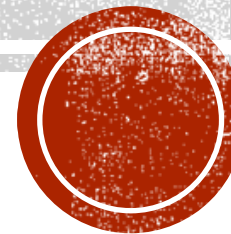


# SPRING BOOT — SESSION AND COOKIE

Trần Văn Thịnh



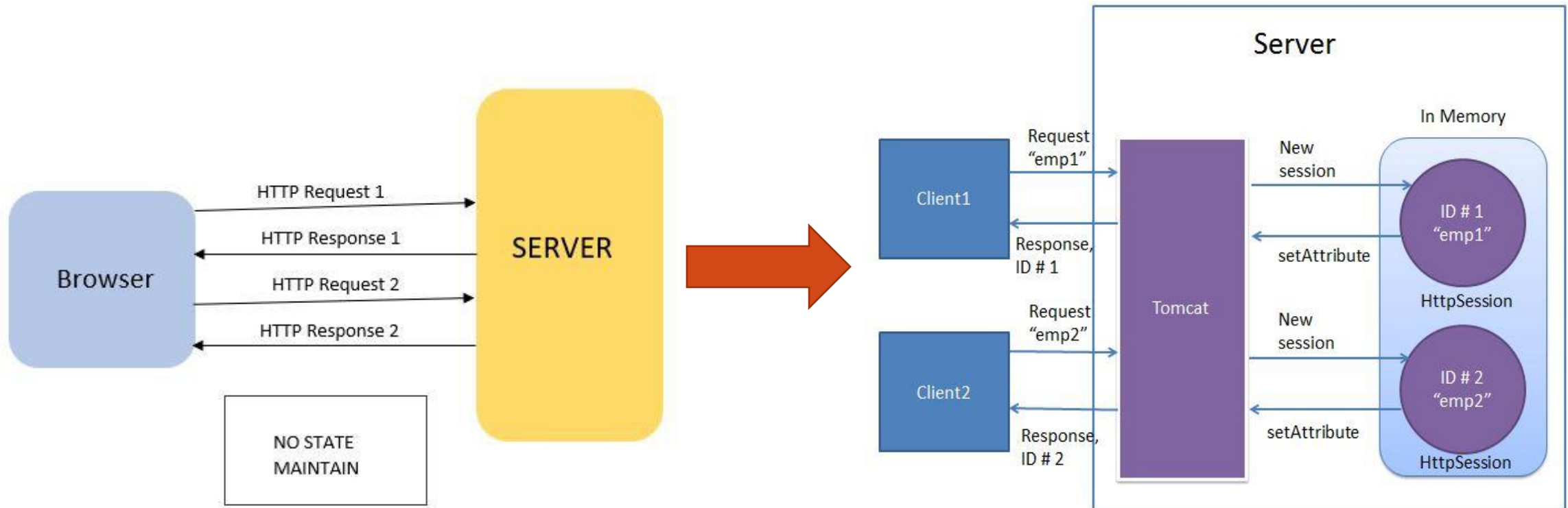


# NỘI DUNG

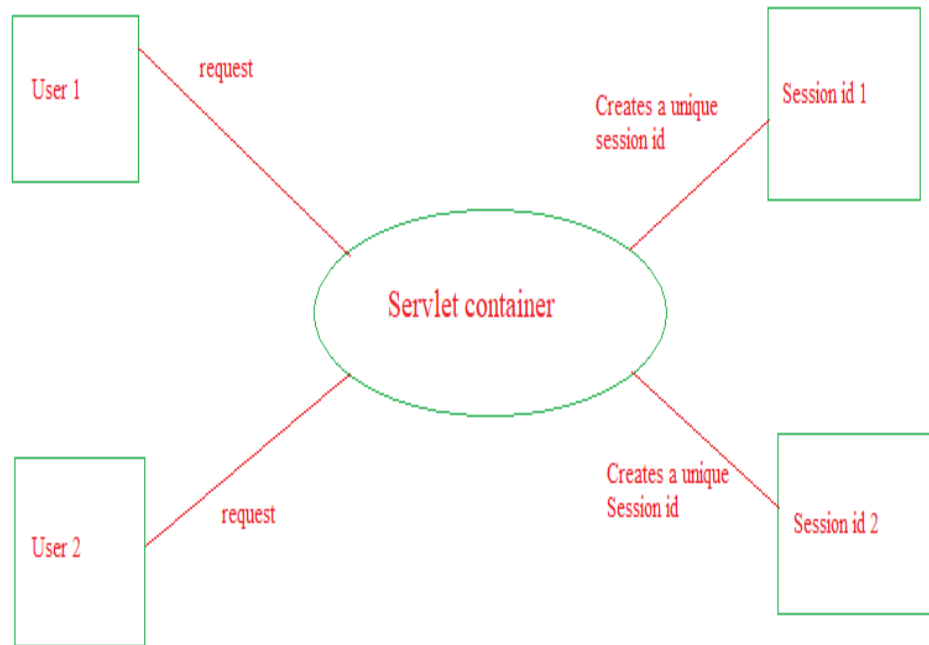
- Giới thiệu về session và cookie
- Quản lý session
- Quản lý session qua HttpSession và Spring Session Module
- Cookie

