


# Sessions

Web Development

## Sessions Introduction



```
graph LR; HttpSession[HttpSession]; Cookies[Cookies]; HiddenField[Hidden Field]; URLRewriting[URL-Rewriting]; Cookies --- PST[Partial Session Tracking]; HiddenField --- PST; HttpSession --- SM[State Management]; URLRewriting --- SM;
```

Keeping track of the currently logged in user is a challenge - as HTTP is, by definition 'stateless'. Hidden form fields, url rewriting and cookies are three common techniques for implementing sessions.

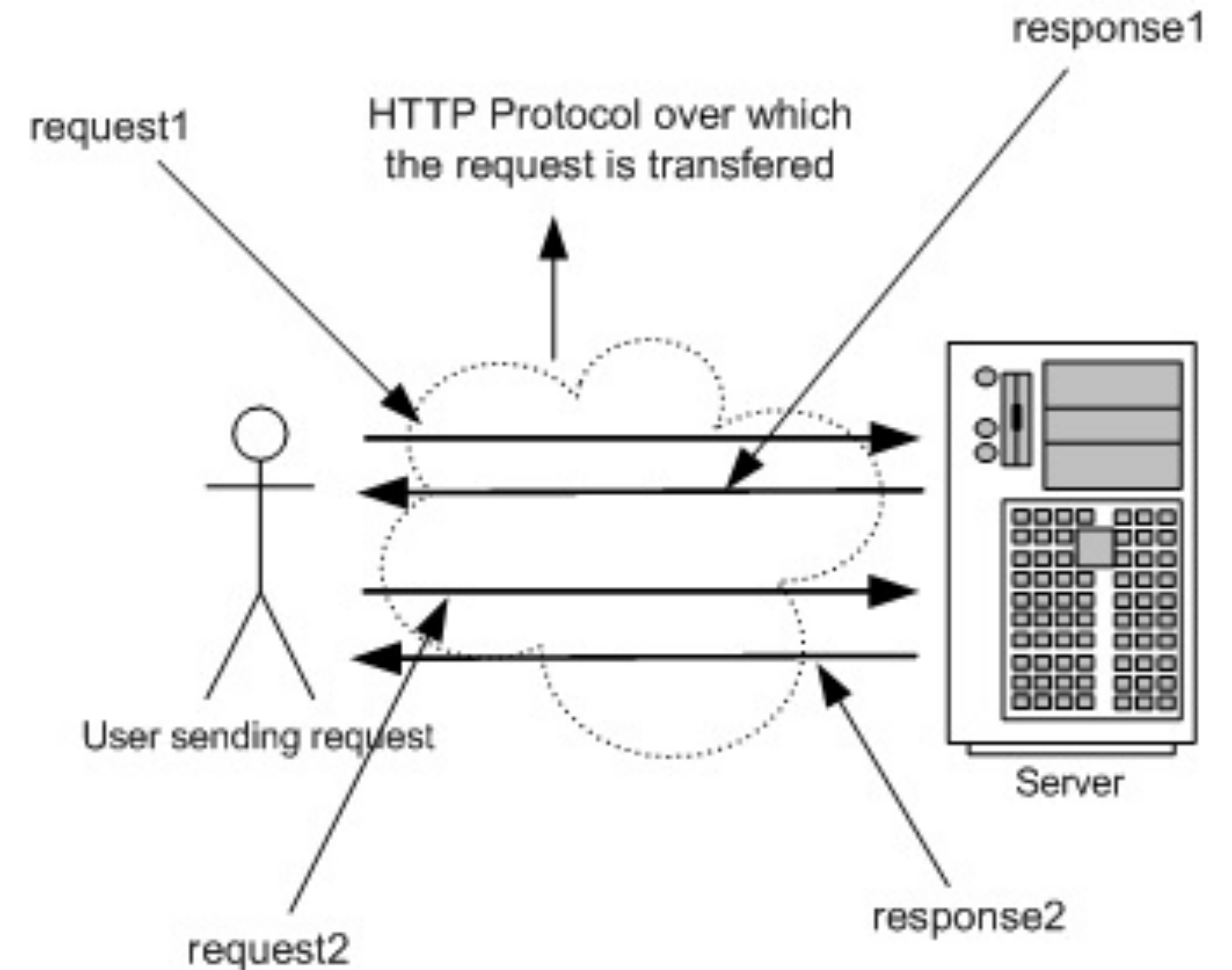
# How to Make an Application out of a Web Page?



