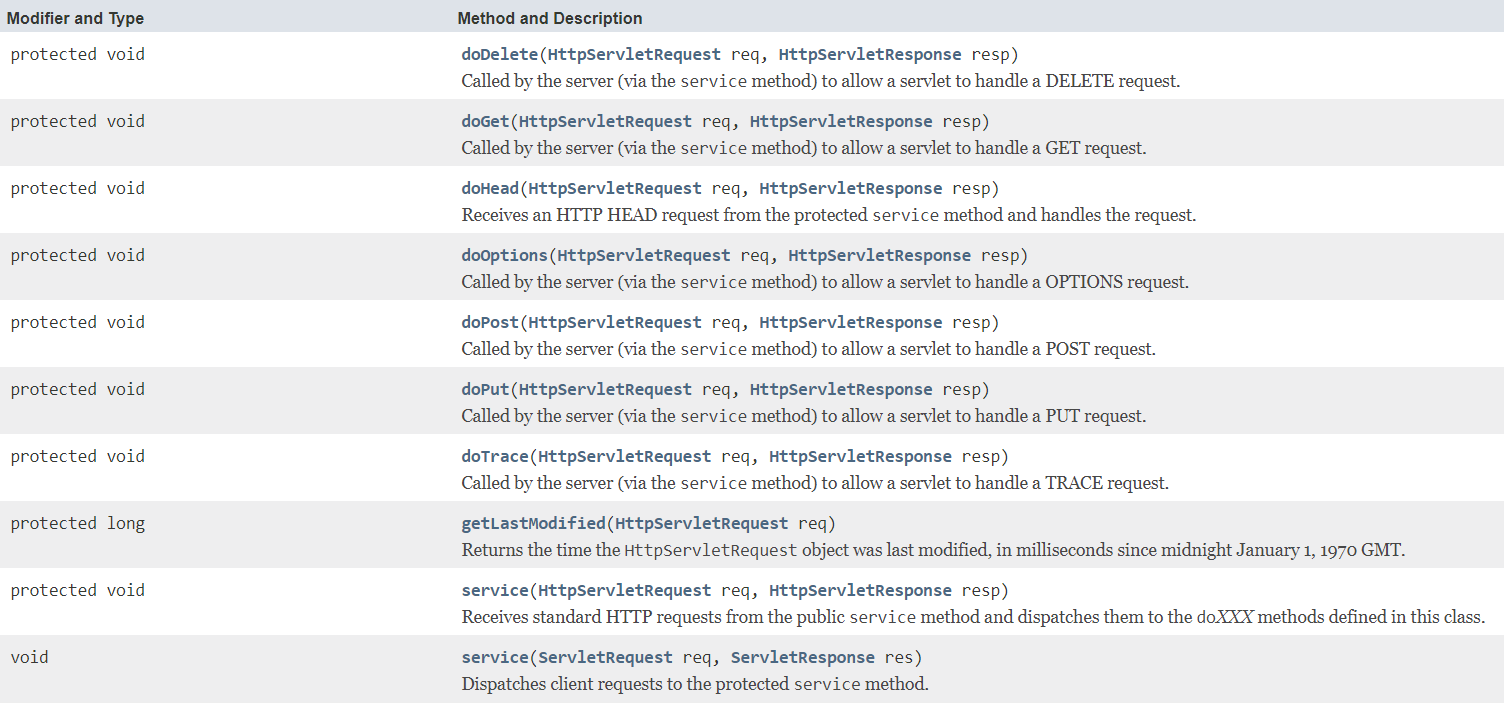
## 1.Tổng quan:

-Kế thừa lớp GenericServlet và sử dụng cho giao thức Http.

-Cung cấp một lớp trừu tượng được phân lớp để tạo ra một HTTP servlet phù hợp cho một trang Web. Một lớp con của HttpServlet phải ghi đè ít nhất một phương thức, thường là một trong những phương thức sau:

* doGet, nếu servlet hỗ trợ các yêu cầu HTTP GET.
* doPost, cho các yêu cầu HTTP POST.
* doPut, cho các yêu cầu HTTP PUT.
* doDelete, cho các yêu cầu HTTP DELETE.
* init và destroy, để quản lý các tài nguyên được lưu giữ cho vòng đời của servlet.
* getServletInfo, mà servlet sử dụng để cung cấp thông tin về chính nó.

