



Môn học

# Lập Trình Mạng

Giảng Viên:

Phạm Minh Tuấn



# Giới thiệu

- ❖ Phạm Minh Tuấn
- ❖ E-mail: [pmtuan@dut.udn.vn](mailto:pmtuan@dut.udn.vn)
- ❖ Tel: 0913230910
- ❖ Khoa Công nghệ thông tin – Trường ĐHBK – ĐHĐN



# Thỏa thuận

## ❖ Đối với giáo viên:

- Dạy đủ tất cả nội dung của môn học.
- Trả lời các câu hỏi của học sinh trong và ngoài giờ giảng liên quan tới môn học.
- Ra bài tập cho học sinh
- Lên lớp đúng giờ



# Thỏa thuận (tiếp)

## ❖ Đối với học sinh:

- Tham gia trên 80% số tiết học.
- Tham gia đóng góp tiết học như phát biểu, trả lời hay đặt câu hỏi cho giáo viên (không nói chuyên riêng)
- Làm bài tập đầy đủ.
- Lên lớp đúng giờ (không được đi trễ hơn giáo viên quá 5 phút)



# Mục tiêu môn học

- ❖ Hiểu được các giao thức mạng.
- ❖ Lập trình giao thức mạng
- ❖ Lập trình đa luồng
- ❖ Lập trình với cơ sở dữ liệu



# Nội dung môn học

## ❖ Khái niệm chung

- Kiến Trúc Mạng
- Lập trình mạng
  - Đối tượng lập trình mạng
  - Phạm vi
- Các loại mạng
- Hệ điều hành
  - Unix, Linux, Windows



# Nội dung môn học (tt)

## ❖ Các mô hình mạng

- Nguyên tắc truyền thông
- Mô hình truyền thông
  - Phương pháp phân tầng
  - Nguyên tắc
- Mô hình 7 tầng OSI
- Mô hình 4 tầng TCP/IP
- Mô hình thu gọn 3 tầng

# Nội dung môn học (tt)

## ❖ Mô hình ứng dụng client/server

- Thành phần và chức năng
- Cách hoạt động
- Đặc trưng mô hình ứng dụng client/server
- Ưu nhược điểm
- Client/server 2 lớp
- Client/server 3 lớp
- Giao thức cho ứng dụng



# Nội dung môn học (tt)

- ❖ Lập trình với TCP
- ❖ Lập trình với UDP
- ❖ Lập trình đa tuyến(luồng)
- ❖ Lập trình với CSDL



# Bài 1:

# Khái niệm chung

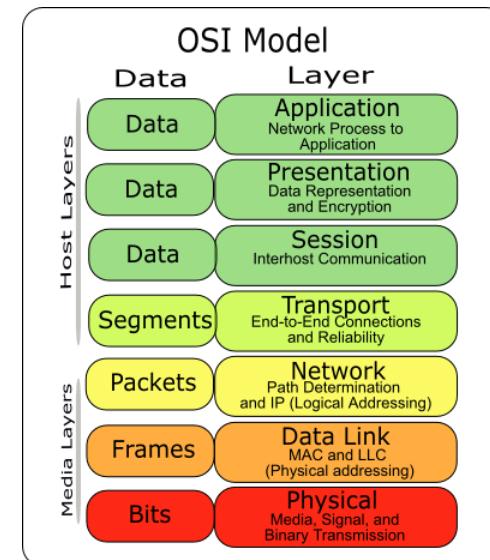
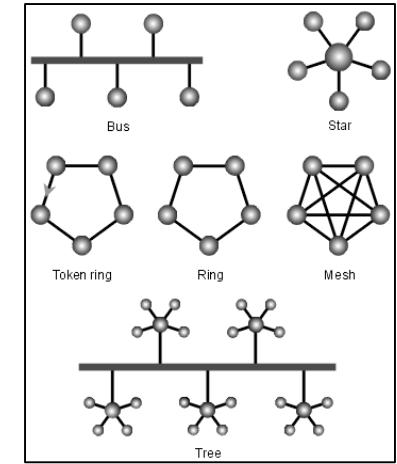
# Kiến Trúc Mạng

## ❖ Các topology

- Là cấu hình kết nối vật lý

## ❖ Kiến trúc phân tầng

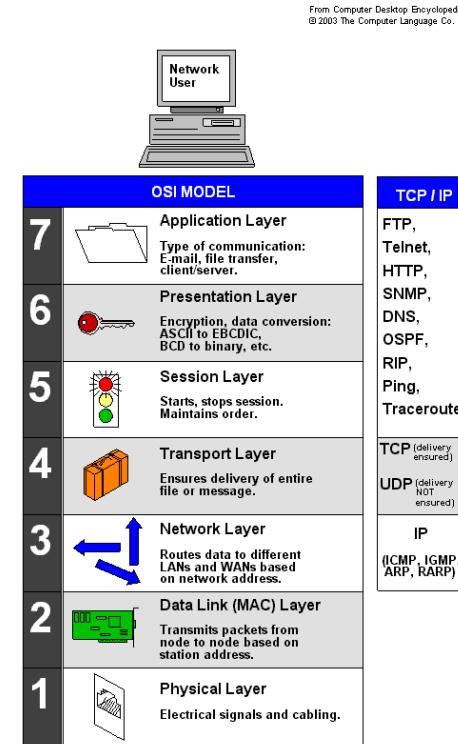
- Hệ thống các tầng giao thức mạng gồm
  - Các thực thể phần cứng
  - Phần mềm
    - Đảm bảo hoạt động của hệ thống
  - Vd: OSI hay TCP/IP



# Lập trình mạng

## ❖ Đối tượng lập trình mạng

- Các thực thể phần mềm thực thi giao thức trong hệ thống mạng
  - Được xây dựng trên nền tảng hệ thống máy tính
  - Phần cứng và hệ điều hành, kiến trúc phân tầng mạng





# Lập trình mạng (tt)

## ❖ Vậy, LT mạng?

- Tạo ra các thực thể phần mềm hoạt động trên một tầng
  - Sử dụng các thực thể ở tầng kè dưới
  - Cung cấp dịch vụ cho các thực thể tầng kè trên
- Chủ yếu, tạo các thực thể phần mềm ở tầng ứng dụng
  - Cung cấp dịch vụ cho người dùng



# Phạm vi môn học

- ❖ Tập trung vào các kỹ thuật lập trình sử dụng dịch vụ tại tầng transport để xây dựng các ứng dụng mạng
- ❖ Lập trình đa luồng
- ❖ Lập trình truyền tải thông tin với CSDL



# Hạ tầng truyền thông

- ❖ Một ứng dụng hay một dịch vụ mạng cần có hạ tầng mạng bên dưới khi hoạt động
- ❖ Tùy theo yếu tố kỹ thuật hay yêu cầu đối với từng ứng dụng mà ta cần phải lựa chọn loại mạng ứng dụng và dịch vụ



# Các loại mạng

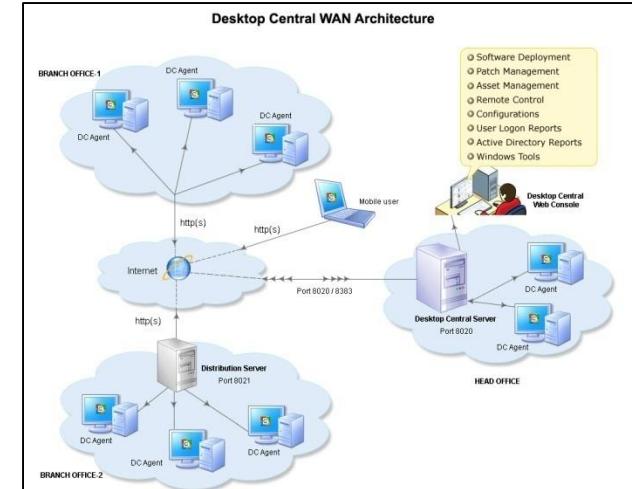
## ❖ Mạng cục bộ LAN

- Băng thông (bandwidth) rộng
- Tỷ lệ lỗi thấp
- Thích hợp với các ứng dụng
  - Email
  - Truyền file
  - Ứng dụng CSDL có độ truy xuất cao
  - Ứng dụng truyền đa phương tiện

# Các loại mạng(tt)

## ❖ Mạng diện rộng WAN

- Có nhiều kỹ thuật để lắp đặt mạng WAN
  - VD: Lease-line, Frame-relay, ISDN, ATM...
  - Mỗi kỹ thuật có băng thông khác nhau
- WAN thường kết nối các mạng LAN ở xa nhau
- Đường WAN sử dụng với mục đích truyền số liệu, kết nối từ xa, VoIP,...





# Câu hỏi???

- ❖ Băng thông là gì?
- ❖ Tại sao phải chú ý đến băng thông?



# Bài tập

## ❖ BT1:

- Để tải một đoạn phim có dung lượng 5Mbyte bằng đường dây có băng thông 1Mbps thì ta phải mất bao nhiêu thời gian?

## ❖ BT2:

- Giả sử bộ phim 5Mbyte có thể xem được 5 phút. Vậy tối đa có bao nhiêu người có thể xem phim cùng một lúc mà không bị giật?

## ❖ BT3:

- Một ngày trung bình có bao nhiêu lượt xem phim thì không bị giật?



# Các loại mạng(tt)

## ❖ Mạng Internet

- Là môi trường kém ổn định và không an toàn so với LAN và WAN
- Các dịch vụ mạng trên internet:
  - Email, Web, thương mại điện tử, Game online...
- **Vấn đề về an ninh mạng**





# Hệ điều hành

❖ Một ứng dụng mạng hoạt động trên một hoặc nhiều hệ thống máy tính.

- Để hoạt động được thì ứng dụng cần môi trường hoạt động.
- Môi trường quan trọng nhất đó là:  
**Hệ Điều Hành**



# Hệ điều hành (tt)

## ❖ Unix

- Bắt đầu được xây dựng tại phòng thí nghiệm Bell Lab.
- Là hệ điều hành đa nhiệm, đa người sử dụng và phục vụ truyền thông rất tốt
- Hạn chế:
  - Có quá nhiều phiên bản
  - Phức tạp trong quản trị
  - Đòi hỏi cấu hình mạnh



# Hệ điều hành (tt)

## ❖ LINUX

- Là một phiên bản thu nhỏ của UNIX
- Có nhiều phiên bản
  - Redhat Linux, Mandrake Linux...
- Dùng cho máy trạm, máy chủ và siêu máy tính
- Là hệ điều hành đa nhiệm, đa người dùng
- Tính ổn định cao
- Hỗ trợ truyền thông cao
- Miễn phí



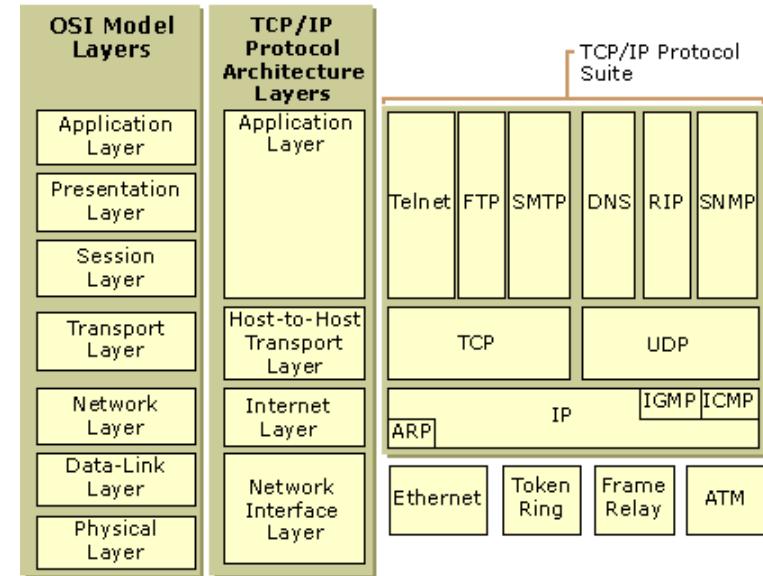
# Hệ điều hành (tt)

## ❖ Windows

- Là hệ điều hành đa nhiệm, đa người sử dụng
- Tính năng hỗ trợ mạng
- Đễ sử dụng
- Có các phiên bản cho máy trạm, máy chủ
- Hỗ trợ nhiều loại dịch vụ
- Hạn chế:
  - Bảo mật kém
  - Ít ổn định
  - Không miễn phí

# Tập giao thức

- ❖ Trong phạm vi môn học này, trọng tâm sử dụng bộ giao thức TCP/IP với lý do:
  - Là bộ giao thức phổ biến, có thể dùng:
    - Mọi loại mạng
      - LAN, WAN, Internet
    - Mọi hệ điều hành
    - Các thiết bị phần cứng





# Ngôn ngữ lập trình và công cụ

## ❖ Ngôn ngữ lập trình

- C/C++
- Java
- .NET
- BASIC

## ❖ Công cụ phát triển

- DevC
- Eclipse
- MVS



# Một số chú ý về kỹ thuật LT mạng

## ❖ LT thủ tục

- Chia chương trình thành các chương trình con
- Hàm thủ tục

## ❖ LT hướng đối tượng

- Thiết kế chương trình theo hướng đối tượng,
  - Tạo thư viện phục vụ LT mạng thành các gói, lớp đối tượng
  - Sử dụng một số thư viện đối tượng có sẵn

## ❖ LT đa tuyến

- Tận dụng tối đa khả năng của bộ vi xử lý
  - Thực hiện đồng thời nhiều tác vụ

## ❖ LT với CSDL



# Bài 2:

# Các mô hình mạng



# Nguyên tắc truyền thông

- ❖ Một máy tính trở thành một môi trường truyền dữ liệu cần có các yếu tố sau:
  - Các máy tính phải được kết nối nhau theo một cấu trúc topology nào đó.
  - Việc chuyển dữ liệu thực hiện dưới những quy định thống nhất gọi là giao thức mạng (protocol).
  - Phân chia hoạt động truyền thông của hệ thống thành nhiều lớp theo các nguyên tắc nhất định.



## Mô hình truyền thông trong kiến trúc mạng

### ❖ Phương pháp phân tầng mạng

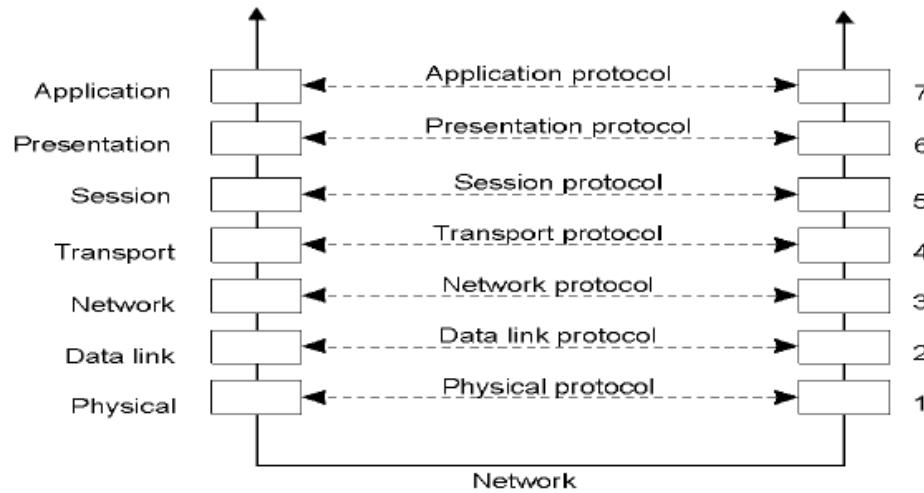
- Tách và xét mô hình mạng thành các môđun độc lập:
  - Giảm độ phức tạp cho việc thiết kế và cài đặt.

### ❖ Nguyên tắc

- Mỗi hệ thống xây dựng như một cấu trúc nhiều tầng và có cấu trúc giống nhau:
  - Số lượng tầng và chức năng của các tầng.

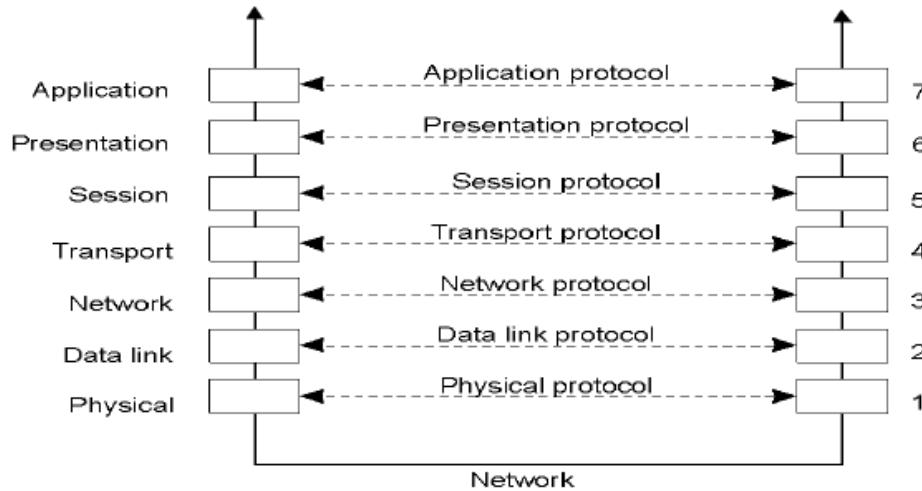


- Dữ liệu chỉ được truyền giữa 2 tầng kề nhau
- Bên gửi: Dữ liệu từ tầng cao nhất lần lượt đến tầng thấp nhất.
- Bên nhận: Dữ liệu từ tầng thấp nhất ngược lên đến tầng cao nhất





- Chỉ có 2 tầng thấp nhất mới có liên kết vật lý với nhau, còn các tầng trên cùng thứ tự chỉ có liên kết logic với nhau.

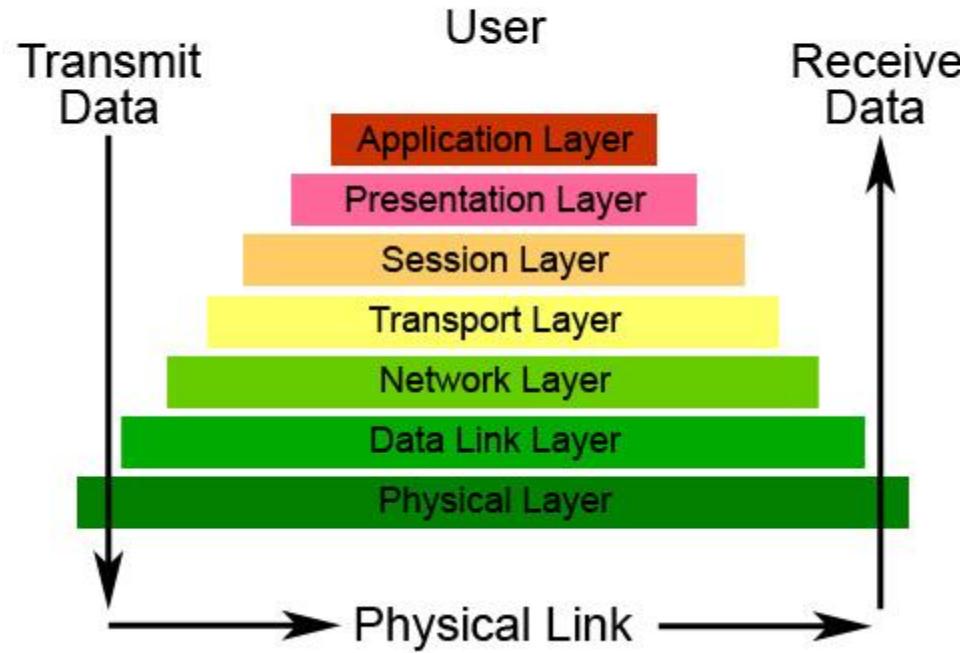




# Mô hình OSI (Open Systems Interconnection)

- ❖ Là mô hình gồm 7 tầng

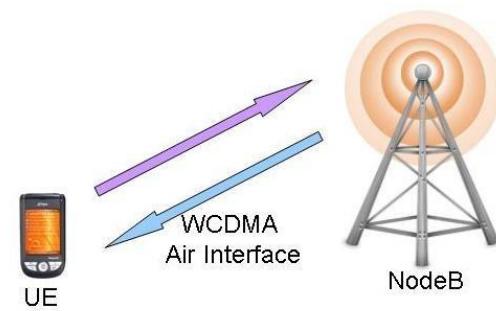
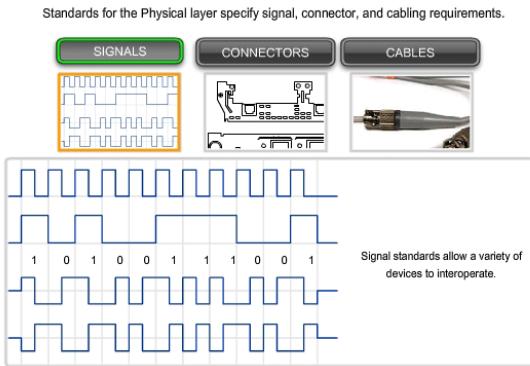
The Seven Layers of OSI





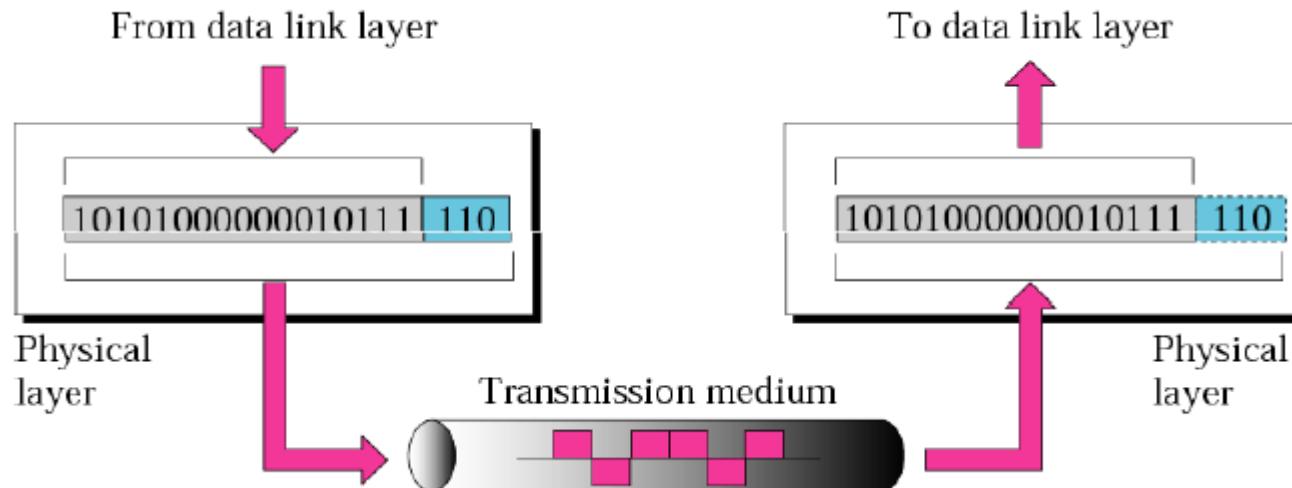
# Tầng vật lý (physical layer)

- ❖ Kiểm soát mức thấp nhất việc truyền thông giữa 2 nút mạng.
- ❖ Card mạng và cáp mạng. Truyền dây các bít giữa 2 nút.
- ❖ Các nhà lập trình không làm việc ở tầng này.
  - Trách nhiệm của các nhà phát triển driver phần cứng và các kỹ sư điện, điện tử.
- ❖ Lỗi có thể xảy ra trong quá trình truyền dữ liệu ở tầng này do điện áp, hay nhiễu đường truyền trên mạng.



# Tầng liên kết dữ liệu(Data link layer)

- ❖ Chịu trách nhiệm truyền dữ liệu tin cậy hơn
- ❖ Nhóm dữ liệu thành các frames.
  - Frames tương tự như các packet dữ liệu, nhưng chúng là các khối dữ liệu, được đặc tả theo kiến trúc phần cứng.
- ❖ Frames có trường kiểm tra lỗi truyền
- ❖ Đảm bảo dữ liệu bị méo không được truyền lên tầng trên.



# Ví dụ về Kiểm tra lỗi (error detection and correction)

## ❖ Bít chẵn lẻ (*parity bit*)

7 bit dữ liệu	byte có bit chẵn lẻ	
	Quy luật số chẵn	Quy luật số lẻ
0000000	00000000	00000001
1010001	10100011	10100010
1101001	11010010	11010011
1111111	11111111	11111110

## ❖ Khối chẵn lẻ (*parity block*)



## Bài tập

- ❖ Bt1: Hãy tính khả năng(xác suất) phát hiện lỗi khi sử dụng bit chẵn lẻ với dữ liệu 7 bit?



# Lời giải

- ❖ Giả sử dữ liệu không bị lỗi
  - 1 trường hợp → Kiểm tra không thành công
- ❖ Giả sử dữ liệu bị 1 bit lỗi
  - 8 trường hợp → Kiểm tra thành công
- ❖ Giả sử dữ liệu 2 bit lỗi
  - 28 trường hợp → Kiểm tra lỗi không thành công
- ❖ ....
- ❖ Giả sử dữ liệu có n bit lỗi ?



# Bài tập

- ❖ Bt1: Hãy tính khả năng(xác suất) phát hiện lỗi khi sử dụng khối chẵn lẽ với khối dữ liệu  $3 \times 7$  bit?



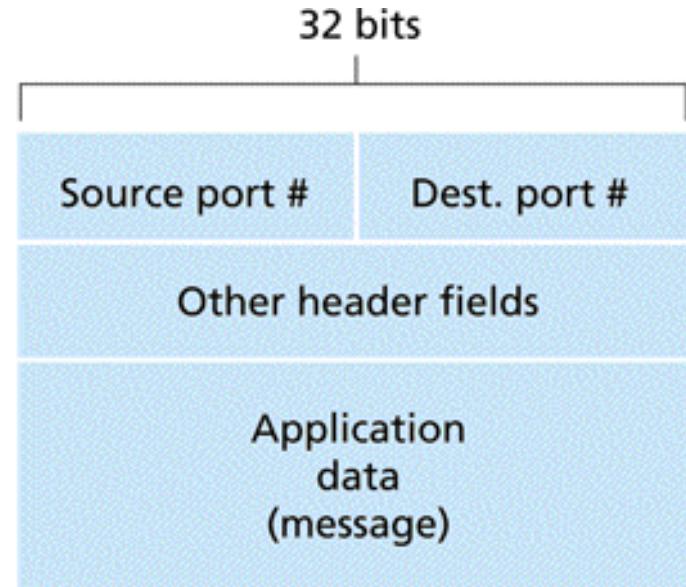
# Tầng mạng (Network layer)

- ❖ Các frames từ tầng Datalink lên hoặc các segments từ tầng transport xuống.
- ❖ Dữ liệu ở dạng packets
- ❖ Phần header chứa các thông tin quan trọng:
  - Địa chỉ mạng (network address)
  - Định tuyến mạng (routing)
- ❖ Packets được gửi qua lại giữa các mạng
- ❖ Các packets thường được định tuyến khác nhau
  - Việc định tuyến do các routers thực hiện
- ❖ **Các lập trình viên rất ít khi làm dịch vụ cho tầng này.**



# Tầng vận chuyển (transport layer)

- ❖ Liên quan đến việc dữ liệu được truyền như thế nào
- ❖ Dữ liệu dạng segments
- ❖ Chịu trách nhiệm
  - Xử lý việc kết nối
  - Phát hiện lỗi tự động
  - Điều khiển luồng dữ liệu





# Tầng phiên (session layer)

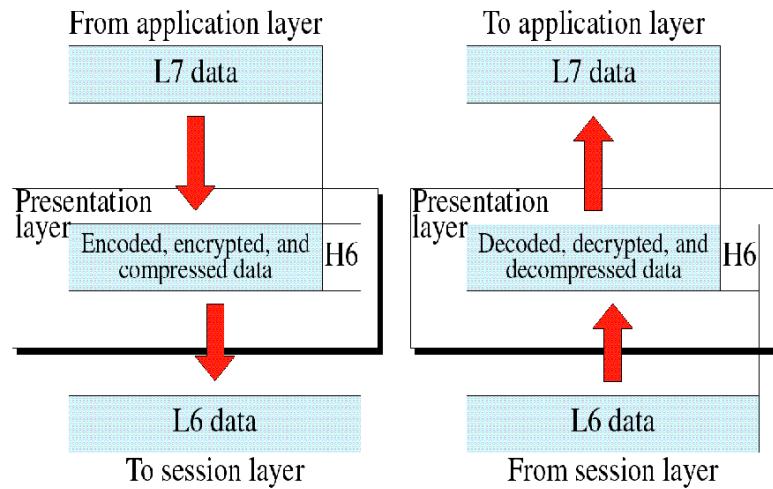
- ❖ Làm cho dễ dàng việc trao đổi dữ liệu
- ❖ Quản lý phiên truyền thông giữa các ứng dụng
  - Thiết lập một phiên
  - Đồng bộ một phiên
  - Thiết lập lại phiên nếu một phiên bị kết thúc đột ngột
- ❖ Không phải tất cả các ứng dụng đều sử dụng giao thức có kết nối
  - Do vậy việc quản lý phiên không phải lúc nào cũng được yêu cầu.



# Tầng trình bày (Presentation layer)

## ❖ Nhiệm vụ đảm bảo hiển thị và chuyển đổi dữ liệu

- Các máy tính khác nhau có thể sử dụng các khía cạnh biểu diễn dữ liệu khác nhau
- Nén giải nén dữ liệu
- Mã hóa giải mã dữ liệu





# Tầng ứng dụng (Application layer)

- ❖ Tầng cao nhất trong mô hình mạng
- ❖ Hầu hết các ứng dụng mạng được viết ở tầng này





# Các giao thức

## ❖ Application

- HTTP, FTP, SMTP, NSF, Telnet, SSH, ECHO, ...

## ❖ Presentation

- SMB, NCP, ...

## ❖ Session

- SSH, NetBIOS, RPC, ...



# Các giao thức (tt)

## ❖ Transport

- TCP, UDP, ...

## ❖ Network

- IP, ICMP, IPX

## ❖ Data link

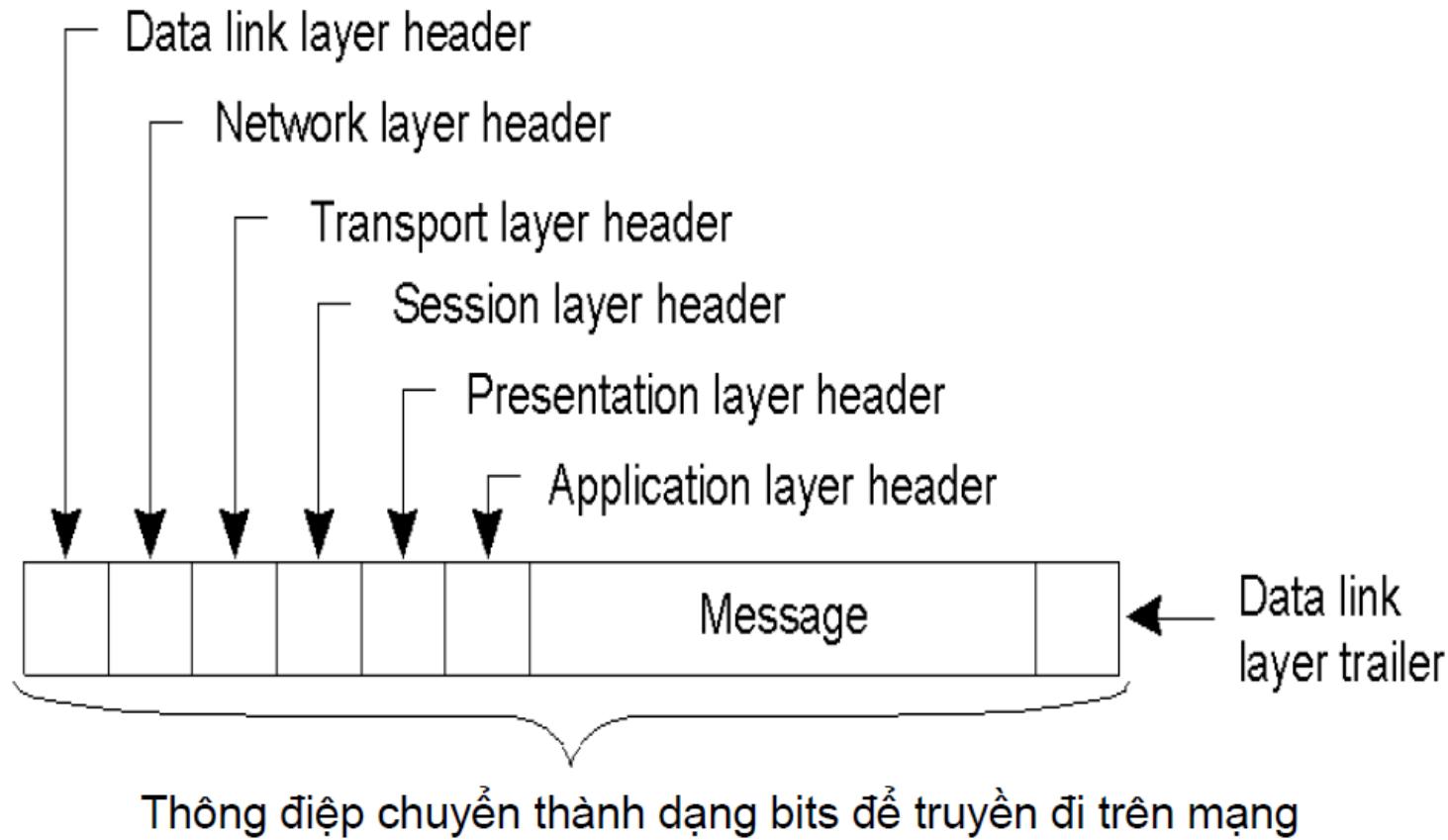
- Ethernet, Token Ring, ISDN, ...