# GIỚI THIỆU VỀ SESSION VÀ COOKIE

- Http là giao thức stateless(không lưu trạng thái)
- Tất cả các request và response đều độc lập với nhau



# GIỚI THIỆU VỀ SESSION VÀ COOKIE

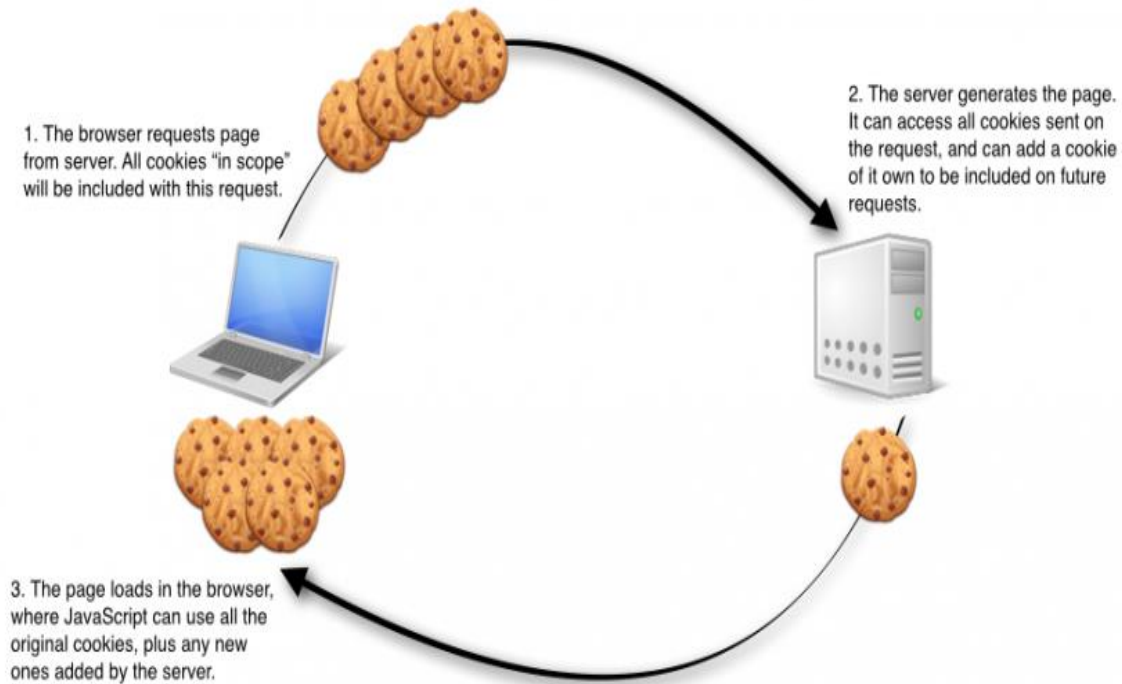


## ▪ Session là gì

- Phiên làm việc, là cách giao tiếp giữa client (trình duyệt web) với server
- Một session bắt đầu khi client gửi request đến sever, tồn tại xuyên suốt từ trang này đến trang khác trong ứng dụng và chỉ kết thúc khi hết thời gian timeout hoặc khi đóng ứng dụng, tắt trình duyệt
- Lưu trữ những thông tin tạm thời trong session
- Mỗi session sẽ được cấp phát một định danh duy nhất SessionID. Khi kết thúc một phiên làm việc và bắt đầu một phiên mới, mình sẽ nhận được một session ID khác