## 2.Các phương thức của lớp HttpServlet:

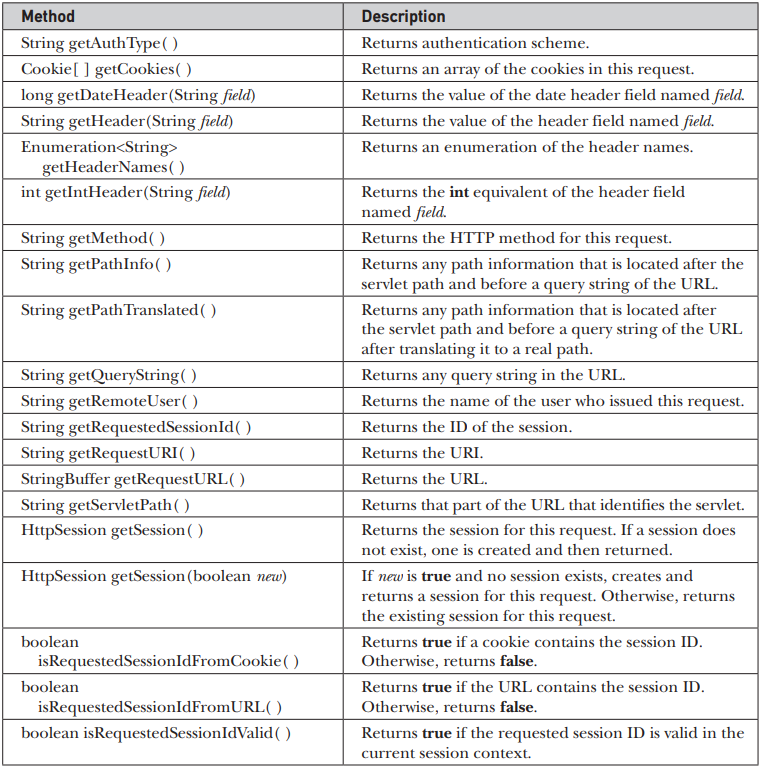


## 3.Phân biệt Get và Post:

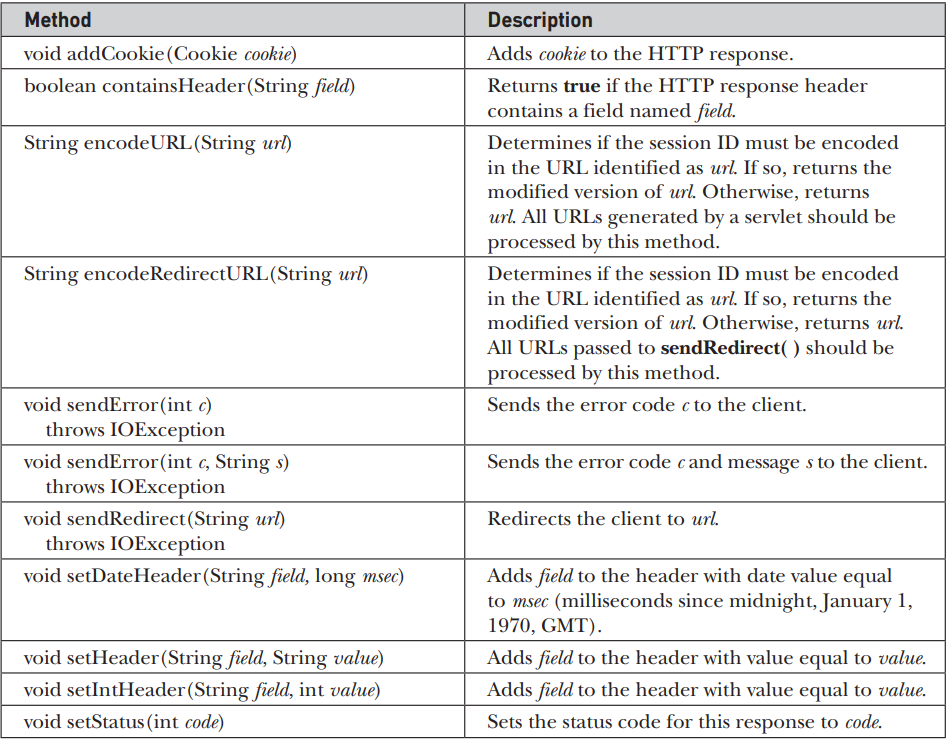
| **Get** | **Post** |
| --- | --- |
| Get gửi tham số yêu cầu dưới dạng chuỗi truy vấn được thêm vào cuối yêu cầu. | Post gửi các tham số yêu cầu như một phần của body của yêu cầu http. |
| Get có kích thước tối đa là 2000 ký tự. | Post có kích thước tối đa là 8 mb. |
| Các thông số không được mã hóa. | Các thông số được mã hóa. |
| Get thường được sử dụng để truy vấn hoặc lấy một số thông tin từ máy chủ. | Post thường được sử dụng để cập nhật hoặc đăng một số thông tin lên máy chủ. |
| Get xử lý thông tin nhanh hơn | Post chậm hơn |
| Vẫn còn trong lịch sử trình duyệt. | Không còn trong lịch sử trình duyệt. |

# V.Các phương thức của lớp HttpServletRequest/Response:

## 1.Các phương thức của lớp HttpServletRequest:

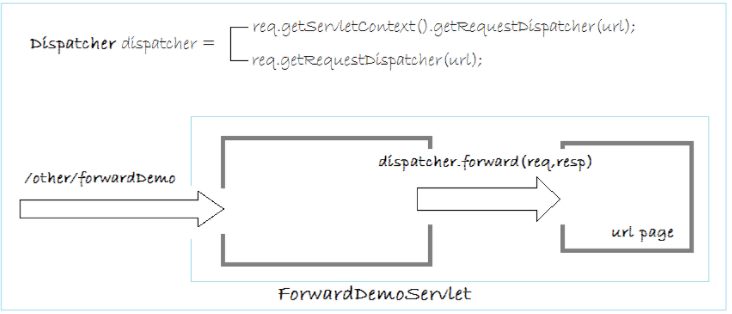


## 2.Các phương thức của lớp HttpServletResponse:

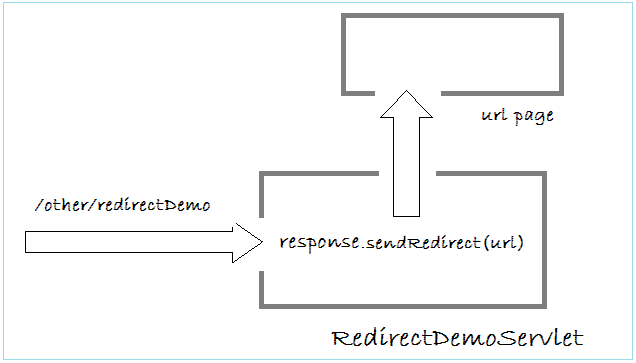


# VI.Forward và Redirect:

## 1.Forward:

* Khi một yêu cầu của trình duyệt được gửi đến một Servlet, nó có thể chuyển tiếp yêu cầu đó đến một trang khác (hoặc một servlet khác). Địa chỉ trên trình duyệt của người dùng vẫn là URL của trang đầu tiên, nhưng nội dung của trang được tạo bởi trang được chuyển tiếp.
* Hoạt ảnh minh họa:
* 
* Trong một ứng dụng java web ,khi chúng ta thực hiện một hành động nào đó mà muốn gửi dữ liệu đi,hoặc muốn chuyển sang một trang mới xử lý dữ liệu thì chúng ta nên dùng forward.