## ❖ Physical

- 100BASE-T, 1000BASE-T, 802.11

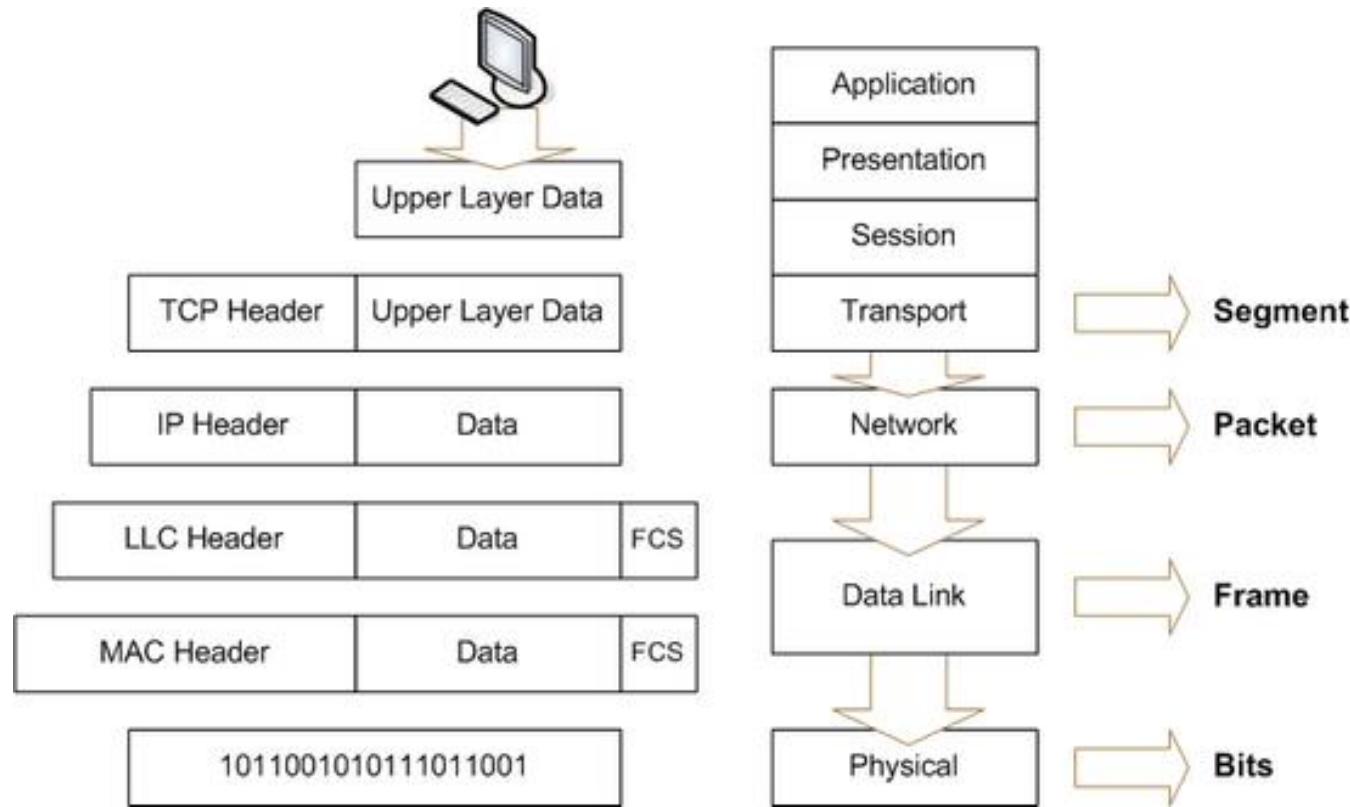


# Metadata trong một thông điệp

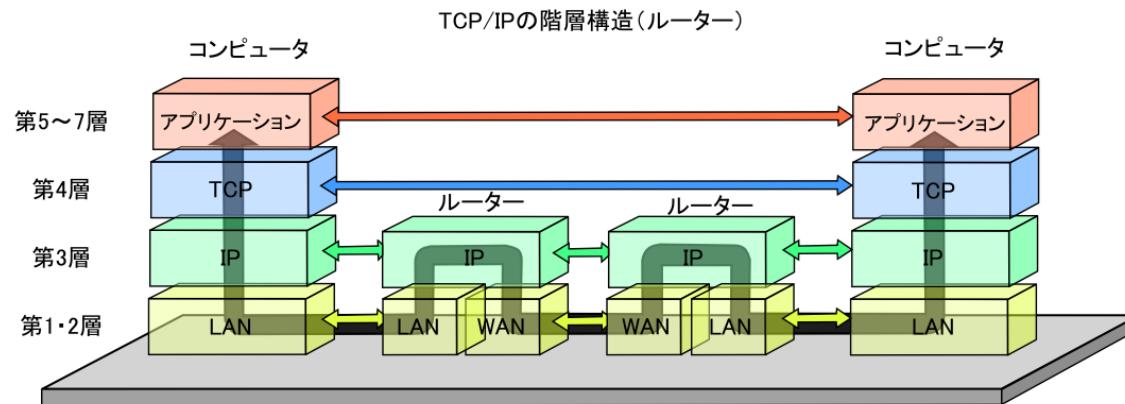
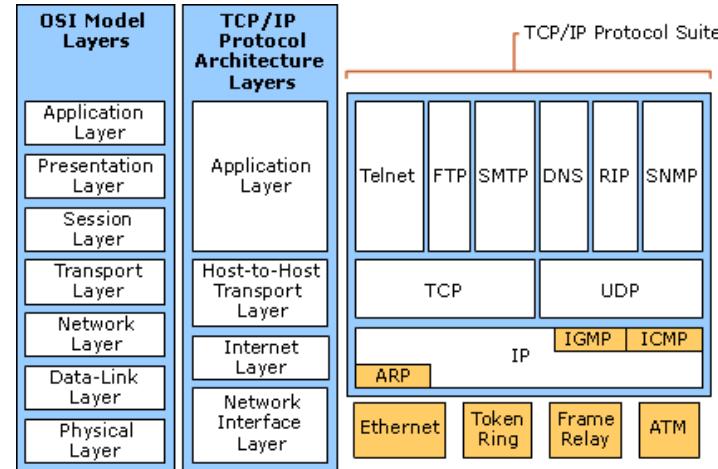




# Đóng gói dữ liệu



# Mô hình 4 tầng TCP/IP





# Mô hình phân tầng thu gọn 3 tầng

## ❖ Một số mô hình được phát triển

- Mô hình 7 tầng OSI
- Mô hình 4 tầng TCP/IP

## ❖ Xét trên phương diện lập trình

- Mô hình truyền thông đơn giản 3 tầng.
  - Tầng ứng dụng
  - Tầng giao vận
  - Tầng mạng





# Mô hình phân tầng thu gọn 3 tầng

## ❖ Các thành phần tham gia trong quá trình truyền thông

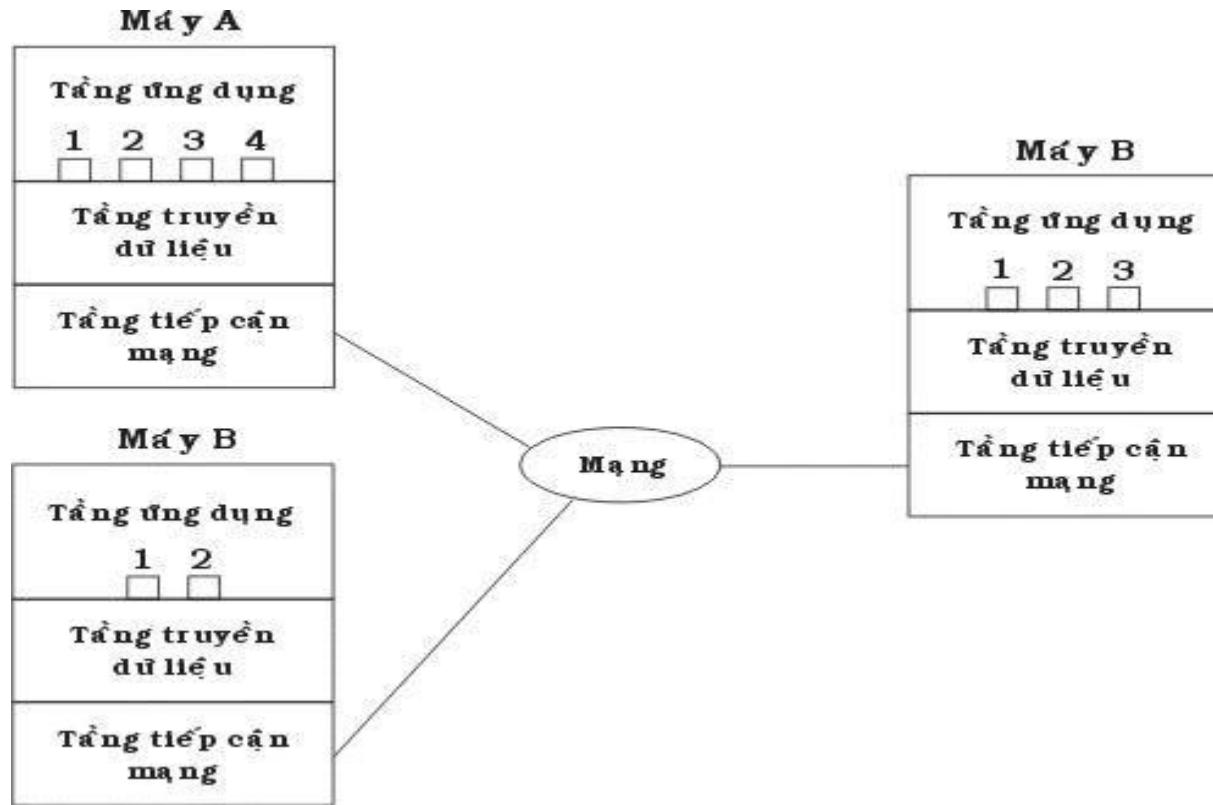
- Các chương trình ứng dụng
- Các chương trình truyền thông
- Các máy tính và các mạng

## ❖ Gửi dữ liệu các ứng dụng

- Máy tính gửi
  - Ứng dụng gửi chuyển dữ liệu cho chương trình truyền thông
  - Chương trình truyền thông sẽ gửi dữ liệu cho máy tính nhận
- Máy tính nhận
  - Chương trình truyền thông sẽ tiếp nhận và kiểm tra dữ liệu
  - Chuyển cho ứng dụng đang chờ dữ liệu



# Ví dụ mô hình truyền thông đơn giản





# Ví dụ mô hình truyền thông đơn giản

## ❖ Máy A:

- Ứng dụng 1 cần gửi một khối dữ liệu
- Dữ liệu được chuyển cho tầng giao vận
  - Chia dl thành nhiều đoạn và đóng gói thành các gói tin (packets)
  - Bổ sung thêm các thông tin điều khiển (header) vào mỗi gói tin
- Dữ liệu tiếp tục được chuyển cho tầng tiếp cận mạng và chuyển cho máy B.

## ❖ Máy B:

- Tầng tiếp cận mạng sẽ tập hợp dữ liệu và chuyển cho tầng giao vận
  - Kiểm tra và ghép dl lại thành khối (nhờ tt header)
  - Khối dữ liệu sẽ được chuyển lên cho tầng ứng dụng



# Bài 3:

# Mô hình ứng dụng

# client/server



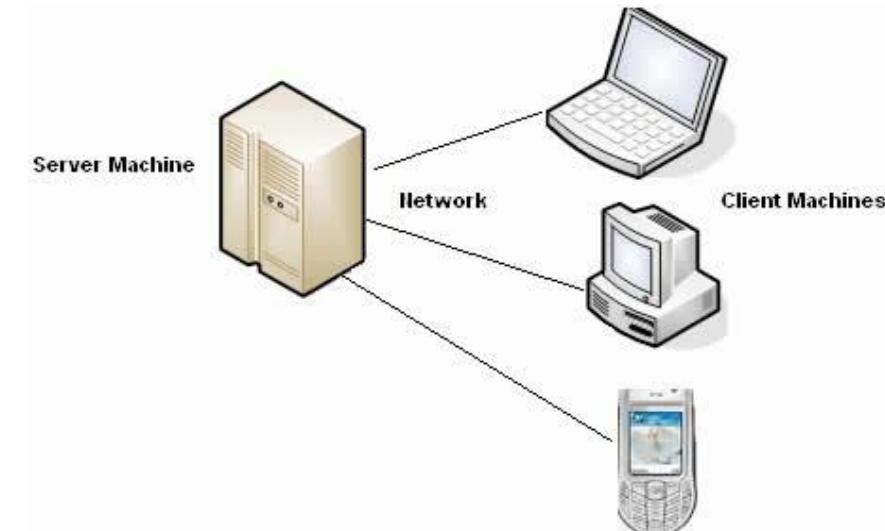
# Tổng quan

- ❖ Mô hình mạng cơ bản nhất hiện nay
- ❖ Dạng phổ biến của mô hình ứng dụng phân tán
- ❖ Đa số các ứng dụng mạng dựa theo mô hình này
- ❖ Thuật ngữ client/server xuất hiện từ đầu thập niên 80
- ❖ Ứng dụng client/server phổ biến
  - Email, FTP, Web



# Thành phần

- ❖ Một tiến trình Server
- ❖ Một hoặc nhiều tiến trình client
  - Các tiến trình clients và servers có thể chạy trên cùng trạm (host) hoặc khác trạm.
  - Là các đối tượng logic tách biệt và liên lạc với nhau qua mạng cùng thực hiện một công việc chung





# Chức năng từng thành phần

## ❖ Server

- Quản lý nguồn tài nguyên nào đó
- Cung cấp dịch vụ và phân phối tài nguyên.

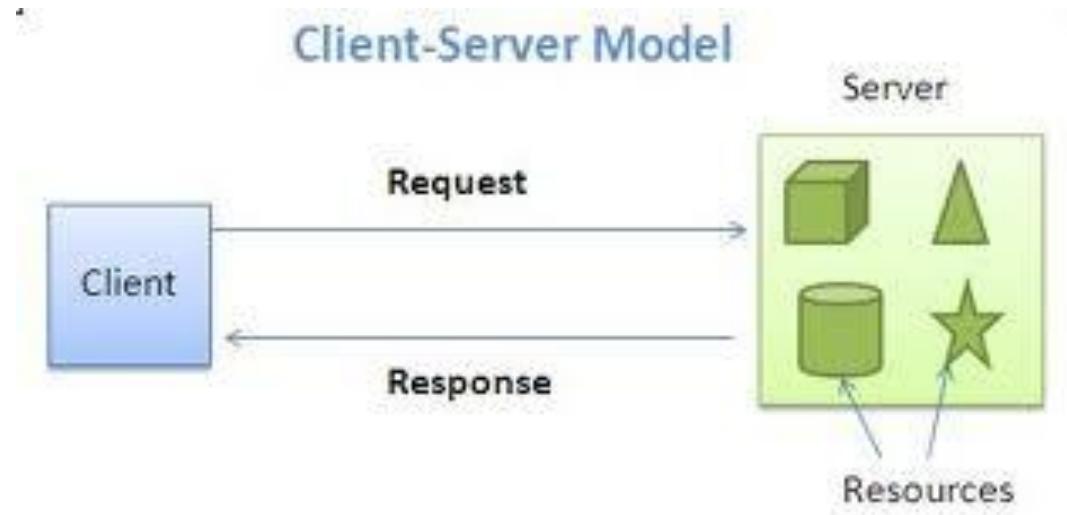
## ❖ Client

- Chương trình giao tiếp với người sử dụng
- Cần yêu cầu về tài nguyên

## ❖ Một tiến trình có thể vừa là server vừa là client



# Cách hoạt động



## ❖ Client

- Khởi tạo liên lạc với server (speaks first)
- Yêu cầu dịch vụ nào đó từ server
- Đối với Web, client được hiện thực trong browser



# Cách hoạt động

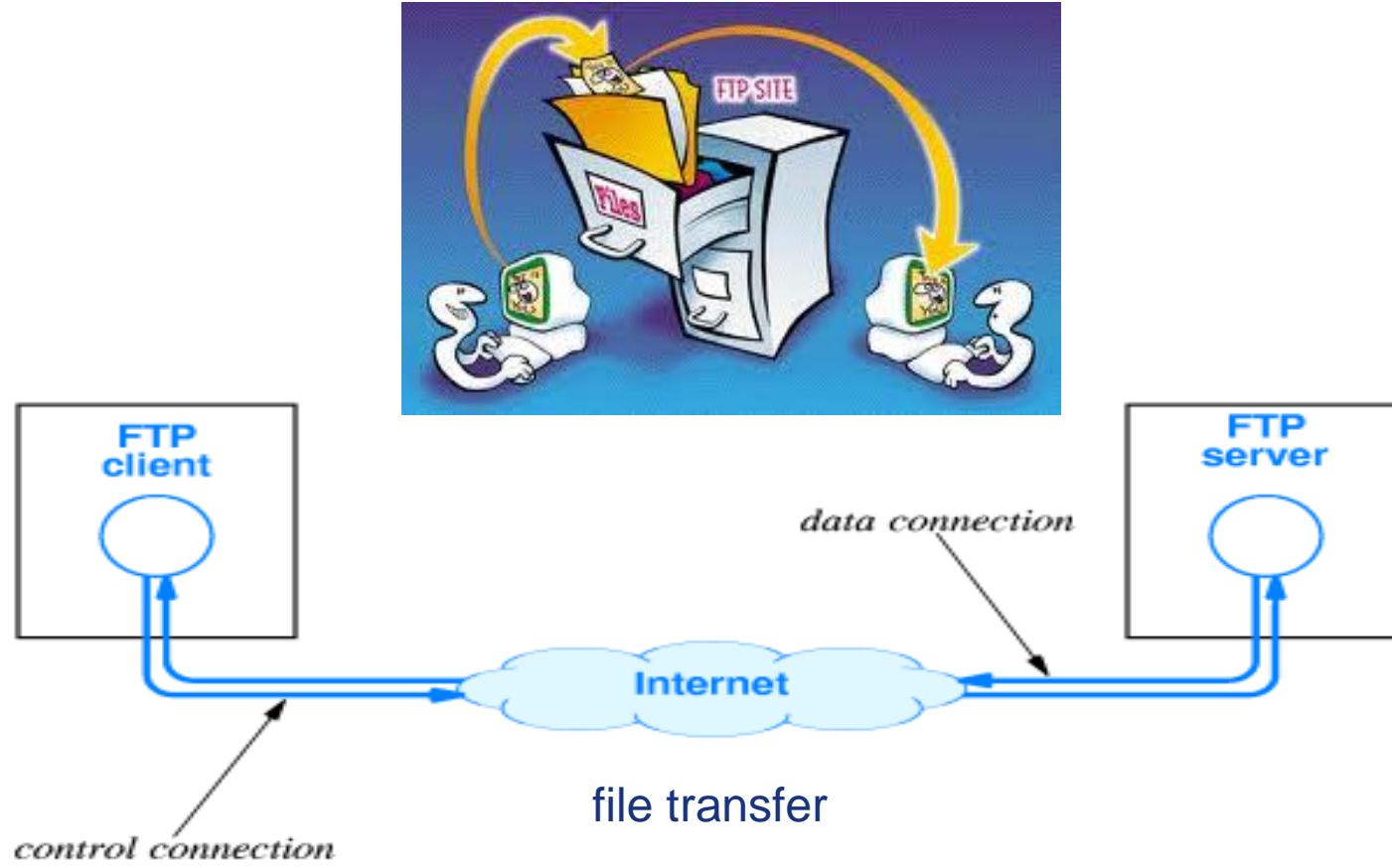


## ❖ Server

- Cung cấp dịch vụ yêu cầu cho client
- Chẳng hạn, Web server gửi Web yêu cầu hay mail server phân phát email



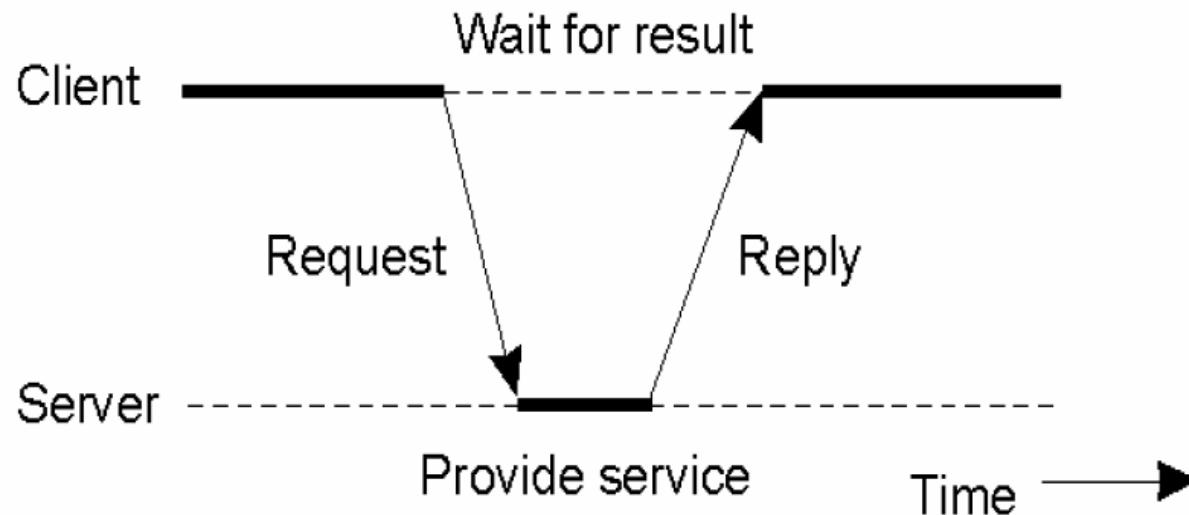
# Ví dụ FTP: The File Transfer Protocol



# Đặc trưng của mô hình ứng dụng C/S

## ❖ Hoạt động theo kiểu giao thức bất đối xứng

- Thể hiện quan hệ một chiều giữa client và một server
- Client bắt đầu phiên hội thoại bằng cách yêu cầu dịch vụ
- Server sẵn sàng chờ các yêu cầu từ client





# Đặc trưng của mô hình ứng dụng C/S

## ❖ Đóng gói dịch vụ

- Server như một chuyên gia
  - Hoàn thành tác vụ đáp ứng lại các yêu cầu từ client
- Server có thể được nâng cấp mà không ảnh hưởng đến client

## ❖ Tính toàn vẹn

- Mã và dữ liệu đối với server được bảo trì tập trung
  - Giảm chi phí và bảo vệ sự toàn vẹn của dữ liệu chung
- Client duy trì tính cá nhân và độc lập



# Ưu điểm của mô hình UD C/S

- ❖ Tính tập trung(Centralization)
  - Truy cập tài nguyên và bảo mật dữ liệu
    - Tập trung thông qua server
- ❖ Tính co giãn (Scalability)
  - Nâng cấp bất cứ thành phần nào khi cần thiết
- ❖ Tính mềm dẻo (Flexibility)
  - Công nghệ mới có thể dễ dàng tích hợp vào hệ thống
- ❖ Tính trao đổi tương tác (Interoperability)
  - Tất cả các thành phần (clients, mạng, servers) cùng nhau làm việc



# Nhược điểm của mô hình UD C/S

- ❖ Quản trị hệ thống khó khăn
  - Duy trì thông tin cấu hình luôn cập nhật và nhất quán giữa tất cả các thiết bị
- ❖ Nâng cấp phiên bản mới khó đồng bộ
- ❖ Phụ thuộc độ tin cậy của mạng
- ❖ Chi phí thiết kế, cài đặt, quản trị và bảo trì cao
- ❖ Phải giải quyết
  - Sự xung đột trong hệ thống
  - Tính tương thích của các thành phần
  - Việc cấu hình hệ thống



## Sự phân lớp trong mô hình UD C/S

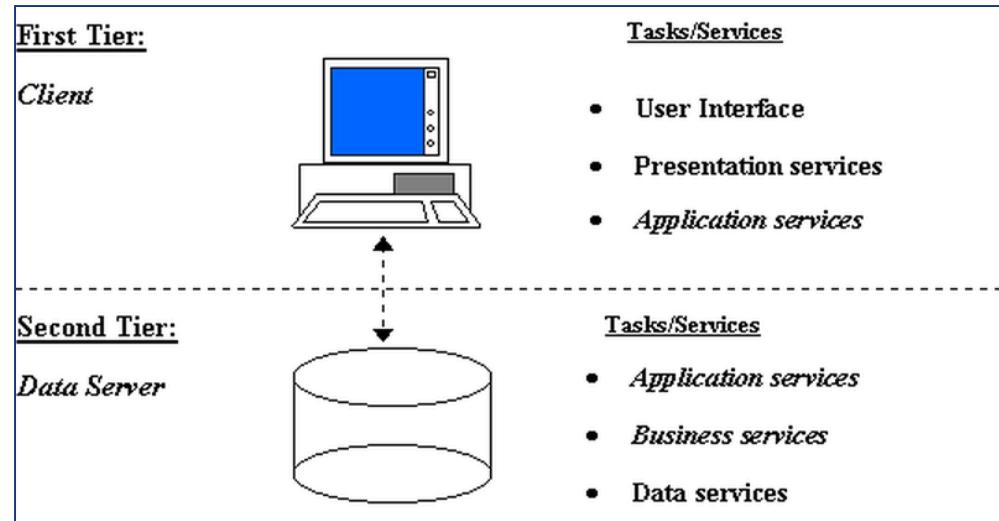
- ❖ Mọi ứng dụng mạng client/server đều có 3 khối chức năng
  - Khối biểu diễn hay giao diện người dùng
  - Khối nghiệp vụ: thuật toán điều khiển thông tin giữa khối biểu diễn và khối dữ liệu
  - Khối dữ liệu



# Kiến trúc UD C/S 2 lớp (2-tier)

## ❖ Khối nghiệp vụ (business logic)

- Được đặt bên trong lớp giao diện người dùng tại client, **hoặc**
- Được đặt bên trong CSDL

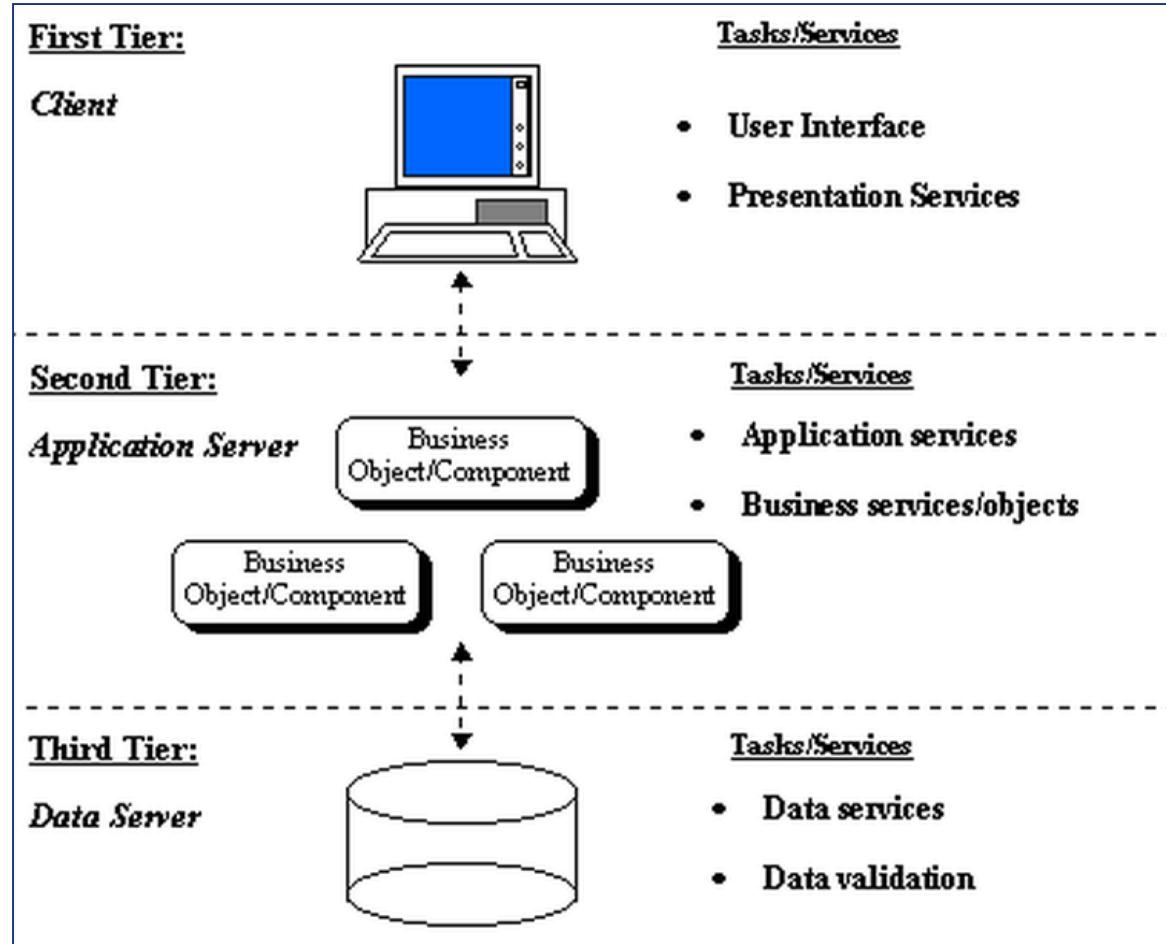




# Kiến trúc UD C/S 3 lớp (3-tier)

- ❖ Phát triển vào thập niên 90'
  - Hệ thống lớn và ổn định
- ❖ Mở rộng mô hình 2 lớp: Tách biệt khỏi nghiệp vụ
  - Nâng cao hiệu năng (performance),
  - Tính linh hoạt (flexibility),
  - Khả năng bảo trì (maintainability),
  - Khả năng dùng lại (reusability)

# Kiến trúc UD C/S 3 lớp (3-tier)





# Kiến trúc UD C/S 3 lớp (3-tier)

## ❖ Lớp trên cùng

- Chứa giao diện dịch vụ cho người dùng

## ❖ Lớp dưới cùng

- Chứa chức năng quản trị CSDL

## ❖ Lớp trung gian

- Chứa khối nghiệp vụ
- Các tiến trình xử lý
- Điều khiển các giao dịch và các hàng đợi
- Gửi các yêu cầu từ client đến csdl
- Được xem như proxy server



# Ứng dụng client/server 3 lớp

## ❖ Ứng dụng Web

- Lớp trên cùng: Web browser
- Lớp trung gian: Web server engine
- Lớp dưới cùng: Hệ CSDL

## ❖ Ứng dụng Struts hoặc JSP (trong Java)

- Lớp trên cùng: Views
- Lớp trung gian: Controllers
- Lớp dưới cùng: Models

## ❖ Kiến trúc client/server n lớp (n-tier)

- Lớp trung gian được chia thành nhiều đơn vị nhỏ



# Giao thức cho ƯD client/server

## ❖ Giao thức?

- Là tập các khuôn dạng bản tin, tập các trạng thái, quy tắc, quy ước trong truyền thông giữa client và server.

## ❖ Khi tạo một ứng dụng client/server

- Phải thiết kế giao thức
- Các giao thức phổ biến FTP, HTTP, ...
  - đã được IETF (Internet Engineering Task Force) chuẩn hóa thành các giao thức chuẩn.

## ❖ Lập trình CSDL

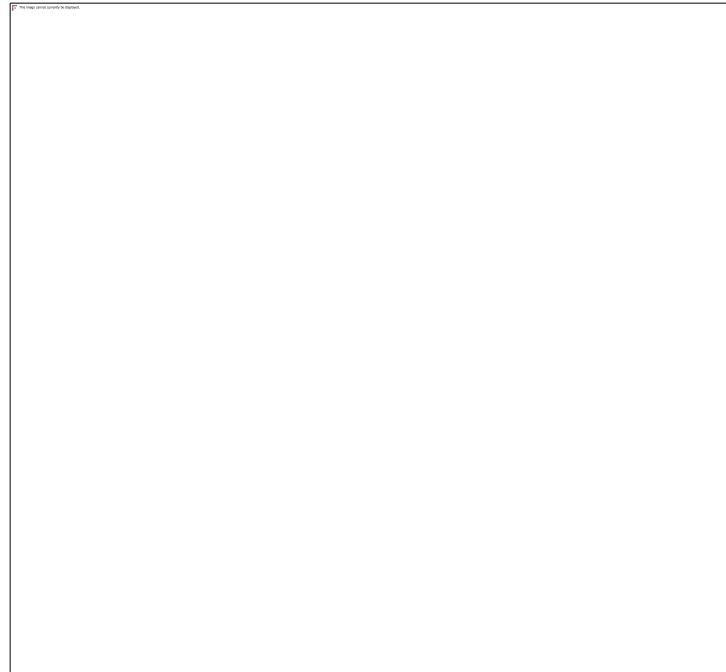
- Được hỗ trợ và quy định bởi hệ quản lý csdl và các thư viện lập trình.