- On the internet, a web page is a web page is a web page...
  - If you surf from ./page1.html to ./page2.html these are two unique requests.
  - The server doesn't know anything about the fact that both pages are visited by the same user.
- Sessions are the technique used to logically group several requests into a "group" (called a session)
  - If you start a session, the server will know that it's still the same user who surfed from ./page1.html to ./page2.html

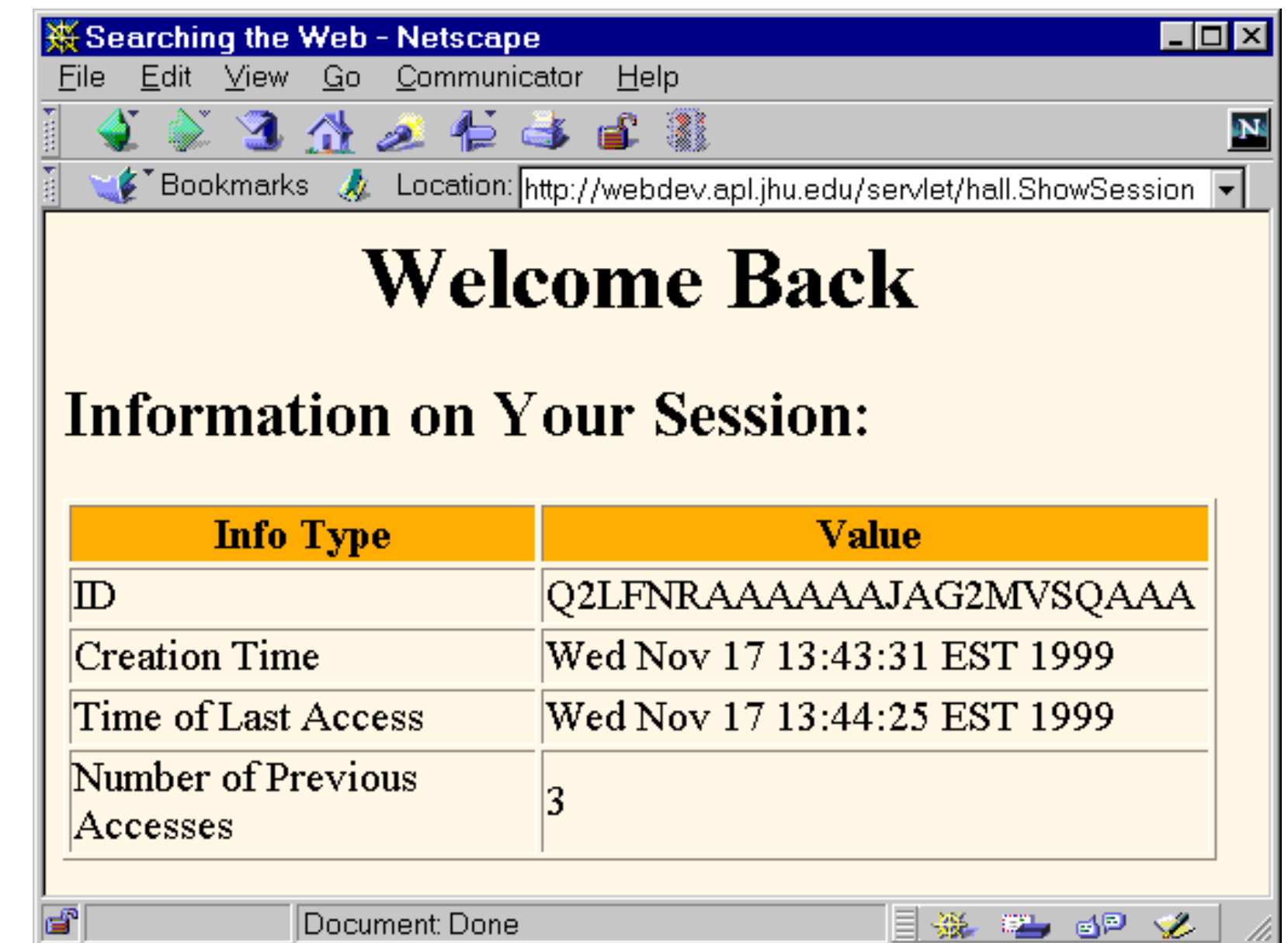
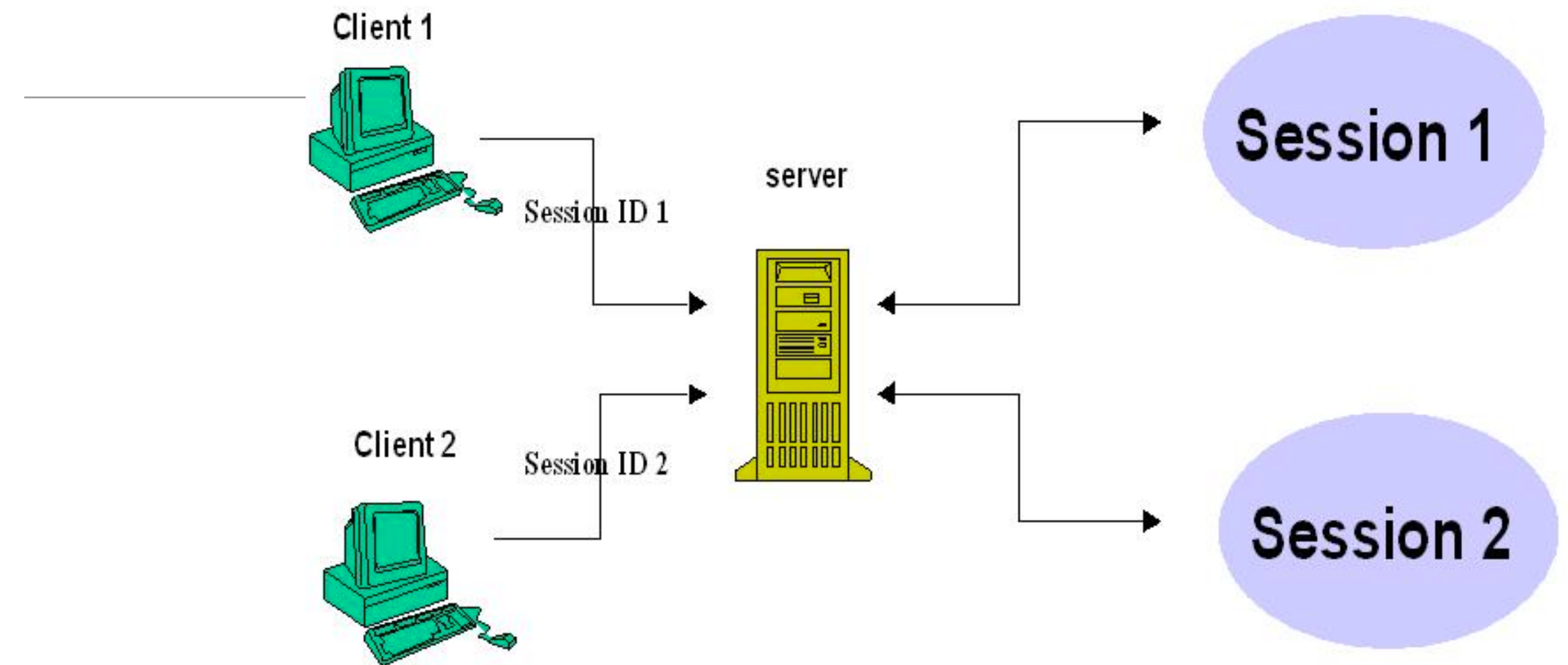
# Sessions