## 2.Redirect:

* Khi một yêu cầu từ người dùng được gửi đến một Servlet (Trang A), servlet này có thể chuyển hướng yêu cầu đến một trang khác (Trang B) và hoàn thành nhiệm vụ của nó. Trang được chuyển hướng có thể là một trang trong ứng dụng của bạn hoặc có thể là bất kỳ trang nào. Địa chỉ trên trình duyệt của người dùng lúc này sẽ hiển thị đường dẫn của trang B.
* Hoạt ảnh minh họa:
* 
* Trong một ứng dụng java web ,khi chúng ta thực hiện một hành động nào đó mà không muốn gửi dữ liệu đi,hoặc muốn chuyển sang một trang mới mà không có bất kỳ xử lý dữ liệu gì thì chúng ta nên dùng Redirect.

## 3.Phân biệt Forward và Redirect:

| **Tham số so sánh** | **Forward** | **Redirect** |
| --- | --- | --- |
| Ý tưởng | được sử dụng khi có nhu cầu chuyển tiếp yêu cầu từ JSP hoặc servlet này sang JSP hoặc servlet khác. | được sử dụng khi có nhu cầu chuyển hướng yêu cầu của client đến một URL khác nằm trên một máy chủ khác. |
| Sự tham gia của client | Web container xử lý yêu cầu trong nội bộ và do đó client không tham gia trực tiếp vào quá trình. | cửa sổ hiện tại được chuyển hướng đến một cửa sổ khác và client nhận được thông tin thích hợp về những gì đang diễn ra, do đó client có quyền tham gia trực tiếp vào quá trình này. |
| Gửi dữ liệu | Sử dụng forward, chúng ta có thể gửi dữ liệu đến một trang được chuyển tiếp bằng request.setAttribute () | Nếu chúng ta muốn gửi dữ liệu, chúng ta phải lưu trữ dữ liệu trong phiên hoặc cùng với URL. |
| Tốc độ xử lý | Lệnh forward hoạt động trong một máy chủ duy nhất, do đó, nó hoạt động với tốc độ nhanh hơn. | Lệnh redirect hoạt động chậm hơn so với lệnh forward vì nó hoạt động trong nhiều máy chủ. |
| Ngữ cảnh sử dụng tốt nhất | hiệu quả nhất khi component thực hiện một logic nghiệp vụ và kết quả được chia sẻ với một component khác. | hoạt động hiệu quả nhất khi client được chuyển hướng từ trang này sang trang khác. |

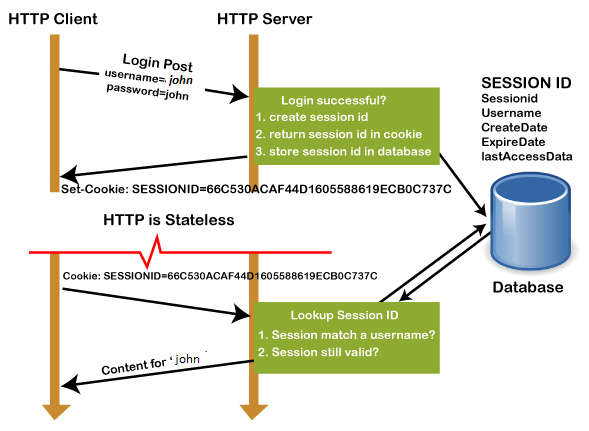
# VII.Session và Cookie:

## 1.Session:

### a.Tổng quan:

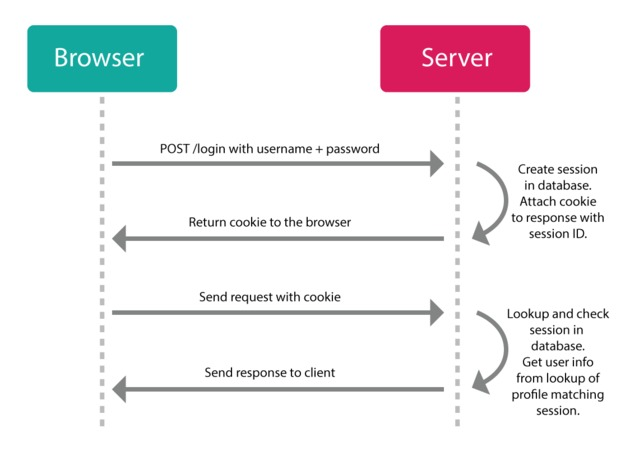
* Session là một khái niệm phổ biến được sử dụng trong lập trình trang web kết nối với cơ sở dữ liệu. Đặc biệt các chức năng như đăng nhập và đăng xuất người dùng sẽ khó thực hiện nếu không sử dụng session.
* Session đơn giản là một cách để chúng ta lưu lại dữ liệu của người dùng đang sử dụng website. Giá trị session được lưu trữ trong một tệp trên máy chủ.
* Thông tin người dùng được lưu trữ trong các biến session và các biến này có thể lưu trữ bất kỳ kiểu giá trị hoặc kiểu dữ liệu nào của đối tượng.
* Mỗi session là duy nhất cho mỗi người dùng và có thể sử dụng bất kỳ số session nào trong một ứng dụng; không có giới hạn cho nó.
* Người dùng được xác định với sự trợ giúp của sessionID, là một số duy nhất được lưu bên trong máy chủ. Nó được lưu dưới dạng cookie, trường biểu mẫu hoặc URL.

### b.Hoạt động của session:



* Trong bước đầu tiên, client yêu cầu máy chủ thông qua phương thức GET hoặc POST.
* SessionID được tạo trên máy chủ và nó lưu sessionID vào cơ sở dữ liệu. Nó trả về sessionId với một cookie như một phản hồi cho client.
* Cookie với sessionID được lưu trữ trên trình duyệt được gửi trở lại máy chủ. Máy chủ đối sánh ID này với sessionID đã lưu và gửi phản hồi HTTP Status 200 (OK).