# Phân loại giao thức

## ❖ Giao thức đồng bộ (Synchronous protocol)

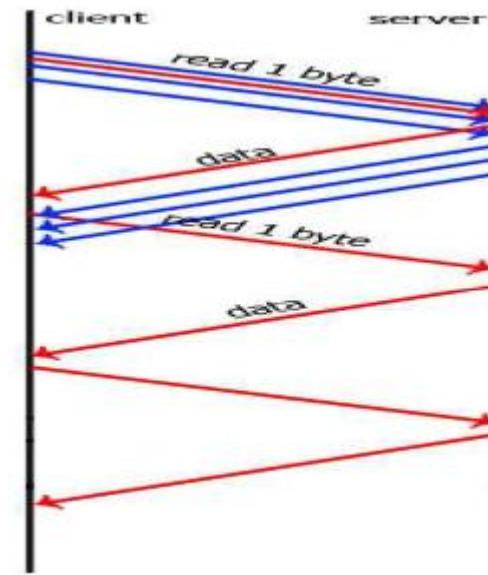
- Truyền thông giữa client và server diễn ra theo 2 chiều nhưng không đồng thời
- Được thực hiện lần lượt
- Các giao thức kiểu này là:
  - HTTP SMTP, POP3





# Phân loại giao thức

- ❖ Giao thức không đồng bộ (Asynchronous protocol)
  - Client và server có thể đồng thời gửi thông tin
  - Các giao thức này như TELNET, RLOGIN,...
  - Ngoài ra, còn có loại giao thức hybrid kết hợp giữa các giao thức trên





# Lập trình mạng trên Java



# LẬP TRÌNH MẠNG TRÊN JAVA

## ❖ Gói java.net

- InetAddress
- ServerSocket
- Socket
- URL
- URLConnection
- DatagramSocket

# LẬP TRÌNH MẠNG TRÊN JAVA

## ❖ InetAddress class

- Class mô tả về địa chỉ IP (Internet Protocol)
- Các phương thức getLocalHost, getByName, hay getAllByName để tạo một InetAddress instance:
  - *public static InetAddress InetAddress.getByName(String hostname)*
  - *public static InetAddress [] InetAddress.getAllByName(String hostname)*
  - *public static InetAddress InetAddress.getLocalHost()*
- Để lấy địa chỉ IP hay tên dùng các phương thức:
  - *getHostAddress()*
  - *getHostName()*



## Ví dụ 1

### ❖ In địa chỉ IP của localhost

```
import java.net.*;
public class HostInfo {
    public static void main(String args[]) {
        try {
            InetAddress myHost = InetAddress.getLocalHost();
            System.out.println(myHost.getHostAddress());
            System.out.println(myHost.getHostName());
        } catch (UnknownHostException ex) {
            System.err.println("Cannot find local host");
        }
    }
}
```



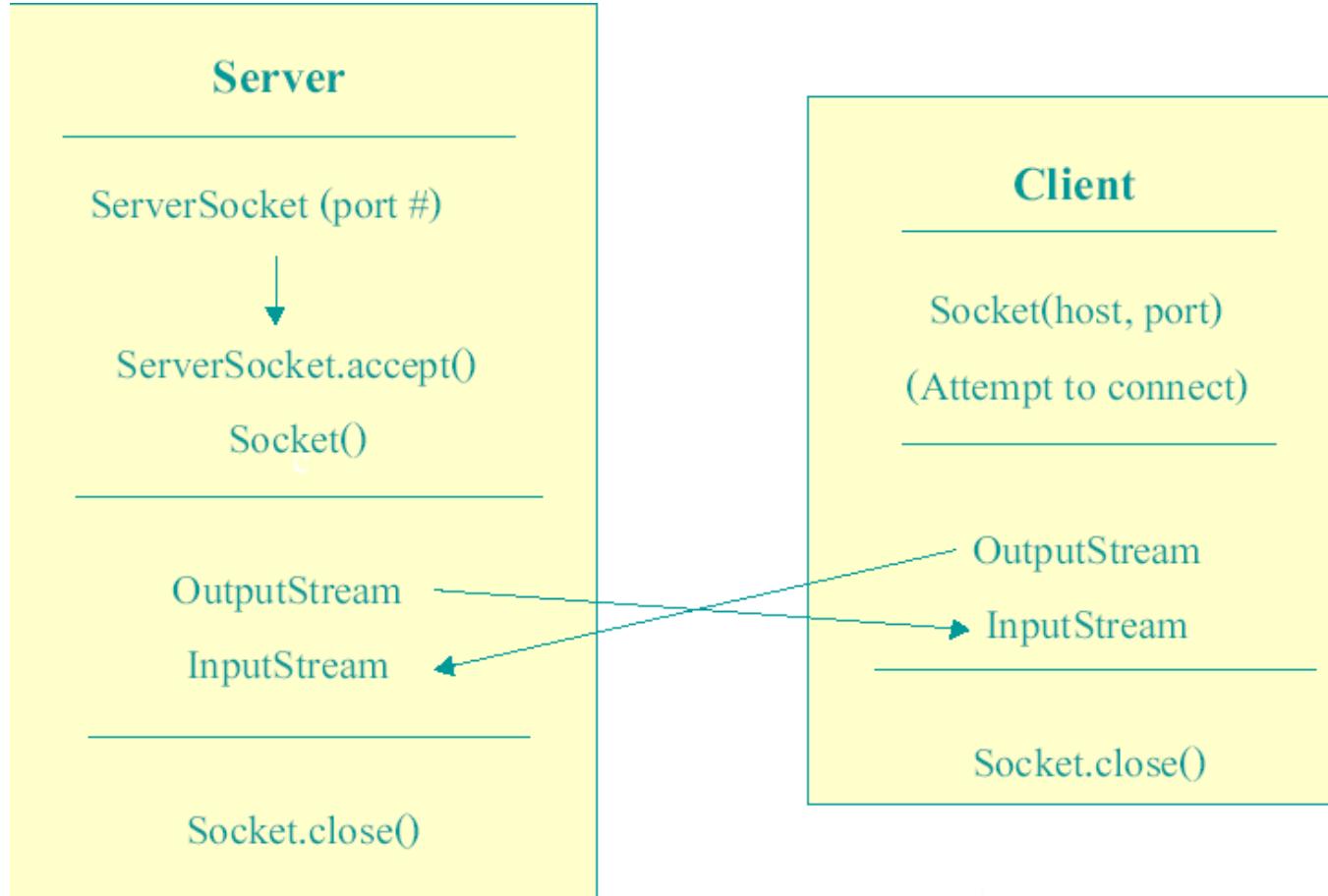
## Ví dụ 2

### ❖ In địa chỉ IP của một địa chỉ bất kỳ

```
import java.net.*;
public class IPAddresses{
    public static void main (String args[]) {
        try {
            InetAddress[] addresses =
            InetAddress.getAllByName(args[0]);
            for (int i = 0; i < addresses.length; i++)
                System.out.println(addresses[i]);
        }
        catch (UnknownHostException e) {
            System.out.println("Could not find" + args[0]);
        }
    }
}
```



# LẬP TRÌNH MẠNG TRÊN JAVA





# LẬP TRÌNH MẠNG TRÊN JAVA

## ❖ **Socket class**

- Class mô tả về *socket*
- Tạo một *socket*
  - **Socket(InetAddress address, int port)**
  - **Socket(String host, int port)**
  - **Socket(InetAddress address, int port, InetAddress, localAddr, int localPort)**
  - **Socket(String host, int port, InetAddress, localAddr, int localPort)**
  - **Socket()**



# LẬP TRÌNH MẠNG TRÊN JAVA

## ❖ **Socket class (tiếp theo)**

- **Lấy thông tin về một socket**
  - InetAddress **getInetAddress()** : trả về địa chỉ mà socket kết nối đến.
  - int **getPort()** : trả về địa chỉ mà socket kết nối đến.
  - InetAddress **getLocalAddress()** : trả về địa chỉ cục bộ.
  - int **getLocalPort()** : trả về địa chỉ cục bộ.
- **Sử dụng Streams**
  - public OutputStream **getOutputStream()** throws IOException  
Trả về một output stream cho việc viết các byte đến socket này.
  - public InputStream **getInputStream()** throws IOException  
Trả về một input stream cho việc đọc các byte từ socket này.



# LẬP TRÌNH MẠNG TRÊN JAVA

## ❖ Kết nối đến 1 số webserver

```
import java.net.*;
import java.io.*;
public class getSocketInfo {
    public static void main(String[] args) {
        for (int i = 0; i < args.length; i++) {
            try {
                Socket theSocket = new Socket(args[i], 80);
                System.out.println("Connected to " +
                theSocket.getInetAddress() +
                " on port " + theSocket.getPort() + " from port " +
                theSocket.getLocalPort() + " of " +
                theSocket.getLocalAddress());
            }
        }
    }
}
```



# Tiếp theo

```
        } catch (UnknownHostException e) {
            System.err.println("I can't find " + args[i]);
        } catch (SocketException e) {
            System.err.println("Could not connect to " + args[i]);
        } catch (IOException e) {
            System.err.println(e);
        }
    } // end for
} // end main
} // end getSocketInfos
```

# ServerSocket class

- ❖ Class mô tả về *ServerSocket*
- ❖ Tạo một *ServerSocket*
  - **ServerSocket(int port)** throws IOException
  - **ServerSocket(int port, int backlog)** throws IOException
  - **ServerSocket(int port, int backlog, InetAddress bindAddr)** throws IOException

# ServerSocket class (tt)

- ❖ Các phương thức trong *ServerSocket*
  - Socket **accept()** throws IOException: Chấp nhận và lắng nghe một kết nối đến Socket này.
  - void **close()** throws IOException: Đóng socket
  - InetAddress **getInetAddress()**: Trả về địa chỉ cục bộ của Socket
  - int **getLocalPort()**: Trả về Local port
  - void **setSoTimeout(int timeout)** throws SocketException

# Ví dụ DateTime Server

```
import java.net.*;
import java.io.*;
import java.util.Date;
public class DayTimeServer {
    public final static int daytimePort = 5000;
    public static void main(String[] args) {
        ServerSocket theServer;
        Socket theConnection;
        try {
            theServer = new ServerSocket(daytimePort);
        (tt)
```

# Ví dụ DateTime Server

```
while (true) {  
    theConnection = theServer.accept();  
    DataOutputStream dos = new DataOutputStream(  
        theConnection.getOutputStream());  
    String time = new Date().toString();  
    dos.writeUTF(time);  
    theConnection.close();  
    theServer.close();  
}  
}catch (IOException e) {  
    System.err.println(e);  
}  
}  
}  
}
```



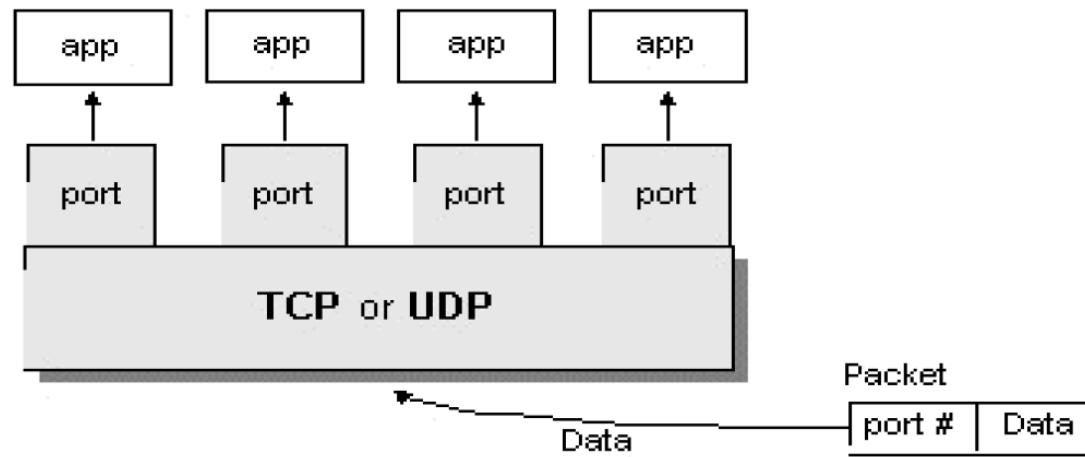
# Lập trình với giao thức TCP

# Giao thức TCP

- ❖ Là giao thức hướng kết nối(connection-oriented)
  - Truyền thông tin cậy
  - Thiết lập kênh kết nối
- ❖ Cung cấp một kênh point-to-point cho các ứng dụng yêu cầu truyền thông tin cậy
- ❖ Các giao thức sau là giao thức hướng kết nối:
  - HTTP
  - FTP
  - Telnet

# Cổng (Port)

- ❖ Là một số (nhân) đặc biệt
  - Được gán cho một tiến trình mạng
- ❖ Mỗi tiến trình mạng đều được gán một cổng duy nhất



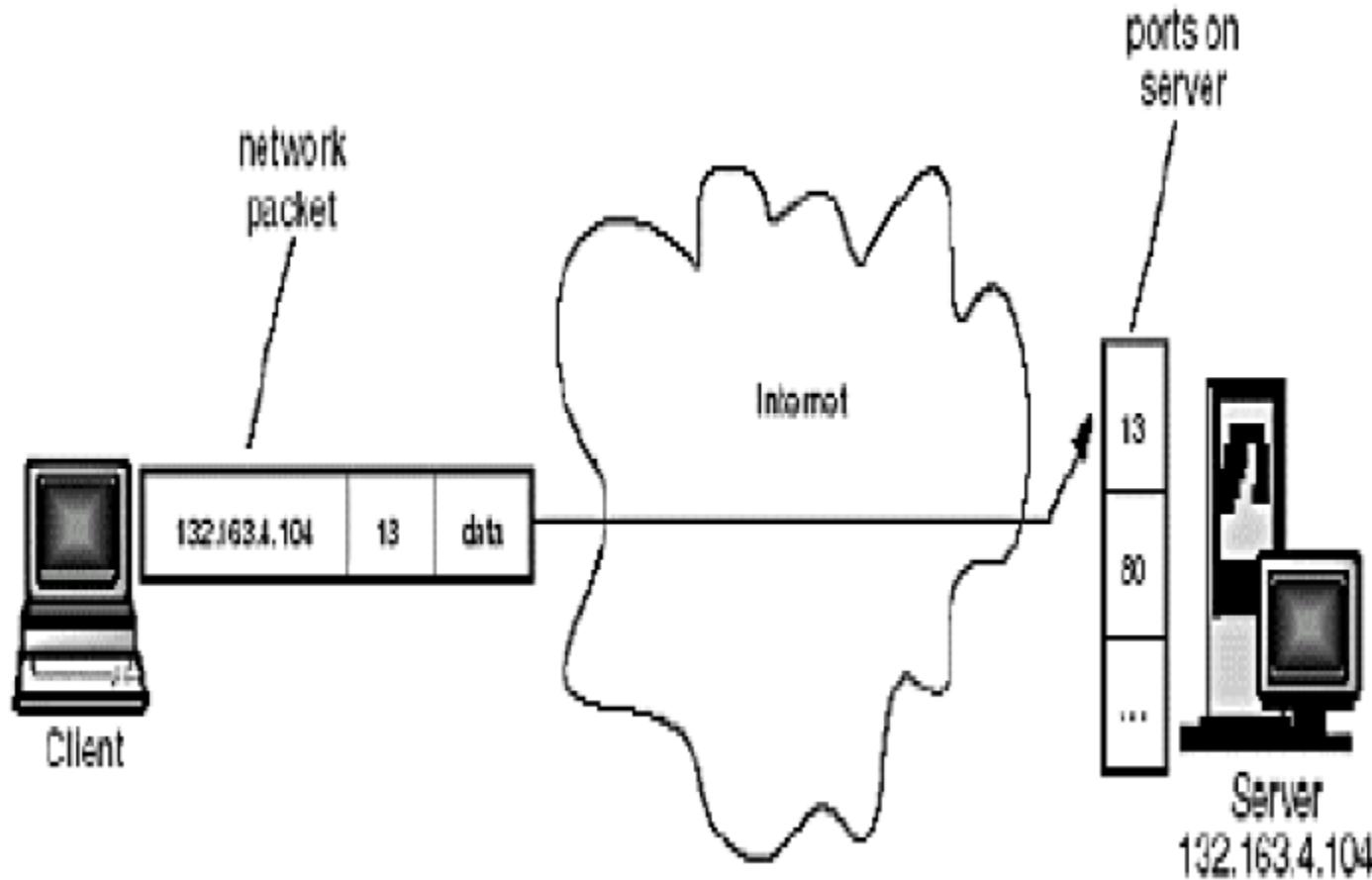


# Cổng (Port)

- ❖ Số cổng là số 2 bytes
  - Các số 0-1023: dành cho các ứng dụng thông dụng
  - Các số 1024-65535: là các cổng động
- ❖ Các server thường sử dụng các cổng nổi tiếng
  - Mục đích bất cứ client có thể nhận biết dễ dàng server/service
  - HTTP=80, FTP=21, Telnet=23,...
- ❖ Client thường dùng các cổng động
  - Gán ở thời điểm chạy chương trình

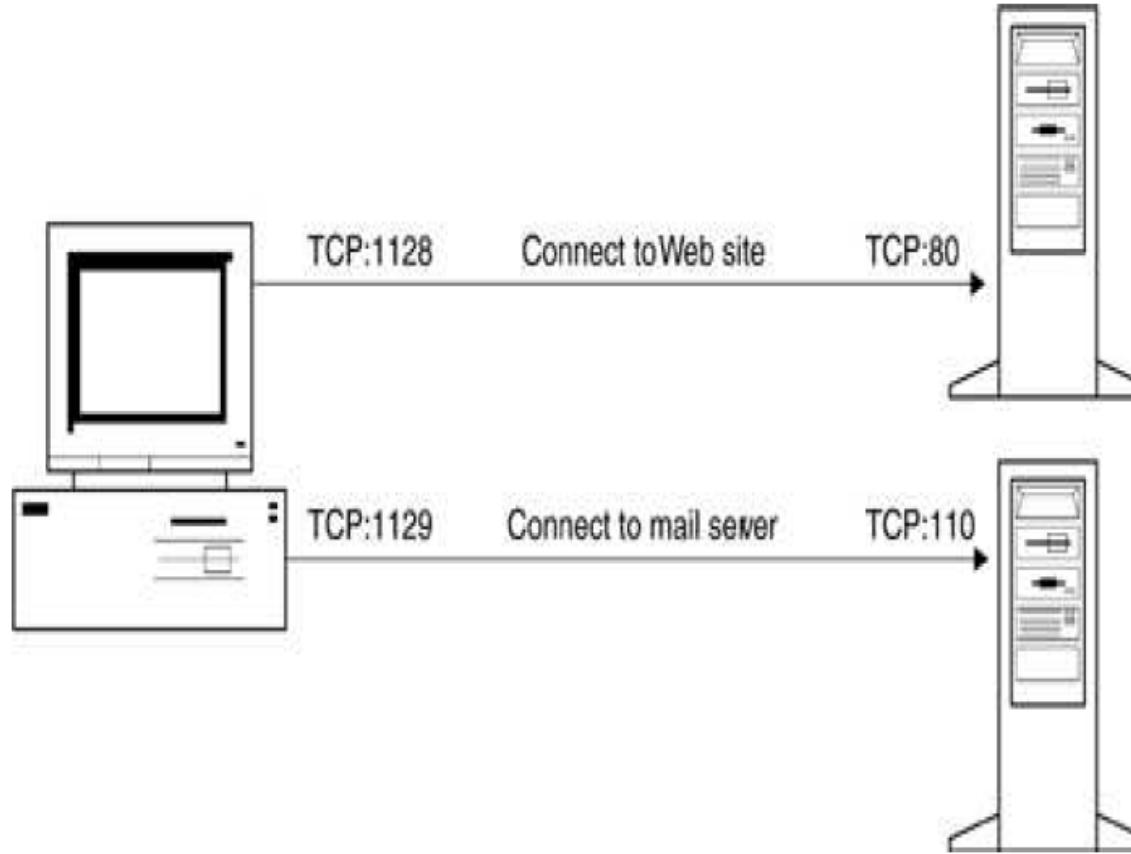


# Cổng (Port)





# 1 ƯD sử dụng nhiều cổng khác nhau

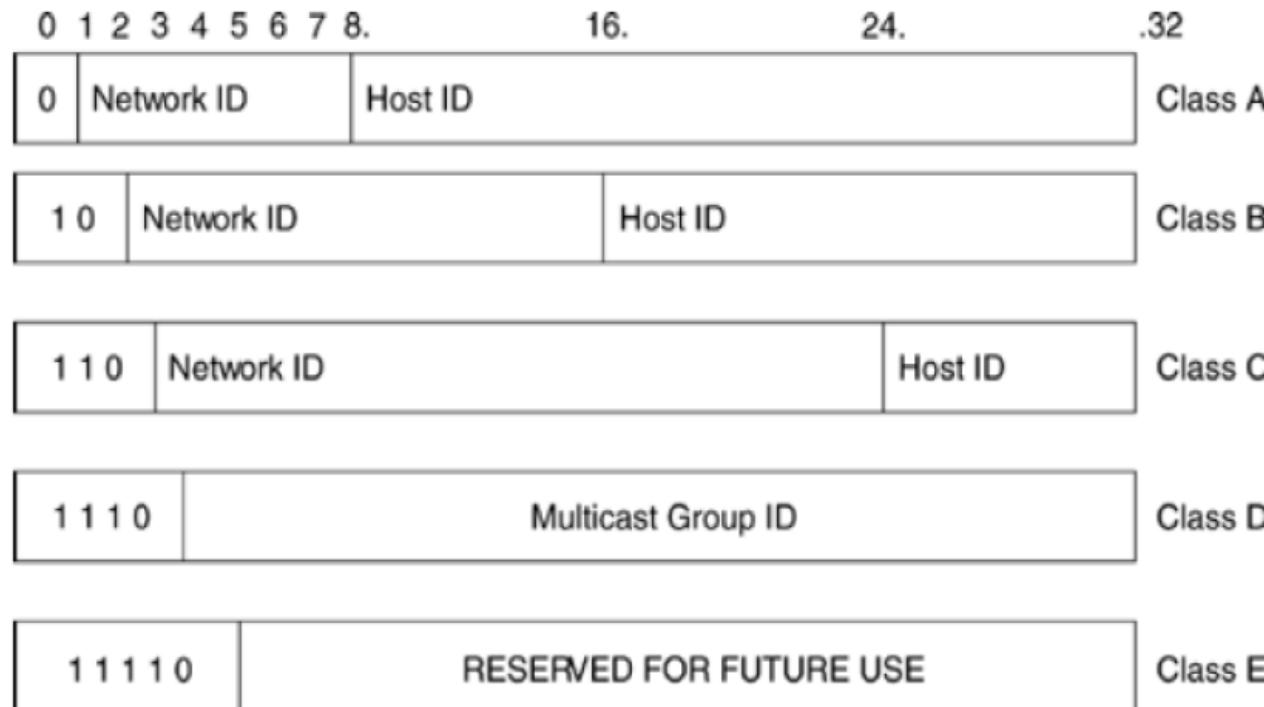


# Một số cổng tiếng (Well-known)

Port	Protocol	RFC
13	DayTime	RFC 867
7	Echo	RFC 862
25	SMTP (e-mail)	RFC 821 (SMTP) RFC 1869 (Extnd SMTP)
		RFC 822 (Mail Format)
		RFC 1521 (MIME)
110	Post Office Protocol	RFC 1725
20	File Transfer Protocol (data)	RFC 959
80	Hypertext Transfer Protocol	RFC 2616

# Địa chỉ IP (Addresses)

- ❖ Mỗi thực thể trên mạng chỉ có một địa chỉ IP duy nhất
  - IPv4: 32 bits, tạo thành từ 4 octets. Vd: 192.169.12.1
  - Địa chỉ IP được chia thành các lớp sau





# Dãy địa chỉ IP theo lớp

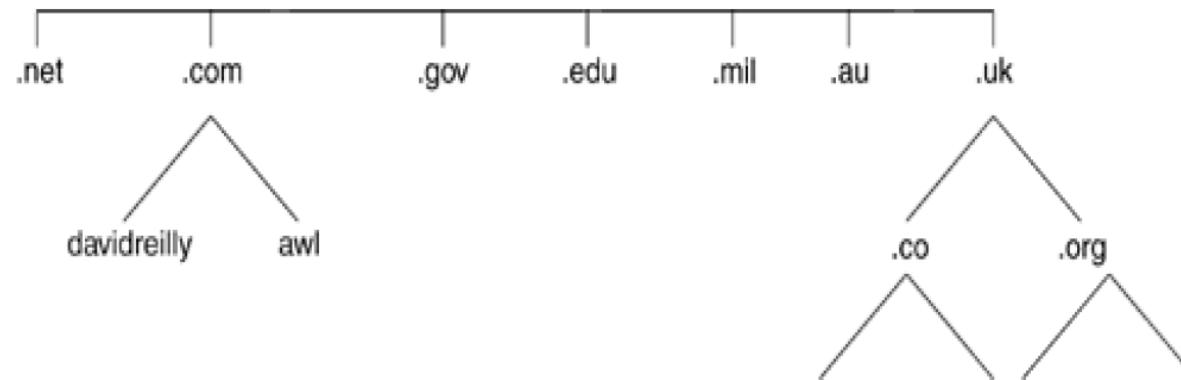
- ❖ Lớp A: 0. 0. 0. 0 – 127.255.255.255
- ❖ Lớp B: 128.0.0.0 – 191.255.255.255
- ❖ Lớp C: 192.0.0.0 – 223.255.255.255
- ❖ Lớp D: 224.0.0.0 – 239.255.255.255
- ❖ Lớp E: 240.0.0.0 – 247.255.255.255

Địa chỉ 127.0.0.1 chỉ địa chỉ IP máy cục bộ



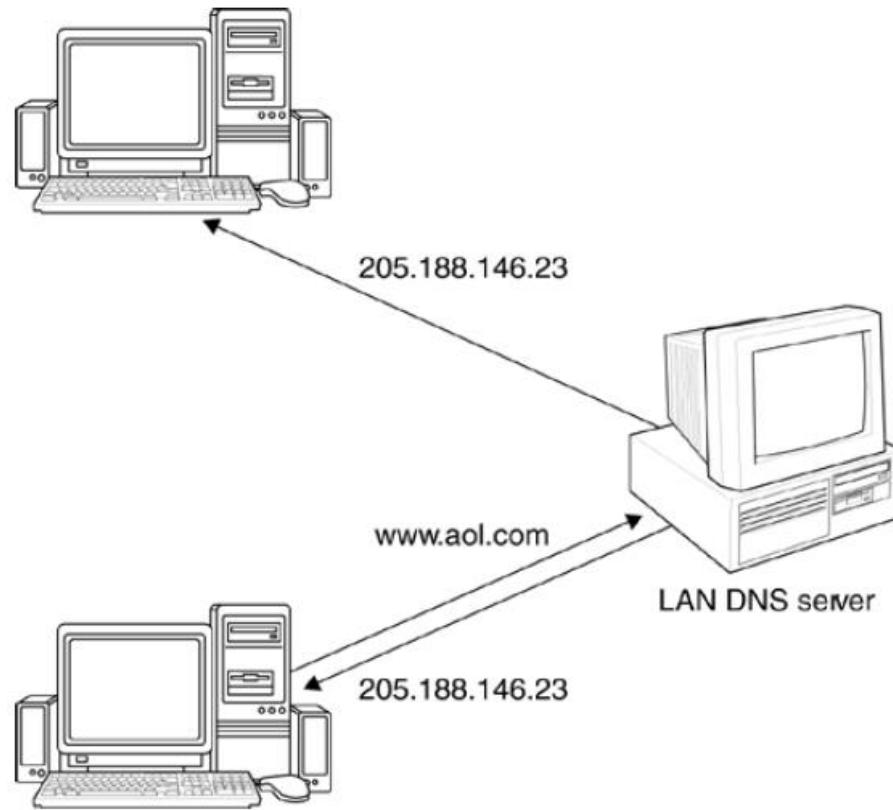
# Tên miền (Domain Name)

- ❖ Chúng ta thường khó nhớ một số dài hơn là một tên
  - DNS(Domain Name Server) cung cấp ánh xạ địa chỉ sang tên
  - Ví dụ ftp.davidreilly, www.davidreilly.com
  - Tên miền cũng được nhóm theo các miền con sau





# Ánh xạ tên miền





# Socket?

- ❖ Một ứng dụng trên mạng được xác định thông qua
  - Địa chỉ IP duy nhất mà nó chạy trên một hệ thống
  - Số hiệu cổng riêng được gán cho nó
- ❖ 2 ứng dụng mạng liên lạc được với nhau cần phải thiết lập kết nối (connection)
  - Mỗi đầu kết nối tương ứng với một Socket



# Socket?

## ❖ VẬY

- Một socket là một đầu cuối của một sự truyền thông 2 chiều, liên kết giữa hai chương trình chạy trên mạng.
- Một socket được gán với một số hiệu cổng(port), vì thế tầng giao vận có thể nhận biết ứng dụng mà dữ liệu được chuyển đến



# Các hoạt động của Socket

- ❖ Kết nối đến máy ở xa
- ❖ Gửi dữ liệu
- ❖ Nhận dữ liệu
- ❖ Đóng kết nối
- ❖ Gắn với một cổng
- ❖ Lắng nghe dữ liệu đến
- ❖ Chấp nhận kết nối từ máy ở xa trên cổng được gán



# Các kiểu socket

## ❖ Có 3 kiểu socket

- Kiểu 1: Stream sockets, tương tự như điện thoại kết nối đến một tổng đài. Gọi là: kiểu hướng kết nối
  - VD: TCP(Transmission Control Protocol) socket
  - Đảm bảo dữ liệu đến đích an toàn và đầy đủ
- Kiểu 2: Datagram sockets, tương tự như mailbox. Gọi là: kiểu phi kết nối (không giữ kênh kết nối trong quá trình truyền thông)
  - VD: UDP(User Datagram Protocol) socket
- Kiểu 3: Raw sockets



# Các bước tạo một TCP

## ❖ Phía Server

- Gán một cổng với Socket
- Chờ và lắng nghe yêu cầu kết nối từ client
- Chấp nhận kết nối, tạo Socket tương ứng
- Truyền nhận dữ liệu thông qua các streams in/out của đối tượng Socket
- Đóng kết nối



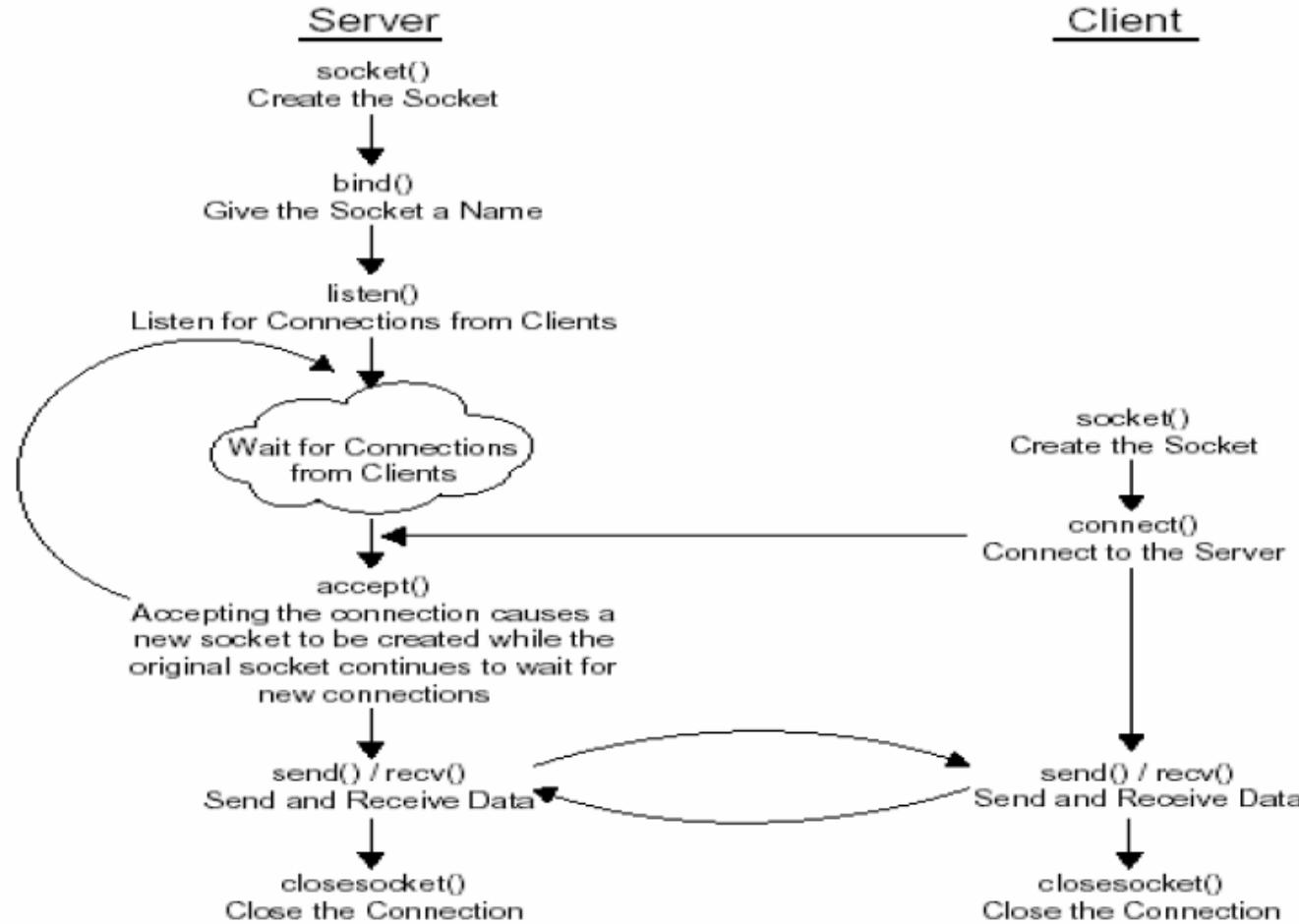
# Các bước tạo một TCP

## ❖ Phía Client

- Tạo một TCP socket với địa chỉ IP và số cổng mà chương trình Server đang chạy
- Thiết lập kết nối đến Server
- Trao đổi dữ liệu với Server
- Đóng kết nối

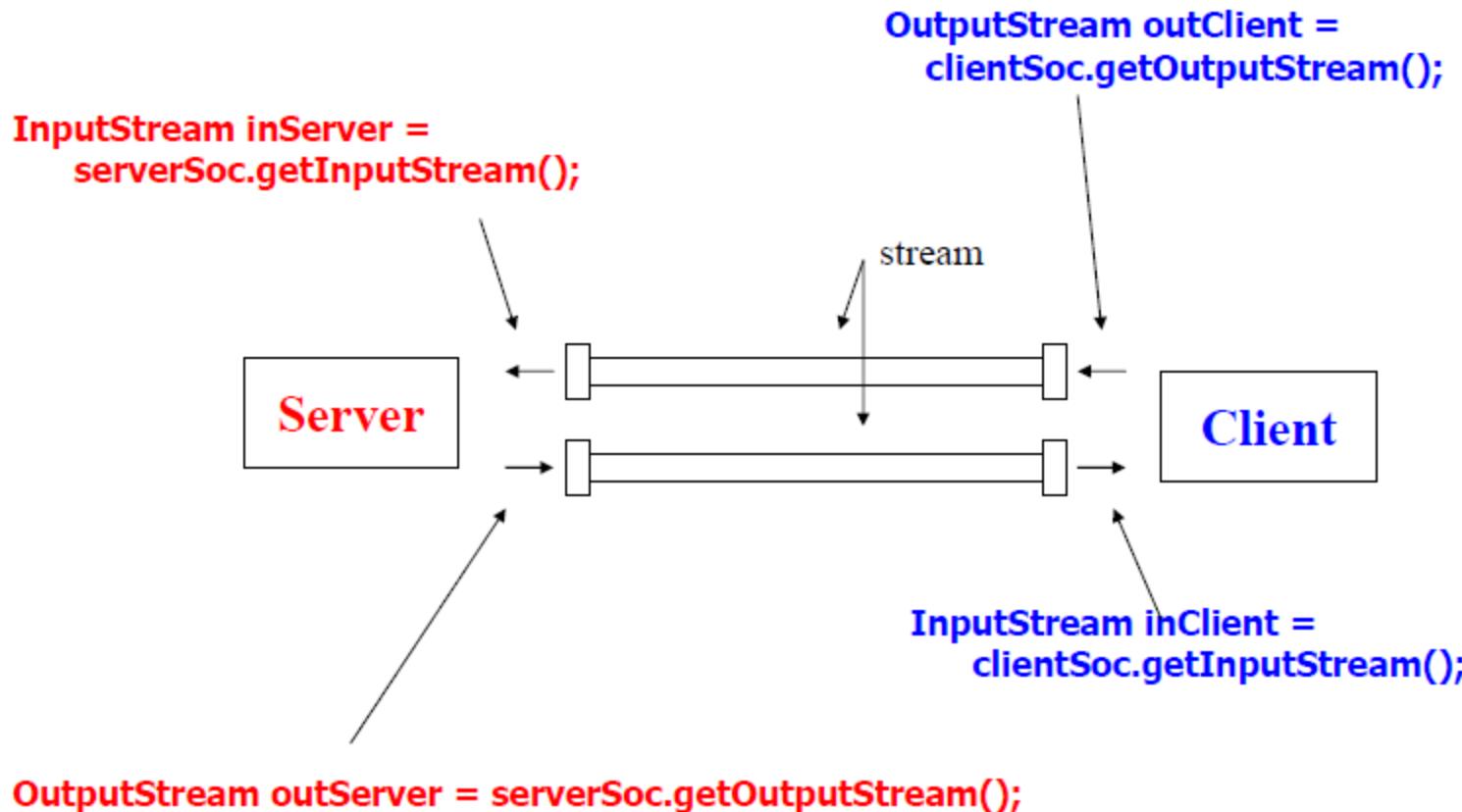


# Các bước tạo một TCP





# Mô tả quá trình trao đổi dữ liệu





# Ví dụ DateTime Server

```
public class DayTimeServer {  
    public final static int daytimePort = 5000;  
    public static void main(String[] args) {  
        ServerSocket theServer;  
        try {  
            theServer = new ServerSocket(daytimePort);  
            while (true) {  
                Socket theConnection = theServer.accept();  
                DataOutputStream dos = new DataOutputStream(  
                    theConnection.getOutputStream());  
                String time = new Date().toString();  
                dos.writeUTF(time);  
                theConnection.close();  
            }  
        }catch (IOException e) {}  
    }  
}
```