# GIỚI THIỆU VỀ SESSION VÀ COOKIE



## ■ Cookie là gì

- Là một đoạn văn bản ghi thông tin được tạo ra và lưu trên trình duyệt của máy người dùng
- Ghi nhớ những thông tin như tên đăng nhập, mật khẩu, các tùy chọn do người dùng lựa chọn đi kèm
- Tập tin cookie sẽ được truyền từ server tới browser và được lưu trữ trên máy tính khi truy cập vào ứng dụng.
- Đóng browser cũng không mất đi các giá trị vì đã lưu trên máy tính.



# GIỚI THIỆU VỀ SESSION VÀ COOKIE

## Cookie

Cookie được lưu trữ trên trình duyệt của người dùng.

Dữ liệu cookie được lưu trữ ở phía máy khách.

Dữ liệu cookie dễ dàng sửa đổi khi chúng được lưu trữ ở phía khách hàng.

Dữ liệu cookie có sẵn trong trình duyệt của chúng ta đến khi hết hạn.

## Session

Session không được lưu trữ trong trình duyệt của người dùng.

Dữ liệu session được lưu trữ ở phía máy chủ.

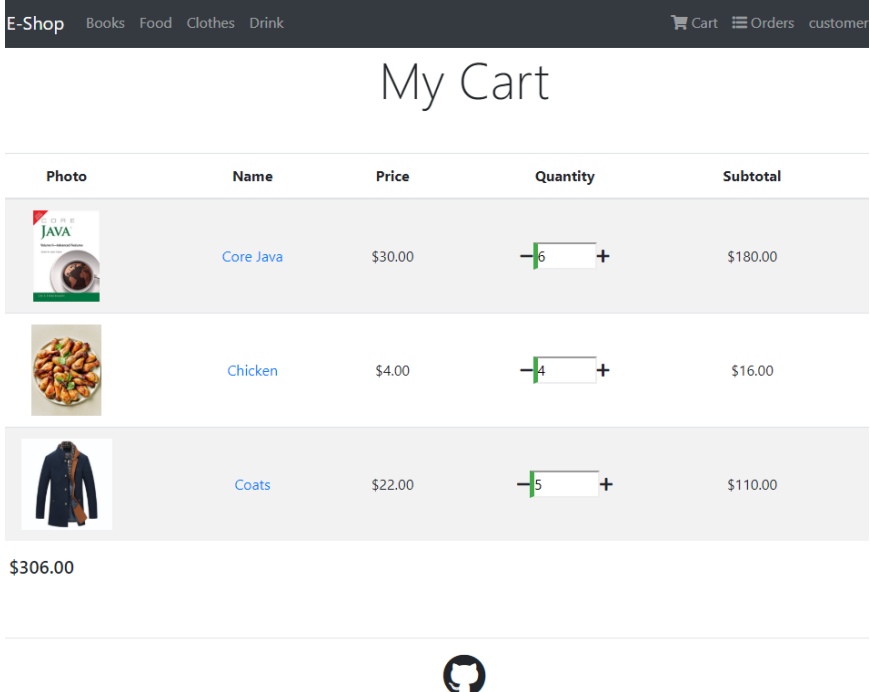
Dữ liệu phiên không dễ dàng sửa đổi vì chúng được lưu trữ ở phía máy chủ.




Dữ liệu phiên có sẵn cho trình duyệt chạy. Sau khi đóng trình duyệt sẽ mất thông tin phiên.