### c.Ứng dụng và tiện ích:

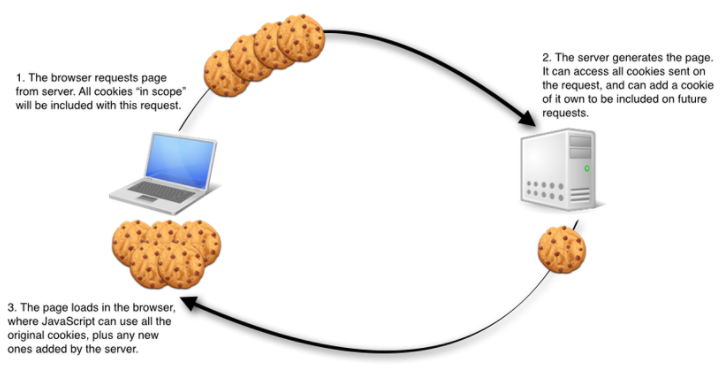
* Các session được sử dụng để lưu trữ thông tin như UserID qua máy chủ một cách an toàn hơn, nơi mà nó không thể được xử lý.
* Một trong các ứng dụng điển hình là việc quản lý Đăng nhập, Đăng xuất của thành viên mà hầu hết các trang Web nào cũng phải có.
* Với những tác vụ cần xác nhận là thành viên mới sử dụng được, chúng ta cần yêu cầu thành viên đăng nhập vào hệ thống. Nhưng nếu chỉ dùng biến thông thường thì mỗi lần cần thực hiện lại phải đăng nhập vào. Trong khi ấy, nếu dùng session thì sau khi đăng nhập, 1 biến session được tạo ra (ví dụ là user\_id), thì biến này sẽ tồn tại từ trang này sang trang khác, như thế khi cần thực hiện tác vụ khác cũng cần đăng nhập, ta chỉ cần kiểm tra xem có tồn tại biến user\_id này hay chưa là đủ. Nếu tồn tại rồi thì thôi, chưa tồn tại thì đăng nhập.
* Nó cũng có thể chuyển thông tin dưới dạng giá trị từ trang web này sang trang web khác.
* 
* Nó có thể được sử dụng thay thế cho cookie cho các trình duyệt không hỗ trợ cookie để lưu trữ các biến theo cách an toàn hơn.

## 2.Cookie:

### a.Tổng quan:

* Cookie là một tệp văn bản nhỏ được lưu trữ trên máy tính của người dùng.
* Cookie giúp các trang web theo dõi lịch sử duyệt web của người dùng hoặc thông tin giỏ hàng khi họ truy cập trang web của họ.
* Nó chỉ lưu trữ kiểu dữ liệu "Chuỗi".
* Thông tin được lưu trữ trong cookie không an toàn vì thông tin này được lưu trữ ở định dạng văn bản ở phía máy khách, bất kỳ ai cũng có thể đọc được.

### b.Hoạt động của Cookie:



* Khác với dữ liệu gửi từ form (POST hay GET) thì cookies sẽ được trình duyệt tự động gửi đi theo mỗi lần truy cập lên máy chủ.
* Khi người dùng truy cập trang web sử dụng cookie, máy chủ web của trang web đó sẽ tự động gửi cookie đến máy tính của người dùng.
* Khi truy cập các trang web sử dụng cookie đã lưu, các cookie này sẽ tự động gửi thông tin của người dùng đến chủ sở hữu của nó (người tạo ra cookie).
* Mỗi cookie thường có một khoảng thời gian chờ nhất định do lập trình viên xác định trước. Các thông tin được lưu trữ trong cookie như thông tin đăng nhập, thao tác của người dùng, tần suất truy cập trang web, thời gian truy cập ... Tất cả đều là thông tin tạm thời và được lưu trữ trong một khoảng thời gian.

### c.Ứng dụng:

* HTTP là một giao thức không trạng thái; do đó nó không lưu trữ bất kỳ thông tin người dùng nào.
* Với mục đích này, chúng ta có thể sử dụng Cookie. Nó cho phép chúng ta lưu trữ thông tin trên máy tính của người dùng và theo dõi trạng thái của các ứng dụng.
* Cá nhân hóa trải nghiệm người dùng - điều này đạt được bằng cách cho phép người dùng chọn tùy chọn của họ.
* Trang được yêu cầu theo dõi được cá nhân hóa dựa trên các tùy chọn đã đặt trong cookie. Theo dõi các trang mà người dùng đã truy cập.
* Người ta thường dùng cookies để lưu trữ các thông tin có liên quan đến nhiều phiên làm việc khác nhau (qua nhiều lần đóng và mở session).

### d.Ưu điểm của Cookie:

* Giúp việc truy cập Website của người dùng nhanh hơn, tiện lợi hơn, không quá mất nhiều thời gian đăng nhập lại nhiều lần.
* Đối với các doanh nghiệp, việc sử dụng Cookie sẽ giúp họ theo dõi được hành vi người dùng, từ đó biết được họ thường truy cập ít hay nhiều, thời gian là bao lâu hay các sở thích khác để có thể tối ưu hóa Website, dịch vụ của mình.
* Ngoài ra, việc lưu trữ Cookie đối với các doanh nghiệp sẽ giúp khách hàng của họ thuận tiện hơn trong việc truy cập hay đơn giản là việc nhập liệu ở Website đó trở nên tiện lợi khi các thông tin đã được lưu trữ.

### e.Nhược điểm của Cookie:

* Vì Cookie là một file dùng để lưu trữ các thông tin, hoạt động sử dụng của người dùng mang tính cá nhân vì vậy sẽ dễ dàng bị các Hacker dòm ngó, tìm cách đột nhập hệ thống Website, máy tính cá nhân để lấy cắp thông tin và sử dụng cho các mục đích xấu mà bạn không thể lường trước được.