# DateTime Client

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.Socket;
public class TimeClient {
    public static void main(String[] args) throws Exception {
        Socket socket = new Socket("localhost", 5000);
        DataInputStream din = new
        DataInputStream(socket.getInputStream());
        String time = din.readUTF();
        System.out.println(time);
    }
}
```



# Threaded Server

- ❖ Ví dụ vừa rồi chỉ đúng với một client kết nối đến server tại một thời điểm
- ❖ Để cho phép nhiều client có thể kết nối đến server cùng lúc thì server phải là chương trình đa tuyến
  - Mỗi tuyến ứng với một kết nối từ client.



# Threaded Server

- Đoạn mã chương trình bằng Java

```
while(true){  
    Socket socket = s.accept( );  
    new ThreadedTimeHandler(socket, i).start();  
    i++;  
}
```

```
class ThreadedTimeHandler extends Thread {  
    Socket incoming;  
    int counter;  
    public ThreadedTimeHandler(Socket i, int c)  
    {  
        incoming = i; counter = c;  
    }  
    public void run() { ....  
        // đoạn mã truyền nhận dữ liệu ở đây  
    }  
}
```





# Bài tập

## ■ Chat Room

- ❑ Lập trình một chương trình chat room sử dụng TCP socket.
- ❑ Client có giao diện đơn giản hay phức tạp tùy vào bạn.
- ❑ Chat Server là một server có khả năng quản lý nhiều clients của chat room.
- ❑ Mỗi thông điệp từ một client gửi đến server, server phải có nhiệm vụ gửi đến tất cả các client còn lại, và tất cả client đều hiển thị thông điệp đó lên màn hình.
- ❑ Server nên dùng một Vector để lưu trữ tất cả các tiến trình clients. Nhờ Vector này mà nó có thể quản lý và truyền thông điệp đến tất cả các clients trong chat room.
- ❑ That's all. Good luck!.



# Lập trình với UDP Socket



# Giao thức UDP

- ❖ UDP (User Datagram Protocol) là giao thức mà cho phép gửi các gói(packets) dữ liệu độc lập, gọi là datagrams,
  - từ một máy tính đến một máy tính khác
  - Không đảm bảo toàn vẹn dữ liệu.
- ❖ UDP là giao thức phi kết nối(tức không giữ kenh trong quá trình truyền)



# Giao thức UDP

- ❖ UDP không cung cấp cơ chế báo nhận
- ❖ UDP không sắp xếp tuần tự các gói tin (datagram) đến và có thể dẫn đến tình trạng mất hoặc trùng dữ liệu mà không có cơ chế báo lỗi cho người gửi
  - Gửi datagrams tương tự như gửi 1 bưu kiện thông qua dịch vụ bưu điện: thứ tự phát gói tin không quan trọng và không đảm bảo.
  - Mỗi thông điệp độc lập với các thông điệp khác.



# Ứng dụng phổ biến của UDP

- ❖ DNS (Domain Name System),
- ❖ Ứng dụng streaming media
- ❖ Voice over IP
- ❖ Trivial File Transfer Protocol (TFTP)
- ❖ Game trực tuyến



# Khuôn dạng UDP datagrams

- ❖ UDP datagrams có tham số đơn giản hơn nhiều so với TCP

+	Bits 0 – 15	16 – 31
0	Source Port	Destination Port
32	Length	Checksum
64		Data



# Header của IP trong TCP

+	Bít 0 – 3	4 – 7	8 – 9	10 – 15	16 – 31			
0	Source address							
32	Destination address							
64	Zeros		Protocol		TCP length			
96	Source Port		Destination Port					
128	Sequence Number							
160	Acknowledgement Number							
192	Data Offset	Reserved	Flags	Window				
225	Checksum			Urgent Pointer				
257	Options (optional)							
257/289+	Data							

# Khuôn dạng UDP datagrams

## ❖ Source port

- Trường này xác định cổng của người gửi thông tin và có ý nghĩa nếu muốn nhận thông tin phản hồi từ người nhận.

## ❖ Destination port: Trường xác định cổng nhận thông tin.

## ❖ Length: Chiều dài của toàn bộ datagram

## ❖ Checksum

- Trường checksum 16 bit dùng cho việc kiểm tra lỗi của phần header và dữ liệu.

+	Bits 0 – 15	16 – 31
0	Source Port	Destination Port
32	Length	Checksum
64		Data



## Tính năng của UDP

- ❖ UDP cũng cung cấp cơ chế gán và quản lý các số hiệu cổng.
- ❖ Do ít chức năng phức tạp nên UDP thường có xu thế hoạt động nhanh hơn so với TCP
- ❖ UDP thường dùng cho các ứng dụng không đòi hỏi độ tin cậy cao trong khi truyền



# Ưu nhược điểm của UDP

Các đặc trưng	UDP	TCP
Hướng liên kết	Không	Có
Sử dụng phiên	Không	Có
Độ tin cậy	Không	Có
Xác thực	Không	Có
Đánh thứ tự	Không	Có
Điều khiển luồng	Không	Có
Bảo mật	Ít	Nhiều hơn

# Khi nào sử dụng UDP

- ❖ Rất nhiều ứng dụng trên Internet sử dụng UDP. Dựa trên các ưu và nhược điểm của UDP chúng ta có thể kết luận UDP có ích khi:
  - Sử dụng cho các phương thức truyền broadcasting và multicasting khi chúng ta muốn truyền tin với nhiều host.
  - Kích thước datagram nhỏ
  - Không cần thiết lập liên kết
  - Không cần truyền lại các gói tin
  - Ứng dụng không gửi các dữ liệu quan trọng
  - Băng thông của mạng đóng vai trò quan trọng



# Các bước tạo ứng dụng UDP socket

❖ Ứng dụng UDP socket không thiết lập kết nối như TCP

- Client

- Client
  - Socket, tạo một điểm cuối truyền thông phía client
  - Truyền và nhận dữ liệu

- Server

- Server
  - Socket, tạo một điểm cuối truyền thông phía server
  - Gắn một cổng đến kết nối
  - Truyền nhận dữ liệu



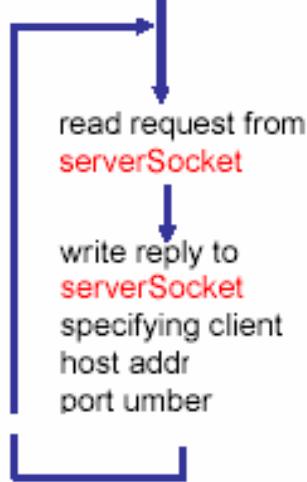
## Lưu ý

- ❖ Client không thiết lập kết nối đến server
  - Kết nối không cần thiết
- ❖ Server không chấp nhận kết nối
  - Chờ và lắng nghe không cần thiết
  - Không tồn tại chấp nhận kết nối

# Lập trình Socket với UDP

## Server (running on hostid)

```
create socket,  
port=x, for  
incoming request:  
serverSocket =  
DatagramSocket()
```



## Client

```
create socket,  
clientSocket =  
DatagramSocket()
```

```
Create, address (hostid, port=x,  
send datagram request  
using clientSocket
```

```
read reply from  
clientSocket  
close  
clientSocket
```

# Ví dụ chương trình Java: Client

```
import java.util.*;
import java.net.*;
public class UDPClient{
    public static void main(String args[]) throws Exception{
        Scanner input=new Scanner(System.in);
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress IPAddress = InetAddress.getByName("localhost");
        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];

        String sentence = input.nextLine();
        sendData = sentence.getBytes();
        DatagramPacket sendPacket = new DatagramPacket(sendData,
                                                       sendData.length, IPAddress, 8876);
        clientSocket.send(sendPacket);
        DatagramPacket receivePacket = new DatagramPacket(receiveData,
                                                       sentence.length());
        clientSocket.receive(receivePacket);
        String modifiedsentence = new String(receivePacket.getData());
        System.out.println("From Server: "+modifiedsentence);
        clientSocket.close();
    }
}
```

# Ví dụ chương trình Java: Server

```
import java.net.*;
public class UDPServer
{
public static void main(String args[]) throws Exception{
DatagramSocket serverSocket = new DatagramSocket(8876);
byte[] receiveData = new byte[1024];
byte[] sendData = new byte[1024];
while(true)
{
DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
serverSocket.receive(receivePacket);
String sentence = new String(receivePacket.getData());
InetAddress IPAddress = receivePacket.getAddress();
int port = receivePacket.getPort();
System.out.println(""+IPAddress+","+port+
": "+sentence);
String capsent=sentence.toUpperCase();
sendData = capsent.getBytes();
DatagramPacket sendPacket =
new DatagramPacket(sendData, capsent.length(), IPAddress, port);
serverSocket.send(sendPacket);
}
}
}
```



# Bài tập lập trình UDP Socket

## ❖ Xây dựng chương trình Exchange Rate

### ▪ ExchangeRateServer

- Khi nhận một chuỗi thì xác định chuỗi đó có cấu trúc sau hay không:
  - “ExchangeRate”+“Loại tiền”+”to”+“Loại tiền”,
  - “Loại tiền”: Yen, VND, USD, ...
- Nếu đúng thì trả về cho máy khách tỉ giá(random) theo ngày giờ của hệ thống

### ▪ ExchangeRateTable

- Cập nhật thông tin về tỉ giá sau mỗi giây.

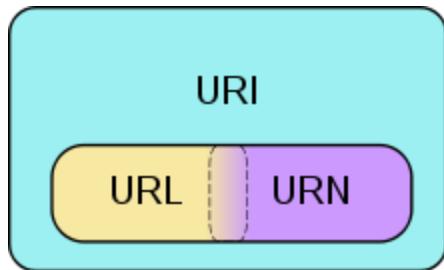


# URI, URL và URLConnection



# URI

- ❖ URI (Uniform Resource Identifier) là một chuỗi ký tự được sử dụng để xác định một tên hoặc một tài nguyên.
- ❖ URI nhằm cho phép tương tác với các thể hiện tài nguyên trên mạng (World Wide Web) sử dụng các giao thức cụ thể.



Uniform Resource Locator(URL)

Uniform Resource Name(URN)



## URI (tt)

- ❖ URI được biểu hiện bởi lớp `java.net.URI`
- ❖ `java.net.URI` có khả năng:
  - Biểu hiện
  - Phân tích
  - Chuẩn hóa



# URI (Ví dụ)

```
import java.net.*;
public class URITest {
    public static void main(String[] argv) throws Exception {
        URI uri1 = new URI("http://www.example.com/hoge/./moge/../../../");
        URI uri2 = new URI("http://www.example.com/");
        uri1 = uri1.normalize();

        System.out.println("uri1: " + uri1);
        System.out.println("uri2: " + uri2);
        System.out.println(uri1.compareTo(uri2));
    }
}
```

Kết quả

```
uri1: http://www.example.com/
uri2: http://www.example.com/
0
```



# URL

- ❖ URL (Uniform Resource Locator) được dùng để tham chiếu tới tài nguyên trên Internet.
- ❖ URL mang lại khả năng siêu liên kết cho các trang mạng.
- ❖ Các tài nguyên khác nhau được tham chiếu tới bằng địa chỉ, chính là URL.



# URL

## ❖ Một URL gồm có nhiều phần :

- URL scheme, thường là Tên giao thức (ví dụ: http, ftp) nhưng cũng có thể là một cái tên khác (ví dụ: news, mailto).
- Tên miền (ví dụ: http://vi.wikipedia.org)
- Chỉ định thêm cổng (có thể không cần)
- Đường dẫn tuyệt đối trên máy phục vụ của tài nguyên (ví dụ: thumuc/trang)
- Các truy vấn (có thể không cần)
- Chỉ định mục con (có thể không cần)



# URL(Ví dụ)

http://vi.wikipedia.org:80/thumuc/trang?timkiem=cauhoi#dautien

URL scheme

tên miền

cổng

đường dẫn

truy vấn

mục con



# Ví dụ

```
import java.net.*;
public class URLTest
{
    public static void main(String[] argv)
        throws Exception
    {
        URI uri = new
            URI("http://www.example.com/hoge/./moge/../../..");
        uri = uri.normalize();

        URL url = uri.toURL();

        System.out.println(url);
    }
}
```

# Phân biệt URI và URL

## ❖ URI

- Dùng để xác định một resource nào đó trên web, về mặt tên (cách gọi nó như thế nào) hoặc địa chỉ (nó nằm ở đâu, làm sao đưa được nó về máy trạm!?)

## ❖ URL

- Là địa chỉ tới một resource nào đó.



# Phân biệt URL và tên miền

❖ Đôi khi có thể hiểu nhầm URL là tên miền.

Ví dụ URL:

- URL:

- <http://www.viphanoi.com/index.php> <- Đây là URL

- Tên miền:

- [www.viphanoi.com](http://www.viphanoi.com) <- Đây là tên miền.

❖ URL là địa chỉ của một đối tượng thường được gõ vào vùng Address của các Web Browser. Về cơ bản, URL là con trỏ chỉ tới vị trí của một đối tượng.



# Khởi tạo URL trong Java

- ❖ `public URL(String url) throws MalformedURLException`
  - `URL u = new URL("http://www.sun.com/index.html");`
- ❖ `public URL(String protocol, String host, String file) throws MalformedURLException`
  - `URL u = new URL("http", "/www.sun.com", "index.html");`
- ❖ `public URL(String protocol, String host, int port, String file) throws MalformedURLException`
  - `URL u = new URL("http", "/www.sun.com", 80, "index.html");`
- ❖ `public URL(URL u, String s) throws MalformedURLException`



## Các thành phần

- ❖ `public String getProtocol()`
- ❖ `public String getHost()`
- ❖ `public int getPort()`
- ❖ `public int getDefaultPort()`
- ❖ `public String getFile()`
- ❖ `public String getRef()`



## Ví dụ

- ❖ Viết chương trình nhập vào một URL từ đối dòng lệnh và hiển thị từng thành phần tạo nên URL lên màn hình.

```
try{  
    URL u = new URL(args[0]);  
    System.out.println("URL is "+u);  
    System.out.println("The protocol part is "+u.getProtocol());  
    System.out.println("The host part is "+u.getHost());  
    System.out.println("The file part is "+u.getFile());  
    System.out.println("The reference part is "+u.getRef());  
}  
catch(MalformedURLException e){}
```



# URLConnection

- ❖ URLConnection là một phương thức thể hiện sự truyền tải tín hiệu của URL

URI → URL → URLConnection

# Ví dụ về URL Connection

```
import java.io.*;
import java.net.*;
import java.util.*;
public class NetCat {
    public static void main(String[] argv) throws Exception {
        URI uri = new URI(argv[0]);
        URLConnection connection = uri.toURL().openConnection();
        Map headers = connection.getHeaderFields();
        for (Object key : headers.keySet()) {
            System.out.println(key + ":" + headers.get(key));
        }
        BufferedReader reader =
            new BufferedReader(new InputStreamReader
                (connection.getInputStream(), "JISAutoDetect"));
        String buffer = reader.readLine();
        System.out.println();
        while (null != buffer) {
            System.out.println(buffer);
            buffer = reader.readLine();
        }
    }
}
```



# Kết quả

```
% java NetCat http://www.example.com/
```

```
Set-Cookie:
```

```
[PREF=ID=xxxxxxxxxxxxxxxx:TM=xxxxxxxxxx:LM=xxxxxxxxxx:S=xxxxxxxxxxxxxxxxx;  
expires=Sun, 17-Jan-2038 19:14:07 GMT; path=/; domain=.example.com]
```

```
null: [HTTP/1.1 200 OK]
```

```
Date: [Mon, 03 Jul 2006 09:22:03 GMT]
```

```
Content-Type: [text/html]
```

```
Server: [XXX/YYY]
```

```
Transfer-Encoding: [chunked]
```

```
Cache-Control: [private]
```

```
<html>  
<head>  
<meta http-equiv="content-type" content="text/html; charset=UTF-  
8"><title>Example</title><script>  
</head>  
<body>  
It is an example.  
</body>  
%
```



# Ưu điểm của URLConnection

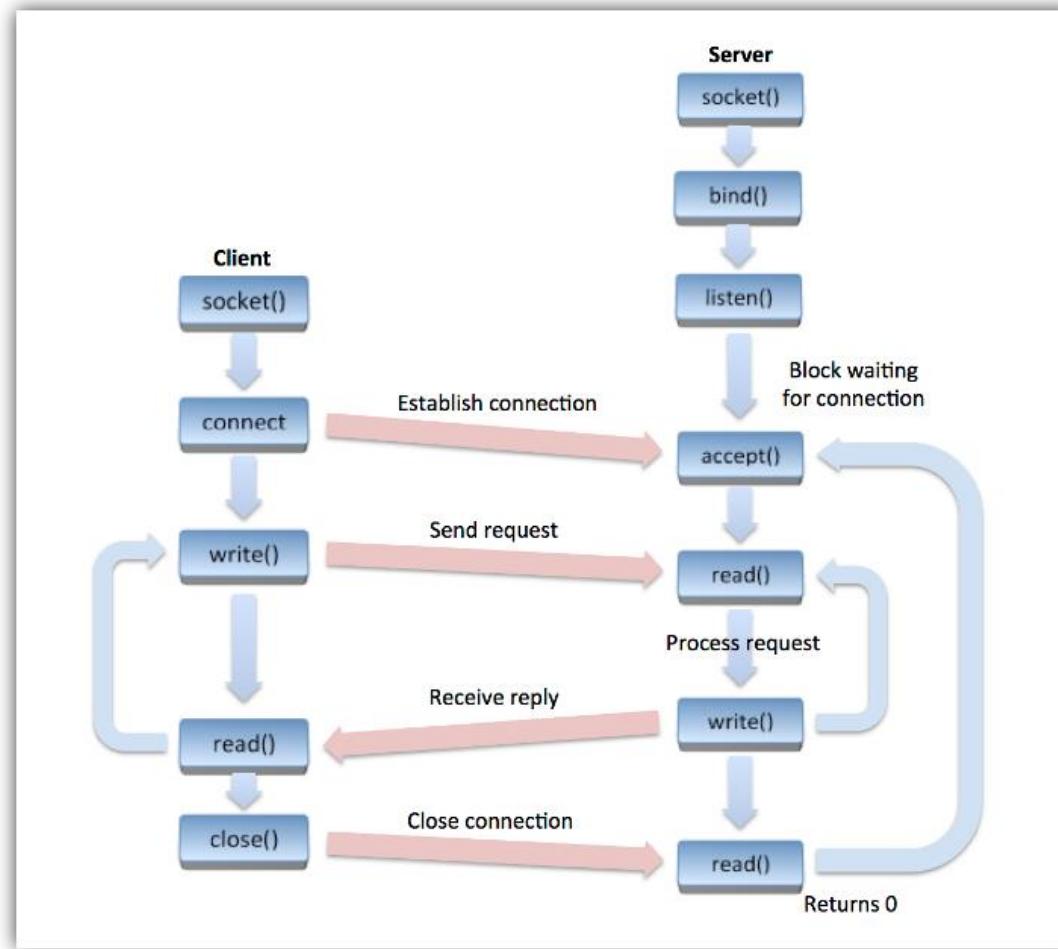
- ❖ Có thể lấy thông tin của các trang Web một cách nhanh chóng mà không cần phải lập trình tương tác sử dụng Socket
- ❖ Không cần điều khiển các luồng thông tin như khi lập trình Socket
- ❖ Nâng cao hiệu quả lập trình.



# Hướng dẫn tạo Chat Room

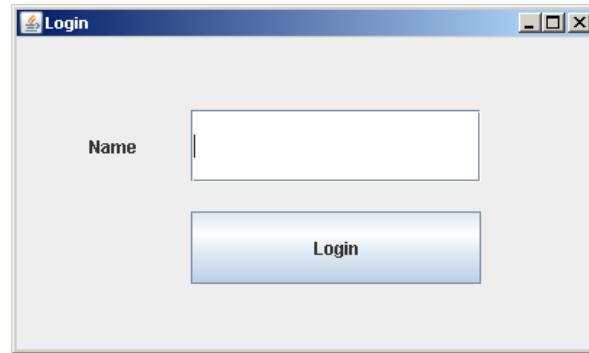


# Ôn lại lập trình giao thức TCP

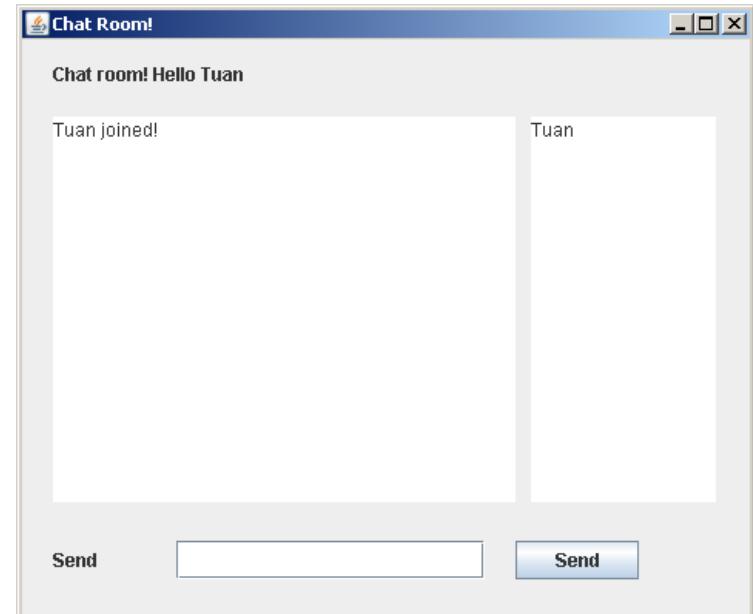


# Chuẩn bị giao diện

## ❖ Giao diện Login



## ❖ Giao diện Chat room





# Giao diện Login

```
import javax.swing.*;
public class LoginFrame {
public JFrame frame;
public LoginFrame(String ms){
    frame = new JFrame("Login");
    frame.setSize(400 ,300);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setLayout(null);

    JLabel lname=new JLabel("Name");
    lname.setBounds(50, 50, 50, 50);
    frame.add(lname);

    final JTextField Name=new JTextField("");
    Name.setBounds(120, 50, 200, 50);
    frame.add(Name);
```



# Giao diện Login (tt)

```
final JLabel msg=new JLabel("Msg:"+ms);
msg.setBounds(120, 200, 200, 50);
frame.add(msg);

JButton OK=new JButton("Login");
OK.setBounds(120, 120, 200, 50);
frame.add(OK);

frame.setVisible(true);
}
public static void main(String[] args){
    new LoginFrame("");
}
}
```

# Giao diện Chat room

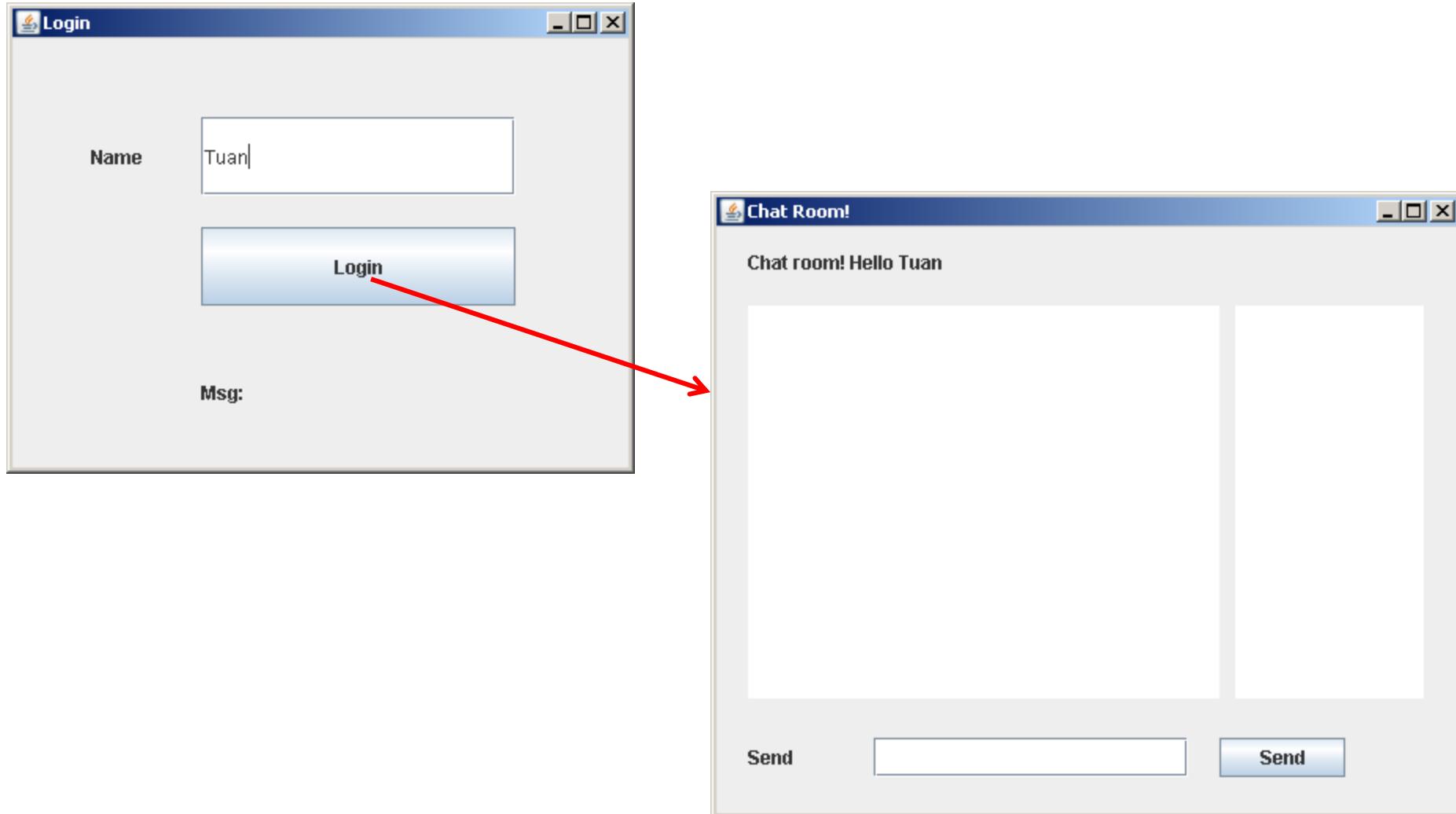
```
public class ChatRoom {  
    public JFrame frame;  
    public JTextArea Room;  
    public JTextField msg;  
    public JTextArea Joiners;  
    public String NickName;  
    public ChatRoom(String NickName){  
        this.NickName = NickName;  
        this.frame = new JFrame("Chat Room!");  
        this.frame.setSize(480 ,400);  
        this.frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        this.frame.setLayout(null);  
        JLabel lr=new JLabel("Chat room! Hello "+this.NickName);  
        lr.setBounds(20, 10, 300, 25);  
        this.frame.add(lr);  
        this.Room=new JTextArea("");  
        this.Room.setBounds(20, 50, 300, 250);  
        this.Room.setEditable(false);  
        this.frame.add(Room);  
    }  
}
```

# Giao diện Chat room(tt)

```
JLabel lsd=new JLabel("Send");
lsd.setBounds(20, 325, 50, 25);
this.frame.add(lsd);
this.msg=new JTextField("");
this.msg.setBounds(100, 325, 200, 25);
this.frame.add(msg);
JButton OK=new JButton("Send");
OK.setBounds(320, 325, 80, 25);
this.frame.add(OK);
JLabel lj=new JLabel("Joiners");
lj.setBounds(620, 10, 50, 50);
this.frame.add(lj);
this.Joiners = new JTextArea("");
this.Joiners.setBounds(330, 50, 120, 250);
this.Joiners.setEditable(false);
this.frame.add(Joiners);
frame.setVisible(true);
}
}
```



# Mở Chat Room trong Login





# Mở Chat Room trong Login

LoginFrame.java

Thêm vào

```
//Lược
OK.setBounds(120, 120, 200, 50);
OK.addActionListener(new ActionListener(){
    @Override
    public void actionPerformed(ActionEvent arg0) {
        if (!Name.getText().equals("")){
            new ChatRoom(Name.getText());
            frame.dispose();
        }
        else msg.setText("Msg: Please input your name!");
    }
});
frame.add(OK);
//Lược
```

# Tạo TCP Server

```
public class ChatRoomServer {  
    public final static int daytimePort = 5000;  
    public ChatRoomServer(){  
        ServerSocket theServer;  
        Socket theConnection;  
        try {  
            theServer = new ServerSocket(daytimePort);  
            while (true) {  
                theConnection = theServer.accept();  
                System.out.println("Have Connection!");  
                new ThreadedHandler(this, theConnection).start();  
            }  
        }catch (IOException e) {  
            System.err.println(e);  
        }  
    }  
    public static void main(String[] args) {new ChatRoomServer();}
```

# Tạo TCP Server (tt)

```
public class ThreadedHandler extends Thread{
    ChatRoomServer crsv;
    public Socket incoming;
    public DataInputStream dis;
    public DataOutputStream dos;
    public ThreadedHandler(ChatRoomServer crsv, Socket i)
    {
        this.csv=crsv;
        this.incoming=i;
        try{
            this.dis = new DataInputStream(
                incoming.getInputStream());
            this.dos = new DataOutputStream(
                incoming.getOutputStream());
        }catch(IOException e){}
    }
    public void run(){} //Chưa thực thi
}
```

# Tạo Socket kết nối với Server

Khai báo trong lớp ChatRoom

```
public Socket soc;
public DataInputStream dis;
public DataOutputStream dos;
```

Tạo Socket kết nối với Server vào cuối

```
public ChatRoom(String NickName){
    //lược

    try{
        soc = new Socket("localhost", 5000);
        this.dis = new DataInputStream(soc.getInputStream());
        this.dos = new DataOutputStream(soc.getOutputStream());
    }catch(IOException e){this.frame.dispose();}

}
```



# Cấu trúc thông điệp của Chat Room

Command

Message

Mục Đích: Phân biệt các thông điệp của client và server

Chuyển các thông tin cần thiết ứng với các thông điệp.



# Cấu trúc thông điệp

## ❖ Phía Client

### ▪ Nhận:

- “Msg,”+[Tin nhắn] : Nhận tin nhắn từ Server
- “Jnr,”+[Tất cả tên User] : Nhận tất cả tên user có trong room từ Server

### ▪ Gửi

- “Join”+[Tên]: Gửi cho Server tên đăng nhập
- “Msg,”+[Tin nhắn] : Gửi cho Server tin nhắn



# Cấu trúc thông điệp

## ❖ Phía Server

### ▪ Gửi:

- “Msg,”+[Tin nhắn] : Gửi cho Client tin nhắn
- “Jnr,”+[Tất cả tên User] : Gửi tất cả tên user có trong room đến Client

### ▪ Nhận:

- “Join”+[Tên]: Nhận tên đăng nhập từ Client
- “Msg,”+[Tin nhắn] : Nhận tin nhắn từ Client



# Hoàn thiện chương trình

## ❖ Phía Server

- Tạo một Vector trong ChatRoomServer quản lý các luồng kết nối.

