

# Session & Cookie

## Bạn đã thật sự hiểu?

Lê Văn Quân

Ngày 31 tháng 7 năm 2025

# Mở đầu câu chuyện

- Ngày xưa, khi chưa có cookie hay session
- Server không thể phân biệt được người dùng nào đang truy cập.
- Lý do:
  - HTTP là giao thức **stateless**
  - Mỗi request độc lập, không “nhớ” request trước.

## Khi không có cookie/session, server làm gì?

- 1 Gắn ID vào URL (URL rewriting):  
`/profile?user_id=123`
- 2 Dùng hidden field trong form:  
`<input type="hidden" value="123" name="user_id">`
- 3 Theo dõi địa chỉ IP

**Nhược điểm:** không bảo mật, dễ giả mạo, không tiện lợi

# Sự ra đời của Cookie

- **1994** – Lou Montulli (kỹ sư tại Netscape) đề xuất cơ chế Cookie
- Mục tiêu: giải quyết việc lưu trạng thái người dùng trong giao thức HTTP (vốn stateless)
- Cookie cho phép lưu một lượng nhỏ dữ liệu trên trình duyệt:
  - ID người dùng
  - Tùy chọn ngôn ngữ, giao diện
  - Phiên đăng nhập
- Sau này được chuẩn hóa trong **RFC 2109** (1997) → **RFC 6265** (2011)

# Sự ra đời của Cookie

## Cách hoạt động:

- 1 Server gửi header: `Set-Cookie: user_id=12345`
- 2 Trình duyệt lưu cookie vào máy
- 3 Trong các request sau, trình duyệt gửi: `Cookie: user_id=12345`

**Nhờ vậy:** Server có thể nhận biết user trong mỗi request

# Vấn đề bảo mật với Cookie

- Cookie được lưu phía người dùng → người dùng có thể sửa!
- Ví dụ: `user_id=12345` tự đổi thành `user_id=1`
- Khi đó, người dùng có thể mạo danh người khác nếu server không kiểm tra cẩn thận
- Ngoài ra:
  - Cookie có thể bị đánh cắp qua XSS
  - Cookie gửi kèm mọi request → rò rỉ thông tin

# Giải pháp ra đời: Session!

- Session bắt đầu xuất hiện và được chuẩn hoá khoảng từ năm 1996–1997
- Session lưu thông tin trên server, chỉ gửi ID cho client
- Cookie chỉ chứa `session_id`, không chứa thông tin quan trọng
- Người dùng không thể sửa dữ liệu session vì nó nằm trên server
- Nhờ đó: bảo mật tốt hơn, tránh mạo danh và can thiệp

# Tình huống thực tế

- Trên một trang web xem phim
- Tôi là tài khoản **free**, còn bạn là **premium**
- Tôi lấy được cookie của bạn (auth\_token, session\_id, v.v.)
- Tôi gán vào trình duyệt của mình và... **truy cập được phim premium!**

**Câu hỏi:** Liệu chỉ dùng session là đủ để ngăn chặn?



## Chỉ dùng Session liệu có đủ?

- Session lưu trên server – tốt hơn cookie thuần túy
- Nhưng: session\_id vẫn gửi qua cookie → **có thể bị đánh cắp**
- Nếu attacker gán session\_id vào trình duyệt họ → Server vẫn tin tưởng!

**Kết luận:** Session là cần thiết, nhưng **không đủ** để ngăn giả mạo!

# Giải pháp chống giả mạo session

- Ràng buộc session với:
  - Địa chỉ IP gốc
  - Trình duyệt (User-Agent)
  - Device fingerprint (nếu cần)
- Hạn chế thời gian sống của session
- Sử dụng access token ngắn hạn + refresh token dài hạn
- Luôn kiểm tra quyền truy cập phía server có được phép xem phim không, trước khi gửi file video về cho trình duyệt

# So sánh nhanh Cookie vs Session

Tiêu chí	Cookie	Session
Lưu ở đâu	Trình duyệt	Server
Bảo mật	Thấp	Cao
Dung lượng	~4KB	Tùy server
Thời gian sống	Tùy cài đặt	Tùy cấu hình

# Ứng dụng của Cookie

- **Lưu tùy chọn người dùng:** Theme (tối/sáng), ngôn ngữ, layout cá nhân
- **Theo dõi giỏ hàng tạm thời:** Lưu ID sản phẩm trước khi người dùng đăng nhập
- **Theo dõi hành vi người dùng:** Dùng bởi Google Analytics, Facebook Pixel, v.v.
- **Ghi nhớ đăng nhập (Remember me):** Duy trì trạng thái đăng nhập qua nhiều lần truy cập
- **Quản lý popup/thông báo:** Ghi nhớ người dùng đã tắt thông báo, không hiển thị lại
- **A/B Testing:** Gán user vào nhóm A hoặc B để thử nghiệm giao diện

# Ứng dụng của Session

- **Giỏ hàng trong e-commerce:** Lưu thông tin sản phẩm, số lượng theo session
- **Tạm lưu dữ liệu form:** Tránh mất dữ liệu nếu submit lỗi
- **Chống gửi form nhiều lần:** Xác thực token chống CSRF
- **Theo dõi user đang hoạt động:** Hiển thị trạng thái “online” trong hệ thống
- **Lưu vị trí điều hướng trước đó:** Redirect về trang trước sau khi đăng nhập
- **Lưu tiến trình thao tác:** Như trong các bước đăng ký nhiều bước (multi-step form)

# Kết luận

- Cookie & Session là nền tảng cho mọi website hiện đại
- Từ chỗ “không thể nhớ ai” → Web cá nhân hóa, bảo mật hơn
- Giúp trải nghiệm người dùng trở nên mượt mà và đáng tin cậy