## 3.Phân biệt Session và Cookie:

| Cookie | Session |
| --- | --- |
| Cookie được lưu trữ trên trình duyệt của người dùng. | Session không được lưu trữ trên trình duyệt của người dùng. |
| Dữ liệu cookie được lưu trữ ở phía máy khách. | Dữ liệu session được lưu trữ ở phía máy chủ. |
| Dữ liệu cookie dễ dàng sửa đổi hoặc lấy cắp khi chúng được lưu trữ ở phía máy khách. | Dữ liệu session không dễ dàng sửa đổi vì chúng được lưu trữ ở phía máy chủ. |
| Dữ liệu cookie có sẵn trong trình duyệt cho đến khi hết hạn. | Sau khi đóng trình duyệt, session sẽ kết thúc phiên làm việc. |
| Cookie được lưu trữ trên máy tính người dùng dưới dạng tệp văn bản. | Một session lưu trữ các biến và giá trị của nó trong một tệp trong thư mục tạm thời của máy chủ. |
| Kích thước tối đa của cookie của các trình duyệt thường là 4KB. | Chúng ta có thể lưu trữ nhiều dữ liệu mà chúng ta muốn trong 1 phiên, nhưng có sự giới hạn về dung lượng tối đa mà 1 tập lệnh có thể sử dụng cùng 1 lúc thường là 128 MB. |

# VIII.RequestDispatcher:

## 1.Định nghĩa:

* Định nghĩa một đối tượng nhận yêu cầu từ máy khách và gửi chúng đến bất kỳ tài nguyên nào (chẳng hạn như tệp servlet, tệp HTML hoặc tệp JSP) trên máy chủ. Vùng chứa servlet tạo đối tượng RequestDispatcher, được sử dụng như một trình bao bọc xung quanh tài nguyên máy chủ nằm tại một đường dẫn cụ thể hoặc được cung cấp bởi một tên cụ thể.
* Giao diện này được thiết kế để bọc các servlet, nhưng một thùng chứa servlet có thể tạo các đối tượng RequestDispatcher để bọc bất kỳ loại tài nguyên nào.

## 2.Tạo một đối tượng RequestDispatcher:

getRequestDispatcher()Phương thức của **ServletRequest** trả về đối tượng của **RequestDispatcher** .



Hoặc:



## 3.Sử dụng RequestDispatcher:

Trong ví dụ này, tôi sẽ chỉ cho bạn cách RequestDispatcher được sử dụng để **forward** or **include** phản hồi của một tài nguyên trong Servlet. Ở đây tôi đang sử dụng **index.html** để lấy tên người dùng và mật khẩu từ người dùng, **Validate** Servlet sẽ xác nhận mật khẩu đã nhập bởi người dùng, nếu người dùng đã nhập vào “kienkiki” làm mật khẩu, thì sẽ được chuyển tiếp tới **Welcome** Servlet khác mà người dùng sẽ ở trên trang index.html và một thông báo lỗi sẽ được hiển thị.

### a.Các tập tin cần được tạo ra:

* **Index.html** sẽ có các trường mẫu để lấy thông tin người dùng.
* **Validate.java** sẽ xác nhận dữ liệu được nhập bởi người dùng.
* **Welcome.java** sẽ là trang chào đón.
* **Web.xml** , descriptor triển khai.

#### Index.html:

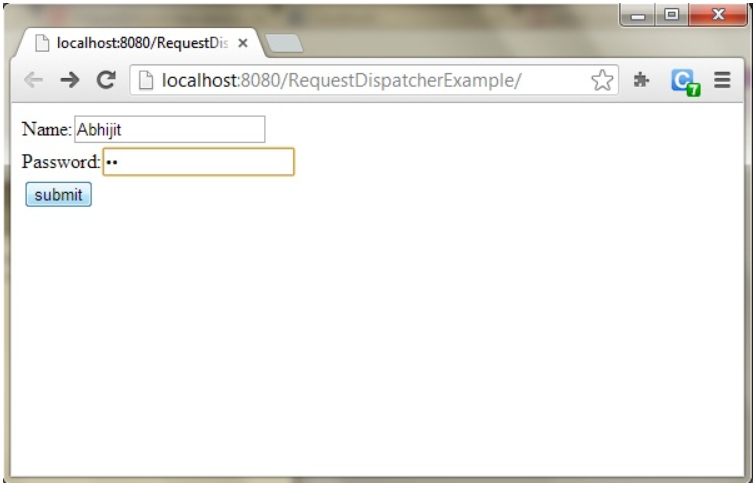
#### Validate.java:

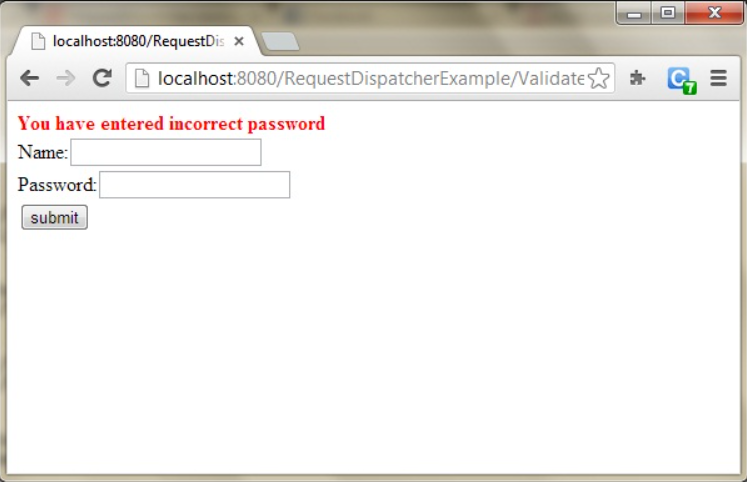
#### Welcome.java:

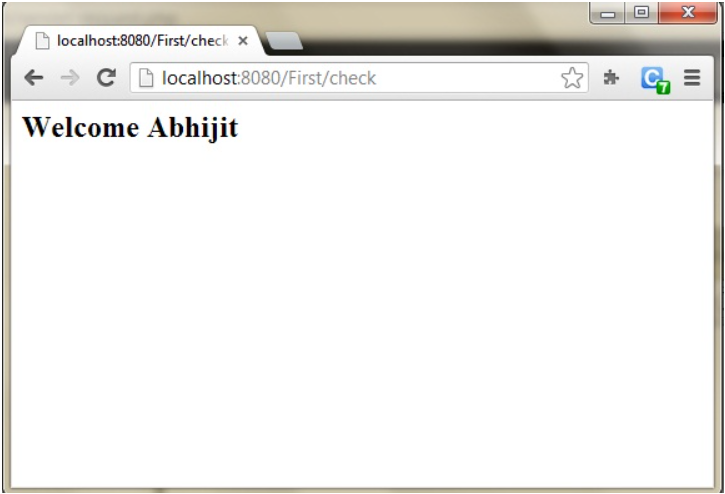
#### Web.xml:

### b.Các màn thể hiện các bước:

Đây sẽ là màn hình đầu tiên. Bạn có thể nhập Tên đăng nhập và Mật khẩu của bạn ở đây.



Khi bạn nhấn vào Submit, mật khẩu sẽ được xác nhận, nếu nó không phải là ‘kienkiki’, thông báo lỗi sẽ được hiển thị.

Nhập bất kỳ Tên đăng nhập, nhưng nhập ‘kienkiki’ làm mật khẩu. Mật khẩu sẽ được xác nhận thành công và bạn sẽ được hướng tới Servlet Chào đón.

# IX.Session tracking:

## 1.Định nghĩa:

* Session tracking hiểu đơn giản là cách để duy trì trạng thái(dữ liệu) của người dùng.
* Vấn đề: Khi một khách hàng được thực hiện mua sắm trực tuyến, ông / bà có thể chọn các mục từ các trang khác nhau và đặt nó trong giỏ hàng. Khi khách hàng nhấp vào một mới trang, các thông tin về các mục đã chọn trước đó bị mất do HTTP là stateless protocol
* Giải pháp: Session tracking

+Cho phép các ứng dụng Web duy trì thông tin với máy chủ miễn là khách hàng không đăng xuất khỏi Website.

+Theo dõi các danh tính của khách hàng và thông tin trạng thái khác cần thiết trong suốt session.

## 2.Các cách session tracking:

### a.Sử dụng URL Rewriting:

#### Tổng quan:

* Uniform Resource Locator (URL) là địa chỉ của một tài nguyên nằm trên Trang web.
* Với URL rewriting, bạn có thể thêm một số dữ liệu thêm vào cuối mỗi URL xác định sessions, và các máy chủ có thể kết hợp mà định danh session với các dữ liệu đã được lưu trữ về session đó.
* 
* Ví dụ: với địa chỉ http://kiendepzai.com?sessionid=12345 định danh session được đính kèm trong sessionid = 12345 có thể truy cập vào máy chủ để xác định thông tin khách hàng có sessionid = 12345.

#### Ưu điểm:

* Có thể được nối khi gửi dữ liệu từ form HTML.
* Có thể được gửi cùng với nội dung động từ một Servlet.
* Là một cách tốt để duy trì phiên khi các trình duyệt không hỗ trợ cookie hoặc người dùng vô hiệu hóa sự hỗ trợ cho các tập tin cookie.

#### Nhược điểm:

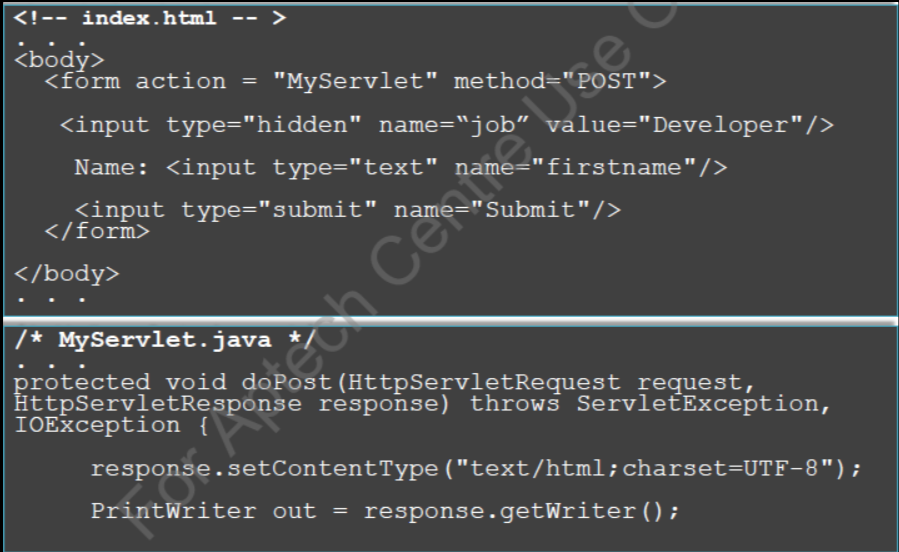
* URL chỉ có thể được gửi thông qua các hyperlink trên trang Web.
* URL dài không thể lưu trữ nhiều thông tin do giới hạn chiều dài URL(255).
* URL có chứa thông tin dữ liệu có thể nhìn thấy, vì vậy nó không phải an toàn.

### b.Sử dụng Hidden form fields:

#### Tổng quan:

* Một máy chủ web có thể gửi một trường mẫu HTML ẩn cùng với một session ID duy nhất như sau:
* 
* Mục nhập này có nghĩa là, khi các mẫu được gửi, tên và giá trị xác định được tự động đưa vào GET hoặc POST dữ liệu. Mỗi lần khi duyệt web sẽ gửi yêu cầu trở lại, sau đó giá trị session\_id có thể được sử dụng để giữ theo dõi của các trình duyệt web khác nhau.
* Chúng được đặt trong một form HTML.
* Chúng là một phần của biểu mẫu HTML tĩnh hoặc thông tin động tạo ra thông qua servlet.
* Chúng có thể được sử dụng để giữ bất kỳ loại dữ liệu.
* Chúng không thể được nhìn thấy bởi người sử dụng và do đó không được giải thích bởi các trình duyệt.