# GIỚI THIỆU VỀ SESSION VÀ COOKIE

- Ví dụ về dùng session và cookie
  - Shopping cart (Giỏ hàng)
    - User chọn sản phẩm mình cần mua sau đó thêm nó vào giỏ hàng, sau đó có thể tiếp tục mua các sản phẩm khác
    - Dùng session để lưu trữ danh sách các sản phẩm
    - Sử dụng chung session.n với cookie để lưu trữ dữ liệu lâu hơn. Cookie sẽ lưu trên máy người dùng nên nó sẽ không mất đi, nó mất đi khi thời gian sống hết hạn(do lập trình viên cấu hình)



My Cart				
Photo	Name	Price	Quantity	Subtotal
	Core Java	\$30.00	<input type="text" value="6"/>	\$180.00
	Chicken	\$4.00	<input type="text" value="4"/>	\$16.00
	Coats	\$22.00	<input type="text" value="5"/>	\$110.00
				\$306.00





# GIỚI THIỆU VỀ SESSION VÀ COOKIE

- Ví dụ về dùng session và cookie
  - Remember-me khi đăng nhập
    - Form đăng nhập có checkbox
    - Chọn checkbox thì lần sau không cần đăng nhập tên và mật khẩu nữa (lưu thông tin vào cookie)
    - Cookie sẽ hết hạn (ví dụ 30 ngày), thì lúc đó user cần đăng nhập lại

**Login with Username and Password**

User:

Password:

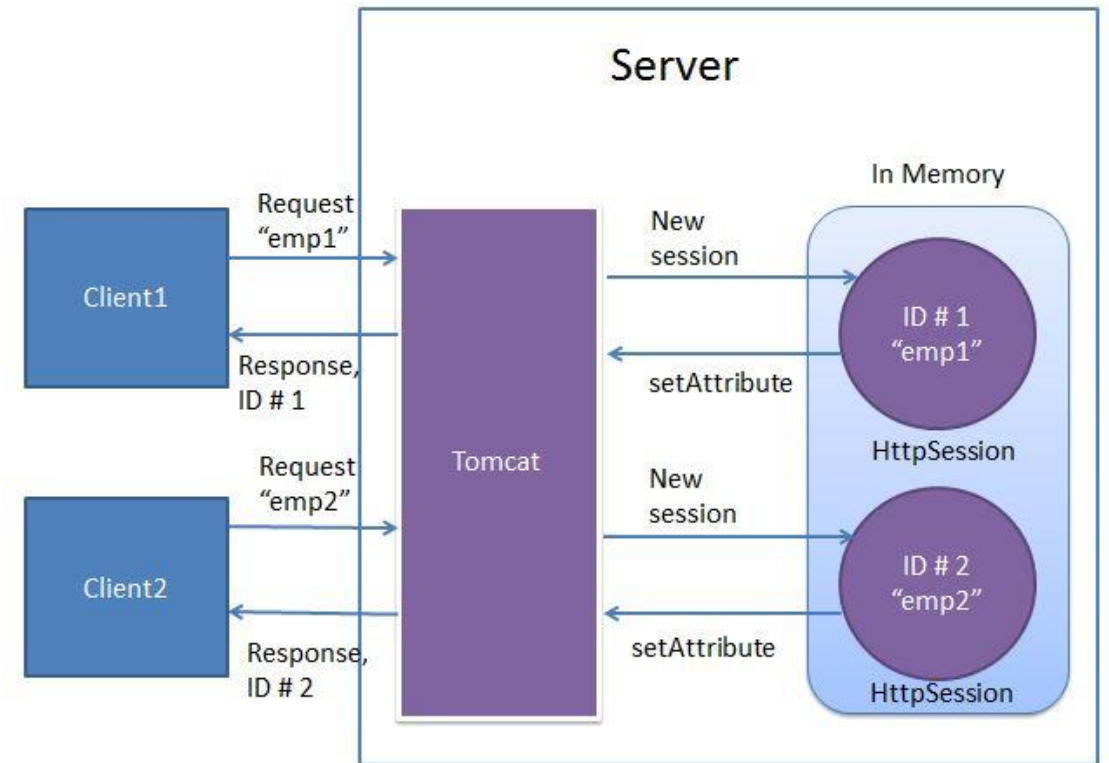
Remember Me: ☐





# QUẢN LÝ SESSION

- Quản lý session là gì
  - Theo dõi, tương tác với hoạt động của client qua nhiều request
  - Là cơ chế sử dụng bởi Web container để lưu trữ thông tin session cho từng user riêng biệt



# QUẢN LÝ SESSION

- Một số cách quản lý session trong java
  - Cookies
  - Hidden form field
  - URL Rewriting
  - HttpSession