```
public Vector<ThreadedHandler> cls=new Vector<ThreadedHandler>();
```

- Trong luồng kết nối (lớp ThreadedHandler)
  - Quản lý tên User bằng cách Khai báo tên

```
public String name;
```
  - Xử lý đăng nhập trong run(){}



# Xử lý đăng nhập trong run(){}

```
String ch="";
try{
    ch = dis.readUTF();
    String cmd=ch.substring(0, ch.indexOf(","));
    String msg=ch.substring(ch.indexOf(",")+1);
    if (!cmd.equals("Join")) incoming.close();
        System.out.println("Hello "+msg);
    this.name=msg;
    this.csv.cls.add(this);

    //Nhận và gửi thông điệp
}catch(IOException e){
    csv.cls.remove(this);
}
```

# Nhận và gửi đoạn chat

```
while (true)
{
    ch = dis.readUTF();
    cmd=ch.substring(0, ch.indexOf(","));
    msg=ch.substring(ch.indexOf(",")+1);
    if (cmd.equals("Msg")) {
        for (int i=0;i<this.csv.cls.size();i++){
            ThreadedHandler temp=this.csv.cls.get(i);
            if (temp!=this){
                temp.dos.writeUTF("Msg,"+this.name+">>>"+msg);
            }
        }
    }
    else{
        incoming.close();
        this.csv.cls.remove(this);
    }
}
```



# Hoàn thiện chương trình (Client)

## ❖ Phía Client

- Tạo luồng nhận thông điệp từ Server

```
public class ThreadedHandler extends Thread{  
    ChatRoom cr;  
    public ThreadedHandler(ChatRoom cr){  
        this.cr=cr;  
    }  
    public void run(){  
        String ch="";  
        try{
```



```
while (true){  
    ch = dis.readUTF();  
    String cmd=ch.substring(0, ch.indexOf(",,"));  
    String msg=ch.substring(ch.indexOf(",,")+1);  
    if (cmd.equals("Msg"))  
        this.cr.Room.setText(msg+"\n"+cr.Room.getText());  
    else  
        this.cr.soc.close();  
}  
}catch(IOException e){cr.frame.dispose();new LoginFrame();}  
}
```



Khởi tạo luồng nhận thông điệp từ Server và Tham gia vào Room

```
public ChatRoom(String NickName){  
    //lược  
  
    try{  
        soc = new Socket("localhost", 5000);  
        this.dis = new DataInputStream(soc.getInputStream());  
        this.dos = new DataOutputStream(soc.getOutputStream());  
        new ThreadedHandler(this).start();  
        this.dos.writeUTF("Join,"+this.NickName);  
  
    }catch(IOException e){this.frame.dispose();}  
}
```

# Tạo ActionListener cho Client

```
public class SendActionListener implements ActionListener{  
    ChatRoom cr;  
    public SendActionListener(ChatRoom cr){  
        this.cr = cr;  
    }  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        if (!cr.msg.getText().equals("")){  
            cr.Room.setText(cr.NickName+">"+  
                            cr.msg.getText()+"\n"+cr.Room.getText());  
            try{  
                this.cr.dos.writeUTF("Msg,"+cr.msg.getText());  
            }catch(IOException e1){  
                cr.frame.dispose();new LoginFrame();}  
            cr.msg.setText("");  
        }  
    }  
}
```



# Bài tập

- ❖ Thêm vào chương trình chat room các chức năng sau:
  - Khi có một người đăng nhập hay thoát ra thì:
    - Server thông báo cho tất cả Clients biết và cập nhật tên của tất cả Users
    - Clients hiển thị tên các Users bên phải Frame



# Lập trình truy xuất Cơ sở dữ liệu



# Lập trình truy xuất cơ sở dữ liệu

## ❖ Java cung cấp thư viện java.sql gồm:

- Các lớp
- Interface

Cho phép chương trình kết nối với các cơ sở dữ liệu để truy xuất và xử lý dữ liệu.

## ❖ Các hệ cơ sở dữ liệu như

- MySQL Server, Access, Oracle,...

Được xem như các chương trình server lưu trữ dữ liệu.



# Lập trình truy xuất cơ sở dữ liệu

- ❖ Để kết nối và trao đổi dữ liệu với các CSDL, chương trình cần nạp Driver tương ứng. Driver có nhiệm vụ:
  - Chuyển đổi các yêu cầu đến CSDL từ chương trình thành định dạng mà hệ CSDL hiểu được
  - Chuyển đổi dữ liệu lên chương trình



# Các bước kết nối và truy xuất

- ❖ Nạp Driver tương ứng với hệ CSDL
- ❖ Thực hiện kết nối đến CSDL
- ❖ Trao đổi dữ liệu với CSDL
  - executeQuery()
  - executeUpdate()
  - execute()
  - ....

# Ví dụ về cài đặt hệ CSDL

## ❖ Cài đặt hệ CSDL MySQL Server

- Tải và cài đặt MySQL Community Server

- <http://downloads.mysql.com/archives/community/>

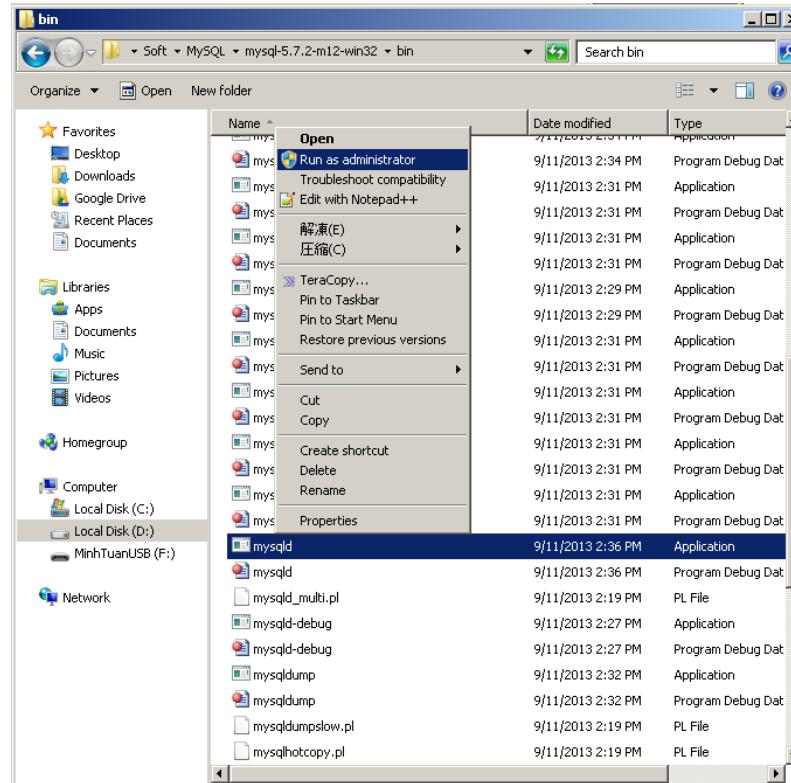
- Tải và cài đặt MySQL Query Browser

- <http://downloads.mysql.com/archives/query/>



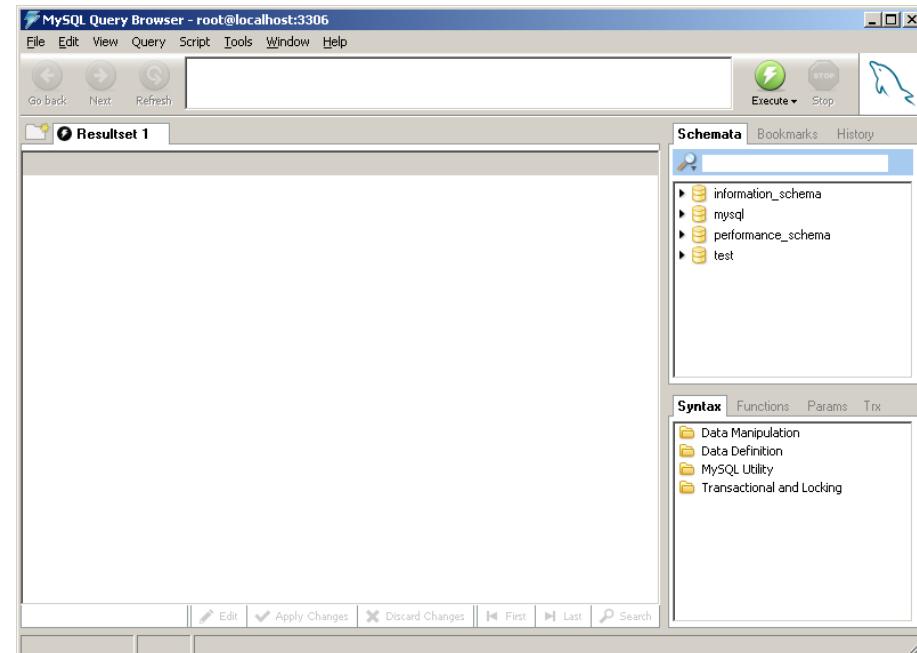
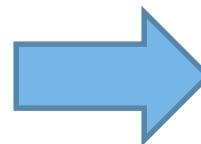
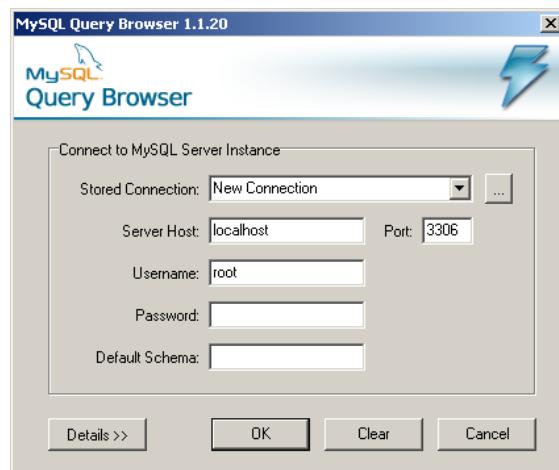
# Khởi động và cấu hình

- ❖ Khởi động MySQL Server
  - Chạy file mysqld.exe với chế độ administrator



# Khởi động và cấu hình (tt)

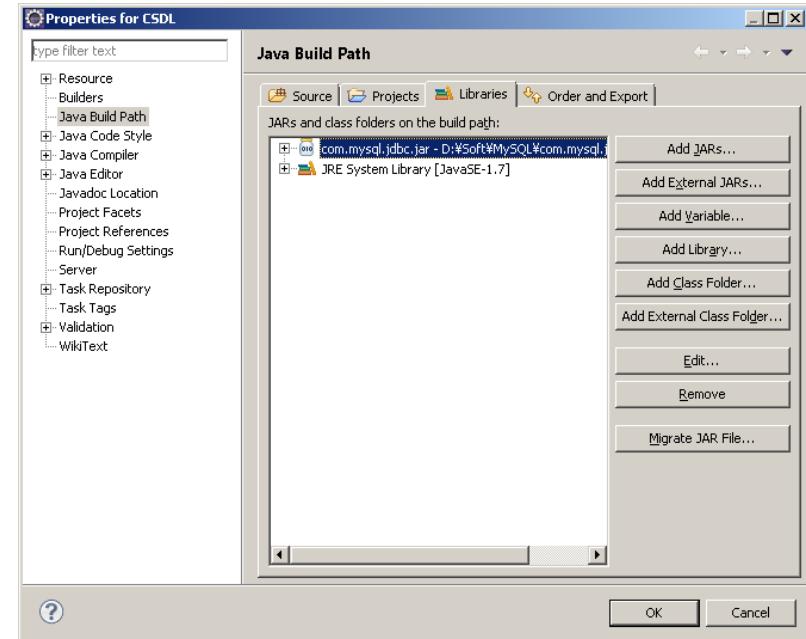
## ❖ Khởi động thiết lập MySQL Query Browser





# Nạp driver

- ❖ Tải driver mysql: com.mysql.jdbc.jar
  - <http://www.java2s.com/Code/Jar/c/Downloadcommysqljdbc515jar.htm>
- ❖ Nạp driver vào Eclipse



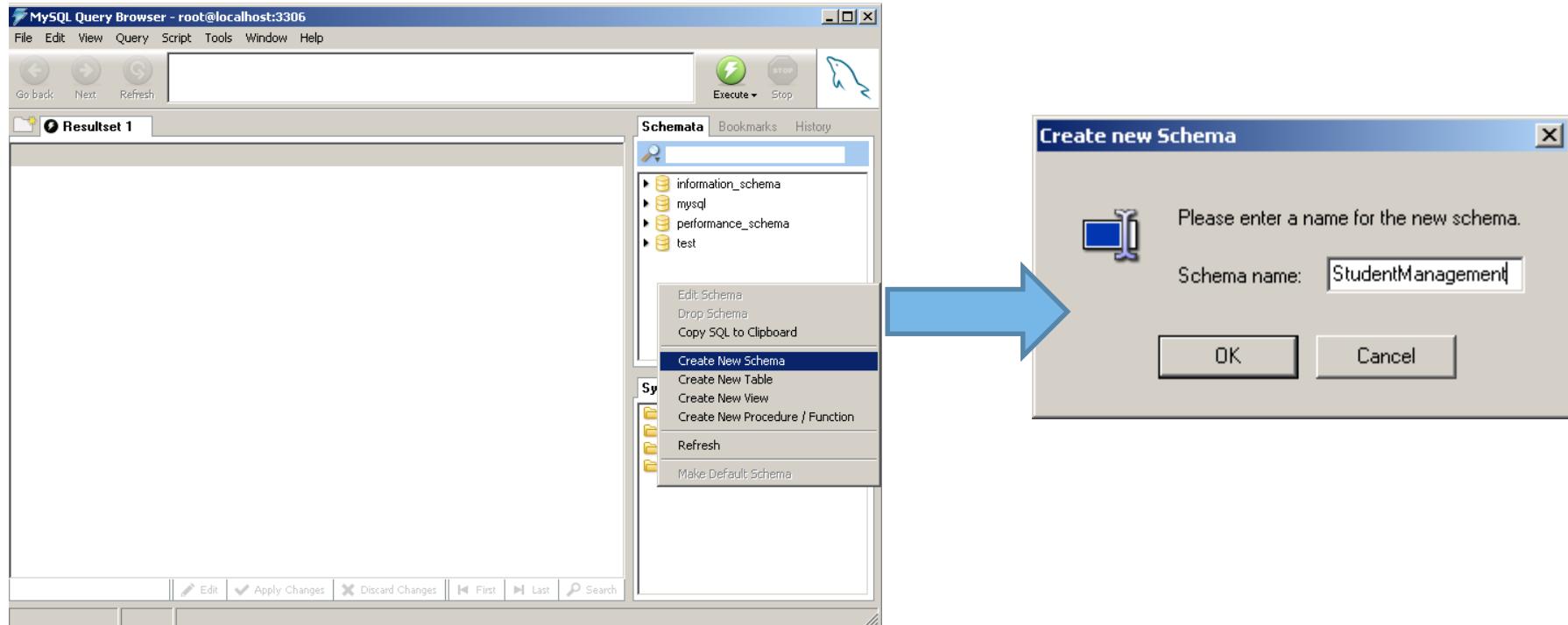


# Bài tập CSDL

- ❖ Tạo cơ sở dữ liệu có tên
  - StudentManagement
- ❖ Tạo bảng Students gồm các trường
  - ID, Name, Math, Phys, Chem và Aver
- ❖ Viết chương trình Java kết nối đến CSDL, truy vấn và hiển thị tên cột của bảng Students

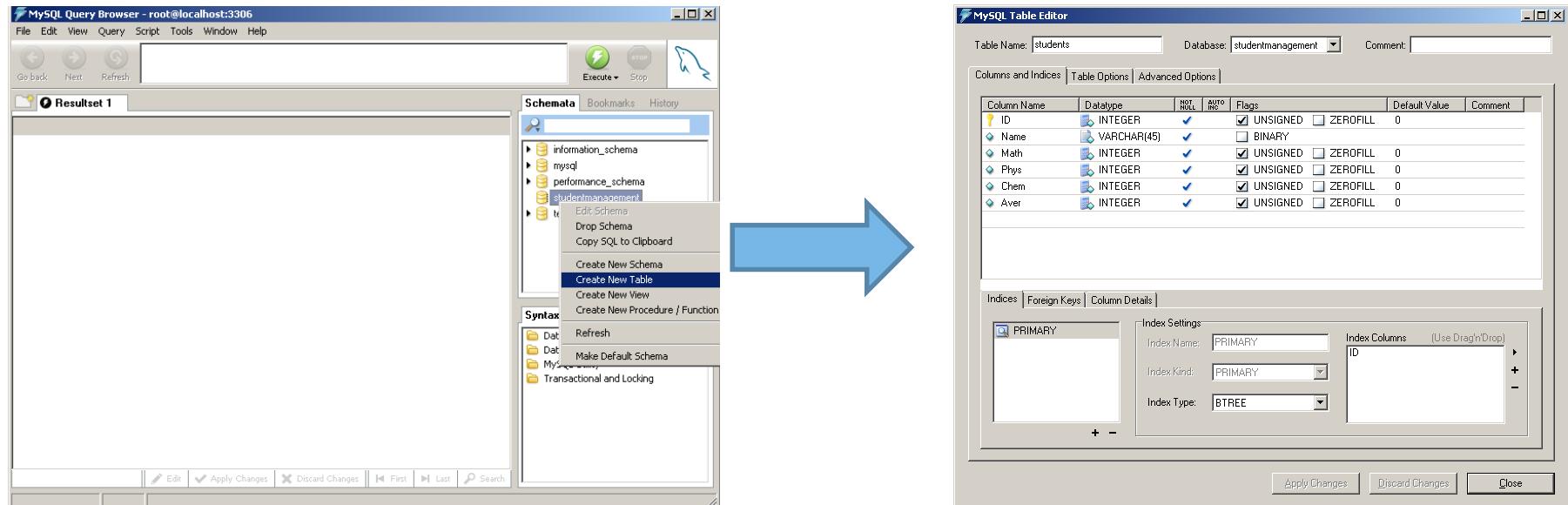
# Đáp án

- ❖ Mở MySQL Query Browser
  - Tạo Schema



# Tiếp theo

## ❖ Tạo Bảng Students



# Chương trình Java

```
import java.sql.*;
public class Ketnoi {
    public static void main(String [] args){
        try{
            //Nap Driver
            Class.forName("com.mysql.jdbc.Driver");
            //Ket noi toi CSDL
            Connection conn = DriverManager.getConnection("jdbc:mysql:" +
                "//localhost:3306/studentmanagement","root","");
            Statement sm=conn.createStatement();
            //Truy van
            ResultSet rs = sm.executeQuery("Select * from students");
            //Lay thong tin
            ResultSetMetaData rsm=rs.getMetaData();
            int cn=rsm.getColumnCount();
            for (int i=1;i<=cn;i++)
                System.out.print(rsm.getColumnLabel(i)+"    ");
        }catch(Exception e){}
    }
}
```



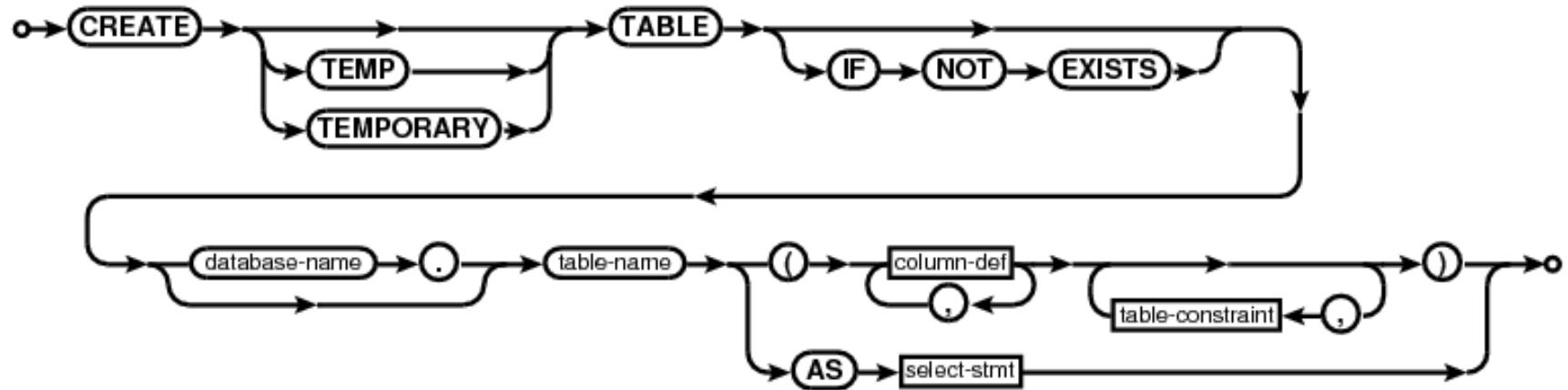
## Bài Tập 2

- ❖ Xây dựng chương trình thêm, xóa, sửa thông tin bảng Students và truy vấn thông tin từ bảng để hiển thị ra màn hình.



# Các lệnh truy xuất thường gặp

## Create



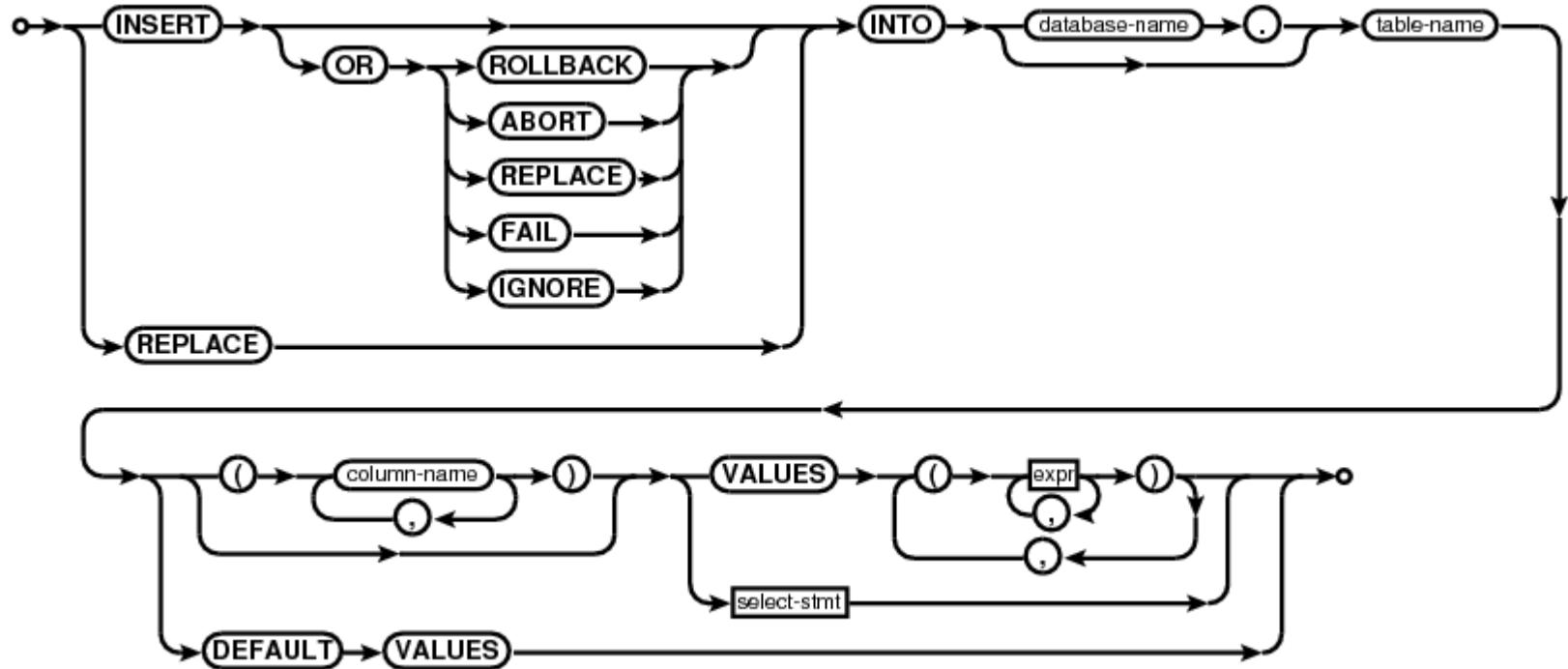
## Delete





# Các lệnh truy xuất thường gặp

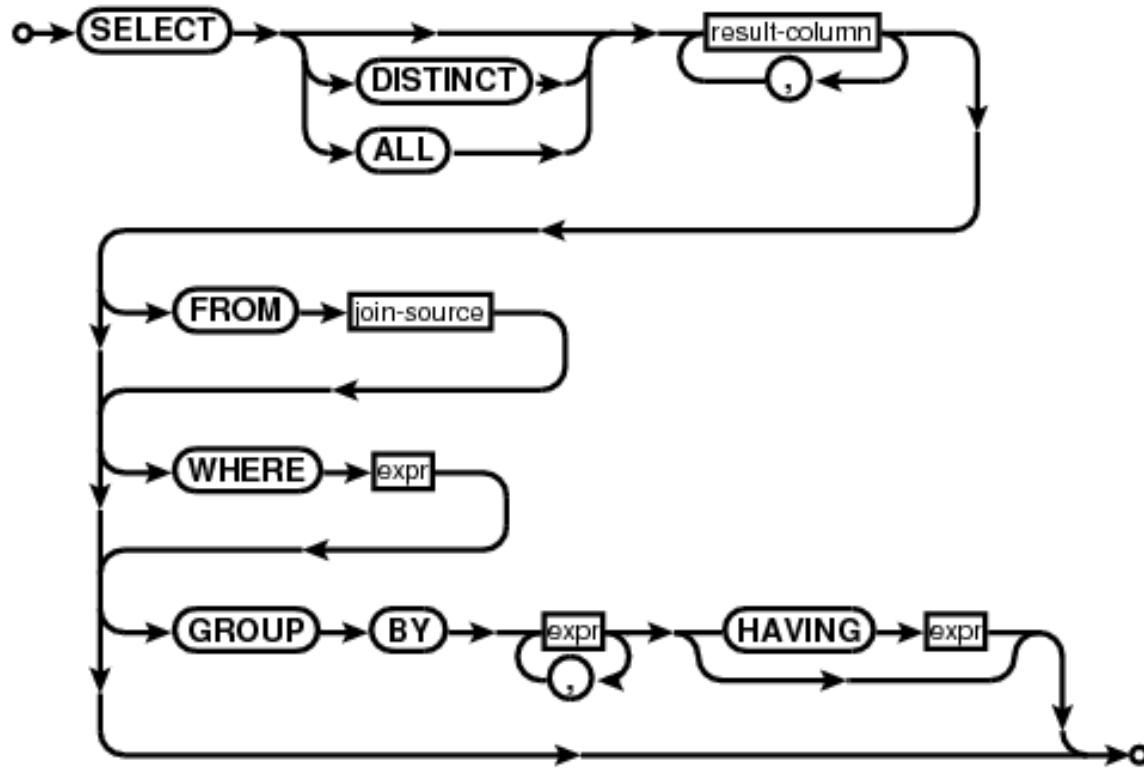
## Insert





# Các lệnh truy xuất thường gặp

## Select



# Đáp Án

```
import java.sql.*;
import java.util.*;
public class Ketnoi {
public static void main(String [] args){
Scanner sc=new Scanner(System.in);
try{
    //Nap Driver
    Class.forName("com.mysql.jdbc.Driver");
    //Ket noi toi CSDL
    Connection conn = DriverManager.getConnection("jdbc:mysql:" +
        "//localhost:3306/studentmanagement","root","");
    Statement sm=conn.createStatement();
    lap:while(true){
        //Truy van
        ResultSet rs = sm.executeQuery("Select * from students");
        //Lay thong tin
        ResultSetMetaData rsm=rs.getMetaData();
        int cn=rsm.getColumnCount();
        for (int i=1;i<=cn;i++)
            System.out.print(rsm.getColumnLabel(i)+"\t");
        System.out.println("");
    }
}
```

```
while(rs.next()){
    for (int i=1;i<=cn;i++)
        System.out.print(rs.getString(i)+"\t");
    System.out.println("");
}
System.out.println("1. Them....2.Xoa....3.Sua....4.Ketthuc");
int chon=sc.nextInt();
int id;
String name;
int math,phys,chem,avr;
switch(chon){
    case 1:
        System.out.print("Nhap ID:");id=sc.nextInt();
        System.out.print("Nhap ten:");name=sc.next();
        System.out.print("Nhap diem toan:");math=sc.nextInt();
        System.out.print("Nhap diem Ly:");phys=sc.nextInt();
        System.out.print("Nhap diem hoa:");chem=sc.nextInt();
        avr=(math+phys+chem)/3;
        try{
            sm.execute("Insert into students values("+id+",\""
                +name+"\"," +math+"," +phys+"," +chem+"," +avr+ ")");
        }catch(Exception e){System.out.println("Error!");}
        break;
}
```

```
case 2:  
    System.out.print("Nhập ID:"); id=sc.nextInt();  
    try{  
        sm.execute("Delete from students where id="+id);  
    }catch(Exception e){System.out.println("Error!");}  
    break;  
case 3:  
    System.out.print("Nhập ID:"); id=sc.nextInt();  
    System.out.print("Nhập tên:"); name=sc.next();  
    System.out.print("Nhập điểm toán:"); math=sc.nextInt();  
    System.out.print("Nhập điểm lý:"); phys=sc.nextInt();  
    System.out.print("Nhập điểm hóa:"); chem=sc.nextInt();  
    avr=(math+phys+chem)/3;  
    try{  
        sm.execute("Update students set Name = '"+  
                  +name+"',Math = "+math+",Phys="  
                  +phys+",Chem="+chem+",Aver="  
                  +avr+" where ID="+id);  
    }catch(Exception e){System.out.println("Error!");}  
    break;  
case 4:break lap;  
default:break;  
}  
}  
}catch(Exception e){}  
}
```



## Bài Tập 3

- ❖ Xây dựng chương trình theo mô hình Client/Server như sau:
  - Server
    - Chứa CSDL
    - Nhận các câu truy vấn từ Client và trả kết quả cho Client
  - Client
    - Truy vấn tới Server
    - Hiển thị kết quả Server trả về



## Bài tập 4

- ❖ Mô hình Client/Server ở bài 3 có gì không ổn?
- ❖ Hãy chỉ ra chỗ không ổn và nêu hướng giải quyết.
- ❖ Lập Trình



Môn học

# Lập Trình Mạng

Giảng Viên:

Phạm Minh Tuấn



# Giới thiệu

- ❖ Phạm Minh Tuấn
- ❖ E-mail: [pmtuan@dut.udn.vn](mailto:pmtuan@dut.udn.vn)
- ❖ Tel: 0913230910
- ❖ Khoa Công nghệ thông tin – Trường ĐHBK – ĐHĐN



## Bài 4:

# HTML/CSS/JavaScript



I T F

# HTML



# Công cụ & Môi trường

- ❖ Môi trường Internet
- ❖ Notepad
- ❖ Web browser
- ❖ FTP Soft



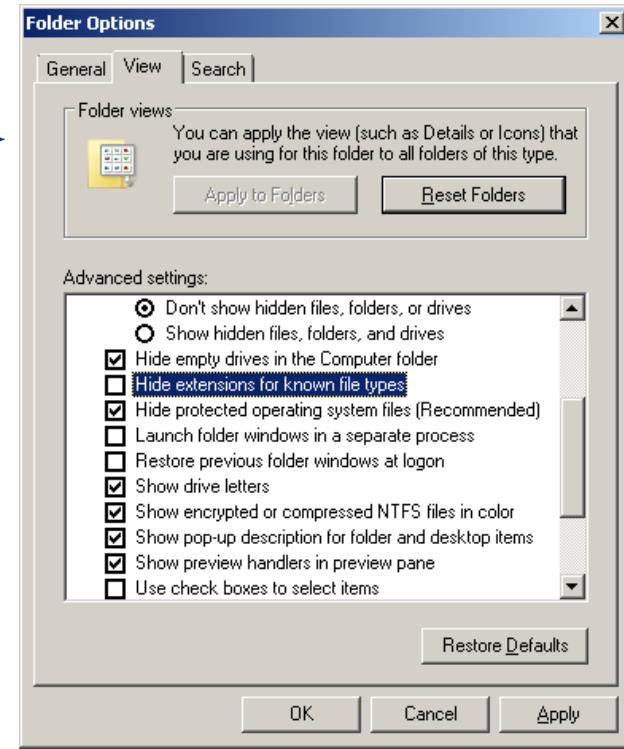
# HTML

## ❖ HTML

- Hyper Text Markup Language

## ❖ Chuẩn bị

- Control Panel → Appearance and Personalization → Folder Option





# Tổng quan HTML

## ❖ Tag

- Bắt đầu: <Tag name>
- Kết thúc: </Tag name>

## ❖ Các Tag cơ bản

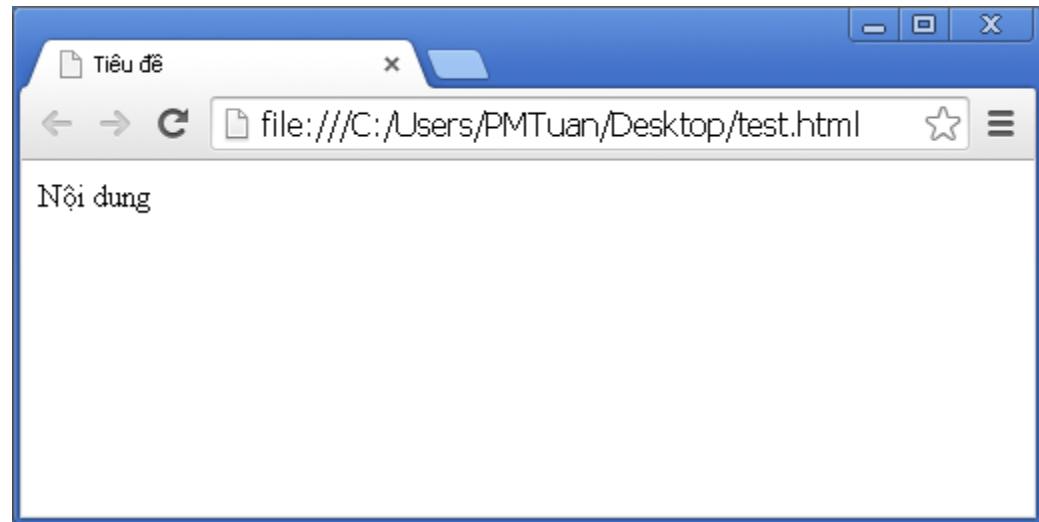
- <html> </html> : Định nghĩa HTML
- <head> </head> : Thông tin cơ bản
- <title> </title>: Tiêu Đề
- <body> </body>: Nội dung



# Ví dụ

```
<html>
<head>
<title> Tiêu đề </title>
</head>

<body> Nội dung
</body>
</html>
```





# Các chỉ định chung

- ❖ <body text="Màu">
- ❖ <basefont size="Kích thước">
- ❖ <body bgcolor="Màu">
- ❖ <body background="URL">
- ❖ <!-- Comment -->



# Ví dụ

```
<html>
<head>
<title>Tiêu Đề</title>
</head>
<body text="blue" bgcolor="yellow">
<basefont size="3">
  HTML TEST
</body>
</html>
```





# Chỉ định từng phần

- ❖ <font color="Màu"> </font>
- ❖ <font size="7"> </font>
- ❖ <br> : Xuống hàng
- ❖ <p> Đoạn </p>
- ❖ <h1></h1>~<h6></h6> : Kích cỡ chữ
- ❖ <b>Đậm</b>; <i>Nghiên</i>
- ❖ <u>Gạch chân</u>; <s>Gạch bỏ</s>
- ❖ <sup></sup>; <sub></sub>
- ❖ <font face="Kiểu"> ...</font>



# Layout

## ❖ Line

- `<hr>`
- `<hr width="450">`
- `<hr width="50%">`
- `<hr size="20">`
- `<hr align="left" width="50%">`
- `<hr noshade>`

❖ `<center></center>`: Canh giữa

❖ `<pre></pre>`: Tự trình bày

❖ Thuộc tính **align**: canh trái, phải, giữa

- Ví dụ `<p align="right">...</p>`



# Link

- ❖  [</a>](..../index.html)  
 [</a>](http://www.yahoo.co.jp/)



## Thêm hình ảnh

- ❖ ``
- ❖ ``
- ❖ ``



# List

## ❖ <ol>

- <li></li>
- <li></li>
- ...
- <li></li>

## ❖ </ol>

## ❖ <ol type="1">

- type="1"
- type="A"
- type="square", "disc", "circle"



# Table

- ❖ **<table></table>: 1 Bảng**
- ❖ **<tr></tr> : 1 Dòng**
- ❖ **<td></td>: 1 ô**

```
<table border="1"  
style="width:100%">  
  <tr>  
    <td>Jill</td>  
    <td>Smith</td>  
    <td>50</td>  
  </tr>  
  <tr>  
    <td>Eve</td>  
    <td>Jackson</td>  
    <td>94</td>  
  </tr>  
</table>
```



I T F

# Cascading Style Sheets

## CSS



# CSS

- ❖ **CSS** stands for **Cascading Style Sheets**
- ❖ **CSS** defines **how HTML elements are to be displayed**
- ❖ Styles were added to HTML 4.0 **to solve a problem**
- ❖ CSS saves a lot of work
- ❖ External Style Sheets are stored in **CSS files**



# Ví dụ



```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-color: #d0e4fe;
}

h1 {
    color: orange;
    text-align: center;
}
```

```
p {
    font-family: "Times New Roman";
    font-size: 20px;
}
</style>
</head>

<body>
    <h1>My First CSS Example</h1>
    <p>This is a paragraph.</p>
</body>
</html>
```



# CSS Syntax

Selector

**h1**

Declaration

{ color:blue; font-size:12px; }

Declaration

Property

Value

Property

Value

```
p {  
    color: red;  
    /* This is a single-line  
comment */  
    text-align: center;  
}  
  
/* This is  
a multi-line  
comment */
```

# CSS Selectors

- ❖ The element selector selects elements based on the element name.

```
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
    text-align: center;
    color: red;
}
</style>
</head>
<body>

<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the
style.</p>

</body>
</html>
```



# CSS Selectors

## ❖ The class Selector

- The class selector selects elements with a specific class attribute.

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
    text-align: center;
    color: red;
}
</style>
</head>
<body>
<h1 class="center">Red and center-aligned heading</h1>
<p class="center">Red and center-aligned paragraph.</p>
</body>
</html>
```



# CSS Selectors

- ❖ CSS can group the selectors, to minimize the code.

```
h1 {  
    text-align: center;  
    color: red;  
}  
  
h2 {  
    text-align: center;  
    color: red;  
}  
  
p {  
    text-align: center;  
    color: red;  
}
```



```
h1, h2, p {  
    text-align: center;  
    color: red;  
}
```



# Insert CSS

- ❖ There are three ways of inserting a style sheet:
  - External style sheet
  - Internal style sheet
  - Inline style



# External Style Sheet

- ❖ An external style sheet is ideal when the style is applied to many pages. With an external style sheet, you can change the look of an entire Web site by changing just one file.