#### Ví dụ:





#### Ưu điểm:

* Được hỗ trợ trong tất cả các trình duyệt.
* Không yêu cầu máy chủ đặc biệt từ khách hàng.
* Không thể nhìn thấy trực tiếp bởi người sử dụng.
* Làm việc với hoặc không cần cookies.

#### Nhược điểm:

* Chỉ hoạt động khi trang nhận được yêu cầu thông qua một submit form.
* Đây có thể là một cách hiệu quả để theo dõi phiên nhưng khi click vào liên kết siêu văn bản (<A HREF...>) thường không dẫn đến việc gửi form, do đó các trường ẩn cũng không thể hỗ trợ theo dõi phiên.

### c.Sử dụng HttpSession:

#### Tổng quan:

* Servlet cung cấp HttpSession interface để xác định một người dùng trên nhiều hơn một trang yêu cầu hoặc truy cập vào một trang web và lưu trữ thông tin về người dùng đó.
* Session được tạo ra giữa một khách hàng HTTP và một máy chủ HTTP bằng servlet container sử dụng HttpSession interface.
* Sau một khoảng thời gian nhất định không hoạt động phiên giao dịch bị phá hủy để ngăn chặn việc số session tăng vô hạn.
* Nó sẽ vẫn hoạt động nếu người dùng vẫn còn hoạt động trong một khoảng thời gian lớn hơn thời gian được cài đặt
* Khoảng thời gian chờ Session có thể được thiết lập hoặc trong file web.xml hoặc có thể được thiết lập theo phương pháp setMaxInactiveInterval ().
* Các cài đặt timeout nên được viết trong file web.xml: <session-config>

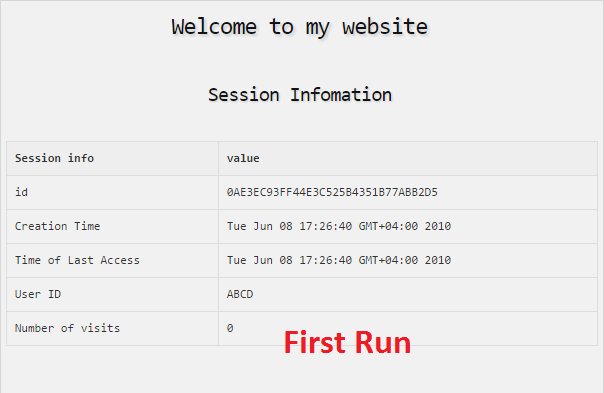
<session-timeout>N</session-timeout>

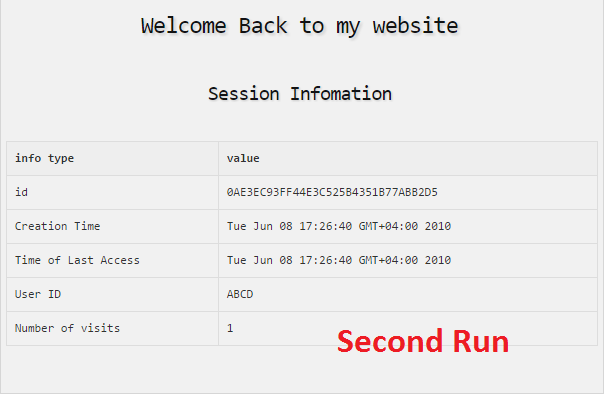
</session-config>

#### Ví dụ:







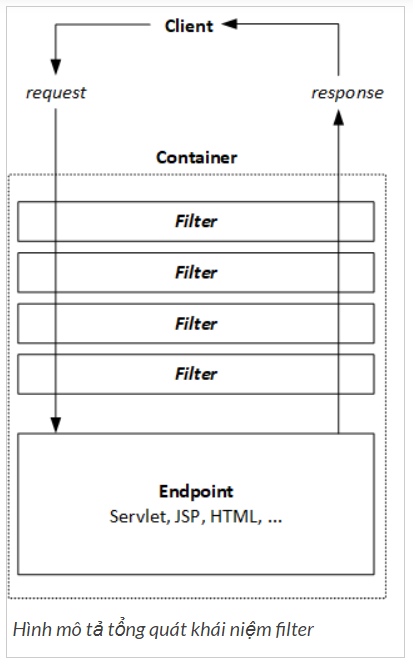


# X.Web filter:

## 1.Định nghĩa:

* Java Servlet 2.3 giới thiệu thành phần Filter. Filter sử dụng để “chặn”, hay thao-tác-trước đối với request và thao-tác-sau đối với response để chuyển đổi hoặc sử dụng thông tin được chứa trong request hoặc response đó.
* Hay dễ hiểu hơn là một đối tượng filter sẽ thực hiện các công việc tiền xử lý (request gửi đến) và hậu xử lý (response trả về).
* Filter trong Servlet là một khái niệm giống như middleware trong Laravel (PHP Framework) hoặc Express (Node.js Framework). Tức là những đối tượng trung gian làm nhiệm vụ thao tác với request trước khi xử lí và response trước khi trả về một cách tự động.
* Một request tới một servlet, JSP hay static file (.htm, .css, .js, image file, …) và response trả về tương ứng đều có thể dùng filter để thao tác.

## 2.Chức năng của Filter:



* Hình trên mô tả một số filter dùng để thao tác với tất cả request và response. Có thể thấy các filter được sắp xếp theo thứ tự, request phải đi qua lần lượt từng filter mới tới được endpoint; tương tự đối với response cũng phải đi qua lần lượt từng filter mới được chính thức trả về cho client. Tại mỗi filter, sẽ có code Java thao tác (can thiệp) cụ thể với request hoặc response.
* Phải hiểu rằng, một request từ client tới một endpoint có thể bị buộc phải đi số lượng filter bất kỳ; theo một thứ tự nhất định; bất kì một filter nào trong số đó cũng đều có thể can thiệp request/response, kể cả chặn đứng request/response hoàn toàn.