# QUẢN LÝ SESSION

- Phương pháp sử dụng Cookies
  - Một web server có thể chỉ định một session ID duy nhất làm một cookie cho mỗi client và các yêu cầu tiếp theo từ client có thể được phân biệt bằng cách sử dụng cookie đã nhận.
  - Hạn chế khi trình duyệt không hỗ trợ cookies
- Phương pháp sử dụng Hidden Fields của HTML Form
  - Một web server có thể gửi một trường ẩn (Hidden Field) cùng với một session ID duy nhất như sau:  
`<input type="hidden" name="sessionid" value="1234567890">`
  - Khi Form được gửi lên, tên và giá trị trường chỉ định được đưa vào dữ liệu. Mỗi khi web browser yêu cầu trở lại, giá trị sessionid có thể được sử dụng để theo dõi các web browser
  - Hạn chế khi click vào link (tag <a href>)



# QUẢN LÝ SESSION

- Phương pháp sử dụng URL Rewriting
  - Có thể thêm một số dữ liệu bổ sung vào cuối mỗi URL xác định session và server có thể liên kết session id với dữ liệu mà nó đã lưu trữ về session đó
  - Ví dụ: với /test.html?sessionid=1234567890, session id được đính kèm dưới dạng sessionid = 1234567890, có thể được truy cập tại web server để xác định client.
  - Viết lại URL là một cách tốt hơn để duy trì session và nó hoạt động ngay cả khi các browser không hỗ trợ cookie. Hạn chế của việc viết lại URL là bạn sẽ phải tự động tạo ra tất cả các URL để chỉ định một session id, ngay cả trong trường hợp của một trang HTML tĩnh đơn giản.



# QUẢN LÝ SESSION

- Session trong Java Servlet – HttpSession
  - Servlet API hỗ trợ quản lý session thông qua HttpSession interface
  - Một số method hỗ trợ
    - HttpSession getSession() trả về HttpSession object mà được đính kèm trong request, hoặc tạo một mới và trả về nếu session không được đính kèm trong request
    - HttpSession getSession(boolean flag) Trả về HttpSession object nếu request có session hoặc null trong trường hợp ngược lại
    - String getId() trả về chuỗi chứa định danh duy nhất mà được gán cho session
    - Object getAttribute(String name); getAttributeNames(), removeAttributes(String name). setAttribute(String name, Object value)
    - ServletContext getServletContext() trả về ServletContext object cho ứng dụng
    - boolean isNew()
    - void invalidate()





# QUẢN LÝ SESSION

- JSESSIONID Cookie
  - Khi sử dụng phương thức `HttpServletRequest getSession()` và tạo mới một request, nó sẽ tạo một `HttpSession` object và cũng thêm vào 1 `Cookie` cho response object với tên `JSEESIONID` và giá trị là session id. Cookie này được dùng để xác định `HttpSession` object trả về trong các request thêm vào từ phía client. Nếu cookies bị vô hiệu hóa ở phía client và chúng ta sử dụng phương pháp URL rewriting thì method này sẽ sử dụng giá trị `jsessionId` từ request URL để tìm session tương ứng.