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

mystyle.css

```
body {
    background-color: lightblue;
}
h1 {
    color: navy;
    margin-left: 20px;
}
```



# Internal Style Sheet

- ❖ An internal style sheet should be used when a single document has a unique style. You define internal styles in the head section of an HTML page, inside the `<style>` tag, like this:

```
<head>
<style>
body {
    background-color: linen;
}
h1 {
    color: maroon;
    margin-left: 40px;
}
</style>
</head>
```



# Inline Styles

- ❖ An inline style loses many of the advantages of a style sheet (by mixing content with presentation).

```
<h1 style="color:blue; margin-left:30px;">This is a heading.</h1>
```



# Background

- ❖ CSS background properties are used to define the background effects of an element.
- ❖ CSS properties used for background effects:
  - background-color:
    - `body {background-color: #e0ffff;}`
  - background-image
    - `body { background-image: url("paper.gif"); }`
  - background-repeat
    - `body {`  
`background-image: url("paper.gif");`  
`background-repeat: repeat-x; }`
  - background-attachment
  - background-position



# CSS Text

## ❖ Text Color

```
body { color: blue; }  
h1 { color: #00ff00; }  
h2 { color: rgb(255,0,0); }
```

## ❖ Text Alignment

```
h1 { text-align: center; }  
p.date { text-align: right; }  
p.main {text-align: justify; }
```

## ❖ Text Decoration

```
h1 {text-decoration: overline; }  
h2 {text-decoration: line-through; }  
h3 {text-decoration: underline; }
```

## ❖ Text Transformation

```
p.uppercase { text-  
transform: uppercase; }  
p.lowercase { text-  
transform: lowercase; }  
p.capitalize {text-  
transform: capitalize; }
```



# CSS Font

## ❖ Font Family

`font-family: "Times New Roman", Times, serif;`

## ❖ Font Style

`font-style: italic;`

`font-style: normal;`

## ❖ Font Size

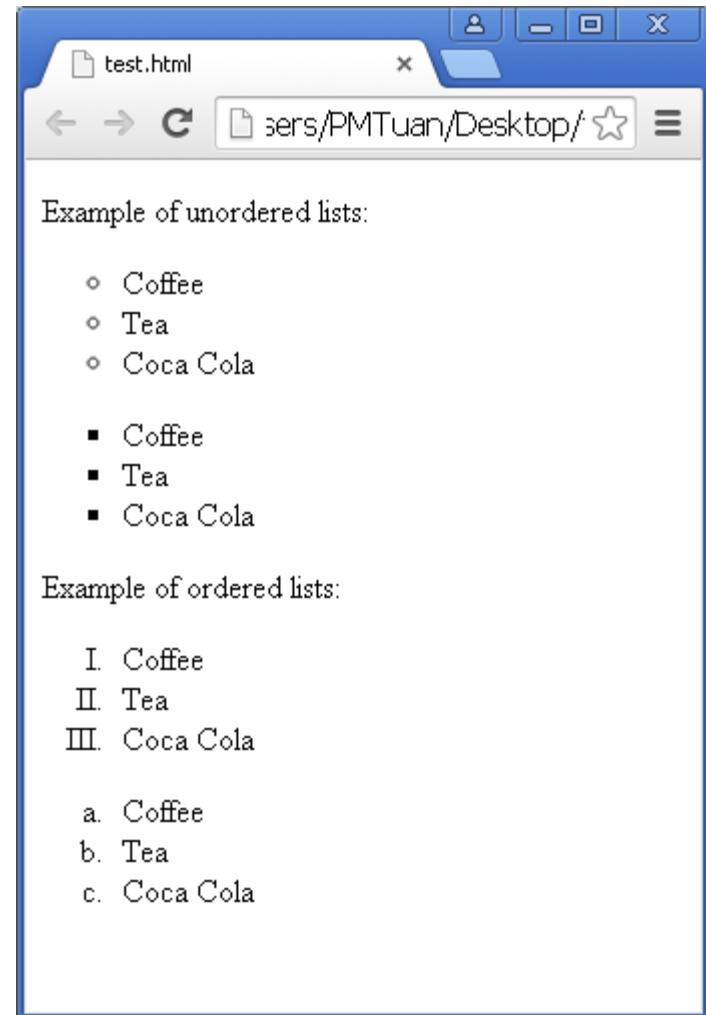
`font-size: 40px;`

`font-size: 2.5em`



# CSS Lists

```
ul.a {  
    list-style-type: circle;  
}  
  
ul.b {  
    list-style-type: square;  
}  
  
ol.c {  
    list-style-type: upper-roman;  
}  
  
ol.d {  
    list-style-type: lower-alpha;  
}
```





# CSS Tables

## ❖ Table Borders

```
table, th, td {  
    border: 1px solid black;  
}
```

## ❖ Table Width, Height

```
table {  
    width: 100%;  
}  
th {  
    height: 50px;  
}
```

## ❖ Horizontal Text Alignment

```
th {  
    text-align: left;  
}
```

## ❖ Vertical Text Alignment

```
td {  
    height: 50px;  
    vertical-align: bottom;  
}
```



I T F

# JavaScript



# JavaScript

- ❖ JavaScript is the programming language of the Web.
- ❖ All modern HTML pages are using JavaScript.
- ❖ JavaScript is easy to learn.
- ❖ This tutorial will teach you JavaScript from basic to advanced.



# Examples

```
<!DOCTYPE html>
<html>
<body>
<h1>My First JavaScript</h1>
<button type="button"
onclick="document.getElementById('demo').in
nerHTML = Date()">
Click me to display Date and Time.</button>
<p id="demo"></p>
</body>
</html>
```





# Why Study JavaScript?

- ❖ JavaScript is one of the **3 languages** all web developers **must** learn:
  - 1. **HTML** to define the content of web pages
  - 2. **CSS** to specify the layout of web pages
  - 3. **JavaScript** to program the behavior of web pages
- ❖ JavaScript is the most popular programming language in the world.



# JavaScript Can Change HTML Content

- ❖ One of many HTML methods is
  - **document.getElementById()**.

```
<!DOCTYPE html>
<html>
<body>
<h1>What Can JavaScript Do?</h1>

<p id="demo">JavaScript can change HTML content.</p>

<button type="button"
onclick="document.getElementById('demo').innerHTML =
'Hello JavaScript!'">Click Me!</button>

</body>
</html>
```





# JavaScript Can Change HTML Styles(CSS)

- ❖ Changing the style of an HTML element, is a variant of changing an HTML attribute:

```
document.getElementById("demo").style.fontSize = "25px";
```

```
<!DOCTYPE html>
<html>
<body>
<h1>What Can JavaScript
Do?</h1>

<p id="demo">JavaScript can
change the style of an HTML
element.</p>

<script>
```

```
function myFunction() {
  var x =
document.getElementById("demo");
  x.style.fontSize = "25px";
  x.style.color = "red";
}
</script>

<button type="button"
onclick="myFunction()">Click Me!</button>
</body>
</html>
```



# JavaScript Can Validate Data

```
<!DOCTYPE html>
<html>
<body>

<h1>JavaScript Can Validate Input</h1>

<p>Please input a number between 1
and 10:</p>

<input id="numb" type="number">

<button type="button"
onclick="myFunction()">Submit</button>

<p id="demo"></p>
```

```
<script>
function myFunction() {
  var x, text;
  x = document.getElementById
    ("numb").value;
  if (isNaN(x) || x < 1 || x > 10) {
    text = "Input not valid";
  } else {
    text = "Input OK";
  }
  document.getElementById
    ("demo").innerHTML = text;
}
</script>

</body>
</html>
```



# JavaScript

- ❖ JavaScript can be placed in the `<body>` and the `<head>` sections of an HTML page.
- ❖ The `<script>` Tag
  - In HTML, JavaScript code must be inserted between `<script>` and `</script>` tags.

```
<script>
document.getElementById("demo").innerHTML = "My First
JavaScript";
</script>
```



# JavaScript

## ❖ JavaScript Functions and Events

- A JavaScript **function** is a block of JavaScript code, that can be executed when "asked" for.
- For example, a function can be executed when an **event** occurs, like when the user clicks a button.

## ❖ JavaScript in <head> or <body>

- You can place any number of scripts in an HTML document.



# JavaScript

## ❖ JavaScript in <head>

```
<!DOCTYPE html>
<html><head>
<script>
function myFunction() {
    document.getElementById("demo").innerHTML = "Paragraph
changed.";
}
</script>
</head>
<body>
<h1>My Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>
</body>
</html>
```



# JavaScript

## ❖ JavaScript in <body>

```
<!DOCTYPE html>
<html>
<body>
<h1>My Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>
<script>
function myFunction() {
    document.getElementById("demo").innerHTML = "Paragraph
changed.";
}
</script>
</body>
</html>
```



# JavaScript

## ❖ External JavaScript

### myScript.js

```
function myFunction() {  
    document.getElementById("demo").innerHTML = "Paragraph  
changed.";  
}
```

```
<!DOCTYPE html>  
<html>  
<body>  
<h1>My Web Page</h1>  
<p id="demo">A Paragraph</p>  
<button type="button" onclick="myFunction()">Try it</button>  
<script src="myScript.js"></script>  
</body>  
</html>
```



# JavaScript Output

- ❖ JavaScript can "display" data in different ways:
  - Writing into
    - an alert box, using **window.alert()**.
    - the HTML output using **document.write()**.
    - an HTML element, using **innerHTML**.
    - the browser console, using **console.log()**.



# JavaScript Output

## ❖ Using `window.alert()`

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
window.alert(5 + 6);
</script>

</body>
</html>
```



# JavaScript Output

## ❖ Using `document.write()`

- This method should be used only for testing.

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
document.write(5 + 6);
</script>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<button onclick="document.write
(5 + 6)">Try it</button>

</body>
</html>
```



# JavaScript Output

## ❖ Using innerHTML

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My First Paragraph</p>
<p id="demo"></p>

<script>
document.getElementById("demo").in
nerHTML = 5 + 6;
</script>

</body>
</html>
```



# JavaScript Output

- ❖ Using `console.log()`
  - browser console with F12

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
console.log(5 + 6);
</script>

</body>
</html>
```

# JavaScript Syntax

## ❖ JavaScript Programs

- A **computer program** is a list of "instructions" to be "executed" by the computer.
- In a programming language, these program instructions are called **statements**.
- JavaScript is a **programming language**.
- JavaScript statements are separated by **semicolon**.

```
var x = 5;  
var y = 6;  
var z = x + y;
```



# JavaScript Syntax

## ❖ JavaScript Statements

- JavaScript statements are composed of: Values, Operators, Expressions, Keywords, and Comments.

## ❖ JavaScript Values

- The JavaScript syntax defines two types of values: Fixed values and variable values.
- Fixed values are called **literals**. Variable values are called **variables**.



# JavaScript Syntax

## ❖ JavaScript Operators

- JavaScript uses an **assignment operator** “=” to **assign** values to variables:

```
var x = 5;  
var y = 6;
```

- JavaScript uses **arithmetic operators** “+-\* /” to **compute** values:

$$(5 + 6) * 10$$



# JavaScript Syntax

## ❖ JavaScript Keywords

- JavaScript **keywords** are used to identify actions to be performed.
- The **var** keyword tells the browser to create a new variable: `var x = 5 + 6;`

## ❖ JavaScript Comments

- Not all JavaScript statements are "executed".
- Code after double slashes `//` or between `/*` and `*/` is treated as a **comment**.



# JavaScript Syntax

## ❖ JavaScript is Case Sensitive

- All JavaScript identifiers are **case sensitive**
- The variables **lastName** and **lastname**, are two different variables.

```
lastName = "Doe";  
lastname = "Peterson";
```

## ❖ JavaScript Character Set

- JavaScript uses the **Unicode** character set.



# JavaScript Data Types

- ❖ JavaScript variables can hold many **data types**: numbers, strings, arrays, objects and more:

```
var length = 16;                                // Number
var lastName = "Johnson";                         // String
var cars = ["Saab", "Volvo", "BMW"];              // Array
var x = {firstName:"John", lastName:"Doe"};        // Object
```



# JavaScript Functions

- ❖ A JavaScript function is a block of code designed to perform a particular task.
- ❖ A JavaScript function is executed when "something" invokes it (calls it).

```
function myFunction(p1, p2) {  
    return p1 * p2; // the function returns the product of p1 and p2  
}
```



# Math Object Methods

- ❖ The Math object includes several mathematical methods.

```
Math.random();           // returns a random number
```

```
Math.E;                 // returns Euler's number
Math.PI                 // returns PI
Math.SQRT2               // returns the square root of 2
Math.SQRT1_2              // returns the square root of 1/2
Math.LN2                 // returns the natural logarithm of 2
Math.LN10                // returns the natural logarithm of 10
Math.LOG2E                // returns base 2 logarithm of E
Math.LOG10E               // returns base 10 logarithm of E
```



# Math Object Methods

Method	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x, in radians
asin(x)	Returns the arcsine of x, in radians
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
atan2(y,x)	Returns the arctangent of the quotient of its arguments
ceil(x)	Returns x, rounded upwards to the nearest integer
cos(x)	Returns the cosine of x (x is in radians)
exp(x)	Returns the value of E <sup>x</sup>
floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm (base E) of x
max(x,y,z,...,n)	Returns the number with the highest value
min(x,y,z,...,n)	Returns the number with the lowest value
pow(x,y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer
sin(x)	Returns the sine of x (x is in radians)
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle

# Conditional Statements

- ❖ In JavaScript we have the following conditional statements:
  - Use **if** to specify a block of code to be executed, if a specified condition is true
  - Use **else** to specify a block of code to be executed, if the same condition is false
  - Use **else if** to specify a new condition to test, if the first condition is false
  - Use **switch** to specify many alternative blocks of code to be executed



# The if Statement

```
if (condition) {  
    block of code to be executed if the condition is true  
}
```

## Example

```
if (time < 20) {  
    greeting = "Good day";  
}
```



# The else Statement

```
if (condition) {  
    block of code to be executed if the condition is true  
} else {  
    block of code to be executed if the condition is false  
}
```

## Example

```
if (time < 20) {  
    greeting = "Good day";  
} else {  
    greeting = "Good evening";  
}
```



# JavaScript Switch Statement

```
switch(expression) {  
    case n:  
        code block  
        break;  
    case n:  
        code block  
        break;  
    default:  
        default code block  
}
```

## Example

```
switch (new Date().getDay()) {  
    case 0:  
        day = "Sunday";break;  
    case 1:  
        day = "Monday";break;  
    case 2:  
        day = "Tuesday";break;  
    case 3:  
        day = "Wednesday";break;  
    case 4:  
        day = "Thursday";break;  
    case 5:  
        day = "Friday";break;  
    case 6:  
        day = "Saturday";break;  
}
```



# JavaScript Loop

## ❖ Different Kinds of Loops:

- **for** - loops through a block of code a number of times
- **for/in** - loops through the properties of an object
- **while** - loops through a block of code while a specified condition is true
- **do/while** - also loops through a block of code while a specified condition is true



# JavaScript For Loop

```
for (i = 0; i < cars.length; i++) {  
    text += cars[i] + "<br>";  
}
```

```
var person = {fname:"John", lname:"Doe", age:25};  
  
var text = "";  
var x;  
for (x in person) {  
    text += person[x];  
}
```



# The While Loop

```
while (condition) {  
    code block to be executed  
}
```

```
do {  
    code block to be executed  
}  
while (condition);
```



# Example

```
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>
<script>
cars = ["BMW", "Volvo", "Saab", "Ford"];
var i = 0;
var text = "";
while (cars[i]) {
  text += cars[i] + "<br>";
  i++;
}
document.getElementById("demo").innerHTML = text;
</script>
</body>
</html>
```

result

BMW  
Volvo  
Saab  
Ford



# JavaScript Events

- ❖ Here are some examples of HTML events:
  - An HTML web page has finished loading
  - An HTML input field was changed
  - An HTML button was clicked

```
<!DOCTYPE html>
<html>
<body>
<button
onclick="getElementById('demo').innerHTML=Date()">The time is?</button>
<p id="demo"></p>
</body>
</html>
```



# Common HTML Events

❖ Here is a list of some common HTML events:

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page



# Event Example 1

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 200px;
  height: 100px;
  border: 1px solid black;
}
</style>
</head>
<body>

<div onmousemove="myFunction(event)" onmouseout="clearCoor()"></div>
<p id="demo"></p>
```

Cont'

# Event Example 1

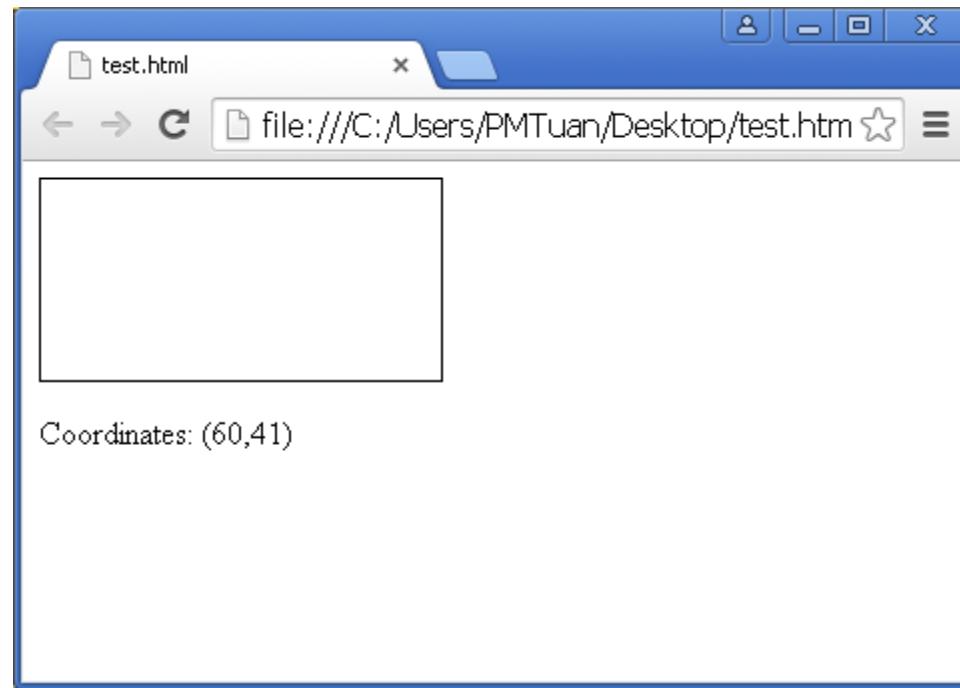
```
<script>
function myFunction(e) {
  var x = e.clientX;
  var y = e.clientY;
  var coor = "Coordinates: (" + x + "," + y + ")";
  document.getElementById("demo").innerHTML = coor;
}

function clearCoor() {
  document.getElementById("demo").innerHTML = "";
}
</script>

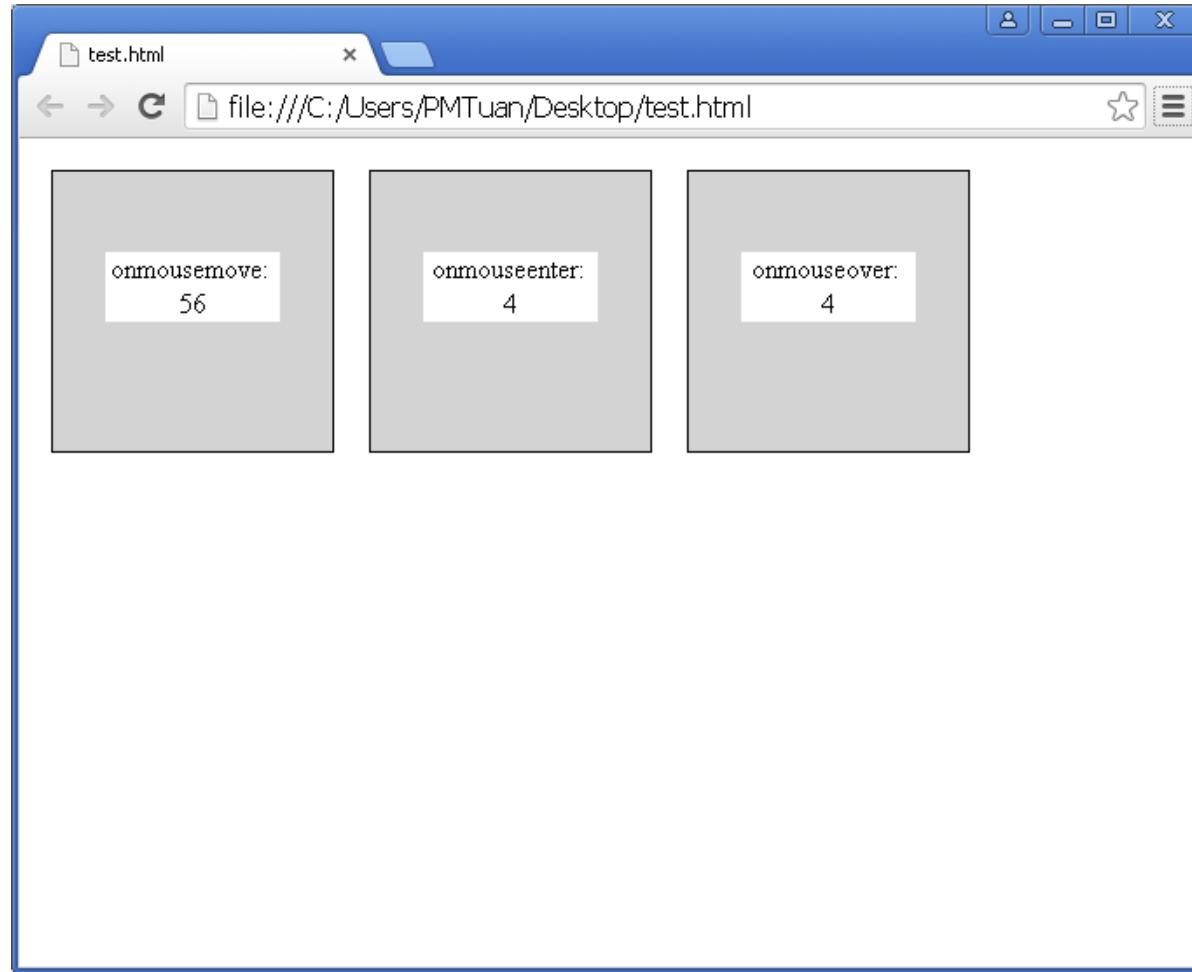
</body>
</html>
```



# Result 1



# Event Example 2



# JavaScript JSON

## ❖ What is JSON?

- JSON stands for **JavaScript Object Notation**
- JSON is lightweight data interchange format
- JSON is language independent \*
- JSON is "self-describing" and easy to understand

```
{"employees": [  
    {"firstName": "John", "lastName": "Doe"},  
    {"firstName": "Anna", "lastName": "Smith"},  
    {"firstName": "Peter", "lastName": "Jones"}  
]
```

# JSON Syntax Rules

- ❖ Data is in name/value pairs

- "firstName":"John"

- ❖ Data is separated by commas

- ❖ Curly braces hold objects

- {"firstName":"John", "lastName":"Doe"}

- ❖ Square brackets hold arrays

- "employees": [  
    {"firstName":"John", "lastName":"Doe"},  
    {"firstName":"Anna", "lastName":"Smith"},  
    {"firstName":"Peter", "lastName":"Jones"}  
]



# Example

```
<!DOCTYPE html>
<html>
<body>
<h2>Create Object from JSON String</h2>
<p id="demo"></p>
<script>
var text = '{"employees":[' +
'{"firstName":"John","lastName":"Doe" },' +
'{"firstName":"Anna","lastName":"Smith" },' +
'{"firstName":"Peter","lastName":"Jones" }]';
obj = JSON.parse(text);
document.getElementById("demo").innerHTML =
obj.employees[1].firstName + " " + obj.employees[1].lastName;
</script>
</body>
</html>
```



I T F

# Thanks



Môn học

# Lập Trình Mạng

Giảng Viên:

Phạm Minh Tuấn



# Giới thiệu

- ❖ Phạm Minh Tuấn
- ❖ E-mail: [pmtuan@dut.udn.vn](mailto:pmtuan@dut.udn.vn)
- ❖ Tel: 0913230910
- ❖ Khoa Công nghệ thông tin – Trường ĐHBK – ĐHĐN



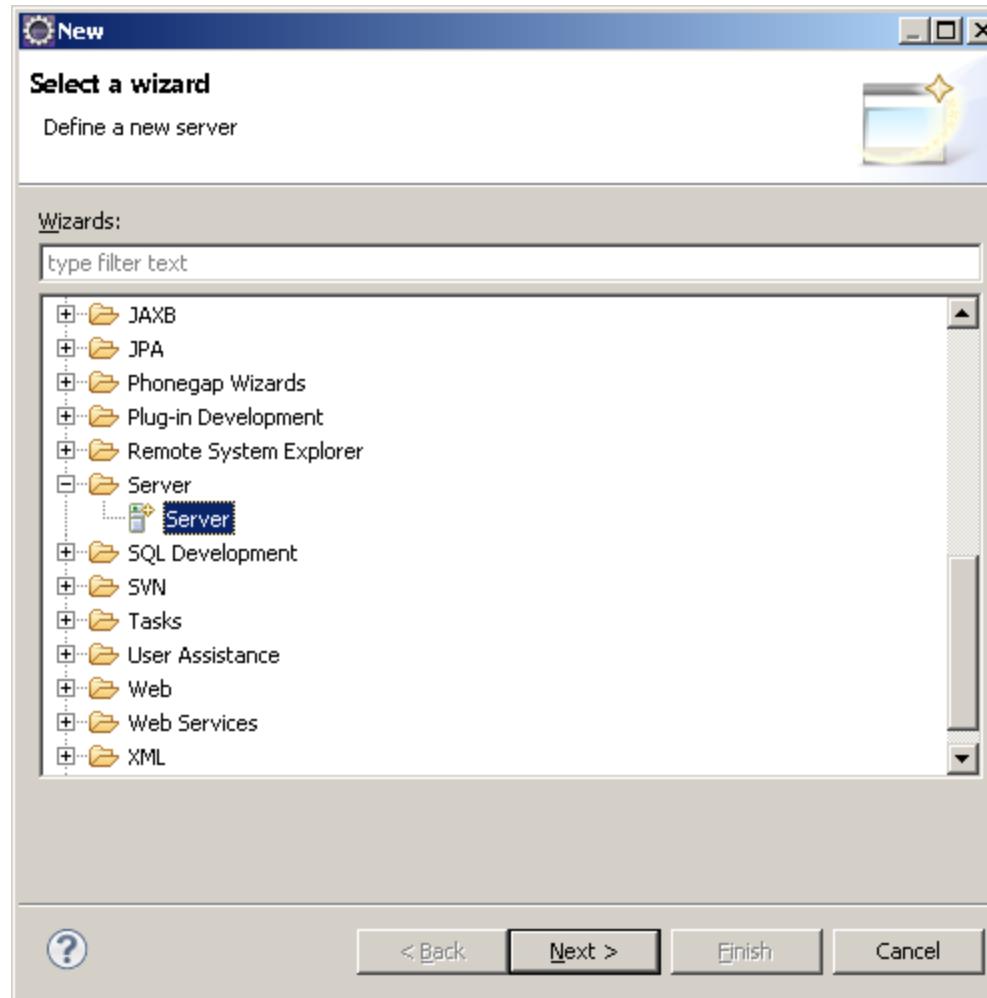
# Bài 7:

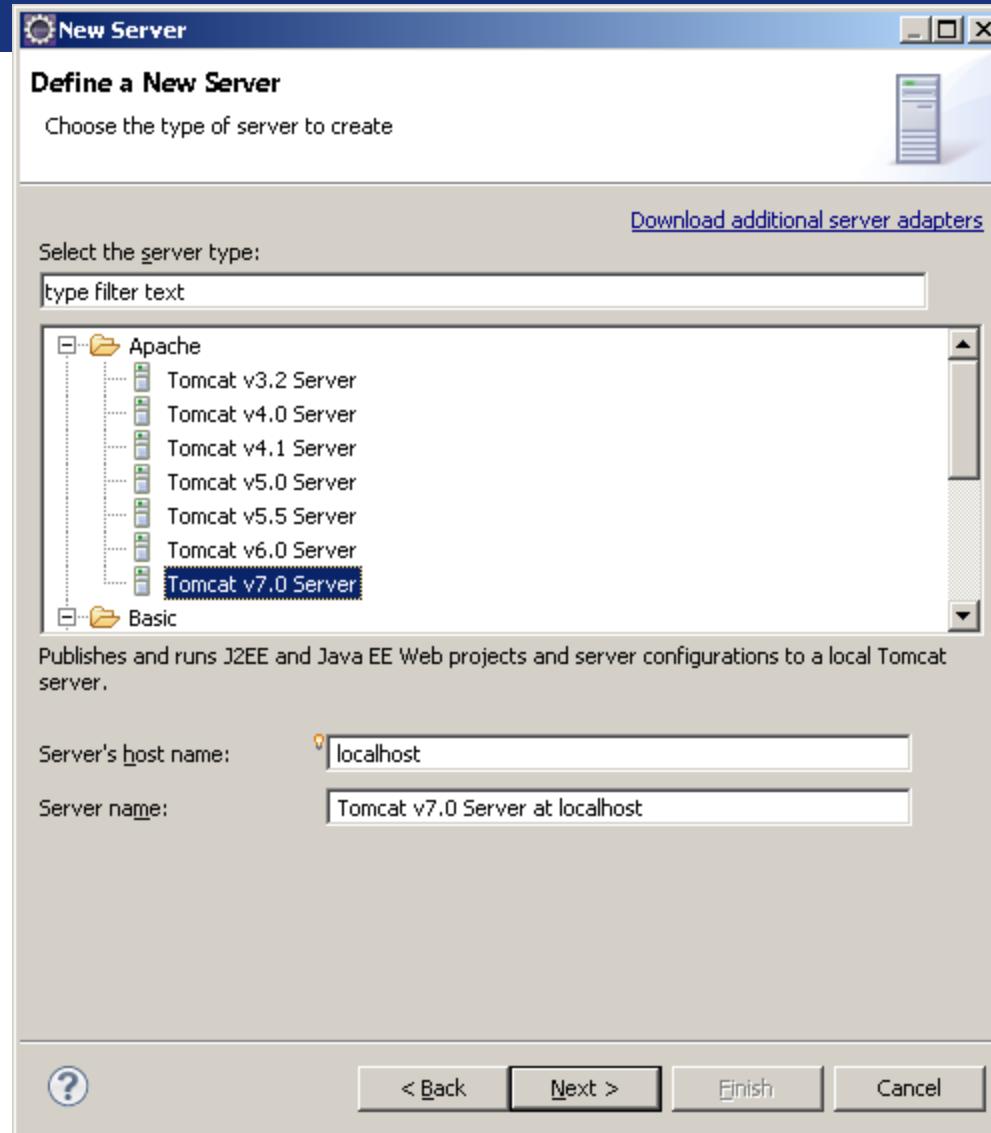
# JSP/Servlet

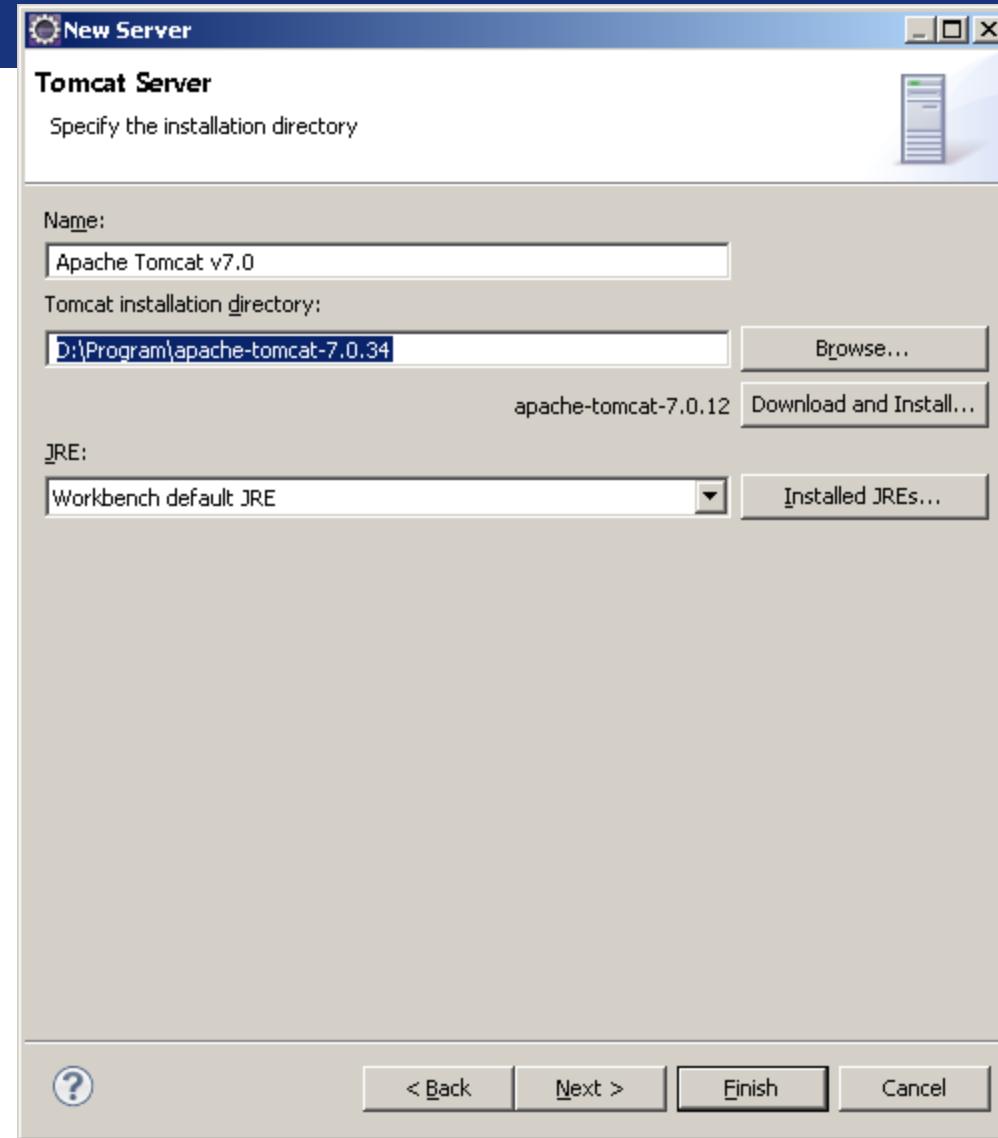
# Servlets & JSP: Setup

- ❖ Setup Tomcat (Java-enabled server)
  - Install Java
  - Unzip Tomcat
- ❖ Download Eclipse(IDE)
- ❖ Tell Eclipse about Tomcat
  - Click on Servers tab at bottom. R-click, New, Server, Apache, Tomcat v7.0, navigate to Tomcat installation folder
- ❖ Test the server

# Tell Eclipse about Tomcat









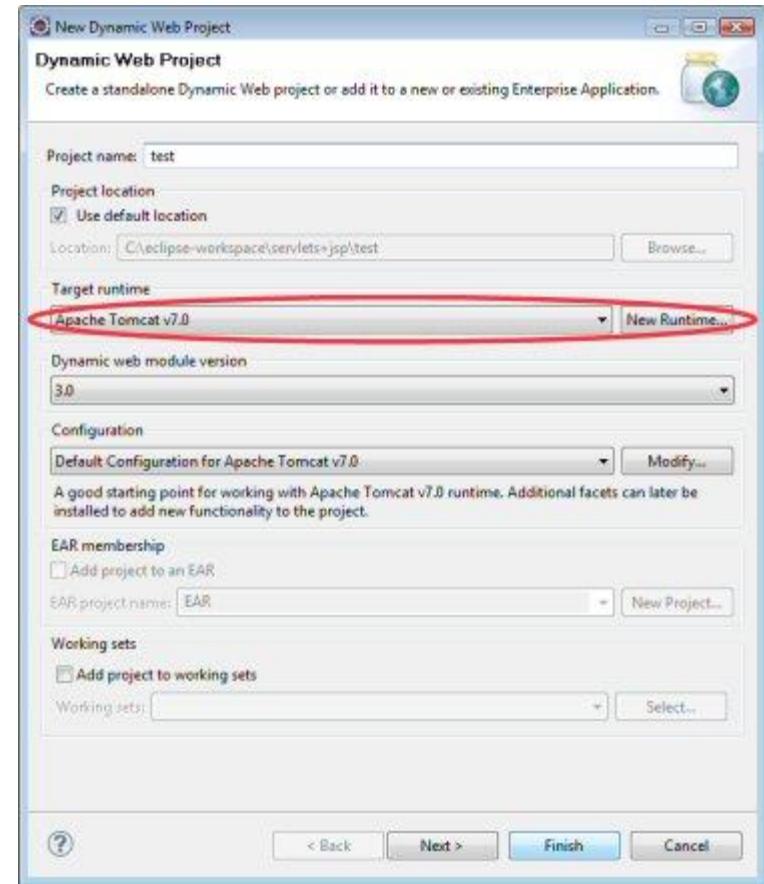
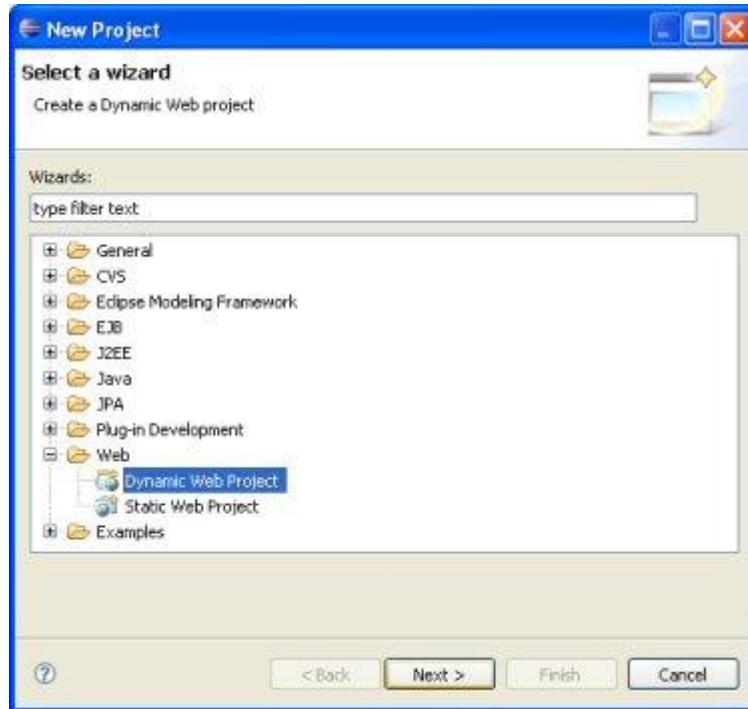
# Create a New Web App in Eclipse

## ❖ Make empty project

- File, New, Project, Web, Dynamic Web Project. Eclipse remembers the recent project types, so once you do this once, you can just do File, New, Dynamic Web Project.
- For "Target Runtime", choose "Apache Tomcat v7.0"
- Give it a name (e.g., "test").
- Accept all other defaults.



# Create a New Web App in Eclipse





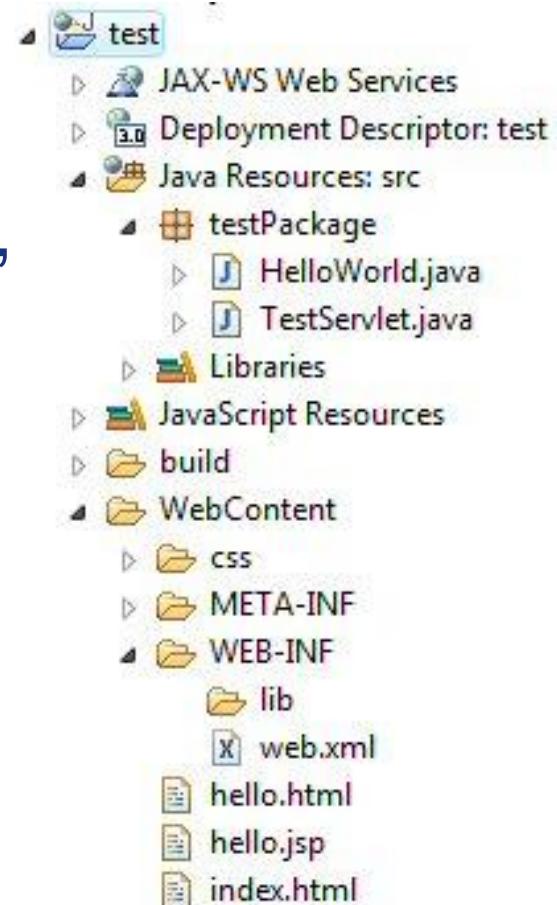
# Add Code to New Apps in Eclipse

## ❖ WebContent

- Regular Web files (HTML, JavaScript, CSS, JSP, images, etc.)

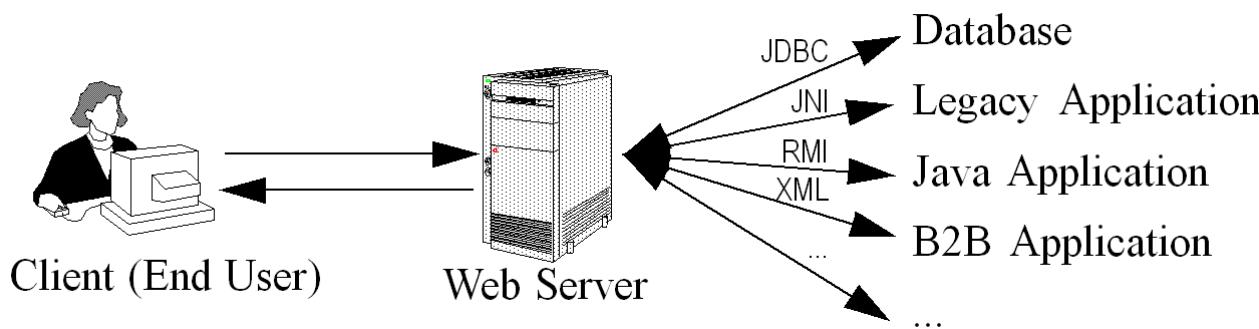
## ❖ src/testPackage

- Java code in testPackage package.



# Servlet Basics

- ❖ Servlet là một lớp Java được nạp tự động và chạy trên những web server có hỗ trợ Java. Những server như thế được gọi là ServletContainer (hay Java Server).
- ❖ Servlet tương tác với web client theo mô hình request-response dựa theo giao thức HTTP. Một số Servlet Container cũng hỗ trợ cả giao thức HTTPS dành cho các giao tác có tính bảo mật.



# LUỒNG XỬ LÝ SERVLET

❖ Luồng xử lý một servlet diễn ra qua các bước sau:

1. Client yêu cầu một servlet, tên của servlet là một phần của URL (vd:<http://www.music.com/music/SearchServlet>).
2. Web server nhận yêu cầu(**request**) và gửi nó tới servlet engine, là nơi quản lý và tạo ra các thể hiện của servlet.
3. Servlet engine sẽ gọi phương thức **service**(hoặc **doPost**, **doGet**) của servlet để xử lý yêu cầu.
4. Servlet sẽ tiếp nhận yêu cầu cùng với các tham số, tài nguyên khác, sau đó xử lý và tạo ra một kết quả trả lời tương ứng(**response**) và chuyển kết quả cho Web server.
5. Web server sẽ gửi trả kết quả này (**response**) cho client (Web Browser).

# Ví dụ servlet đơn giản

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.annotation.*;
import javax.servlet.http.*;
@WebServlet("/hello")
public class HelloWorld extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        out.println("Hello World");
    }
}
```