**Vậy filter có thể can thiệp được những gì?**

* Truy vấn request và xử lý phù hợp.
* Chặn request và response, không cho đi xa hơn (tới filter khác; tới endpoint hoặc client).
* Chỉnh sửa / thay đổi request header và request data, nghĩa là cung cấp một phiên bản request đã bị tùy chỉnh (không phải nguyên gốc từ client).
* Chỉnh sửa / thay đổi response header và response data, nghĩa là cung cấp một phiên bản response đã bị tùy chỉnh (không phải nguyên gốc do endpoint trả về).

## 3.Khai báo và triển khai một filter:

### a.Khai báo:



* WebFilter là một annotation dùng để kí hiệu đây là một khai báo Filter, được cung cấp trong package *“javax.servlet.annotation.WebFilter“*, bên trong có các parameter như *filterName* để đặt tên cho Filter, *urlPatterns* là một mảng chứa các mẫu url dẫn tới servlet, JSP page, static file, … (gọi chung là endpoint). Miễn là là url đến một endpoint khớp với pattern thì filter sẽ được áp dụng.
* Nếu như không sử dụng annotation, thì phải sử dụng file web.xml (gọi là deployment descriptor) để triển khai Filter.
* (phần @WebFilter annotation và web.xml sẽ giải thích rõ hơn trong phần Triển khai Filter).
* Filter là một class thực thi interface *Filter*, gồm 3 hàm:
* *init*: Được gọi bởi container, khi Filter được sử dụng.
* *doFilter*: Được gọi bởi container, mỗi khi có request/response phải đi quá Filter đó.
* *destroy*: Được gọi bởi container, khi Filter bị loại bỏ, không được sử dụng.

Hàm *doFilter()* của Filter thực hiện những thao tác xử lý đối với cặp đối tượng request/response. Ví dụ:

Chẳng hạn, để xử lí các HTTP request parameters mà client gửi lên, như kiểm tra parameter có hợp lệ hay không, hay biến đổi giá trị các parameters đó cho phù hợp, ta hay dùng các hàm của đối tượng request như sau:

* String getParameter(String name): Để lấy HTTP request parameter.
* void setAttribute(String name, Object o): Để tạo một Attribute, có thể là giá trị một parameter đã bị điều chỉnh.
* Object getAttribute(String name): Để lấy giá trị của một Attribute.

| Parameter vs. Attribute (trong ngữ cảnh của servlet):   * Parameter là HTTP request parameter, có giá trị kiểu String, tự động có khi request được gửi lên từ client. * Attribute cũng là “thuộc tính”, phải tự tạo, sử dụng nội bộ trong server-side, có thể mang nhiều kiểu dữ liệu khác nhau. |
| --- |

Ngoài ra trong hàm *doFilter()*, để ý thấy còn một đối tượng nữa là *FilterChain*, biểu thị một chain (chuỗi) các Filter liên tục nhau, sử dụng hàm *doFilter()* của *FilterChain* để gọi hàm *doFilter()* của *Filter* tiếp theo trong chuỗi Filter, cho đến đến khi chạm tới endpoint (servlet, JSP, static file, …), thì endpoint sẽ được thực thi.

### b.Triển khai 1 filter sử dụng web.xml:

Khi:

* Nếu sử dụng một Filter được cài đặt trong một thư viện khác (file .jar) thì phải dùng cách này.
* Nếu muốn kiểm soát thứ tự (order) của các Filter trong FilterChain.
* …

Tạo file *web.xml* trong NetBeans: Trong project Java Web Application, click phải vào thư mục WEB-INF > New > Standard Deployment Descriptor (web.xml) > Finish.

Thay đổi tiến hành bên trong tag <web-app>.

Muốn sử dụng một Filter, phải định nghĩa bằng tag <filter>:



Bắt buộc phải có các tag thuộc tính sau:

* *<filter-name>*: Tên Filter
* *<filter-class>*: Fully qualified name (tên class bao gồm cả package) của class Filter.

Ngoài ra còn nhiều tag khác như:

* *<display-name>*
* *<description>*
* *<init-param>*
* *<icon>*

Để mapping, tức là gắn một Filter vào các *urlPatterns*, hay servlet nào đó, sử dụng tag *<filter-mapping>*:



Bắt buộc phải có các tag thuộc tính sau:

* *<filter-name>*: Tên Filter muốn sử dụng.
* *<url-pattern>*: URL pattern, khi request đến url khớp với URL pattern thì Filter sẽ được chạy.
  + Có thể thêm nhiều tag <url-pattern>

Hoặc nếu như muốn chỉ định Filter cho một Servlet cụ thể, sử dụng tag: *<servlet-name>*, chứa tên của Servlet mong muốn.

Tất nhiên, phải định nghĩa Servlet đó trước, bằng *@WebServlet* annotation hoặc ngay trong *web.xml* với tên cụ thể.

**Giá trị của url-pattern, có thể:**

* Bắt đầu bằng *“/”* và kết thúc bằng *“/\*”* sử dụng để mapping cho một mẫu url nào đó. Chẳng hạn: /myservlet/\*. Nghĩa là toàn bộ những request tới endpoint có URL khớp với mẫu trên, như: /myservlet/helloworld đều phải qua Filter muốn mapping.
* “\*.” sử dụng cho những request có URL dẫn tới endpoint là một static file, như: \*.html, \*.js, …
* URL chính xác, trực tiếp.
* “/” sử dụng cho endpoint mặc định.



Thứ tự của các tag *<filter-mapping>* chính là thứ tự mà container sẽ áp dụng cho endpoint. Như ví dụ trên, container sẽ áp dụng *MyFilter* trước, sau đó mới đến cho *AnotherFilter*.

Có thể sử dụng kết hợp cả *@WebFilter* annotation và *web.xml*, bằng cách:

* Có thể định nghĩa *filterName* trong *@WebFilter*.
* Và tiến hành mapping trong *web.xml*.