- HTTP itself is “stateless”
  - no state stored on the server between requests from the same client
- but many web apps are stateful
  - necessary to connect requests from the same user / browser / browser-window, e.g. shopping cart, appointments calendar etc...
- *Session*
  - multiple requests performed in a stateful context
- *Session tracking*
  - technique that allows sessions in stateless environments



- User surfs to <http://demo.com>
- Server (on 1st request / if no sessionID stored on client)
  - generates unique session id, which is mapped to ...
  - ... a session-object
    - stored in memory (lost on shutdown), in a file or in database
    - can contain anything (list of articles, game state, counters, ...)
- Session id is added to the response
- from now on:
  - each subsequent request from the same user (browser) must contain the session id ...
  - ... which is used by the server to map to the session-object
- No data gets stored on the client, except SessionID

# Session Tracking



# Session Tracking Techniques

---

**HttpSession**

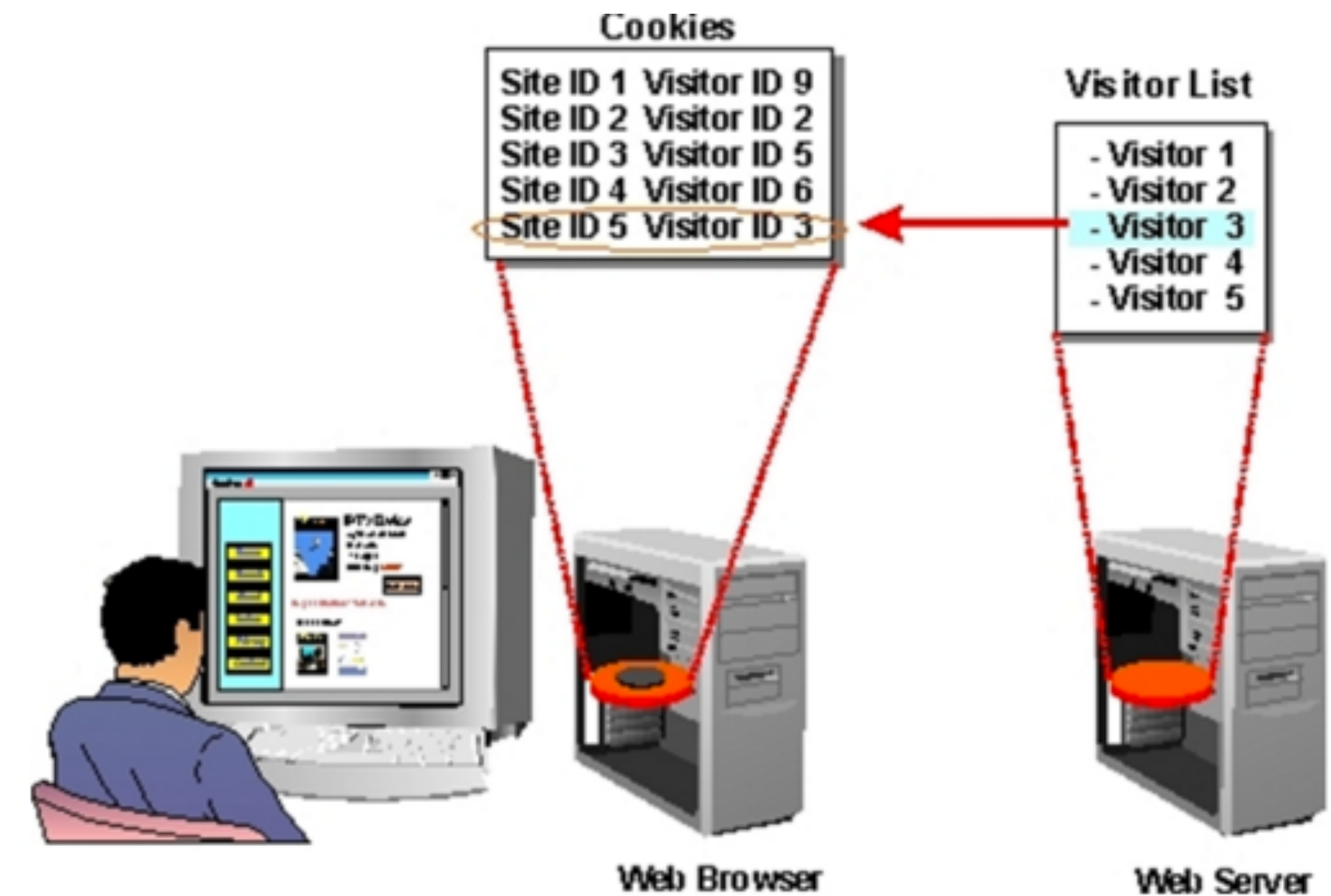
**Cookies**

**Hidden Field**

**URL-Rewriting**

# Cookies

1. Server creates a cookie with session-id on first request
2. Server maps id to a new user-specific session object
3. The session-id is sent to the client with the first response
4. ..and automatically added by the browser on each further request (to the same address/ domain/...)
5. Server receives request + cookie with session-id
6. Server maps session-id to session-object



# URL Rewrite

---

- Basic idea:
  - Server adds the session-id to all links the user can follow
    - `http://server/myhome`
  - is changed to
    - `http://server/myhome?sessionid=123`
  - sessionid must be dynamically added
    - functionality usually offered by scripting frameworks

# Hidden Form Fields

---

- In HTML, we can define "hidden" fields in a form
  - `<input type="hidden" name="sessionid" value="123">`
- These fields are not visible and cannot be changed by the client
- Usage:
  - server creates a session-object for each client and generates a unique ID
  - When HTML documents are created and sent back, the hidden form field is automatically generated containing the actual ID
  - Upon form submit, the session ID is automatically sent back to the server
  - The server can associate this call with an already existing session



# Web Frameworks

---

- Cookies generally preferred.
- However, framework will try to ‘abstract away’ specific session management technology, and deliver simpler abstraction to the programmer
- Framework may in fact be able to switch between different techniques depending on circumstances.