# Interpreting HelloWorld Servlet

## ❖ **@WebServlet("/address")**

- This is the URL *relative to the app name. More later.*

## ❖ **doGet**

- Code for an HTTP GET request. doPost also common.

## ❖ **HttpServletRequest**

- Contains anything that comes *from the browser*

## ❖ **HttpServletResponse**

- Used to send stuff *to the browser. Most common is getWriter for a PrintWriter that points at browser.*

## ❖ **@Override**

- General best practice when overriding inherited methods
  - But, I will omit on many of my PowerPoint slides to conserve space. Downloadable source has @Override.



# A Servlet That Generates HTML

- ❖ **Tell the browser that you're sending it HTML**
  - `response.setContentType("text/html");`
- ❖ **Modify the println statements to build a legal Web page**
  - Print statements should output HTML tags
- ❖ **Check your HTML with a formal syntax validator**



# HTML 5 Document Format

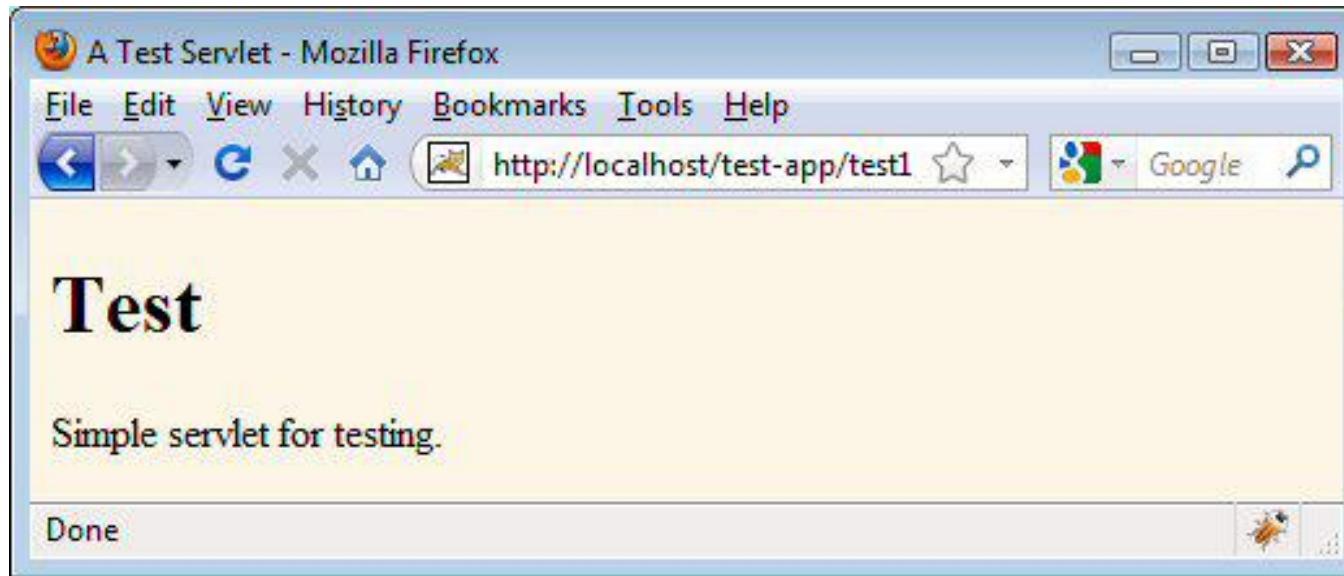
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8"/>
<link href="css/some-stylesheet.css"
      rel="stylesheet"/>
<script src="scripts/some-
script.js"></script>
</head>
<body>
...
</body>
</html>
```

# A Servlet That Generates HTML

```
@WebServlet("/test1")
public class TestServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println
            ("<!DOCTYPE html>¥n" +
             "<html>¥n" +
             "<head><title>A Test Servlet</title></head>¥n" +
             "<body bgcolor=#fdf5e6>¥n" +
             "<h1>Test</h1>¥n" +
             "<p>Simple servlet for testing.</p>¥n" +
             "</body></html>");
    }
}
```



# Kết quả





# A Servlet That Generates HTML - 2

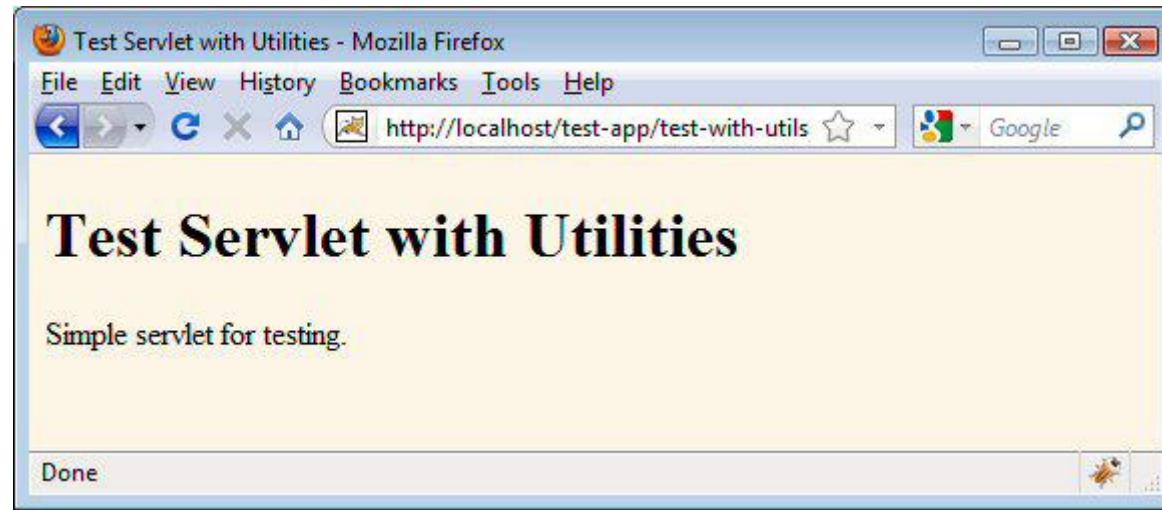
```
public class ServletUtilities {  
    public static String headWithTitle(String title) {  
        return("<!DOCTYPE html>\n" +  
               "<html>\n" +  
               "<head><title>" + title + "</title></head>\n");  
    }  
}
```

# A Servlet That Generates HTML - 2

```
@WebServlet("/test1")
public class TestServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        tring title = "Test Servlet with Utilities";
        out.println
            (ServletUtilities.headWithTitle(title)+
             "<body bgcolor="#fdf5e6">\n" +
             "<h1>Test</h1>\n" +
             "<p>Simple servlet for testing.</p>\n" +
             "</body></html>");
    }
}
```



# Kết quả





# Các phương thức của HttpServlet

- ❖ `doGet(HttpServletRequest req, HttpServletResponse resp)`: để xử lý yêu cầu HTTP GET
- ❖ `doPost(HttpServletRequest req, HttpServletResponse resp)`: để xử lý yêu cầu HTTP POST
- ❖ `init()` : khởi tạo các tham số, tham biến.
- ❖ `destroy()`: huỷ (giải phóng tài nguyên).



# 7.2:

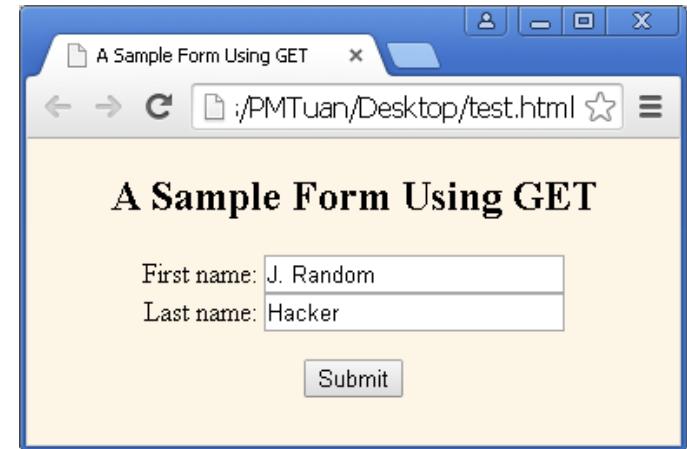
# HTML FORMS

# Servlet và HTML Forms

```
<!DOCTYPE html>
<HTML>
<HEAD>
<TITLE>A Sample Form Using GET</TITLE>
</HEAD>
```

```
<BODY BGCOLOR="#FDF5E6">
<H2 ALIGN="CENTER">A Sample Form Using GET</H2>
<FORM ACTION="http://localhost:8088/SomeProgram">
<CENTER>
First name: <INPUT TYPE="TEXT" NAME="firstName" VALUE="J. Random"><BR>
Last name: <INPUT TYPE="TEXT" NAME="lastName" VALUE="Hacker"><P>
<INPUT TYPE="SUBMIT"> <!-- Press this to submit form -->
</CENTER>
</FORM>

</BODY></HTML>
```





# Đọc Form Data từ Servlet

## ❖ **request.getParameter("name")**

- Returns URL-decoded value of first occurrence of name in query string
- Works identically for GET and POST requests
- Returns null if no such parameter is in query data

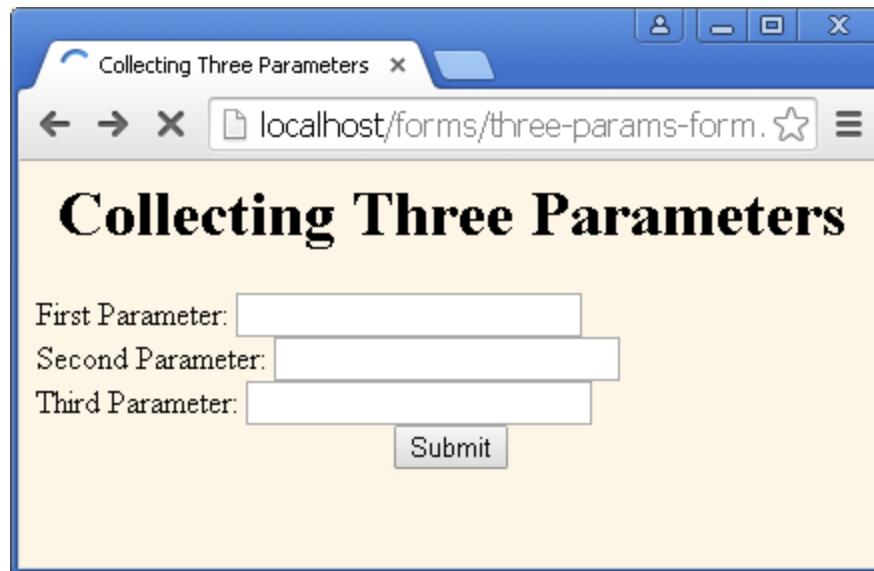
## ❖ **request.getParameterValues("name")**

- Returns an array of the URL-decoded values of *all* occurrences of name in query string
- Returns a one-element array if param not repeated
- Returns null if no such parameter is in query

## ❖ **request.getParameterNames()**

# Ví dụ - Form

```
<FORM ACTION="three-params">  
  First Parameter: <INPUT TYPE="TEXT" NAME="param1"><BR>  
  Second Parameter: <INPUT TYPE="TEXT" NAME="param2"><BR>  
  Third Parameter: <INPUT TYPE="TEXT" NAME="param3"><BR>  
  <CENTER><INPUT TYPE="SUBMIT"></CENTER>  
</FORM>
```



# Servlet đọc các tham số từ Form

```
public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    out.println(docType +
        "<HTML>\n" +
        "<HEAD><TITLE>" + title + "</TITLE></HEAD>\n" +
        "<BODY BGCOLOR=\\"#FDF5E6\\">\n" +
        "<H1 ALIGN=\\"CENTER\\">" + title + "</H1>\n" +
        "<UL>\n" +
        "  <LI><B>param1</B>: " +
        + request.getParameter("param1") + "\n" +
        "  <LI><B>param2</B>: " +
        + request.getParameter("param2") + "\n" +
        "  <LI><B>param3</B>: " +
        + request.getParameter("param3") + "\n" +
        "</UL>\n" +
        "</BODY></HTML>");
}
```



7.3:

# HTTP Request Headers



# HTTP Request Headers

## ❖ Nội dung:

- Đọc các đầu đề của các yêu cầu HTTP
- Hiểu các loại đầu đề khác nhau
- Tăng tốc độ download bằng cách nén trang



# Đọc đầu đề của HTTP Request

## ❖ Phương thức HttpServletRequest

### ▪ General

- getHeader
- getHeaders
- getHeaderNames

### ▪ Specialized

- getCookies
- getContentType
- getDateHeader
- ...



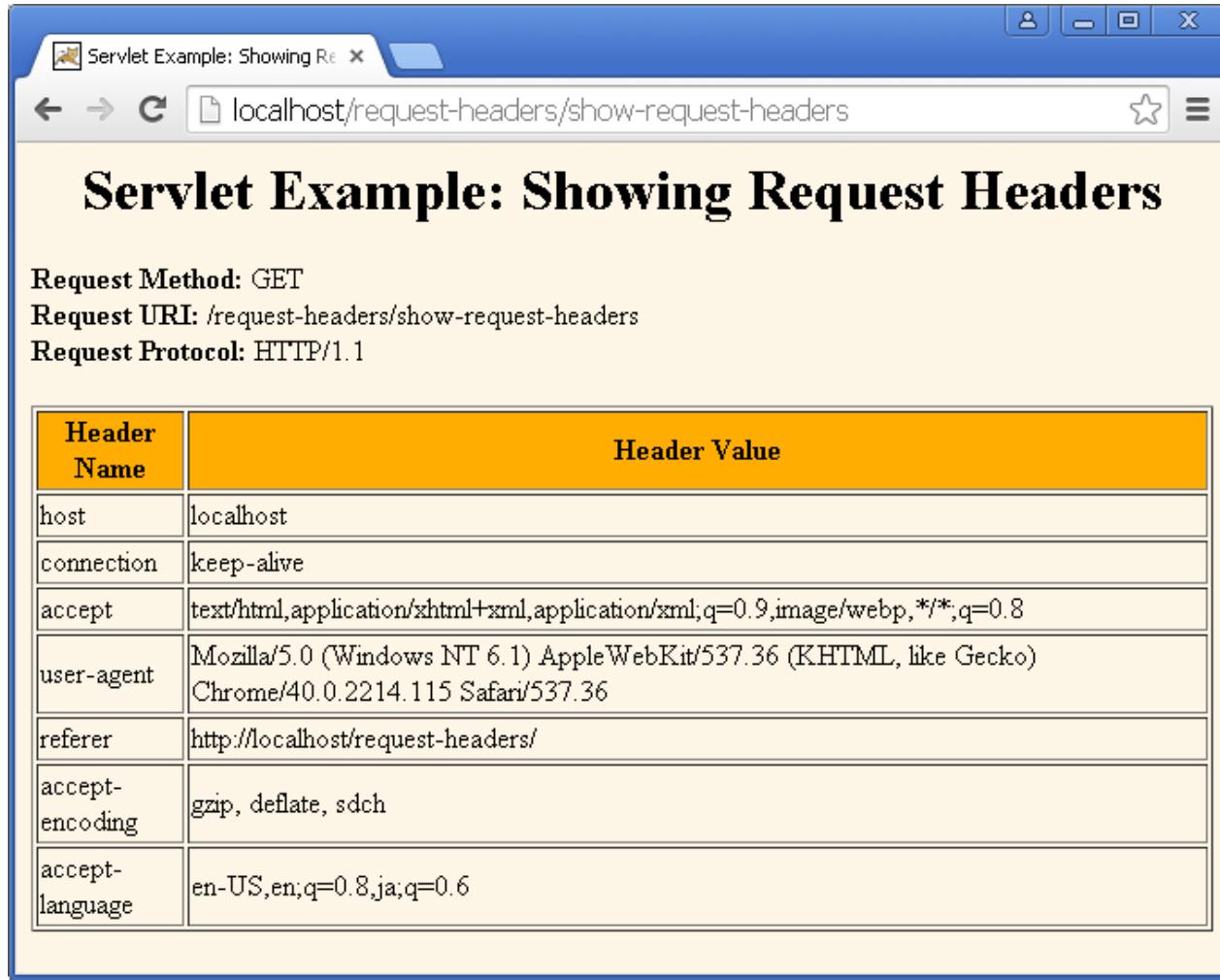
# Ví dụ

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Servlet Example: Showing Request Headers";
    String docType =
        "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 \" +
        "Transitional//EN\">\n";
    out.println(docType +
        "<HTML>\n" +
        "<HEAD><TITLE>" + title + "</TITLE></HEAD>\n" +
        "<BODY BGCOLOR=\"#FDF5E6\">\n" +
        "<H1 ALIGN=\"CENTER\">" + title + "</H1>\n" +
```

# Ví dụ (tiếp)

```
"<B>Request Method: </B>" +
request.getMethod() + "<BR>\n" +
"<B>Request URI: </B>" +
request.getRequestURI() + "<BR>\n" +
"<B>Request Protocol: </B>" +
request.getProtocol() + "<BR><BR>\n" +
"<TABLE BORDER=1 ALIGN=\"CENTER\">\n" +
"<TR BGCOLOR=\"#FFAD00\">\n" +
"<TH>Header Name<TH>Header Value");
Enumeration<String> headerNames = request.getHeaderNames();
while(headerNames.hasMoreElements()) {
    String headerName = headerNames.nextElement();
    out.println("<TR><TD>" + headerName);
    out.println("      <TD>" + request.getHeader(headerName));
}
out.println("</TABLE>\n</BODY></HTML>");
}
```

# Kết quả



**Servlet Example: Showing Request Headers**

**Request Method:** GET  
**Request URI:** /request-headers/show-request-headers  
**Request Protocol:** HTTP/1.1

Header Name	Header Value
host	localhost
connection	keep-alive
accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
user-agent	Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.2214.115 Safari/537.36
referer	http://localhost/request-headers/
accept-encoding	gzip, deflate, sdch
accept-language	en-US,en;q=0.8,ja;q=0.6



# Các loại đầu đề thông dụng

## ❖ Accept

- Can send different content to different clients.

## ❖ Accept-Encoding

- Indicates encodings (e.g., gzip or compress) browser can handle.

## ❖ Connection

- In HTTP 1.0, keep-alive means browser can handle persistent connection.



# Các loại đầu đề thông dụng

## ❖Cookie

- Gives cookies previously sent to client.

## ❖Host

- Indicates host given in original URL

## ❖Referer

- URL of referring Web page

## ❖User-Agent

- Best used for identifying *category of client*



# Sending Compressed Pages

- ❖ Servlet có khả năng gửi cho client trang web đã nén
- ❖ Giúp cho thời gian download giảm đáng kể.
- ❖ Đầu đề accept-encoding
  - gzip

# GzipUtilities

```
public class GzipUtilities {  
    public static boolean isGzipSupported  
        (HttpServletRequest request) {  
        String encodings = request.getHeader("Accept-Encoding");  
        return((encodings != null) && (encodings.contains("gzip")));  
    }  
    public static boolean isGzipDisabled  
        (HttpServletRequest request) {  
        String flag = request.getParameter("disableGzip");  
        return((flag != null) && (!flag.equalsIgnoreCase("false")));  
    }  
    public static PrintWriter getGzipWriter  
        (HttpServletResponse response) throws IOException {  
        return(new PrintWriter(new GZIPOutputStream  
            (response.getOutputStream())));  
    }  
}
```

# Gửi trang

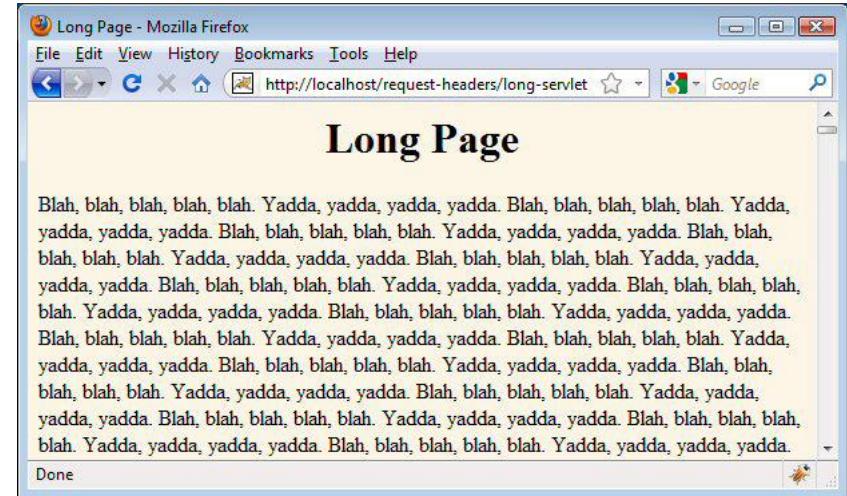
```
public class LongServlet extends HttpServlet {  
    @Override  
    public void doGet(HttpServletRequest request,  
                      HttpServletResponse response)  
        throws ServletException, IOException {  
        response.setContentType("text/html");  
  
        // Change the definition of "out" depending on whether  
        // or not gzip is supported.  
        PrintWriter out;  
        if (GzipUtilities.isGzipSupported(request) &&  
            !GzipUtilities.isGzipDisabled(request)) {  
            out = GzipUtilities.getGzipWriter(response);  
            response.setHeader("Content-Encoding", "gzip");  
        } else {  
            out = response.getWriter();  
        }  
    }  
}
```

# Gửi trang

```
String docType =
    "<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 " +
    "Transitional//EN\">\n";
String title = "Long Page";
out.println
(docType +
    "<HTML>\n" +
    "<HEAD><TITLE>" + title + "</TITLE></HEAD>\n" +
    "<BODY BGCOLOR=#FDF5E6>\n" +
    "<H1 ALIGN=CENTER>" + title + "</H1>\n");
String line = "Blah, blah, blah, blah, blah. " +
    "Yadda, yadda, yadda, yadda.";
for(int i=0; i<10000; i++)
    out.println(line);
out.println("</BODY></HTML>");
out.close(); // Needed for gzip; optional otherwise.
}
}
```

# Khảo sát

- ❖ Uncompressed (28.8K modem), Firefox and Internet Explorer:
    - > 50 seconds
  - ❖ Compressed (28.8K modem), Firefox and Internet Explorer:
    - < 5 seconds





## 7.4:

# HTTP Response Headers



# HTTP Request/Response

## ❖ Request

```
GET /servlet/SomeName HTTP/1.1
Host: ...
Header2: ...
...
HeaderN:
(Blank Line)
```

## ❖ Response

```
HTTP/1.1 200 OK
Content-Type: text/html
Header2: ...
...
HeaderN: ...
(Blank Line)
<!DOCTYPE ...>
<HTML>
<HEAD>...</HEAD>
<BODY>
...
</BODY></HTML>
```



# Phương thức Response Headers

- ❖ `response.setHeader(String headerName, String headerValue)`
  - Sets an arbitrary header
- ❖ `response.setDateHeader(String name, long millisecs)`
- ❖ `response.setIntHeader(String name, int headerValue)`
- ❖ `addHeader, addDateHeader, addIntHeader`



# Phương thức Response Headers

## ❖ setContentType

- Sets the Content-Type header.
- See table of common MIME types.

## ❖ addCookie

## ❖ sendRedirect

- Sets the Location header



# Common MIME Types

**Type**

application/msword  
application/octet-stream  
application/pdf  
application/postscript  
application/vnd.ms-excel  
application/vnd.ms-powerpoint  
application/x-gzip  
application/x-java-archive  
application/x-java-vm  
application/zip  
audio/basic  
audio/x-aiff  
audio/x-wav  
audio/midi  
text/css  
text/html  
text/plain  
text/xml  
image/gif  
image/jpeg  
image/png  
image/tiff  
video/mpeg  
video/quicktime

**Meaning**

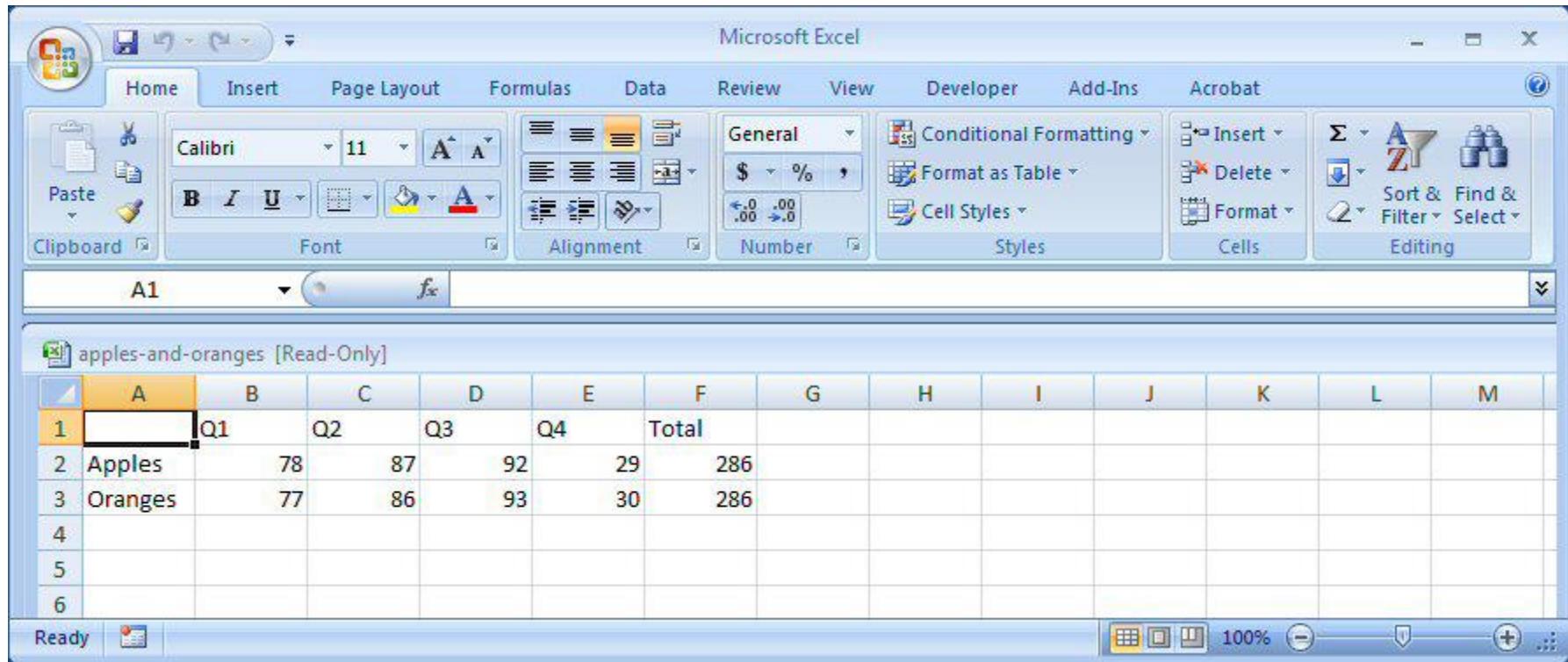
Microsoft Word document  
Unrecognized or binary data  
Acrobat (.pdf) file  
PostScript file  
Excel spreadsheet  
Powerpoint presentation  
Gzip archive  
JAR file  
Java bytecode (.class) file  
Zip archive  
Sound file in .au or .snd format  
AIFF sound file  
Microsoft Windows sound file  
MIDI sound file  
HTML cascading style sheet  
HTML document  
Plain text  
XML document  
GIF image  
JPEG image  
PNG image  
TIFF image  
MPEG video clip  
QuickTime video clip



# Ví dụ

```
@WebServlet("/apples-and-oranges")
public class ApplesAndOranges extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("application/vnd.ms-excel");
        PrintWriter out = response.getWriter();
        out.println("\tQ1\tQ2\tQ3\tQ4\tTotal");
        out.println("Apples\t78\t87\t92\t29\t=SUM(B2:E2)");
        out.println("Oranges\t77\t86\t93\t30\t=SUM(B3:E3)");
    }
}
```

# Kết quả



The screenshot shows a Microsoft Excel window with the title "Microsoft Excel". The ribbon menu is visible with tabs: Home, Insert, Page Layout, Formulas, Data, Review, View, Developer, Add-Ins, and Acrobat. The "Home" tab is selected. The ribbon also includes a "Clipboard" group, a "Font" group (with Calibri, 11pt, A, B, I, U buttons), an "Alignment" group, a "Number" group (with General, \$, %, , buttons), a "Styles" group (with Conditional Formatting, Format as Table, Cell Styles buttons), a "Cells" group (with Insert, Delete, Format buttons), and an "Editing" group (with Sort & Find, Filter, Select buttons). The active cell is A1. The worksheet is titled "apples-and-oranges [Read-Only]". The data is as follows:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		Q1	Q2	Q3	Q4	Total							
2	Apples	78	87	92	29	286							
3	Oranges	77	86	93	30	286							
4													
5													
6													

The status bar at the bottom shows "Ready" and "100%".