# QUẢN LÝ SESSION

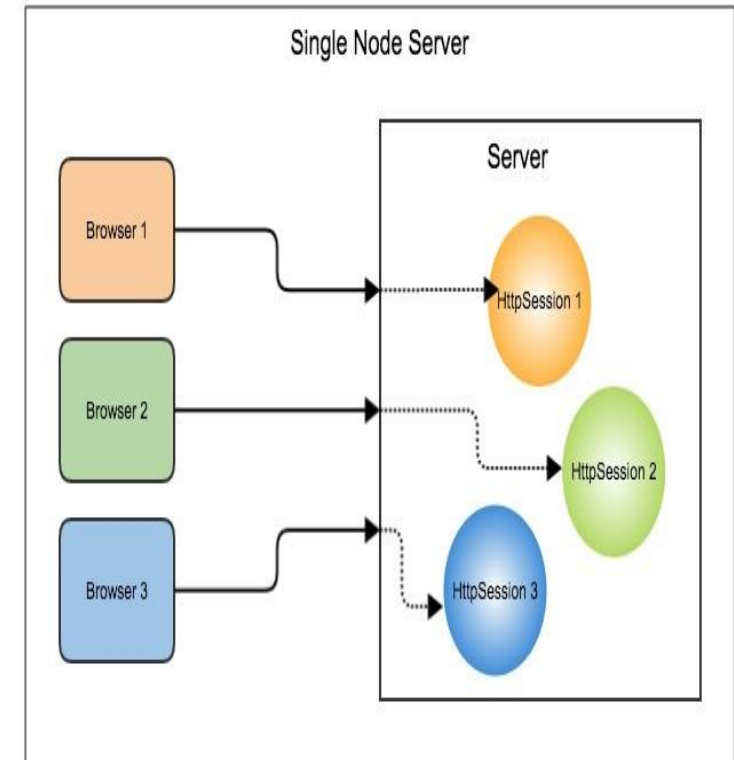
- Session trong môi trường được phân nhóm (clustered environment - CE)
  - Trong các hệ thống với số lượng người dùng ít thì bạn chỉ cần deploy 1 webserver (tomcat...) nhưng nếu trong các ứng dụng với lượng người sử dụng nhiều thì vấn đề cần sử dụng nhiều instance tomcat(nhiều tomcat để chia tải) là điều hiển nhiên, việc tạo thêm các instance này được gọi là cluster(cụm).
  - Trên môi trường product, thường chúng ta có nhiều server node với một bộ load balancer phía trước chúng, và mọi client traffic đều sẽ đi qua một trong các server node. Do đó, cần có một số cơ chế để làm cho dữ liệu session của user dùng được cho mỗi client trong CE
  - Một số kĩ thuật để quản lý session
    - Single Node Server
    - Multi-Node Server with Load Balancer and Sticky sessions
    - Multi-Node Server with Load Balancer and Session Replication
    - Multi-Node Server with Load Balancer and Session Data in a Persistent DataStore



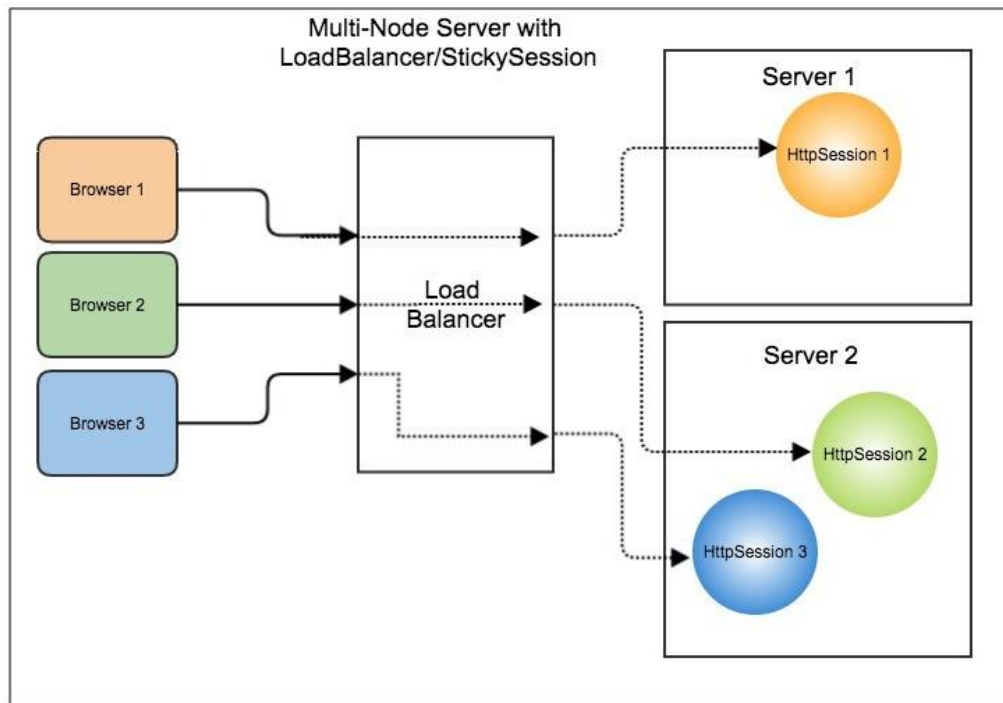


# QUẢN LÝ SESSION

- **Single Node Server (Máy chủ nút đơn)**
  - Nếu ứng dụng của bạn không phải là một dịch vụ quan trọng đối với doanh nghiệp của bạn, sẽ không có quá nhiều người dùng đồng thời và một vài thời gian ngừng hoạt động (downtime) được chấp nhận thì chúng ta có thể triển khai Single Node Server
  - Trong mô hình này, với mỗi client browser, một session object được tạo ra trên server (HttpSession trong hệ thống dung java) và SESSION\_ID được đặt vào một cookie trên browser để xác định session object.
  - Mô hình này thường không được chấp nhận cho hầu hết các ứng dụng vì nếu server sập (go down) thì service cũng sập (go down)