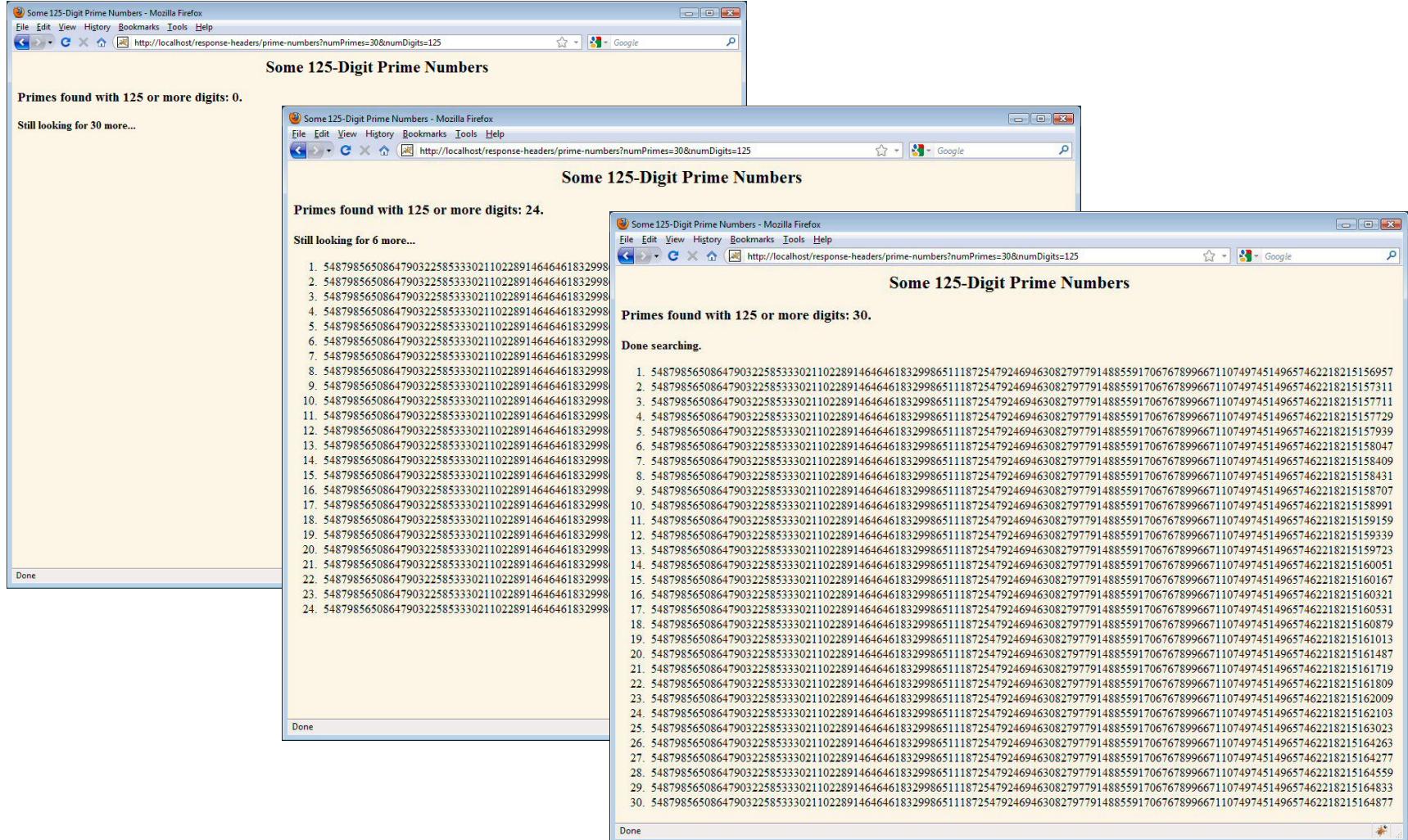
# Auto-Reloading Pages

- ❖ Idea: generate list of large (e.g., 150-digit) prime numbers
  - Show partial results until completed
  - Let new clients make use of results from others
- ❖ Demonstrates use of the Refresh header
- ❖ Shows how easy it is for servlets to maintain state between requests.

# VD: Tìm số nguyên tố lớn

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
    throws ServletException, IOException {
    int numPrimes = ServletUtilities.getIntParameter(request, "numPrimes", 50);
    int numDigits = ServletUtilities.getIntParameter(request, "numDigits", 120);
    PrimeList primeL = findPrimeList(primeListCollection, numPrimes, numDigits);
    if (primeL == null) {
        primeL = new PrimeList(numPrimes, numDigits, true);
        synchronized(primeListCollection) {
            if (primeListCollection.size() >= maxPrimeLists)
                primeListCollection.remove(0);
            primeListCollection.add(primeL);
        }
    }
    List<BigInteger> currentPrimes = primeL.getPrimes();
    int numCurrentPrimes = currentPrimes.size();
    int numPrimesRemaining = (numPrimes - numCurrentPrimes);
    boolean isLastResult = (numPrimesRemaining == 0);
    if (!isLastResult) response.setIntHeader("Refresh", 5);
```

# Kết quả



Index	Prime Number	Action
1.	54879856508647903225853302110228914646461832998	Done
2.	54879856508647903225853302110228914646461832998	Done
3.	54879856508647903225853302110228914646461832998	Done
4.	54879856508647903225853302110228914646461832998	Done
5.	54879856508647903225853302110228914646461832998	Done
6.	54879856508647903225853302110228914646461832998	Done
7.	54879856508647903225853302110228914646461832998	Done
8.	54879856508647903225853302110228914646461832998	Done
9.	54879856508647903225853302110228914646461832998	Done
10.	54879856508647903225853302110228914646461832998	Done
11.	54879856508647903225853302110228914646461832998	Done
12.	54879856508647903225853302110228914646461832998	Done
13.	54879856508647903225853302110228914646461832998	Done
14.	54879856508647903225853302110228914646461832998	Done
15.	54879856508647903225853302110228914646461832998	Done
16.	54879856508647903225853302110228914646461832998	Done
17.	54879856508647903225853302110228914646461832998	Done
18.	54879856508647903225853302110228914646461832998	Done
19.	54879856508647903225853302110228914646461832998	Done
20.	54879856508647903225853302110228914646461832998	Done
21.	54879856508647903225853302110228914646461832998	Done
22.	54879856508647903225853302110228914646461832998	Done
23.	54879856508647903225853302110228914646461832998	Done
24.	54879856508647903225853302110228914646461832998	Done

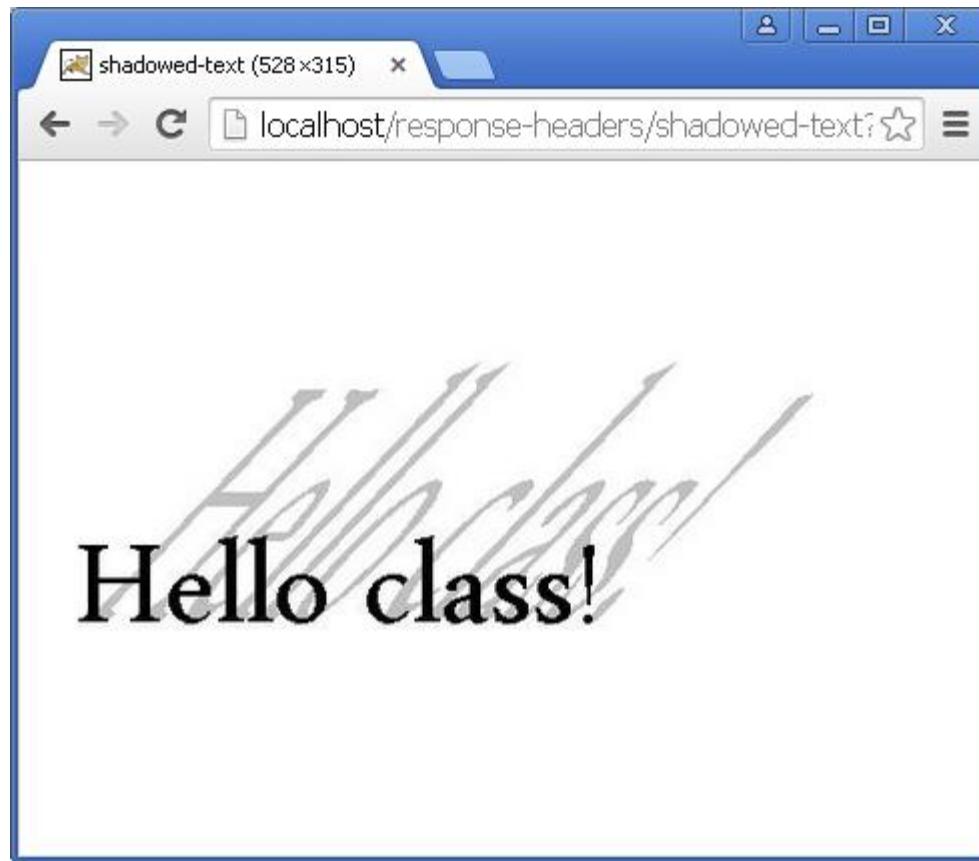


# Generate JPEG Images

- ❖ Create a BufferedImage
- ❖ Draw into the BufferedImage
  - Use normal AWT or Java 2D drawing methods
- ❖ Set the Content-Type response header
  - `response.setContentType("image/jpeg");`
- ❖ Get an output stream
  - `OutputStream out = response.getOutputStream`
- ❖ Send the BufferedImage in JPEG format to the output stream
  - `ImageIO.write(image, "jpg", out);`



# Ví dụ





# 7.5: Cookies



# Ý nghĩa của Cookies

## ❖ Ý tưởng

- Servlet sends a simple name and value to client.
- Client returns same name and value when it connects to same site

## ❖ Các kiểu sử dụng Cookies

- Identifying a user during an e-commerce session
- Avoiding username and password
- Customizing a site
- Focusing advertising

# Cookies and Focused Advertising

Amazon.com: Online Shopping for Electronics, Apparel, Computers, Books, DVDs & more.

Shop All Departments

Books

Movies, Music & Games

Digital Downloads

Kindle

Computers & Office

Electronics

Home & Garden

Grocery, Health & Beauty

Toys, Kids & Baby

Clothing, Shoes & Jewelry

Sports & Outdoors

Tools, Auto & Industrial

Check This Out

Selling on Amazon

Spring Deals

Textbook Buyback

Sell Back Your Books

Trade-In Your Stuff

Warehouse Deals

Kindle. #1 Bestselling Product on Amazon

#1 Bestselling Product on Amazon

Order now

Kindle. More 5-star reviews than any other product on Amazon:

This product is fantastic. I love using it. I have been an avid book reader for at least 40 years and it is hard to believe that it took me this long to buy this product. I may never go back to paperback/hardcover books again.

L. F. "book lover" (Utah) ★★★★☆

What Other Customers Are Looking At Right Now

Native Flutes for Relaxation

RED DEAD REDEMPTION

CHVRICE

Amazon.com home page  
for new visitor or visitor  
with cookies disabled.

Amazon.com home page for  
repeat visitor. Books shown  
are based on prior history.

Amazon.com: Online Shopping for Electronics, Apparel, Computers, Books, DVDs & more.

Shop All Departments

Books

Movies, Music & Games

Digital Downloads

Kindle

Computers & Office

Electronics

Home & Garden

Grocery, Health & Beauty

Toys, Kids & Baby

Clothing, Shoes & Jewelry

Sports & Outdoors

Tools, Auto & Industrial

Check This Out

Selling on Amazon

Spring Deals

Textbook Buyback

Sell Back Your Books

Trade-In Your Stuff

Warehouse Deals

Kindle. #1 Bestselling Product on Amazon

#1 Bestselling Product on Amazon

Order now

Kindle. More 5-star reviews than any other product on Amazon:

This product is fantastic. I love using it. I have been an avid book reader for at least 40 years and it is hard to believe that it took me this long to buy this product. I may never go back to paperback/hardcover books again.

L. F. "book lover" (Utah) ★★★★☆

What Other Customers Are Looking At Right Now

Native Flutes for Relaxation

RED DEAD REDEMPTION

CHVRICE

More Items to Consider

You viewed

Customers who viewed this also viewed

LOOK INSIDE!

Core Servlets and JavaServer Pages

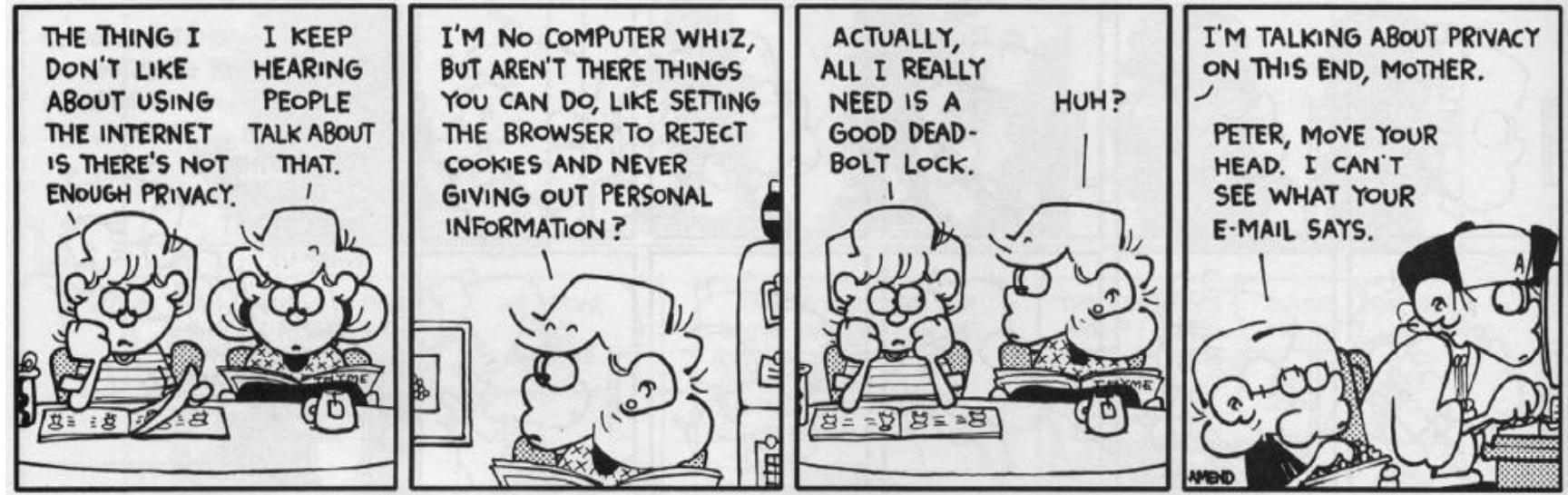
Murach's Java Servlets and JSP

Core Servlets and Javaserver Pages...

Core Servlets and JavaServer Pages...



# Cookies and Privacy



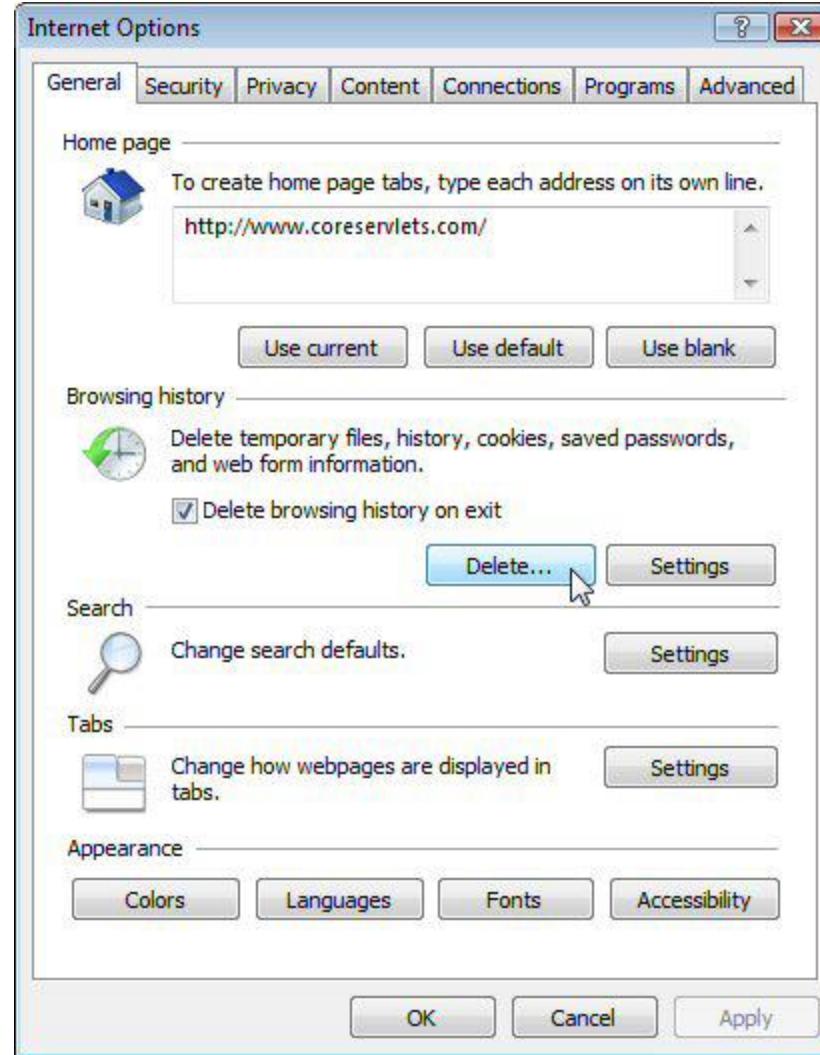
FoxTrot © 1998 Bill Amend.



# Một số vấn đề của Cookies

- ❖ The problem is privacy, not security.
  - Servers can remember your previous actions
  - If you give out personal information, servers can link that information to your previous actions
  - Servers can share cookie information through use of a cooperating third party like doubleclick.net
  - Poorly designed sites store sensitive information like credit card numbers directly in cookie

# Cách xóa Cookies



# Gửi Cookies tới Client

- ❖ Create a Cookie object.
  - Call the Cookie constructor with a cookie name and a cookie value, both of which are strings.
  - `Cookie c = new Cookie("userID", "a1234");`
- ❖ Set the maximum age.
  - To tell browser to store cookie on disk instead of just in memory, use `setMaxAge`
  - `c.setMaxAge(60*60*24*7); // One week`
- ❖ Place the Cookie into the HTTP response
  - Use `response.addCookie`
  - `response.addCookie(c);`



# Đọc Cookies từ Client

- ❖ Call `request.getCookies`
  - This yields an array of `Cookie` objects
- ❖ Loop down the array, calling `getName` on each entry until you find the cookie of interest
  - Use the value in application-specific way

```
String cookieName = "userID";
Cookie[] cookies = request.getCookies();
if (cookies != null)
    for(Cookie cookie: cookies)
        if (cookieName.equals(cookie.getName()))
            doSomethingWith(cookie.getValue());
```

# Ví dụ

```
public void doGet(HttpServletRequest request,
                   HttpServletResponse response)
    throws ServletException, IOException {
    for(int i=0; i<3; i++) {
        Cookie cookie = new Cookie("Session-Cookie-" + i,
                                    "Cookie-Value-S" + i);
        response.addCookie(cookie);
        cookie = new Cookie("Persistent-Cookie-" + i,
                            "Cookie-Value-P" + i);
        cookie.setMaxAge(3600);
        response.addCookie(cookie);
    }
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String docType =
        "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 \" +
        \"Transitional//EN\">\n";
    String title = "Active Cookies";
```

# Ví dụ (tiếp)

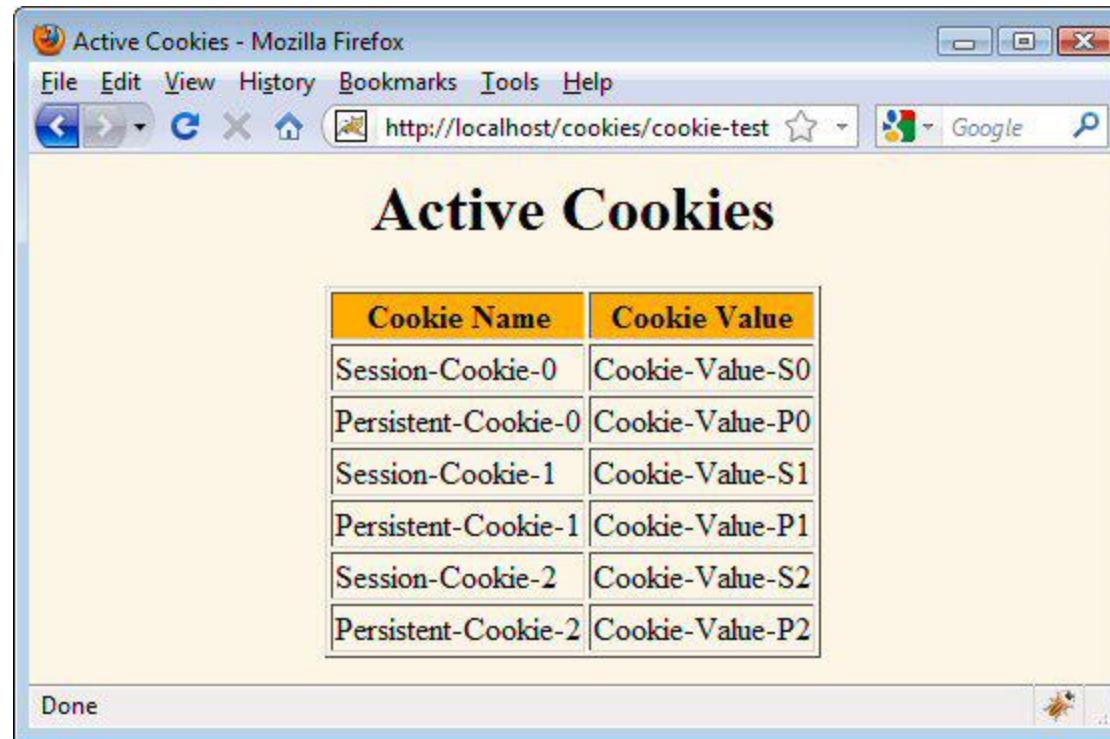
```
out.println(docType + "<HTML>\n" +
            "<HEAD><TITLE>" + title + "</TITLE></HEAD>\n" +
            "<BODY BGCOLOR=\"#FDF5E6\">\n" +
            "<H1 ALIGN=\"CENTER\">" + title + "</H1>\n" +
            "<TABLE BORDER=1 ALIGN=\"CENTER\">\n" +
            "<TR BGCOLOR=\"#FFAD00\">\n" +
            "  <TH>Cookie Name\n" +
            "  <TH>Cookie Value");
Cookie[] cookies = request.getCookies();
if (cookies == null) {
    out.println("<TR><TH COLSPAN=2>No cookies");
} else
    for(Cookie cookie: cookies)
        out.println("<TR>\n" +
                    "  <TD>" + cookie.getName() + "\n" +
                    "  <TD>" + cookie.getValue());
    out.println("</TABLE></BODY></HTML>");
}
```



# Kết quả - Kết nối lần 1



# Kết nối lần 2





## 7.6:

# JAVA Server Page - JSP



# Nội dung

- ❖ JSP và các khái niệm
- ❖ Tìm hiểu các thẻ JSP
- ❖ Truy xuất CSDL với JSP

# JSP LÀ GÌ?

- ❖ JSP (Java Server Page) cho phép ta chèn các mã lệnh Java vào các tập tin định dạng HTML hay XML để thực hiện các trang web động
  - Trái ngược với Servlet – vì sao?
- ❖ Công nghệ JSP là một thành phần trong họ Java, sử dụng ngôn ngữ Java như là ngôn ngữ kịch bản (script)



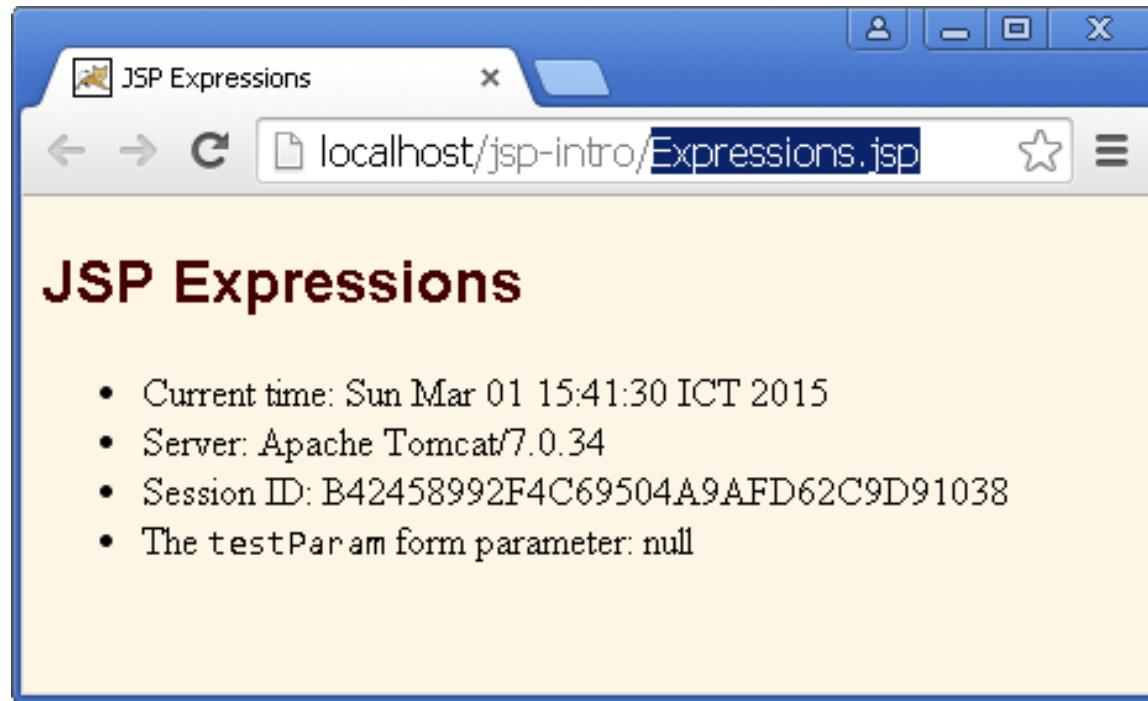
# Ví dụ một trang JSP đơn giản

Expressions.jsp

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<TITLE>JSP Expressions</TITLE>
</HEAD>
<BODY>
<H2>JSP Expressions</H2>
<UL>
    <LI>Current time: <%= new java.util.Date() %>
    <LI>Server: <%= application.getServerInfo() %>
    <LI>Session ID: <%= session.getId() %>
    <LI>The <CODE>testParam</CODE> form parameter:
        <%= request.getParameter("testParam") %>
</UL>
</BODY></HTML>
```



# Kết quả



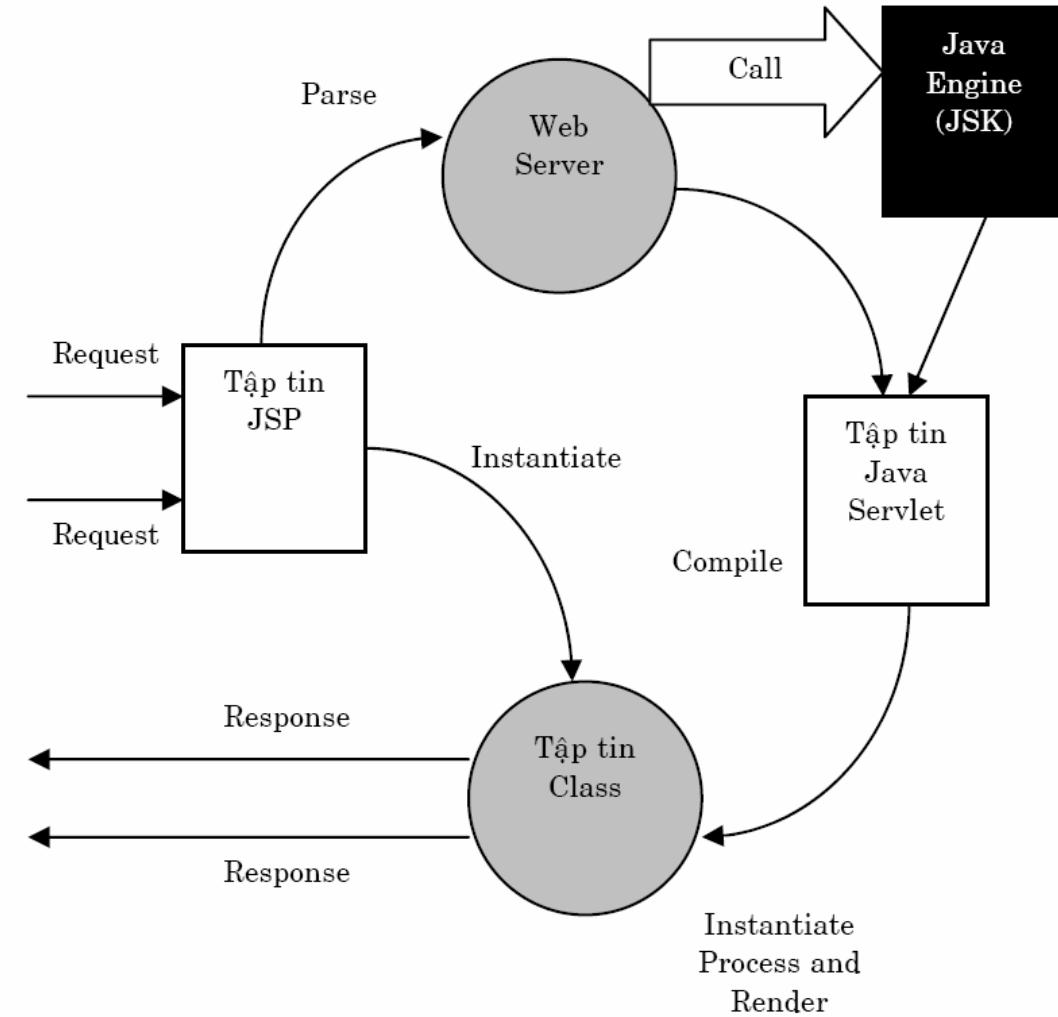


# JSP hay Servlet

- ❖ Viết JSP đơn giản và gọn hơn Servlet
- ❖ Servlet thường được dùng cho các chức năng xử lý phức tạp
  - Giao tiếp với Applet,
  - Bảo mật tài nguyên
  - Chứng thực mật khẩu,...
- ❖ JSP được dùng cho các thao tác đơn giản
  - Trình bày giao diện
  - Gọi các thành phần Servlet,...

# Cơ chế làm việc của JSP

- ❖ Khi gọi trang JSP lần đầu tiên, Web Server gọi trình biên dịch trang JSP (JDK) thành tập tin Java, sau đó biên dịch thành các class. Chạy các class để sinh ra các trang HTML





# JSP/Servlets in the Real World

## ❖ Airlines

- Delta Airlines, American Airlines,...

## ❖ Travel Sites

- Hotels.com, CheapTickets.com,...

## ❖ Financial Services

## ❖ Retail

- Nike.com, Sears.com,...

## ❖ Entertainment

## ❖ Military and Federal Government



# Cú pháp cơ bản

## ❖ HTML Text

- `<H1>Blah</H1>`
- servlet code: `out.print("<H1>Blah</H1>");`

## ❖ HTML Comments

- `<!-- Comment -->`

## ❖ JSP Comments

- `<%-- Comment --%>`



# Các thẻ lệnh JSP cơ bản

## ❖ Expressions

- Hiển thị biểu thức
- Format: <%= expression %>

## ❖ Scriptlets

- Đưa đoạn lệnh Java vào Html
- Format: <% code %>

## ❖ Declarations

- Khai báo biến
- Format: <%! code %>



# JSP Expressions <%= value %>

## ❖ Format

- <%= Java Expression %>

## ❖ Result

- Expression evaluated, converted to String, and placed into HTML page at the place it occurred in JSP page

## ❖ Examples

- <%= new java.util.Date() %>
- <%= request.getRemoteHost() %>



# JSP Scriptlets <% Code %>

## ❖ Format

- <% Java Code %>

## ❖ Example

- <% String queryData = request.getQueryString(); %>
- Attached GET data: <%= queryData %>
- <% response.setContentType("text/plain"); %>



I T F

# Thanks