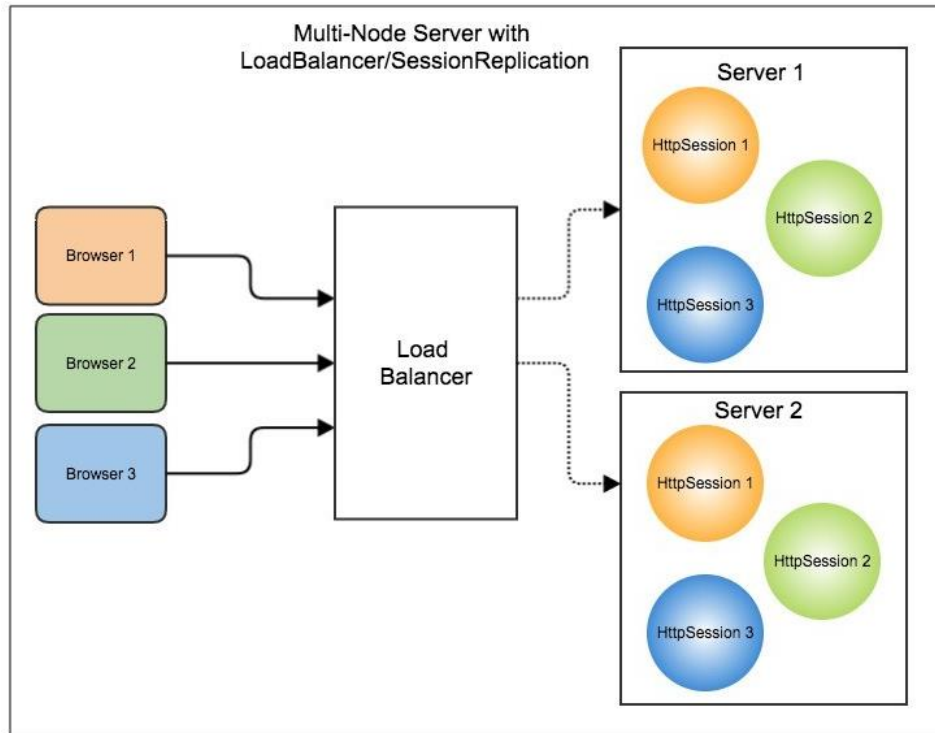
# QUẢN LÝ SESSION



- **Multi-Node Server with Sticky Sessions (Máy chủ nhiều nút với sticky session)**
  - Để làm cho ứng dụng của chúng ta có tính khả dụng, sẵn sàng cao và tăng số lượng user, chúng ta có mô hình multi node server nằm phía sau một hệ thống load balancer (bộ cân bằng tải). Trong cách tiếp cận Sticky Session, chúng ta cấu hình load balancer để định hướng tất cả các request từ cùng một user tới cùng node
  - User session được khởi tạo trên một server node và tất cả các request thêm vào từ phía client sẽ được gửi tới cùng node đó. Nhưng vấn đề của cách tiếp cận này là nếu một server node sập thì kéo theo tất cả dữ liệu user session trên server đó cũng ra đi.



# QUẢN LÝ SESSION



- Multi-Node Server with Session Replication (Máy chủ nhiều nút với Session Replication (sao chép))
  - Trong mô hình này, dữ liệu user session sẽ được sao chép (replicated) trên tất cả các server node, cho nên mỗi request có thể được định tuyến đến bất kỳ node nào. Khi server node có sập thì client request vẫn có thể được xử lý bởi node khác
  - Nhưng mô hình này yêu cầu hỗ trợ về phần cứng và làm tăng việc phải cấu hình cho các server



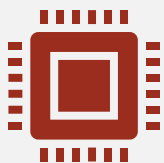


QUẢN LÝ SESSION  
QUA HTTPSESSION  
VÀ SPRING  
SESSION MODULE

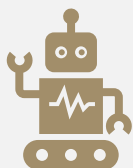
Session với  
java/spring  
MVC

Session với  
Spring Session  
Module





Với java EE, java servlet, chúng ta sử dụng HttpSession với các phương thức hỗ trợ như ở trên



Với Spring/Spring MVC, chúng ta có một số cách để làm việc với session

Sử dụng HttpSession từ request, servlet hoặc qua argument của Controller method

Sử dụng scoped proxy

Sử dụng @SessionAttributes

# SESSION VỚI JAVA/SPRING MVC



# SESSION VỚI JAVA/SPRING MVC

- Cách sử dụng Scoped proxy
  - Khi setup 1 bean (@Bean) với session-scoped, bean đó được hỗ trợ bởi một proxy. Đồng nghĩa với chúng ta có thể inject bean này vào trong singleton-scoped @Controller
  - Bởi vì ngay từ đầu không có session khi khởi tạo context, Spring sẽ tạo ra một proxy của class đó để inject vào như một dependency. Target instance của class đó sẽ được khởi tạo nếu cần khi được yêu cầu bởi request
  - @Bean @Scope( value = WebApplicationContext.SCOPE\_SESSION, proxyMode = ScopedProxyMode.TARGET\_CLASS)
  - Cách này không làm ảnh hưởng đến khai báo của method request mapping, khả năng readability cao hơn khi sử dụng @SessionAttributes

# SESSION VỚI JAVA/SPRING MVC

- Cách sử dụng @SessionAttributes annotation
  - @SessionAttributes được sử dụng ở cấp độ class
    - @SessionAttributes và @ModelAttribute (method annotation): Thuộc tính khai báo trong @SessionAttribute sẽ được tìm trong HttpSession, nếu không tìm thấy thì phương thức @ModelAttribute với cùng giá trị sẽ được gọi để điền vào session
    - @SessionAttributes và @ModelAttribute( argument annotation) Spring sẽ lấy giá trị của thuộc tính từ session và gán giá trị cho tham số

```
@Controller
@SessionAttributes("visitor")
@RequestMapping("/trades")
public class TradeController {

    @RequestMapping("/")
    public String handleRequestById (@ModelAttribute("visitor") Visitor visitor,
                                     Model model,
                                     HttpServletRequest request) {
        model.addAttribute("msg", "trades request, serving page " +
                                request.getRequestURI());
        visitor.addPageVisited(request.getRequestURI());
        return "traders-page";
    }

    @ModelAttribute("visitor")
    public Visitor getVisitor (HttpServletRequest request) {
        return new Visitor(request.getRemoteAddr());
    }
}
```





# SESSION VỚI JAVA/SPRING MVC

- Cách sử dụng @SessionAttributes annotation
  - Các thuộc tính session như được chỉ ra bằng cách sử dụng annotation này tương ứng với các thuộc tính của model, được lưu trữ minh bạch trong một conversational session. Các thuộc tính đó sẽ bị xóa sau khi trình xử lý cho biết đã hoàn thành conversational session của nó. Do đó, hãy sử dụng phương tiện này cho các conversational attributes mà được lưu trữ trong một session tạm thời trong một conversational attributes của một quá trình cụ thể
  - Đối với các thuộc tính session cố định, ví dụ: một đối tượng xác thực người dùng, hãy sử dụng phương thức session.setAttribute truyền thống để thay thế.
  - Để báo hiệu một conversational session sẽ kết thúc ở bước nào đó, truyền argument Session status vào controller method và sử dụng status.setComplete()



# SPRING SESSION MODULE

- Spring Session Core
- Spring Session Data Redis
- Spring Session JDBC
- Spring Session Hazelcast
- ...





**Reading  
HTTP Cookie**



**Setting HTTP  
Cookie**



**Reading All  
Cookies**



**Cookie  
Expiration**



**Secure  
Cookie**



**HttpOnly  
Cookie**



**Cookie Scope**



**Deleting  
Cookie**

**SPRING  
SECURITY  
REMEMBER ME**

